Apéndice C

Recursos informáticos

En este último apéndice describiremos algunas rutinas útiles desarrolladas a partir de líneas de programa escritas en la aplicación **R** *para Windows, versión 2.2.1, que permiten simular de forma muy flexible y analizar de forma sistemática la mayoría de los problemas que hemos definido a lo largo de esta tesis. Las instrucciones y comandos aquí descritos*[1] *creemos que podrían resultar de utilidad para explorar extensiones en los alcances de lo estudiado en este trabajo.*

C.1. Sobre la elección de R

A la hora de decidir con qué paquete de aplicaciones informáticas daríamos soporte y agilidad a los cálculos que utilizamos en este trabajo, elegimos el programa **R** (v. 2.2.1), un entorno de programación muy flexible y de uso cada vez más extendido en el ámbito de la Estadística orientada a la investigación[2]. Las razones que hemos encontrado no son de nuestra exclusiva "cosecha", sin embargo hemos encontrado que mucha de la literatura específica de reciente edición apoya sus desarrollos y sus formas de calcular y presentar los datos en este programa[3].

La lógica empleada por este programa para escribir códigos es muy ordenada y potente, siendo una ventaja el hecho de tener disponible la llamada "programación orientada a objetos" que tiene el programa.

Finalmente, diremos también que al ser un programa de libre distribución, encontramos que el número de nuevas contribuciones (macros, programas, extensiones, etc.) que van realizando tanto los programadores más avanzados como los recién iniciados, es relativamente alto y muy dinámico. El carácter gratuito y la posibilidad de disponer del mismo mediante internet, hace que no haya restricciones relacionadas con altos costos de licencias ni preocuparse por sus renovaciones periódicamente.

[1] Por motivos de compatibilidad de fuentes del entorno LaTeX, es probable que no aparezcan algunos acentos al reproducir algunas líneas de comando.

[2] El website en donde se encuentra disponible es: http://www.r-project.org. La versión con la que hemos ejecutado todos los programas escritos en este trabajo es la 2.2.1, de diciembre de 2005. Una descripción general del programa y muy comprensible puede verse en: http://cran.r-project.org/doc/FAQ/R-FAQ.html#What-is-R_003f

[3] En Faraway (2005) y Verzani (2005), por ejemplo, pueden encontrarse algunos motivos sobre la conveniencia en el uso de este programa, los cuales compartimos con ambos autores.

Las secciones que siguen contienen la descripción general de cada familia de programas utilizados en este trabajo, todos escritos en **R**, los cuales se agrupan en series. Ya que en una misma serie los programas difieren solamente en parámetros puntuales, solamente reproduciremos las líneas de código en **R** de un solo programa, siendo el resto de los de la misma serie, análogos. En una gran parte de las líneas de programa, también hemos introducido algunos comentarios ilustrativos que explican con un poco más de detalle el propósito de cada línea referida, como así también algunas aclaraciones sobre el funcionamiento de ciertas funciones de **R**.

C.2. Serie de programas `BETAS`

C.2.1. Descripción general

Esta serie de programas corresponde a la generación de modelos de forma aleatoria, que serán aquellos que representarán nuestros procesos generadores de datos binarios, a partir de los cuales ajustaremos modelos y evaluaremos las estrategias de exploración de *MSR* que proponemos en nuestro trabajo. Para manejar adecuadamente las entradas y salidas que componen el mismo, los mismos se encuentran organizados por `MÓDULOS`.

La serie comprende 5 programas, uno para cada valor del Factor de centro, w. Si bien resulta más eficiente escribir uno solo en donde se pueda dar libremente el valor deseado a este factor, preferimos dejar escrito un programa para cada uno de ellos. En esta sección describimos solamente el primero de ellos, para $w = 5\,\%$.

C.2.2. Resumen de entradas y salidas

Inputs o entradas principales

Para un valor específico del Factor de centro[4], w, se ingresan los 3 niveles posibles que podrán tomar cada uno de los coeficientes β_j del vector de parámetros del modelo teórico completo de segundo orden. En el caso que hemos desarrollado, tomamos como niveles a los valores $-2, 0, 2$.

Outputs o salidas principales

- Determinación automática de los modelos disponibles que representen superficies de respuesta con máximos (matriz `M7`).
- Determinación automática del primer centro de experimentación (punto `O.1`) sobre la curva de nivel correspondiente al factor de centro elegido.
- Determinación automática de la distancia que separa el punto máximo del modelo teórico del primer centro de experimentación, ambos medidos en el plano de los factores x_1 y x_2.
- Para un punto cualquiera $\mathbf{x}_0 = (x_{01}, x_{02})'$, el programa permite calcular el valor de la función teórica en dicho punto para cualquier modelo que se elija de

[4] Como mencionáramos anteriormente, tomaremos los siguientes 5 niveles para este factor: 5 %, 10 %, 15 %, 20 % y 25 %.

los disponibles, $\pi(\mathbf{x}_0, \boldsymbol{\beta})$. En el programa, esta función está identificada como `f.02(x1,x2)`.

C.2.3. Detalle de entradas y salidas por módulos

En la tabla siguiente, describimos los inputs y outputs principales que componen cada uno de los módulos que conforman los programas de esta serie. Cada uno de estos programas comprende 8 módulos, para los cuales describiremos qué función desempeñan. Ésta puede verse en la figura (C.1):

PROGRAMAS DE LA SERIE "BETAS"		
INPUTS	MÓDULOS	OUTPUTS
• Niveles para cada uno de los β_j.	1	Matriz `M1`: matriz (729 x 6) en donde se detallan todos los posibles modelos que pueden formarse con los 3 niveles elegidos para los β_j, j = 1 a 6.
• Matriz `M1`	2 y 3	Matriz `M4`: matriz (81 x 8) en que aparecen sólo los modelos que corresponden a máximos, indicando los valores propios de cada modelo.
• Matriz `M4`	4	Matriz `M5`: matriz (81 x 10), similar a `M4`, pero con el agregado de las coordenadas del punto estacionario para cada fila (modelo).
• Matriz `M5`	5	Matriz `M6`: matriz (81 x 12), similar a M5, pero con el agregado del valor de la función teórica en el punto estacionario.
• Matriz `M6`	6	Matriz `M7`: matriz (60 x 12), similar a M6, pero sin los casos en que el punto estacionario coincida con (0, 0)′.
• Fila (modelo) cualquiera de `M7` • Punto cualquiera $\mathbf{x} = (x_{01}, x_{02})'$	7	Valor de la función teórica para el punto considerado, $\pi(\mathbf{x}, \beta)$, identificada como `f.02(x1, x2)`.
• Fila (modelo) cualquiera de `M7` • Nivel para el factor de centro, w	8	Coordenadas del primer centro de experimentación (punto `O.1`) y distancia entre `O.1` y el punto máximo de la función teórica (`d.OM`).

Figura C.1: Características principales de los programas de la serie `BETAS`.

C.2.4. Líneas de programa en R

Detallamos a continuación las líneas del programa `BETAS-W005-UNI-v3.txt`.

```
# PROGRAMA:                                        BETAS-W005-UNI-v3.txt
# FECHA REVISION:                                            02 ene 2006

# GENERALIDADES:
# Se trata de configurar una matriz de modelos con 2 factores de varia-
# bilidad, cuya respuesta tenga distribucion binomial. Para ello, se dan
# niveles arbitrarios al vector de coeficientes beta del modelo logisti-
```

```
# co, de tal modo que su representacion resulte en un maximo.
# Luego, y a partir de ese maximo, se ira buscando el optimo mediante u-
# na adaptacion de la MSR clasica, cuyos modelos de ajuste y criterios
# de seleccion de terminos se desarrollan en el programa CRJER-FXX-vX.txt.

# DETALLES:
# Se parte de un vector de 3 niveles para cada una de las p=6 betas que
# componen el predictor lineal, y se construye una matriz de 6 columnas
# por 3^6=729 filas, que seran todas las configuraciones posibles que
# cabe considerar para el vector de betas cuando cada uno de ellos tiene
# 3 niveles. Se clasifican luego los modelos mediante el criterio de los
# autovalores, y se definen funciones para calcular el valor de la fun-
# cion, para calcular maximos y para graficar superficies de respuesta y
# curvas de nivel.

# REQUISITOS:
# Ninguno.

# WORKING DIRECTORY:
setwd('C:/Documents and Settings/Arturo/Mis documentos/Tesis/Programas/W005')

# ---------------------------------------------------------------------
# MODULO 1
#
# Generacion de todos valores posibles del vector de coeficientes 'beta',
# con 3 niveles cada uno.
# ---------------------------------------------------------------------

# Elijo 3 valores posibles para los niveles que dare a beta, arbitraria-
# mente:
nb.1 <- -2
nb.2 <- 0
nb.3 <- -nb.1
nb <- c(nb.1, nb.2, nb.3)
nb -> b00 -> b01 -> b02 -> b11 -> b22 -> b12
M1 <- expand.grid(b00, b01, b02, b11, b22, b12)
# Doy nombres a las columnas:
c.M1 <- c("B.00","B.01","B.02","B.11","B.22","B.12")
names(M1) <- c.M1
dim(M1) # debe dar: [1] 729   6
```

```
# ------------------------------------------------------------------------
# MODULO 2
#
# Clasificacion de las SR en escala logistica a partir de la matriz de
# betas de 2do. orden.
# ------------------------------------------------------------------------

# Construccion de la matriz B, de coeficientes de segundo orden:
#     [B.11      0.5*B.12]
# B = [                  ], dim(B) = 2 x 2.
#     [0.5*B.12      B.22]
# M1 es la matriz que contiene las 729 configuraciones que puede tomar un
# vector de parametros beta correspondientes al modelo completo. En otras
# palabras, M1 contiene 729 modelos distintos. De alli saco los valores
# para construir B.
# Construyo un vector auxiliar aux.1 que me ponga todos los valores de
# la matriz B en una misma fila:
aux.1 <- c(M1[,"B.11"], 0.5*M1[,"B.12"], 0.5*M1[,"B.12"],M1[,"B.22"])
length(aux.1) # debe dar: 2916 (NB: 2916 = 729 x 4)
# Es un vector, con los elementos de segundo orden como columnas. Lo
# guardo como una matriz, aux.2:
aux.2 <- matrix(aux.1, nc=4)
c.aux.2 <- c("B11","0.5*B12","0.5*B12","B22")
dimnames(aux.2) <- list(NULL,c.aux.2)
dim(aux.2) # debe dar: [1] 729   4

# ------------------------------------------------------------------------
# MODULO 3
#
# Lineas de programa para calcular los autovalores de cada fila de la
# matriz de betas de 2\U{ba} orden.
# ------------------------------------------------------------------------

# Calculo los autovalores y autovectores de una matriz formada por una
# fila cualquiera de la matriz aux.2.
# Defino un loop para que me calcule los autovalores de cada una de las
# matrices que se formen a partir de cada fila de la matriz aux.2.
# Tomo una fila cualquiera, p. ej. la fila 178:
aux.2[178,]
# Los pongo en una matriz auxiliar aux.3:
aux.3 <- matrix(aux.2[178,], nc=2, byrow=T)
# Y calculo autovalores y autovectores de la matriz:
eigen(aux.3)
# Nota: esta funcion es una lista. Su primer elemento, [[1]] devuelve los
#       autovectores '$values', y el segundo, [[2]], los autovectores co-
```

```
#       rrespondientes, '$vectors'.
# Ahora hay que extender esto a todas las filas de la matriz aux.2, es
# decir, lleva a considerar tantas matrices B como filas tenga esta ma-
# triz aux.2.
# Antes de definir el loop, tengo que "declarar la existencia de la va-
# riable" de iteracion: av.1, a la que le asigno un modo numerico y un
# valor por defecto igual a cero. (Asi funciona R).
av.1 <- numeric(0)
for(i in 1:nrow(aux.2))
{
av.1[i] <- eigen(matrix(aux.2[i,], nc=2, byrow=T))[1]
}
# Nota: el [1] al final de la funcion eigen() le indica al loop que se
# quede con el 1\U{ba} elemento de la lista solamente, que son los autovalo-
# res.
length(av.1) # debe dar: [1] 729
# Resultado: 729 elementos, de 2 componentes cada uno, que son los auto-
# valores correspondientes a cada fila de la matriz M1.
# El objeto av.1 es una lista. Como tal, no sirve de mucho. Si los "des-
# listo", recupero los 729 x 2 = 1458 elementos en forma de vector colum-
# na. Poniendole delante el "invisible", no me los muestra al ejecutar el
# programa.
invisible(unlist(av.1))
# Para ver los elementos "deslistados":
# unlist(av.1)
# Estos elementos estan mostrados de a dos, de modo que con esta funcion
# se muestran los 729 pares de autovalores, que los puedo reagrupar en una
# matriz de 2 columnas, byrow=T. Redondeo a 4 cifras y lo guardo en la
# matriz av.2:
av.2 <- matrix(round(unlist(av.1), 4), nc=2, byrow=T)
# Las columnas de esta matriz av.2 seran los lambdas o autovalores.
c.av.2 <- c("Lam.1", "Lam.2")
dimnames(av.2) <- list(NULL, c.av.2)
dim(av.2) # debe dar: [1] 729 2
# Si los dos lambdas son < 0, la configuracion me da un maximo. Le digo
# al programa que me devuelva un "1" si es maximo y un "0" si no lo es:
aux.4 <- numeric(0)
aux.5 <- numeric(0)
for(i in 1:nrow(av.2))
{
aux.4[i] <- if(av.2[i,1] < 0 & av.2[i,2] < 0) (aux.5[i] <- 1)
else (aux.5[i] <- 0)
tipo <- as.vector(aux.5)
}
length(tipo) # debe dar: [1] 729
```

```
# Ya que hay compatibilidad dimensional, pego este vector a la matriz
# av.2. Tengo una matriz auxiliar av.3 de 3 columnas, que me da la natu-
# raleza de cada par de los autovalores correspondientes a los coeficien-
# tes beta de segundo orden.
av.3 <- cbind(av.2, tipo)
dim(av.3) # debe dar: [1] 729   3
# Creo una matriz M2, en donde pego por columnas a la matriz base, M1,
# con av.3:
M2 <- cbind(M1, av.3)
colnames(M2)
# debe dar:
# [1] "B.00"  "B.01"  "B.02"  "B.11"  "B.22"  "B.12" "Lam.1" "Lam.2" "tipo"
dim(M2) # debe dar: [1] 729   9
# Filtro esta matriz, para que figuren solo los maximos:
M3 <- M2[M2[,"tipo"]==1,]
nr.M3 <- nrow(M3)
nc.M3 <- ncol(M3)
dim(M3) # debe dar: [1] nr.M3 nc.M3
# Se ha simplificado la matriz, de 729 filas a
nr.M3 # filas
# Reescribo la matriz como M4, sin la columna tipo, que son las columnas
# 1 a 8:
M4 <- M3[,1:8]
colnames(M4)
# debe dar:
# [1] "B.00"  "B.01"  "B.02"  "B.11"  "B.22"  "B.12"  "Lam.1"  "Lam.2"
dim(M4) # debe dar: [1] 81  8

# ------------------------------------------------------------------------
# MODULO 4
#
# Deteriminacion de las coordenadas del punto estacionario correspondien-
# te a cada beta de la matriz M4.
# ------------------------------------------------------------------------

# Esto puede calcularse a partir de la matriz M4, que contiene solo ma-
# ximos.
# Para calcular los puntos estacionarios, se utiliza la formula:
# Xs=(x1s,x2s)'= -0.5 * B^-1 * b, (es matricial),
# que coinciden todos con los hallados para la SR. Estos puntos deben sa-
# tisfacer la condicion clasica:
# d/dx[ST()] = 0, (es vectorial).
# Evaluando este punto en la funcion, resulta el maximo de la misma:
# ST(Xs) = exp(B.00 + X's*b + X's*B*Xs)/[1+exp(B.00 + X's*b + X's*B*Xs)].
# (a) Matriz B y su inversa:
```

```
# Tomo como punto de partida a la matriz M4, haciendo algo similar a lo
# que se hizo con el vector aux.1:
aux.6 <- c(M4[,"B.11"], 0.5*M4[,"B.12"], 0.5*M4[,"B.12"], M4[,"B.22"])
aux.7 <- matrix(aux.6, nc=4, byrow=F)
dim(aux.7) # debe dar: [1] 81  4
# Ahora hay que armar las 81 matrices B para cada elemento de aux.7.
# Solo armarlos como matriz, pues ya estan multiplicados por 0.5.
aux.08 <- numeric(0)
for(i in 1:nrow(aux.7))
{
aux.08[i] <- list(matrix(c(aux.7[i,1], aux.7[i,2], aux.7[i,3],
aux.7[i,4]),
nc=2, byrow=T))
}
# El resultado es una lista, de 81 elementos, cada uno de los cuales con-
# sistente en la matriz B que corresponde a cada fila de la matriz aux.7.
# Calculo de las inversas de aux.08:
aux.09 <- lapply(aux.08, solve)
# Prueba que los respectivos productos de los elementos de ambas listas
# dan como resultado la matriz identidad:
# Para un elemento cualquiera:
aux.08[[1]]%*%aux.09[[1]] # debe dar la matriz identidad.
# Ahora para todos los elementos:
aux.10 <- numeric(0)
for(i in 1:length(aux.08))
{
aux.10[i] <- list(aux.09[[i]]%*%aux.08[[i]])
}
# Si deslisto esta lista, y si el test esta bien, al reagruparlos como
# una matriz de 4 columnas, con byrow=T, deberia obtener en todos los ca-
# sos el vector 1 0 0 1.
aux.11 <- round(matrix(unlist(aux.10), nc=4, byrow=T), 1)
aux.12 <- numeric(0)
aux.13 <- numeric(0)
for(i in 1:nrow(aux.11))
{
aux.12[i] <- if(
aux.11[i,1]==1 & aux.11[i,2]==0 & aux.11[i,3]==0 & aux.11[i,4]==1)
assign("deslist", "OK") else assign("deslist", "NG")
aux.13 <- aux.12[i]
}
# Los resultados se pueden ver en la variable 'deslist'. Deben ser todos
# OK.
# Nota: en todos los casos debe obtenerse el vector 1 0 0 1
# (b) Vector b:
```

```
# La matriz de partida sigue siendo la M4, la completa. Conviene ponerla
# como una lista:
aux.14 <- numeric(0)
for(i in 1:nrow(M4))
{
aux.14[i] <- list(c(M4[i,"B.01"], M4[i,"B.02"]))
}
# (c) Determinacion de los Xs:
# Hay que calcular:
# Xs = (x1s,x2s)'= -0.5 * B^-1 * b.
# La inversa B^-1 corresponde a aux.09 y b corresponde a aux.14, que son
# los mismos que los calculados en la SR.
# Y creo una lista para los 81 elementos:
aux.15 <- numeric(0)
for(i in 1:length(aux.09))
{
aux.15[i] <- list(-0.5*aux.14[[i]]%*%aux.09[[i]])
}
length(aux.15) # debe dar: [1] 81
aux.16 <- matrix(round(unlist(aux.15), 4), nc=2, byrow=T)
c.aux.16 <- c("X1.S","X2.S")
dimnames(aux.16) <- list(NULL, c.aux.16)
# Compruebo que la derivada primera de la ST se anula en Xs:
# b + 2*B*Xs = 0,
# en donde b es aux.14, B es aux.08 y Xs es aux.15.
# Para elementos sueltos:
aux.14[[1]]+2*aux.08[[1]]%*%t(aux.15[[1]])
# debe dar un vector de dos ceros.
aux.17 <- numeric(0)
for(i in 1:length(aux.14))
{
aux.17[i] <- list(aux.14[[i]]+2*aux.08[[i]]%*%t(aux.15[[i]]))
}
aux.18 <- unlist(aux.17)
M.D <- round(matrix(aux.18, nc=2, byrow=T), 2)
# Nota: en la matriz M.D se guardan todos estos datos, que deben ser to-
# dos ceros. Si esto es asi, los puntos estacionarios estan bien calcu-
# lados. Pego todo en una nueva matriz nueva, M5:
M5 <- cbind(M4,aux.16)
colnames(M5)
# Debe dar:
# [1]  "B.00"  "B.01"  "B.02"  "B.11"  "B.22"  "B.12"  "Lam.1"  "Lam.2"
# [9] "X1.S"   "X2.S"
dim(M5) # debe dar: [1] 81 10
# OK.
```

```
# ------------------------------------------------------------------------
# MODULO 5
#
# Calculo de los valores maximos de las SR teoricas.
# ------------------------------------------------------------------------

# Todo debe calcularse a partir de la ultima matriz del modelo, M5. As
# usual, declaro primero todas las variables, diciendo que son numericas
# y que su valor por default es cero.
eta.M5 <- numeric(0)
Pi.M5 <- numeric(0)
Pi.Max <- numeric(0)
for(i in 1:nrow(M5))
{
eta.M5[i] <- M5[i,"B.00"]+
M5[i,"B.01"]*M5[i,"X1.S"]+
M5[i,"B.02"]*M5[i,"X2.S"]+
M5[i,"B.11"]*M5[i,"X1.S"]*M5[i,"X1.S"]+
M5[i,"B.22"]*M5[i,"X2.S"]*M5[i,"X2.S"]+
M5[i,"B.12"]*M5[i,"X1.S"]*M5[i,"X2.S"]
Pi.M5[i] <- exp(eta.M5[i])/(1+exp(eta.M5[i]))
Pi.Max[i] <- round(Pi.M5[i], 5)
}
# Nota: la funcion Pi.Max da exactamente igual a Pi.M5, pero redondeado.
summary(Pi.Max)
# Agrego una primera columna indizadora de la matriz:
M6 <- cbind(ID=1:nrow(M5), M5, Pi.Max)
colnames(M6)
# debe dar:
# [1] "ID"       "B.00"     "B.01"     "B.02"     "B.11"    "B.22"
# [9] "B.12"     "Lam.1"    "Lam.2"    "X1.S"     "X2.S"     "Pi.Max"
dim(M6)    # debe dar: [1] 81 12

# ------------------------------------------------------------------------
# MODULO 6
#
# Lineas de programa para filtrar la matriz M6, de modo que la misma no
# contenga  modelos en los que el punto (0, 0) del plano de los factores
# sea el punto estacionario.
# ------------------------------------------------------------------------

# Se evitan aquellos modelos que tengan por maximo al punto (0, 0) para
# no dificultar calculos -sobre todo, los que tengan que ver con inver-
# siones de matrices y tambien porque no tiene sentido conjeturar que un
```

```
# proceso tiene su maximo en el punto Pi = 0.
M6[M6$X1.S==0 & M6$X2.S==0,c("ID", "X1.S","X2.S")]
si.cero <- paste("Numero de modelos que contienen al punto (0, 0):",
nrow(M6[M6$X1.S==0 & M6$X2.S==0,c("ID", "X1.S","X2.S")]))
no.cero <- paste("Numero de modelos que NO contienen al punto (0, 0):",
nrow(M6[M6$X1.S!=0 & M6$X2.S!=0,]))
si.cero
no.cero
# Redefino esta nueva matriz y le pongo el nombre M7:
M70 <- M6[M6$X1.S!=0 & M6$X2.S!=0, 2:ncol(M6)]
# Al poner el rango de columnas 2:ncol(M6) se omite la antigua columna
# de "ID".
# Creo una nueva, con los casos ya depurados:
ID <- 1:nrow(M70)
M7 <- cbind(ID,M70)
dim(M7)
# Debe dar:  [1] 60 12

# ---------------------------------------------------------------------
# MODULO 7
#
# Evaluacion del valor de la funcion logistica para un beta elegido de
# la matriz completa y filtrada del modelo, M7, y para un valor generico
# X = (x1, x2)'.
# ---------------------------------------------------------------------

k.2 <- numeric(0)
# Ahora hay que elegir una fila de M7 ("k.2"), 1 <= k.2 <= nrow(M7)
# Un caso particular:
k.2 <- 45
# Y sino, que elija el programa una al azar:
# k.2 <- sample(M7[,"ID"],1)
# Guardo toda la info en un vector:
b.k.2 <- M7[k.2,]
# Y defino la funcion:
f.02 <- function(x1, x2)
{
eta.k.2 <- M7[k.2,"B.00"]+
M7[k.2,"B.01"]*x1+
M7[k.2,"B.02"]*x2+
M7[k.2,"B.11"]*x1*x1+
M7[k.2,"B.22"]*x2*x2+
M7[k.2,"B.12"]*x1*x2
Pi.k.2 <- exp(eta.k.2)/(1+exp(eta.k.2))
round(Pi.k.2, 6)
```

```
}
# Ahora se puede evaluar f.02 para cualquier par de valores x1,x2 y obte-
# ner asi el valor de la funcion teorica correspondiente al modelo dado
# por k.2 y evaluado en el par de puntos elegidos:
# f.02(x1, x2)
# Para un mismo k (k.1=k.2), probar que f.02 y f.01 dan lo mismo para un
# mismo par de puntos (x1,x2):
# f.01(M7[k.1,"X1.S"], M7[k.1,"X2.S"])
# f.02(M7[k.1,"X1.S"], M7[k.1,"X2.S"])
# Preparo los valores de X1 y X2 para graficar la superficie y las curvas
# de nivel en el programa MEDCA-UNI-v1.txt:
# Defino las escalas de los ejes X1 y X2 para el grafico de persp(): tomo
# un entorno de amplitud arbitraria, "escala", alrededor de cada coorde-
# nada del punto estacionario de ambos ejes.
# Para graficar curvas de nivel, esto tambien funciona bien.
escala <- 2.5
X1.inf <- -(abs(b.k.2[,"X1.S"])+escala)
X1.sup <- -X1.inf
X1 <- seq(from=X1.inf, to=X1.sup, length=50)
X2.inf <- -(abs(b.k.2[,"X2.S"])+escala)
X2.sup <- -X2.inf
X2 <- seq(from=X2.inf, to=X2.sup, length=50)
# Queda definida una grilla X1,X2 de valores para los cuales se evalua-
# ra la funcion 'outer()', con la que se podran graficar las superficies
# y las curvas de nivel.
# Con la funcion 'outer()' preparo los puntos para graficarlos:
out.f.01 <- outer(X1, X2, f.02)
# Su utilizacion se hara en el programa MEDCA antes mencionado.

# ------------------------------------------------------------------------
# MODULO 8
#
# Calculo de los puntos de interseccion entre la recta R1 que pasa por
# (0,0) y por X.max y que corta a la curva de nivel correspondiente a
# w%*Pi.max=cte
# ------------------------------------------------------------------------

# El hecho de especificar el modulo como "w% Pi Max" significa que este
# valor w sera el porcentaje del valor teorico Pi.Max con el cual se de-
# fine un plano paralelo al plano de los factores, que corta a la super-
# ficie teorica a una altura igual a w%*Pi.Max, definiendo una curva de
# nivel sobre dicha superficie.
# Se busca cortar dicha curva de nivel con una recta que una al maximo
# teorico con el origen, y asi determinar un primer centro de experimen-
# tacion.
```

```
# Ecuacion de la recta R1 que pasa por X.Max y por (0,0):
# x2 = mR1 * x1. Tomo el mismo k.2 que anteriormente:
k.2 <- 45
m.R1 <- M7[k.2, "X2.S"]/M7[k.2, "X1.S"]
# Asigno un valor para w, que lo guardo en w.00:
w.00 <- 5/100
Pi.00 <- w.00*M7[k.2,"Pi.Max"]
logit.00 <- log(Pi.00/(1-Pi.00))
b.00 <- M7[k.2,"B.00"]
b.01 <- M7[k.2,"B.01"]
b.02 <- M7[k.2,"B.02"]
b.11 <- M7[k.2,"B.11"]
b.22 <- M7[k.2,"B.22"]
b.12 <- M7[k.2,"B.12"]
cc.01 <- b.00-logit.00 # Terminos independientes
bb.01 <- b.01+b.02*m.R1 # Coeficientes de x1
aa.01 <- b.11+b.22*((m.R1)^2)+b.12*m.R1   # Coeficientes de x1^2
R.x1 <- polyroot(c(cc.01, bb.01, aa.01)) # Atenti: orden creciente.
r1.x1 <- Re(R.x1[1])
r2.x1 <- Re(R.x1[2])
# Evaluo los x2 correspondientes a cada x1 solucion:
r1.x2 <- m.R1*r1.x1
r2.x2 <- m.R1*r2.x1
# Los primeros dos puntos solucion:
AA.01 <- c(r1.x1, r1.x2)
BB.01 <- c(r2.x1, r2.x2)
AA.01
BB.01
# Ambos puntos, pertenecen a w.00%*Pi.max?
# Los evaluo en f.02:
f.02(AA.01[1], AA.01[2]) # debe coincidir con Pi.00
f.02(BB.01[1], BB.01[2]) # debe coincidir con Pi.00
Pi.00
# Si la diferencia absoluta calculada con la establecida en Pi.00 da me-
# nor que 1e-6, diremos que el resultado es "OK". Else, "NG".
ifelse((f.02(AA.01[1], AA.01[2])-Pi.00)<(1e-6), "OK AA.01", "NG AA.01")
ifelse((f.02(BB.01[1], BB.01[2])-Pi.00)<(1e-6), "OK BB.01", "NG BB.01")
# Averiguo el punto que se encuentre a mayor distancia de X.max, y lo e-
# lijo como primer centro de experimentacion, O.01:
d.AM <- sqrt((AA.01[1]-M7[k.2, "X1.S"])^2+(AA.01[2]-M7[k.2, "X2.S"])^2)
d.BM <- sqrt((BB.01[1]-M7[k.2, "X1.S"])^2+(BB.01[2]-M7[k.2, "X2.S"])^2)
ifelse((d.AM)<(d.BM), "---> Se elige B.01 como primer centro",
"---> Se elige AA.01 como primer centro")
invisible(ifelse((d.AM)<(d.BM), O.01 <- BB.01, O.01 <- AA.01))
# Renombro la variable:
```

```
0.1 <- 0.01
# Guardo el valor d.BM como d.OM (letra O, letra M):
d.OM <- d.BM
```

C.3. Serie de programas CRJER

C.3.1. Descripción general

Esta serie de programas corresponde a la generación secuencial de puntos de diseño (15 en total) y al ajuste de modelos logísticos siguiendo el criterio jerárquico de selección de términos. Para manejar adecuadamente las entradas y salidas que componen el mismo, los mismos se encuentran organizados por MÓDULOS. Los puntos de diseño secuencialmente elegidos parten del valor $w = 5\,\%$ para el factor de centro. Cada programa genera los 15 cuadros de puntos que se comentaron en capítulos anteriores, los cuales se resumen en estadísticos de localización y de dispersión. Debido a la enorme extensión que representaría colocar aquí todas las líneas utilizadas para los programas de una misma serie, solamente reproduciremos las correspondientes a las que se utilizarían en un solo cuadro de puntos, siendo el resto análogas para los demás cuadros.

La serie comprende 25 programas para cada valor de w, uno para cada fila de la matriz **LS**. Los nombres de cada programa tienen la forma CRJER-W005-FXX-v4.txt, en donde XX denota el número de fila de la matriz **LS** que se ha considerado para su estudio.

C.3.2. Resumen de entradas y salidas

Inputs o entradas principales

Ejecución completa del programa correspondiente de la serie BETAS. En el caso de las líneas de programa que comentamos para la serie CRJER, el programa de la serie BETAS correspondiente es el BETAS-W005-UNI-v3.txt.

Outputs o salidas principales

- Las mismas que para los programas de la serie BETAS.
- Matriz **LS** para el valor considerado de w.
- Coordenadas en x_1 y x_2 de los puntos del primer diseño (5 puntos de 100 observaciones "éxito-fracaso" cada una). Todo esto replicado 15 veces (15 cuadros de puntos).
- Coeficientes del modelo logístico ajustado para el primer diseño y cosenos directores del primer salto, tomando la información de los puntos del primer diseño.
- Coordenadas en x_1 y x_2 de los puntos del segundo diseño (5 puntos de 100 observaciones "éxito-fracaso" cada una). Todo esto replicado 15 veces (15 cuadros de puntos).

- Coeficientes del modelo logístico ajustado para el primer diseño y cosenos directores del segundo salto, tomando la información de los puntos del primero y segundo diseño.
- Coordenadas en x_1 y x_2 de los puntos del tercer diseño (5 puntos de 100 observaciones "éxito-fracaso" cada una). Todo esto replicado 15 veces (15 cuadros de puntos).
- Coeficientes del modelo logístico final, tomando la información de los 3 diseños (15 puntos de diseño).
- Todos los modelos ajustados se clasifican según ellos representen máximos, mínimos, puntos de silla o planos, mediante el criterio de los autovalores (o valores propios).
- Para el valor de w considerado, cálculo de estadisticos de calidad de ajuste en función del criterio I de cantidad de información, a partir de los determinantes de la Matriz Observada de Informacion de Fisher, para cada uno de los modelos ajustados (15 en total). Se obtienen los determinantes teniendo en cuenta tanto `pr.obs` como `pi.hat` para estimar la probabilidad de éxito. Para cada uno de ellos tanto como para su logaritmo, se obtienen las medidas de posición y dispersión básicas. Se obtienen también las respectivas matrices **W** de "pesos" para cada caso.
- Para el valor de w considerado, cálculo de estadisticos de calidad de ajuste en función del criterio II de proximidad al máximo, a partir del cálculo de los puntos `x.hat.max` para el modelo teórico y para cada uno de los modelos ajustados (15 en total). Una vez calculadas las probabilidades en la superficie teórica, se realizan las respectivas transformaciones (logaritmo, logit, angular y Freeman-Tukey). Para cada una de ellas se obtienen las medidas de posición y dispersión básicas.
- Calculo de otros estadísticos de calidad de ajuste generales para cada uno de los modelos ajustados: log-likelihood, devianza residual y AIC.
- Tablas de resumen de todos los estadísticos de calidad de ajuste para cada uno de los modelos ajustados.
- Gráficas de la estrategia secuencial de aproximación al máximo mediante factoriales, una para cada uno de los 15 cuadros de puntos. Gráficos de los factoriales sucesivos, con sus ángulos y saltos a escala.

C.3.3. Detalle de entradas y salidas por módulos

PROGRAMAS DE LA SERIE "CRJER"		
INPUTS	MÓDULOS	OUTPUTS
• Valor del factor de centro, w. • Distancia entre el primer centro y el máximo teórico, d_{OM} (programa de la serie BETAS). • Valor del parámetro binomial m.	1	• Valor absoluto del salto (S) y longitud del lado (L) de los factoriales de toda la exploración. • Matriz **LS** completa para el valor w elegido.
• Módulo 1.	2	• Puntos del primer diseño y valor de la ST de cada uno. • Coordenadas de los puntos generados aleatoriamente "sin ruido" en la ST. • Ajuste automático de los puntos del primer diseño (criterio jerárquico de selección) mediante un modelo logístico. • Cosenos directores estimados del primer salto.
• Módulo 2.	3	• Puntos del segundo diseño y valor de la ST de cada uno. • Coordenadas de los puntos generados aleatoriamente "sin ruido" en la ST. • Ajuste automático de los puntos del primero + segundo diseño (criterio jerárquico de selección) mediante un modelo logístico. • Cosenos directores estimados del segundo salto.
• Módulo 3.	4	• Puntos del tercer diseño y valor de la ST de cada uno. • Coordenadas de los puntos generados aleatoriamente "sin ruido" en la ST. • Ajuste automático de los puntos del primero + segundo + tercer diseño (criterio jerárquico de selección) mediante un modelo logístico.
• Coeficientes de los modelos para todos los cuadros de puntos.	5, 6 y 7	• Cálculo de estadísticos de calidad de ajuste en función de los criterios I y II. • Cálculo de otros estadísticos de calidad de ajuste (MLGs).
• Todos los diseños. • Todos los modelos ajustados.	8 y 9	• Representaciones gráficas de las aproximaciones al máximo de cada modelo.

Figura C.2: Características principales de los programas de la serie CRJER.

C.3.4. Líneas de programa en R

```
# PROGRAMA:                                          CRJER-W005-F01-v4.txt
# FECHA REVISION:                                                04 ene 2006

# GENERALIDADES:
# Se trata de partir del primer centro de experimentacion, O.1, y de ar-
# mar un disenyo factorial a su alrededor, teniendo un total de 5 puntos
# de disenyo. Para cada uno de esos puntos, se recogen 100 observaciones
# de la respuesta a partir del modelo teorico del proceso, luego de lo
# cual se ajusta un modelo logistico.
# Con el modelo ajustado se determina la direccion de maximo crecimiento
# de dicho modelo ajustado, de manera de determinar el segundo centro de
# experimentacion.
# Alrededor del segundo centro, se arma otro factorial y se realizan las
# observaciones de la respuesta. Con los puntos del primer disenyo suma-
# dos a los del segundo, ensayo un segundo ajuste, que me dara a su vez
# un nuevo gradiente.
# Siguiendo dicho gradiente, determino el tercer centro y construyo un
# nuevo factorial a su alrededor.
# Finalmente, con los puntos de los 3 disenyos se ajusta un modelo final,
# con el que se calcularan algunos estadisticos muestrales de medida de
# bondad del ajuste y de valoracion de la estrategia elegida.

# PARAMETROS:
# Como no se conoce de entrada que valores tomar para el lado del facto-
# rial, L, se determina un vector de valores posibles que seran funcion
# de la distancia d.OM, que es la que separa el maximo del primer centro.
# Tampoco se conoce que valor absoluto tendra el salto S que separara el
# primer centro del segundo. Tambien se determina un vector de valores
# posibles para S, con los que se iran realizando experimentos.
# Con ambos vectores de L y S, se forma una matriz LS con todos los ca-
# sos posibles que se evaluaran. El objetivo es definir una serie de es-
# tadisticos muestrales y decidir que valores de L y S son los que hacen
# que la estrategia secuencial de disenyos sea la mas conveniente. En es-
# te programa se evaluan estos estadisticos solamente para una fila de
# dicha matriz, es decir, para un valor determinado de L y de S.

# VALORES A MODIFICAR PARA EVALUAR NUEVOS CASOS:
# 1. Hay que hacer un reemplazo de la fila F01 (por ejemplo), por la que
#    que corresponda al caso nuevo.
# 2. A la variable 'fila.LS' hay que asignarle el valor que corresponda a
#    la nueva fila a analizar.
# 3. Cambiar los nombres del ficheros-resumen, de acuerdo con el w.00
#    elegido, por ejemplo: "list.W005.F01" -> cambiar el 'W005' y el
```

```
#    'F01' por los casos que correspondan.

# METODO DE AJUSTE DE MODELOS:
# * Los modelos que se van ajustando toman todos los puntos experimenta-
#   les observados hasta el momento.
# * Se consideran solamente los terminos que resulten seleccionados de a-
#   quellos modelos ajustados mediante el criterio jerarquico.

# DATA FRAMES DE PUNTOS:
# Se consideraran de partida unos 15 conjuntos de puntos con los que res-
# pectivamente se armaran 15 primeros factoriales. Siguiendo la estrate-
# gia propuesta, se llega a los terceros factoriales correspondientes a
# cada data frame de puntos, con los que se determinan los respectivos 15
# modelos ajustados.

# WORKING DIRECTORY:
setwd('C:/Documents and Settings/Arturo/Mis documentos/Tesis/Programas/W005')

# REQUISITOS:
# Ejecutar completamente el programa BETAS

# Verificar que en el working directory este disponible dicho programa.
# En el caso en que lo este, se lo puede ejecutar abreviadamente hacien-
# do:

source('BETAS-W005-UNI-v3.txt')

# Siguiendo estos pasos, bastara solamente ejecutar el presente programa
# y prescindir de la ejecucion individual del programa de las betas.
# ------------------------------------------------------------------------
# MODULO 1
#
# Niveles generales para los parametros de la estrategia: L, S y M
# ------------------------------------------------------------------------

# (a) Longitud del lado del disenyo, L:
#     Conocida la distancia que une el centro de experimentacion 0.1 con
#     el maximo de la superficie teorica, d.OM, se define el lado del
#     factorial, denotado por L, como una funcion de dicha distancia
#     d.OM. En este caso, consieramos directamente:
L.00 <- 0.25 * d.OM
# (b) Longitud del salto entre centros:
S.00 <- d.OM
# (c) Submultiplos de L y S considerados:
# Se consideraran fracciones de L y de S con los que se arma una matriz.
```

```
# Se establecen escalas fraccionales para los niveles de L y de S:
LL <- seq(from=0.6, to=1.4, length=5) * L.00
SS <- seq(from=0.2, to=1.0, length=5) * S.00
# Al tratarse de 5 niveles cada uno, se forma una matriz de 25 filas por
# 2 columnas:
LS <- expand.grid(LL, SS)
colnames(LS) <- c("Valores de L", "Valores de S")
# Lo que sigue es repetir todo el estudio que se desarrolle en estas li-
# neas de programa para cada uno de los 25 pares de valores (L, S).
# (d) Estudio de un caso:
# En este programa se estudiara el caso de la fila (F01) de la matriz
# LS:
fila.LS <- 1        # Debe coincidir con el FXX del nombre del archivo.
L <- LS[fila.LS, "Valores de L"]# Columna 1
S <- LS[fila.LS, "Valores de S"]# Columna 2
# (e) Valor del index de la familia de distribuciones binomiales de las
#     que provendran los datos simulados (# de exitos y proporciones):
# Se tomara en todos los casos m = 100:
M <- 100

# ------------------------------------------------------------------------
# MODULO 2.1
#
# Puntos del primer disenyo
# ------------------------------------------------------------------------

# (a) Coordenadas en x1 y en x2 de los puntos del primer disenyo
# Los puntos iniciales del primer disenyo seran 5: C.1, D.1. E.1, F.1 y el
# central O.1 (letra O cero uno), que fue calculado en el MODULO 1:
C.1 <- c((O.1[1]-0.5*L),(O.1[2]-0.5*L))
D.1 <- c((O.1[1]+0.5*L),(O.1[2]-0.5*L))
E.1 <- c((O.1[1]+0.5*L),(O.1[2]+0.5*L))
F.1 <- c((O.1[1]-0.5*L),(O.1[2]+0.5*L))
# y el centro, O.1
# (b) Valor de la ST en los puntos del primer disenyo, o valores sin rui-
#     :do
Pi.C.1 <- f.02(C.1[1], C.1[2])
Pi.D.1 <- f.02(D.1[1], D.1[2])
Pi.E.1 <- f.02(E.1[1], E.1[2])
Pi.F.1 <- f.02(F.1[1], F.1[2])
Pi.O.1 <- f.02(O.1[1], O.1[2])
# (c) Simulacion de los numeros de exitos
# Para el valor de M que se sigue en este estudio, se generan exitos para
# cada punto del disenyo, a partir de una distribucion binomial(M, Pi),
# siendo Pi la funcion teorica, que se toma como probabilidad de exito.
```

```
# Genero 15 exitos para cada punto de disenyo:
ex.C.1 <- rbinom(15, M, Pi.C.1)
ex.D.1 <- rbinom(15, M, Pi.D.1)
ex.E.1 <- rbinom(15, M, Pi.E.1)
ex.F.1 <- rbinom(15, M, Pi.F.1)
ex.O.1 <- rbinom(15, M, Pi.O.1)
ex.1 <- matrix(c(ex.C.1, ex.D.1, ex.E.1, ex.F.1, ex.O.1), nc=5, byrow=F)
# (d) Disposicion de todas las cantidades del primer disenyo en data
#     frames:
#     Habra 15 data frames, uno para cada fila de ex.1. Esto se traduce
#     como 15 corridas diferentes del experimento para cada uno de los
#     5 puntos que forman parte del primer factorial.

# ------------------------------------------------------------------
# MODULO 2.1.01
#
# Puntos del primer disenyo para el DF.01.01
# ------------------------------------------------------------------

c1.01.01 <- round(c(C.1[1], D.1[1], E.1[1], F.1[1], O.1[1]), 5)
c2.01.01 <- round(c(C.1[2], D.1[2], E.1[2], F.1[2], O.1[2]), 5)
c3.01.01 <- round(c(Pi.C.1, Pi.D.1, Pi.E.1, Pi.F.1, Pi.O.1), 5)
c4.01.01 <- rep(M, length(c1.01.01))
c5.01.01 <- c(ex.1[1,1], ex.1[1,2], ex.1[1,3], ex.1[1,4], ex.1[1,5])
c6.01.01 <- c5.01.01/M
DF.01.01 <- data.frame(c1.01.01, c2.01.01, c3.01.01, c4.01.01, c5.01.01,
c6.01.01)
rownames(DF.01.01) <- c("C.1", "D.1", "E.1", "F.1", "O.1")
colnames(DF.01.01) <- c("x1", "x2", "Pi", "M", "y", "pr")
```

Los módulos 2.1.02 al 2.1.15 contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# ------------------------------------------------------------------
# MODULO 2.2
#
# Ajuste de modelos a partir del primer disenyo: criterio jerarquico
# ------------------------------------------------------------------

# Para cada uno de los 5 data frames, se busca ajustar modelos mediante
# el criterio jerarquico de seleccion de factores. Se incluyen los coefi-
# cientes del modelo final ajustado y el valor de los cosenos directore
# para el valor del salto considerado, S.
```

```
# De cada data frame se estudia un caso, para corroborar los resultados.
# Se crea una variable llamada p.valor, que establece el valor de acep-
# tacion y rechazo de nuevos terminos en el modelo ajustado, sobre la ba-
# se de su valor de significacion.
# De entrada, se considera el mismo valor que el que utiliza el Minitab:
# 15%
p.valor <- 0.15

# ------------------------------------------------------------------------
# MODULO 2.2.01
#
# Ajuste automatico para el DF.01.01
# ------------------------------------------------------------------------

rm(uno, x.01, x.02, x.12, x.11, Pi, m, y, pr)
uno <- rep(1, nrow(DF.01.01))
x.01 <- DF.01.01$x1
x.02 <- DF.01.01$x2
x.12 <- (DF.01.01$x1)*(DF.01.01$x2)
x.11 <- (DF.01.01$x1)*(DF.01.01$x1)
Pi <- DF.01.01$Pi
m <- DF.01.01$M
y <- ex.1[1,]   # El caso estudiado.
pr <- y/m
mod.5100 <- glm(pr ~x.01 + x.02 + x.12 + x.11, weights = m, binomial)
sum.5100 <- summary.glm(mod.5100)$coefficients
SIG51 <- sum.5100["x.11", "Pr(>|z|)"] <= p.valor
I51 <- ifelse(SIG51==T, assign("mod.5500A", glm(pr ~x.01 + x.02 +
x.12 + x.11, weights = m, binomial)), assign("mod.5200",
glm(pr ~x.01 + x.02 + x.12, weights = m, binomial)))
sum.5200 <- summary.glm(mod.5200)$coefficients
SIG52 <- sum.5200["x.12", "Pr(>|z|)"] <= p.valor
I52 <- ifelse(SIG52==T, assign("mod.5500B", glm(pr ~x.01 + x.02 +
x.12, weights = m, binomial)), assign("mod.5300",
glm(pr ~x.01 + x.02, weights = m, binomial)))
sum.5300 <- summary.glm(mod.5300)$coefficients
p.01.5300 <- sum.5300["x.01", "Pr(>|z|)"]
p.02.5300 <- sum.5300["x.02", "Pr(>|z|)"]
p.5300 <- c("x.01"=p.01.5300, "x.02"=p.02.5300)
x0M.5300 <- get(names(p.5300)[which.max(p.5300)[[1]]])
x0m.5300 <- get(names(p.5300)[which.min(p.5300)[[1]]])
x.0M <- x0M.5300
x.0m <- x0m.5300
mod.5400 <- glm(pr ~x.0m + x.0M, weights = m, binomial)
sum.5400 <- summary.glm(mod.5400)$coefficients
```

```
SIG54 <- sum.5400["x.0M", "Pr(>|z|)"] <= p.valor
SIG55 <- sum.5400["x.0m", "Pr(>|z|)"] <= p.valor
SIG56 <- sum.5400["x.0m", "Pr(>|z|)"] <= p.valor
V55 <- if(SIG52==F & SIG54==T & SIG55==T) assign("mod.5500C",
glm(pr ~x.0m + x.0M, weights = m, binomial))
F55 <- if(SIG52==F & SIG54==T & SIG55==F) assign("mod.5500D",
glm(pr ~x.0M, weights = m, binomial))
V56 <- if(SIG52==F & SIG54==F & SIG56==T) assign("mod.5500E",
glm(pr ~x.0m, weights = m, binomial))
F56 <- if(SIG52==F & SIG54==F & SIG56==F) assign("mod.5500F",
glm(pr ~1, weights = m, binomial))
# Atenti: hay que cambiar el nombre "mod.5901.1" por el que corresponda
# en el caso.
# El modelo final estara guardado en el mod.5901.1:
invisible(if(summary.glm(mod.5500A)$aic != 0) assign("mod.5901.1",
mod.5500A))
invisible(if(summary.glm(mod.5500B)$aic != 0) assign("mod.5901.1",
mod.5500B))
invisible(if(summary.glm(mod.5500C)$aic != 0) assign("mod.5901.1",
mod.5500C))
invisible(if(summary.glm(mod.5500D)$aic != 0) assign("mod.5901.1",
mod.5500D))
invisible(if(summary.glm(mod.5500E)$aic != 0) assign("mod.5901.1",
mod.5500E))
invisible(if(summary.glm(mod.5500F)$aic != 0) assign("mod.5901.1",
mod.5500F))
sum.5901.1 <- summary.glm(mod.5901.1)$coefficients
IN.5901.1 <- attr(mod.5901.1$terms, "intercept")
AT.5901.1 <- attr(mod.5901.1$terms, "term.labels")
IF1 <- if(AT.5901.1[1]=="x.0m" & AT.5901.1[2]=="x.0M") "x.0m y x.0M"
IF2 <- if(AT.5901.1[1]=="x.0M" & is.na(AT.5901.1[2])) "solo x.0M"
IF3 <- if(AT.5901.1[1]=="x.0m" & is.na(AT.5901.1[2])) "solo x.0m"
IF4 <- if(IN.5901.1==1 & is.na(AT.5901.1[1]) & is.na(AT.5901.1[2]))
"solo termino independiente"
IF5 <- if(length(AT.5901.1)==3) "modelo 5500B"
IF6 <- if(length(AT.5901.1)==4) "modelo 5500A"
eti.5901 <- if(IF1=="x.0m y x.0M") IF1
eti.5901 <- if(IF2=="solo x.0M") IF2
eti.5901 <- if(IF3=="solo x.0m") IF3
eti.5901 <- if(IF4=="solo termino independiente") IF4
eti.5901 <- if(IF5=="modelo 5500B") IF5
eti.5901 <- if(IF6=="modelo 5500A") IF6
if(eti.5901=="solo x.0m" & all(get(AT.5901.1[1])==x.01))
assign("mod.5901.2", glm(pr ~x.01, binomial, weights = m))
if(eti.5901=="solo x.0m" & all(get(AT.5901.1[1])==x.02))
```

```
assign("mod.5901.2", glm(pr ~x.02, binomial, weights = m))
if(eti.5901=="solo x.0M" & all(get(AT.5901.1[1])==x.01))
assign("mod.5901.2", glm(pr ~x.01, binomial, weights = m))
if(eti.5901=="solo x.0M" & all(get(AT.5901.1[1])==x.02))
assign("mod.5901.2", glm(pr ~x.02, binomial, weights = m))
if(eti.5901=="x.0m y x.0M") assign("mod.5901.2",
glm(pr ~x.01 + x.02, binomial, weights = m))
if(eti.5901=="solo termino independiente") assign("mod.5901.2",
glm(pr ~1, binomial, weights = m))
if(eti.5901=="modelo 5500B") assign("mod.5901.2",
glm(pr ~x.01 + x.02 + x.12, binomial, weights = m))
if(eti.5901=="modelo 5500A") assign("mod.5901.2",
glm(pr ~x.01 + x.02 + x.12 + x.11, binomial, weights = m))
# En el modelo 5901.2 converge la solucion del modelo ajustado:
sum.5901.2 <- summary.glm(mod.5901.2)$coefficients
AT.5901.2 <- attr(mod.5901.2$terms, "term.labels")
# Armado del data frame collage:
# Modelo candidato: 5500A
# Parametros: b00, x.01, x.02, x.12, x.11
if(length(AT.5901.2)==4)
assign("DF5.A.1", data.frame(
c(sum.5901.2["(Intercept)","Estimate"],
sum.5901.2["x.01","Estimate"],   sum.5901.2["x.02","Estimate"],
sum.5901.2["x.12","Estimate"],   sum.5901.2["x.11","Estimate"]),
c(sum.5901.2["(Intercept)","Std. Error"],
sum.5901.2["x.01","Std. Error"], sum.5901.2["x.02","Std. Error"],
sum.5901.2["x.12","Std. Error"], sum.5901.2["x.11","Std. Error"]),
c(sum.5901.2["(Intercept)","z value"],
sum.5901.2["x.01","z value"],    sum.5901.2["x.02","z value"],
sum.5901.2["x.12","z value"],    sum.5901.2["x.11","z value"]),
c(sum.5901.2["(Intercept)","Pr(>|z|)"],
sum.5901.2["x.01","Pr(>|z|)"],   sum.5901.2["x.02","Pr(>|z|)"],
sum.5901.2["x.12","Pr(>|z|)"],   sum.5901.2["x.11","Pr(>|z|)"])))
colnames(DF5.A.1) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF5.A.1) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11")

# Modelo candidato: 5500B
# Parametros: b00, x.01, x.02, x.12
if(length(AT.5901.2)==3)
assign("DF5.B.1", data.frame(
c(sum.5901.2["(Intercept)","Estimate"],
sum.5901.2["x.01","Estimate"],   sum.5901.2["x.02","Estimate"],
sum.5901.2["x.12","Estimate"],   0),
c(sum.5901.2["(Intercept)","Std. Error"],
sum.5901.2["x.01","Std. Error"], sum.5901.2["x.02","Std. Error"],
```

```
sum.5901.2["x.12","Std. Error"], 0),
c(sum.5901.2["(Intercept)","z value"],
sum.5901.2["x.01","z value"],    sum.5901.2["x.02","z value"],
sum.5901.2["x.12","z value"],    0),
c(sum.5901.2["(Intercept)","Pr(>|z|)"],
sum.5901.2["x.01","Pr(>|z|)"],   sum.5901.2["x.02","Pr(>|z|)"],
sum.5901.2["x.12","Pr(>|z|)"],   0)))
colnames(DF5.B.1) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF5.B.1) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11")

# Modelo candidato: 5500C
# Parametros: b00, x.01, x.02
if(length(AT.5901.2)==2)
assign("DF5.C.1", data.frame(
c(sum.5901.2["(Intercept)","Estimate"],
sum.5901.2["x.01","Estimate"],   sum.5901.2["x.02","Estimate"],   0, 0),
c(sum.5901.2["(Intercept)","Std. Error"],
sum.5901.2["x.01","Std. Error"], sum.5901.2["x.02","Std. Error"], 0, 0),
c(sum.5901.2["(Intercept)","z value"],
sum.5901.2["x.01","z value"],    sum.5901.2["x.02","z value"],    0, 0),
c(sum.5901.2["(Intercept)","Pr(>|z|)"],
sum.5901.2["x.01","Pr(>|z|)"],   sum.5901.2["x.02","Pr(>|z|)"],   0, 0)))
colnames(DF5.C.1) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF5.C.1) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11")

# Modelo candidato: 5500D
# Parametros: b00, x.0m
if(length(AT.5901.2)==1 & AT.5901.2[1]=="x.02")
assign("DF5.D.1", data.frame(
c(sum.5901.2["(Intercept)","Estimate"],
0, sum.5901.2["x.02","Estimate"],   0, 0),
c(sum.5901.2["(Intercept)","Std. Error"],
0, sum.5901.2["x.02","Std. Error"], 0, 0),
c(sum.5901.2["(Intercept)","z value"],
0, sum.5901.2["x.02","z value"],    0, 0),
c(sum.5901.2["(Intercept)","Pr(>|z|)"],
0, sum.5901.2["x.02","Pr(>|z|)"],   0, 0)))
colnames(DF5.D.1) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF5.D.1) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11")

# Modelo candidato: 5500E
# Parametros: b00, x.0M
if(length(AT.5901.2)==1 & AT.5901.2[1]=="x.01")
assign("DF5.E.1", data.frame(
c(sum.5901.2["(Intercept)","Estimate"],
```

```
sum.5901.2["x.01","Estimate"],   0, 0, 0),
c(sum.5901.2["(Intercept)","Std. Error"],
sum.5901.2["x.01","Std. Error"], 0, 0, 0),
c(sum.5901.2["(Intercept)","z value"],
sum.5901.2["x.01","z value"],    0, 0, 0),
c(sum.5901.2["(Intercept)","Pr(>|z|)"],
sum.5901.2["x.01","Pr(>|z|)"],   0, 0, 0)))
colnames(DF5.E.1) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF5.E.1) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11")

# Modelo candidato: 5500F
# Parametros: b00
if(length(AT.5901.2)==0)
assign("DF5.F.1", data.frame(
c(sum.5901.2["(Intercept)","Estimate"],   0, 0, 0, 0),
c(sum.5901.2["(Intercept)","Std. Error"], 0, 0, 0, 0),
c(sum.5901.2["(Intercept)","z value"],    0, 0, 0, 0),
c(sum.5901.2["(Intercept)","Pr(>|z|)"],   0, 0, 0, 0)))
colnames(DF5.F.1) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF5.F.1) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11")
DF5.01 <- DF5.A.1
DF5.01 <- DF5.B.1
DF5.01 <- DF5.C.1
DF5.01 <- DF5.D.1
DF5.01 <- DF5.E.1
DF5.01 <- DF5.F.1
# El modelo ajustado automaticamente converge en un solo data frame,
# DF5.01.
```

Los módulos 2.2.02 al 2.2.15 contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# ------------------------------------------------------------------------
# MODULO 2.3
#
# Vectores de los coeficientes estimados para cada ajuste
# ------------------------------------------------------------------------

# Definicion del gradiente, evaluado en el primer centro 0.1 (letra 0
# punto uno):
# eta = b.hat5.00 + b.hat5.01*x1 + b.hat5.02*x2 + b.hat5.12*x1*x2
# + b.hat5.11*x1*x1
# D5.eta.x1 = b.hat5.01 + b.hat5.12*x2 + 2*b.hat5.11*x1
# D5.eta.x2 = b.hat5.02 + b.hat5.12*x1
```

```
# Los coeficientes b.hat5 provendran del modelo completo ajustado.
# Si el b.hat5.00 es el unico coeficiente que forma parte del modelo, no
# es posible determinar el salto hacia el nuevo centro. Se conviene en
# hacer lo siguiente:
# Decretar el salto igual a cero y los nuevos puntos se definen rotados
# a 45grados con respecto al primer disenyo.

# ------------------------------------------------------------------------
# MODULO 2.3.01
#
# Angulo y salto para el DF.01.01
# ------------------------------------------------------------------------

rm(uno, x.01, x.02, x.12, x.11, Pi, m, y, pr)
uno <- rep(1, nrow(DF.01.01))
x.01 <- DF.01.01$x1
x.02 <- DF.01.01$x2
x.12 <- (DF.01.01$x1)*(DF.01.01$x2)
x.11 <- (DF.01.01$x1)*(DF.01.01$x1)
Pi <- DF.01.01$Pi
m <- DF.01.01$M
y <- ex.1[1,]   # El caso estudiado.
pr <- y/m
mod5.01 <- glm(pr ~x.01 + x.02 + x.12 + x.11, weights = m, binomial)
sum5.01 <- summary.glm(mod5.01)$coefficients
# Si este nuevo ajuste sigue teniendo a beta 0 como unico factor signifi-
# cativo, se decreta "salto igual a cero" y el segundo disenyo se super-
# pone con el centro del primero, y se rotan los puntos del cuadrado unos
# 45grados, clockwise.
# Se guardan los coeficientes del modelo ajustado en las variables co-
# rrespondientes de acuerdo con el summary del modelo completo:
b.00.hat5.01 <- sum5.01["(Intercept)", "Estimate"]
b.01.hat5.01 <- sum5.01["x.01", "Estimate"]
b.02.hat5.01 <- sum5.01["x.02", "Estimate"]
b.12.hat5.01 <- sum5.01["x.12", "Estimate"]
b.11.hat5.01 <- sum5.01["x.11", "Estimate"]
# Las derivadas del predictor lineal evaluadas en el centro:
D5x1.01 <- b.01.hat5.01 + (b.12.hat5.01)*(0.1[2]) + 2*(b.11.hat5.01)*(0.1[1])
D5x2.01 <- b.02.hat5.01 + (b.12.hat5.01)*(0.1[1])
# Valor absoluto del angulo del vector gradiente. El resultado da en
# radianes:
theta5.01 <- abs(atan(D5x2.01/D5x1.01))
# Cuadrante en donde se encuentra el vector (D5x1.01, Dx2.01): el angulo
# se toma en sentido antihorario a partir del eje x1.
signo.D5x1.01 <- sign(D5x1.01)
```

```
signo.D5x2.01 <- sign(D5x2.01)
if(signo.D5x1.01==1 & signo.D5x2.01==1) assign ("phi5.01", theta5.01)
if(signo.D5x1.01==-1 & signo.D5x2.01==1) assign ("phi5.01", pi-theta5.01)
if(signo.D5x1.01==-1 & signo.D5x2.01==-1) assign ("phi5.01", pi+theta5.01)
if(signo.D5x1.01==1 & signo.D5x2.01==-1) assign ("phi5.01", -theta5.01)
# Casos especiales con componentes igual a cero:
if(signo.D5x1.01==0 & signo.D5x2.01==0) assign ("phi5.01", 0.0*pi)
if(signo.D5x1.01==1 & signo.D5x2.01==0) assign ("phi5.01", 0.0*pi)
if(signo.D5x1.01==0 & signo.D5x2.01==1) assign ("phi5.01", 0.5*pi)
if(signo.D5x1.01==-1 & signo.D5x2.01==0) assign ("phi5.01", 1.0*pi)
if(signo.D5x1.01==0 & signo.D5x2.01==-1) assign ("phi5.01", 1.5*pi)
# Conversion de radianes a grados sexagesimales:
phi5.r.01 <- phi5.01          # en radianes
phi5.g.01 <- phi5.01*180/pi   # en grados sexagesimales
# Componentes del gradiente en ambos ejes (el modulo del vector vale "S")
s5.1.01 <- sign(D5x1.01) * S * cos(theta5.01)
s5.2.01 <- sign(D5x2.01) * S * sin(theta5.01)
# Casos especiales:
if(all(sum5.01[2:5, "Estimate"]==0)) assign ("s5.1.01", 0)
if(all(sum5.01[2:5, "Estimate"]==0)) assign ("s5.2.01", 0)
if(s5.1.01=="NaN") assign ("s5.1.01", 0)
if(s5.2.01=="NaN") assign ("s5.2.01", 0)
# Vector gradiente:
S5.01 <- c(s5.1.01, s5.2.01)
```

Los módulos 2.3.02 al 2.3.15 contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# ------------------------------------------------------------------------
# MODULO 3.1
#
# Puntos del segundo disenyo correspondiente a cada data frame
# ------------------------------------------------------------------------

# Para aquellos modelos que hayan conducido a un salto que tenga al me-
# nos una de sus dos componentes distintas de cero, se trasladan los pun-
# tos al segundo disenyo segun:
# PUNTO.2 = PUNTO.1 + SALTO   (todos vectores de dimension 2)
# Si ambas coordenadas del salto fuesen iguales a cero, el segundo dise-
# nyo se obtiene rotando 45grados los puntos del primero.
# Para cada data frame, los puntos obtenidos en el segundo disenyo suma-
# dos a los del primer disenyo, configuraran el segundo data frame, desde
# el cual se ajustara el modelo siguiente.
```

```
# ------------------------------------------------------------------------
# MODULO 3.1.01
#
# Del (DF.01.01) al (DF.01.01) + (DF.02.01.ind) ---> DF.02.01
# ------------------------------------------------------------------------

# El salto que corresponde es S5.01. Si este fuera cero en sus dos compo-
# nentes, entonces se rota el disenyo mediante las formulas:
R.O <- c( 1, 1 )
R.C <- c( (O.1[1]-L/sqrt(2))/(O.1[1]-0.5*L) ,
(O.1[2])/(O.1[2]-0.5*L)
R.D <- c( (O.1[1])/(O.1[1]+0.5*L)           ,
(O.1[2]-L/sqrt(2))/(O.1[2]-0.5*L) )
R.E <- c( (O.1[1]+L/sqrt(2))/(O.1[1]+0.5*L) ,
(O.1[2])/(O.1[2]+0.5*L)           )
R.F <- c( (O.1[1])/(O.1[1]-0.5*L)           ,
(O.1[2]+L/sqrt(2))/(O.1[2]+0.5*L) )
# Traslacion del centro O.1 al O.2 mediante el salto S5.01:
O.20 <- O.1 + S5.01
# Una vez trasladado el centro, reubico los puntos del cuadrado con las
# mismas formulas usadas en el primer disenyo:
C.20 <- c((O.20[1]-0.5*L),(O.20[2]-0.5*L))
D.20 <- c((O.20[1]+0.5*L),(O.20[2]-0.5*L))
E.20 <- c((O.20[1]+0.5*L),(O.20[2]+0.5*L))
F.20 <- c((O.20[1]-0.5*L),(O.20[2]+0.5*L))
# Si el salto es cero en ambas componentes, roto los puntos:
O.2 <- O.20 * if(S5.01[1]==0 && S5.01[2]==0) R.O else c(1,1)
C.2 <- C.20 * if(S5.01[1]==0 && S5.01[2]==0) R.C else c(1,1)
D.2 <- D.20 * if(S5.01[1]==0 && S5.01[2]==0) R.D else c(1,1)
E.2 <- E.20 * if(S5.01[1]==0 && S5.01[2]==0) R.E else c(1,1)
F.2 <- F.20 * if(S5.01[1]==0 && S5.01[2]==0) R.F else c(1,1)
# Atenti 1: utilizando ifelse() en lugar de if(), no funciona.
# Atenti 2: hacerlo de esta forma, a partir del centro, mantiene para-
#           lelos a los ejes los futuros lados de los nuevos disenyos.
# (b) Valor de la ST en los puntos del primer disenyo, o valores sin
#     ruido
Pi.C.2 <- f.02(C.2[1], C.2[2])
Pi.D.2 <- f.02(D.2[1], D.2[2])
Pi.E.2 <- f.02(E.2[1], E.2[2])
Pi.F.2 <- f.02(F.2[1], F.2[2])
Pi.O.2 <- f.02(O.2[1], O.2[2])
# (c) Simulacion de los numeros de exitos
ex.C.2 <- rbinom(1, M, Pi.C.2)
ex.D.2 <- rbinom(1, M, Pi.D.2)
ex.E.2 <- rbinom(1, M, Pi.E.2)
```

```
ex.F.2 <- rbinom(1, M, Pi.F.2)
ex.O.2 <- rbinom(1, M, Pi.O.2)
ex.2 <- matrix(c(ex.C.2, ex.D.2, ex.E.2, ex.F.2, ex.O.2), nc=5, byrow=F)
# (f) Armado del data frame solo con los puntos del segundo disenyo
#     para el DF.02.01 individual, denotado por DF.02.01.ind:
DF.02.01.ind <- rbind(
c(C.2[1], C.2[2], Pi.C.2, M, ex.2[1], ex.2[1]/M),
c(D.2[1], D.2[2], Pi.D.2, M, ex.2[2], ex.2[2]/M),
c(E.2[1], E.2[2], Pi.E.2, M, ex.2[3], ex.2[3]/M),
c(F.2[1], F.2[2], Pi.F.2, M, ex.2[4], ex.2[4]/M),
c(O.2[1], O.2[2], Pi.O.2, M, ex.2[5], ex.2[5]/M) )
rownames(DF.02.01.ind) <- c("C.2", "D.2", "E.2", "F.2", "O.2")
colnames(DF.02.01.ind) <- c("x1", "x2", "Pi", "M", "y", "pr")
# (e) Armado del data frame con todos los puntos del primero y segundo
#     disenyos:
DF.02.01 <- rbind(DF.01.01,
c(C.2[1], C.2[2], Pi.C.2, M, ex.2[1], ex.2[1]/M),
c(D.2[1], D.2[2], Pi.D.2, M, ex.2[2], ex.2[2]/M),
c(E.2[1], E.2[2], Pi.E.2, M, ex.2[3], ex.2[3]/M),
c(F.2[1], F.2[2], Pi.F.2, M, ex.2[4], ex.2[4]/M),
c(O.2[1], O.2[2], Pi.O.2, M, ex.2[5], ex.2[5]/M) )
rownames(DF.02.01) <- c("C.1", "D.1", "E.1", "F.1", "O.1", "C.2", "D.2",
"E.2", "F.2", "O.2")
colnames(DF.02.01) <- c("x1", "x2", "Pi", "M", "y", "pr")
```

Los módulos 3.1.02 al 3.1.15 contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# ------------------------------------------------------------------
# MODULO 3.2
#
# Ajuste de modelos para los puntos del segundo disenyo mediante el cri-
# terio jerarquico
# ------------------------------------------------------------------

# Se ajustan modelos contemplando todos los puntos obtenidos hasta el mo-
# mento agrupados correspondientemente de acuerdo con cada data frame.
# Esto es: dado que cada data frame de puntos contiene 500 puntos experi-
# mentales, se toman los dos primeros disenyos, es decir, 1000 puntos, y
# con todos ellos se ajusta un modelo.
# Los coeficientes de este modelo determinara el angulo del salto entre
# el segundo y el tercer centro.
# La seleccion de terminos para el modelo ajustado se hace siguiendo el
# mismo criterio jerarquico utilizado en el primer ajuste.
```

```
# ------------------------------------------------------------------------
# MODULO 3.2.01
#
# Ajuste automatico de modelos para el DF.02.01
# ------------------------------------------------------------------------

rm(uno, x.01, x.02, x.12, x.11, x.22, Pi, m, y, pr)
uno <- rep(1, nrow(DF.02.01))
x.01 <- DF.02.01$x1
x.02 <- DF.02.01$x2
x.12 <- (DF.02.01$x1)*(DF.02.01$x2)
x.11 <- (DF.02.01$x1)*(DF.02.01$x1)
x.22 <- (DF.02.01$x2)*(DF.02.01$x2)
Pi <- DF.02.01$Pi
m <- DF.02.01$M
y <- DF.02.01$y
pr <- DF.02.01$pr
mod.60000 <- glm(pr ~x.01 + x.02 + x.12 + x.11 + x.22, weights = m,
binomial)
sum.60000 <- summary.glm(mod.60000)$coefficients
p.11.60000 <- sum.60000["x.11", "Pr(>|z|)"]
p.22.60000 <- sum.60000["x.22", "Pr(>|z|)"]
p.60000 <- c("x.11"=p.11.60000, "x.22"=p.22.60000)
xMM.60000 <- get(names(p.60000)[which.max(p.60000)[[1]]])
xmm.60000 <- get(names(p.60000)[which.min(p.60000)[[1]]])
x.MM <- xMM.60000
x.mm <- xmm.60000
mod.61000 <- glm(pr ~x.01 + x.02 + x.12 + x.mm + x.MM, weights = m,
binomial)
sum.61000 <- summary.glm(mod.61000)$coefficients
SIG610 <- sum.61000["x.MM", "Pr(>|z|)"] <= p.valor
SIG620 <- sum.61000["x.mm", "Pr(>|z|)"] <= p.valor
SIG630 <- sum.61000["x.mm", "Pr(>|z|)"] <= p.valor
I610 <- ifelse(SIG610==T, ifelse(SIG620==T, assign("mod.69001A",
glm(pr ~x.01 + x.02 + x.12 + x.11 + x.22, weights = m, binomial)),
assign("mod.61100", glm(pr ~x.01 + x.02 + x.12 + x.MM, weights = m,
binomial))), ifelse(SIG630==T, assign("mod.62100",
glm(pr ~x.01 + x.02 + x.12 + x.mm, weights = m, binomial)),
assign("mod.63100", glm(pr ~x.01 + x.02 + x.12, weights = m,
binomial))))
sum.61100 <- summary.glm(mod.61100)$coefficients
sum.62100 <- summary.glm(mod.62100)$coefficients
sum.63100 <- summary.glm(mod.63100)$coefficients
SIG632 <- sum.61100["x.MM", "Pr(>|z|)"] <= p.valor
```

```
SIG633 <- sum.62100["x.mm", "Pr(>|z|)"] <= p.valor
I632 <- ifelse(SIG632==T, assign("mod.69001B",
glm(pr ~x.01 + x.02 + x.12 + x.MM, weights = m, binomial)),
if(SIG632==F) assign("mod.63400", glm(pr ~x.01 + x.02 + x.12,
weights = m, binomial)))
I633 <- ifelse(SIG633==T, assign("mod.69001C",
glm(pr ~x.01 + x.02 + x.12 + x.mm, weights = m, binomial)),
if(SIG633==F) assign("mod.63500", glm(pr ~x.01 + x.02 + x.12,
weights = m, binomial)))
if(SIG632==F) assign("mod.63400", glm(pr ~x.01 + x.02 + x.12,
weights = m, binomial))
if(SIG633==F) assign("mod.63500", glm(pr ~x.01 + x.02 + x.12,
weights = m, binomial))
# El que no de error corresponde al modelo elegido.
# Primero, hay que verificar si alguno de los modelos "finales" ha sido
# elegido o no. Si esto no ocurriera, hay que elegir uno de los 3 mode-
# los: 63100, 63400 y 63500.
# El que no de error, sera el modelo elegido, y definira el mod.64000:
if(summary.glm(mod.63100)$aic != 0) assign("mod.64000",
glm(pr ~x.01 + x.02 + x.12, weights = m, binomial))
if(summary.glm(mod.63400)$aic != 0) assign("mod.64000",
glm(pr ~x.01 + x.02 + x.12, weights = m, binomial))
if(summary.glm(mod.63500)$aic != 0) assign("mod.64000",
glm(pr ~x.01 + x.02 + x.12, weights = m, binomial))
sum.64000 <- summary.glm(mod.64000)$coefficients
aic.64000 <- summary.glm(mod.64000)$aic
SIG650 <- sum.64000["x.12", "Pr(>|z|)"] <= p.valor
I650 <- ifelse(SIG650==T, assign("mod.69001D",
glm(pr ~x.01 + x.02 + x.12, weights = m, binomial)),
if(SIG650==F & aic.64000 != 0) assign("mod.64100",
glm(pr ~x.01 + x.02, weights = m, binomial)))
sum.64100 <- summary.glm(mod.64100)$coefficients
p.01.410 <- sum.64100["x.01", "Pr(>|z|)"]
p.02.410 <- sum.64100["x.02", "Pr(>|z|)"]
p.64100 <- c("x.01"=p.01.410, "x.02"=p.02.410)
x0M.64100 <- get(names(p.64100)[which.max(p.64100)[[1]]])
x0m.64100 <- get(names(p.64100)[which.min(p.64100)[[1]]])
x.0M <- x0M.64100
x.0m <- x0m.64100
mod.64200 <- glm(pr ~x.0m + x.0M, weights = m, binomial)
sum.64200 <- summary.glm(mod.64200)$coefficients
SIG660 <- sum.64200["x.0M", "Pr(>|z|)"] <= p.valor
SIG670 <- sum.64200["x.0m", "Pr(>|z|)"] <= p.valor
SIG680 <- sum.64200["x.0m", "Pr(>|z|)"] <= p.valor
I660 <- if(SIG660==T) (if(SIG650==F & SIG660==T & SIG670==T)
```

```
assign("mod.69001E", glm(pr ~x.0m + x.0M, weights = m, binomial))) else
(if(SIG650==F & SIG660==T & SIG670==F) assign("mod.69001F",
glm(pr ~x.0M, weights = m, binomial)))
I680 <- if(SIG680==T) (if(SIG650==F & SIG660==F & SIG670==T)
assign("mod.69001G", glm(pr ~x.0m, weights = m, binomial))) else
(if(SIG650==F & SIG660==F & SIG670==F) assign("mod.69001H",
glm(pr ~1, weights = m, binomial)))

# ++++++++++++++++++++++++++++
# Desglose del I660 y del I680:
# I660 <- if(SIG660==T) "V660" else "F660"
# V660 <- if(SIG650==F & SIG660==T & SIG670==T) assign("mod.69001E",
glm(pr ~x.0m + x.0M, weights = m, binomial))
# F660 <- if(SIG650==F & SIG660==T & SIG670==F) assign("mod.69001F",
glm(pr ~x.0M, weights = m, binomial))
# I680 <- if(SIG680==T) "V680" else "F680"
# V680 <- if(SIG650==F & SIG660==F & SIG670==T) assign("mod.69001G",
glm(pr ~x.0m, weights = m, binomial))
# F680 <- if(SIG650==F & SIG660==F & SIG670==F) assign("mod.69001H",
glm(pr ~1, weights = m, binomial))
# ++++++++++++++++++++++++++++

# El modelo final correspondiente a DF.02.0X estara guardado en el
# mod.690X.1.
# Para el DF.02.01, sera mod.69001.1
invisible(if(summary.glm(mod.69001A)$aic != 0) assign("mod.69001.1",
mod.69001A))
invisible(if(summary.glm(mod.69001B)$aic != 0) assign("mod.69001.1",
mod.69001B))
invisible(if(summary.glm(mod.69001C)$aic != 0) assign("mod.69001.1",
mod.69001C))
invisible(if(summary.glm(mod.69001D)$aic != 0) assign("mod.69001.1",
mod.69001D))
invisible(if(summary.glm(mod.69001E)$aic != 0) assign("mod.69001.1",
mod.69001E))
invisible(if(summary.glm(mod.69001F)$aic != 0) assign("mod.69001.1",
mod.69001F))
invisible(if(summary.glm(mod.69001G)$aic != 0) assign("mod.69001.1",
mod.69001G))
invisible(if(summary.glm(mod.69001H)$aic != 0) assign("mod.69001.1",
mod.69001H))
summary.glm(mod.69001.1)$coefficients
# Ahora hay que cambiar las variables: de x.mm y x.MM a terminos de x.11
# y x.22.
IN.69001.1 <- attr(mod.69001.1$terms, "intercept")
```

```
AT.69001.1 <- attr(mod.69001.1$terms, "term.labels")
# Uno solo de estos "IFX" tiene que resultar verdadero:
IF1 <- if(length(AT.69001.1)==5) "cuadratico en ambos"
IF2 <- if(length(AT.69001.1)==4 & AT.69001.1[4]=="x.MM")
"cuadratico en x.MM"
IF3 <- if(length(AT.69001.1)==4 & AT.69001.1[4]=="x.mm")
"cuadratico en x.mm"
IF4 <- if(length(AT.69001.1)==3) "lineal con interaccion"
IF5 <- if(length(AT.69001.1)==2) "lineal sin interaccion"
IF6 <- if(length(AT.69001.1)==1 & AT.69001.1[1]=="x.0M")
"lineal en x.0M"
IF7 <- if(length(AT.69001.1)==1 & AT.69001.1[1]=="x.0m")
"lineal en x.0m"
IF8 <- if(IN.69001.1==1 & is.na(AT.69001.1[1]) & is.na(AT.69001.1[2]))
"termino independiente"
eti.69001 <- if(IF1=="cuadratico en ambos") IF1
eti.69001 <- if(IF2=="cuadratico en x.MM") IF2
eti.69001 <- if(IF3=="cuadratico en x.mm") IF3
eti.69001 <- if(IF4=="lineal con interaccion") IF4
eti.69001 <- if(IF5=="lineal sin interaccion") IF5
eti.69001 <- if(IF6=="lineal en x.0M") IF6
eti.69001 <- if(IF7=="lineal en x.0m") IF7
eti.69001 <- if(IF8=="termino independiente") IF8
# El modelo resultante se guarda en un solo modelo, el mod.69001.2:
if(eti.69001=="cuadratico en ambos") assign("mod.69001.2",
glm(pr ~x.01 + x.02 + x.12 + x.11 + x.22, binomial, weights = m))
if(eti.69001=="cuadratico en x.MM" & all(get(AT.69001.1[4])==x.11))
assign("mod.69001.2", glm(pr ~x.01 + x.02 + x.12 + x.11, binomial,
weights = m))
if(eti.69001=="cuadratico en x.MM" & all(get(AT.69001.1[4])==x.22))
assign("mod.69001.2", glm(pr ~x.01 + x.02 + x.12 + x.22, binomial,
weights = m))
if(eti.69001=="cuadratico en x.mm" & all(get(AT.69001.1[4])==x.11))
assign("mod.69001.2", glm(pr ~x.01 + x.02 + x.12 + x.11, binomial,
weights = m))
if(eti.69001=="cuadratico en x.mm" & all(get(AT.69001.1[4])==x.22))
assign("mod.69001.2", glm(pr ~x.01 + x.02 + x.12 + x.22, binomial,
weights = m))
if(eti.69001=="lineal con interaccion") assign("mod.69001.2",
glm(pr ~x.01 + x.02 + x.12, binomial, weights = m))
if(eti.69001=="lineal sin interaccion") assign("mod.69001.2",
glm(pr ~x.01 + x.02, binomial, weights = m))
if(eti.69001=="lineal en x.0M" & all(get(AT.69001.1[1])==x.01))
assign("mod.69001.2", glm(pr ~x.01, binomial, weights = m))
if(eti.69001=="lineal en x.0M" & all(get(AT.69001.1[1])==x.02))
```

```
assign("mod.69001.2", glm(pr ~x.02, binomial, weights = m))
if(eti.69001=="lineal en x.0m" & all(get(AT.69001.1[1])==x.01))
assign("mod.69001.2", glm(pr ~x.01, binomial, weights = m))
if(eti.69001=="lineal en x.0m" & all(get(AT.69001.1[1])==x.02))
assign("mod.69001.2", glm(pr ~x.02, binomial, weights = m))
if(eti.69001=="termino independiente") assign("mod.69001.2",
glm(pr ~1, binomial, weights = m))
sum.69001.2 <- summary.glm(mod.69001.2)$coefficients
AT.69001.2 <- attr(mod.69001.2$terms, "term.labels")
# Armado del data frame collage:

# Modelo candidato: 69001A
# Parametros: b00, x.01, x.02, x.12, x.11, x.22
if(length(AT.69001.2)==5)
assign("DF6.A.01", data.frame(
c(sum.69001.2["(Intercept)","Estimate"],
sum.69001.2["x.01","Estimate"],   sum.69001.2["x.02","Estimate"],
sum.69001.2["x.12","Estimate"],   sum.69001.2["x.11","Estimate"],
sum.69001.2["x.22","Estimate"]),
c(sum.69001.2["(Intercept)","Std. Error"],
sum.69001.2["x.01","Std. Error"], sum.69001.2["x.02","Std. Error"],
sum.69001.2["x.12","Std. Error"], sum.69001.2["x.11","Std. Error"],
sum.69001.2["x.22","Std. Error"]),
c(sum.69001.2["(Intercept)","z value"],
sum.69001.2["x.01","z value"],    sum.69001.2["x.02","z value"],
sum.69001.2["x.12","z value"],    sum.69001.2["x.11","z value"],
sum.69001.2["x.22","z value"]),
c(sum.69001.2["(Intercept)","Pr(>|z|)"],
sum.69001.2["x.01","Pr(>|z|)"],   sum.69001.2["x.02","Pr(>|z|)"],
sum.69001.2["x.12","Pr(>|z|)"],   sum.69001.2["x.11","Pr(>|z|)"],
sum.69001.2["x.22","Pr(>|z|)"])))
colnames(DF6.A.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF6.A.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 69001B
# Parametros: b00, x.01, x.02, x.12, x.11
if(length(AT.69001.2)==4 & AT.69001.2[4]=="x.11")
assign("DF6.B.01", data.frame(
c(sum.69001.2["(Intercept)","Estimate"],
sum.69001.2["x.01","Estimate"],   sum.69001.2["x.02","Estimate"],
sum.69001.2["x.12","Estimate"],   sum.69001.2["x.11","Estimate"],   0),
c(sum.69001.2["(Intercept)","Std. Error"],
sum.69001.2["x.01","Std. Error"], sum.69001.2["x.02","Std. Error"],
sum.69001.2["x.12","Std. Error"], sum.69001.2["x.11","Std. Error"], 0),
```

```
c(sum.69001.2["(Intercept)","z value"],
sum.69001.2["x.01","z value"],    sum.69001.2["x.02","z value"],
sum.69001.2["x.12","z value"],    sum.69001.2["x.11","z value"],    0),
c(sum.69001.2["(Intercept)","Pr(>|z|)"],
sum.69001.2["x.01","Pr(>|z|)"],   sum.69001.2["x.02","Pr(>|z|)"],
sum.69001.2["x.12","Pr(>|z|)"],   sum.69001.2["x.11","Pr(>|z|)"],    0)))
colnames(DF6.B.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF6.B.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 69001C
# Parametros: b00, x.01, x.02, x.12, x.22
if(length(AT.69001.2)==4 & AT.69001.2[4]=="x.22")
assign("DF6.C.01", data.frame(
c(sum.69001.2["(Intercept)","Estimate"],
sum.69001.2["x.01","Estimate"],   sum.69001.2["x.02","Estimate"],
sum.69001.2["x.12","Estimate"],   0, sum.69001.2["x.22","Estimate"]),
c(sum.69001.2["(Intercept)","Std. Error"],
sum.69001.2["x.01","Std. Error"], sum.69001.2["x.02","Std. Error"],
sum.69001.2["x.12","Std. Error"], 0, sum.69001.2["x.22","Std. Error"]),
c(sum.69001.2["(Intercept)","z value"],
sum.69001.2["x.01","z value"],    sum.69001.2["x.02","z value"],
sum.69001.2["x.12","z value"],    0, sum.69001.2["x.22","z value"]),
c(sum.69001.2["(Intercept)","Pr(>|z|)"],
sum.69001.2["x.01","Pr(>|z|)"],   sum.69001.2["x.02","Pr(>|z|)"],
sum.69001.2["x.12","Pr(>|z|)"],   0, sum.69001.2["x.22","Pr(>|z|)"])))
colnames(DF6.C.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF6.C.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 69001D
# Parametros: b00, x.01, x.02, x.12
if(length(AT.69001.2)==3)
assign("DF6.D.01", data.frame(
c(sum.69001.2["(Intercept)","Estimate"],
sum.69001.2["x.01","Estimate"],   sum.69001.2["x.02","Estimate"],
sum.69001.2["x.12","Estimate"],   0, 0),
c(sum.69001.2["(Intercept)","Std. Error"],
sum.69001.2["x.01","Std. Error"], sum.69001.2["x.02","Std. Error"],
sum.69001.2["x.12","Std. Error"], 0, 0),
c(sum.69001.2["(Intercept)","z value"],
sum.69001.2["x.01","z value"],    sum.69001.2["x.02","z value"],
sum.69001.2["x.12","z value"],    0, 0),
c(sum.69001.2["(Intercept)","Pr(>|z|)"],
sum.69001.2["x.01","Pr(>|z|)"],   sum.69001.2["x.02","Pr(>|z|)"],
```

```
sum.69001.2["x.12","Pr(>|z|)"],   0, 0)))
colnames(DF6.D.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF6.D.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 69001E
# Parametros: b00, x.01, x.02
if(length(AT.69001.2)==2)
assign("DF6.E.01", data.frame(
c(sum.69001.2["(Intercept)","Estimate"],
sum.69001.2["x.01","Estimate"],   sum.69001.2["x.02","Estimate"],
0, 0, 0),
c(sum.69001.2["(Intercept)","Std. Error"],
sum.69001.2["x.01","Std. Error"], sum.69001.2["x.02","Std. Error"],
0, 0, 0),
c(sum.69001.2["(Intercept)","z value"],
sum.69001.2["x.01","z value"],    sum.69001.2["x.02","z value"],
0, 0, 0),
c(sum.69001.2["(Intercept)","Pr(>|z|)"],
sum.69001.2["x.01","Pr(>|z|)"],   sum.69001.2["x.02","Pr(>|z|)"],
0, 0, 0)))
colnames(DF6.E.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF6.E.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 69001F
# Parametros: b00, x.01
if(length(AT.69001.2)==1 & AT.69001.2[1]=="x.01")
assign("DF6.F.01", data.frame(
c(sum.69001.2["(Intercept)","Estimate"],
sum.69001.2["x.01","Estimate"],   0, 0, 0, 0),
c(sum.69001.2["(Intercept)","Std. Error"],
sum.69001.2["x.01","Std. Error"], 0, 0, 0, 0),
c(sum.69001.2["(Intercept)","z value"],
sum.69001.2["x.01","z value"],    0, 0, 0, 0),
c(sum.69001.2["(Intercept)","Pr(>|z|)"],
sum.69001.2["x.01","Pr(>|z|)"],   0, 0, 0, 0)))
colnames(DF6.F.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF6.F.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 69001G
# Parametros: b00, x.02
if(length(AT.69001.2)==1 & AT.69001.2[1]=="x.02")
assign("DF6.G.01", data.frame(
```

```
c(sum.69001.2["(Intercept)","Estimate"],
0, sum.69001.2["x.02","Estimate"],   0, 0, 0),
c(sum.69001.2["(Intercept)","Std. Error"],
0, sum.69001.2["x.02","Std. Error"], 0, 0, 0),
c(sum.69001.2["(Intercept)","z value"],
0, sum.69001.2["x.02","z value"],    0, 0, 0),
c(sum.69001.2["(Intercept)","Pr(>|z|)"],
0, sum.69001.2["x.02","Pr(>|z|)"],   0, 0, 0)))
colnames(DF6.G.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF6.G.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 69001H
# Parametros: b00
if(length(AT.69001.2)==0)
assign("DF6.H.01", data.frame(
c(sum.69001.2["(Intercept)","Estimate"],   0, 0, 0, 0, 0),
c(sum.69001.2["(Intercept)","Std. Error"], 0, 0, 0, 0, 0),
c(sum.69001.2["(Intercept)","z value"],    0, 0, 0, 0, 0),
c(sum.69001.2["(Intercept)","Pr(>|z|)"],   0, 0, 0, 0, 0)))
colnames(DF6.H.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF6.H.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")
DF6.01 <- DF6.A.01
DF6.01 <- DF6.B.01
DF6.01 <- DF6.C.01
DF6.01 <- DF6.D.01
DF6.01 <- DF6.E.01
DF6.01 <- DF6.F.01
DF6.01 <- DF6.G.01
DF6.01 <- DF6.H.01
# El modelo ajustado automaticamente converge en un solo data frame,
# DF6.01.
```

Los módulos `3.2.02` al `3.2.15` contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# ------------------------------------------------------------------------
# MODULO 3.3.01
#
# Angulo y salto para el modelo ajustado en DF.02.01
# ------------------------------------------------------------------------

# Se sigue la misma marcha de calculos que la del primer salto, pero aqui
```

```
# se le agrega un termino mas al modelo, x.22:

rm(uno, x.01, x.02, x.12, x.11, x.22, Pi, m, y, pr)
uno <- rep(1, nrow(DF.02.01))
x.01 <- DF.02.01$x1
x.02 <- DF.02.01$x2
x.12 <- (DF.02.01$x1)*(DF.02.01$x2)
x.11 <- (DF.02.01$x1)*(DF.02.01$x1)
x.22 <- (DF.02.01$x2)*(DF.02.01$x2)
Pi <- DF.02.01$Pi
m <- DF.02.01$M
y <- ex.1[1,]   # El caso estudiado.
pr <- y/m
mod6.01 <- glm(pr ~x.01 + x.02 + x.12 + x.11 + x.22, weights = m,
binomial)
sum6.01 <- summary.glm(mod6.01)$coefficients
b.00.hat6.01 <- sum6.01["(Intercept)", "Estimate"]
b.01.hat6.01 <- sum6.01["x.01", "Estimate"]
b.02.hat6.01 <- sum6.01["x.02", "Estimate"]
b.12.hat6.01 <- sum6.01["x.12", "Estimate"]
b.11.hat6.01 <- sum6.01["x.11", "Estimate"]
b.22.hat6.01 <- sum6.01["x.22", "Estimate"]
D6x1.01 <- b.01.hat6.01 + b.12.hat6.01*(O.2[2]) + 2*b.11.hat6.01*(O.2[1])
D6x2.01 <- b.02.hat6.01 + b.12.hat6.01*(O.2[1]) + 2*b.22.hat6.01*(O.2[2])
theta6.01 <- abs(atan(D6x2.01/D6x1.01))
signo.D6x1.01 <- sign(D6x1.01)
signo.D6x2.01 <- sign(D6x2.01)
if(signo.D6x1.01==1 & signo.D6x2.01==1) assign ("phi6.01", theta6.01)
if(signo.D6x1.01==-1 & signo.D6x2.01==1) assign ("phi6.01", pi-theta6.01)
if(signo.D6x1.01==-1 & signo.D6x2.01==-1) assign ("phi6.01", pi+theta6.01)
if(signo.D6x1.01==1 & signo.D6x2.01==-1) assign ("phi6.01", -theta6.01)
if(signo.D6x1.01==0 & signo.D6x2.01==0) assign ("phi6.01", 0.0*pi)
if(signo.D6x1.01==1 & signo.D6x2.01==0) assign ("phi6.01", 0.0*pi)
if(signo.D6x1.01==0 & signo.D6x2.01==1) assign ("phi6.01", 0.5*pi)
if(signo.D6x1.01==-1 & signo.D6x2.01==0) assign ("phi6.01", 1.0*pi)
if(signo.D6x1.01==0 & signo.D6x2.01==-1) assign ("phi6.01", 1.5*pi)
phi6.r.01 <- phi6.01          # en radianes
phi6.g.01 <- phi6.01*180/pi   # en grados sexagesimales
s6.1.01 <- sign(D6x1.01) * S * cos(theta6.01)
s6.2.01 <- sign(D6x2.01) * S * sin(theta6.01)
if(all(sum6.01[2:5, "Estimate"]==0)) assign ("s6.1.01", 0)
if(all(sum6.01[2:5, "Estimate"]==0)) assign ("s6.2.01", 0)
if(s6.1.01=="NaN") assign ("s6.1.01", 0)
if(s6.2.01=="NaN") assign ("s6.2.01", 0)
S6.01 <- c(s6.1.01, s6.2.01)
```

Los módulos 3.3.02 al 3.3.15 contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# Medidas resumidas para el segundo salto:

sum.6.W005.F01 <- matrix(c(
c(D6x1.01, D6x2.01, theta6.01, phi6.01, phi6.g.01, S6.01[1], S6.01[2]),
c(D6x1.02, D6x2.02, theta6.02, phi6.02, phi6.g.02, S6.02[1], S6.02[2]),
c(D6x1.03, D6x2.03, theta6.03, phi6.03, phi6.g.03, S6.03[1], S6.03[2]),
c(D6x1.04, D6x2.04, theta6.04, phi6.04, phi6.g.04, S6.04[1], S6.04[2]),
c(D6x1.05, D6x2.05, theta6.05, phi6.05, phi6.g.05, S6.05[1], S6.05[2]),
c(D6x1.06, D6x2.06, theta6.06, phi6.06, phi6.g.06, S6.06[1], S6.06[2]),
c(D6x1.07, D6x2.07, theta6.07, phi6.07, phi6.g.07, S6.07[1], S6.07[2]),
c(D6x1.08, D6x2.08, theta6.08, phi6.08, phi6.g.08, S6.08[1], S6.08[2]),
c(D6x1.09, D6x2.09, theta6.09, phi6.09, phi6.g.09, S6.09[1], S6.09[2]),
c(D6x1.10, D6x2.10, theta6.10, phi6.10, phi6.g.10, S6.10[1], S6.10[2]),
c(D6x1.11, D6x2.11, theta6.11, phi6.11, phi6.g.11, S6.11[1], S6.11[2]),
c(D6x1.12, D6x2.12, theta6.12, phi6.12, phi6.g.12, S6.12[1], S6.12[2]),
c(D6x1.13, D6x2.13, theta6.13, phi6.13, phi6.g.13, S6.13[1], S6.13[2]),
c(D6x1.14, D6x2.14, theta6.14, phi6.14, phi6.g.14, S6.14[1], S6.14[2]),
c(D6x1.15, D6x2.15, theta6.15, phi6.15, phi6.g.15, S6.15[1], S6.15[2])),
nc=5, byrow=T)
colnames(sum.6.W005.F01) <- c('D6x1', 'D6x2', 'theta6', 'phi6', 'phi6.g',
'S6.x1', 'S6.x2')
rownames(sum.6.W005.F01) <- c('DF.03.01', 'DF.03.02', 'DF.03.03', 'DF.03.04',
'DF.03.05', 'DF.03.06', 'DF.03.07', 'DF.03.08', 'DF.03.09', 'DF.03.10',
'DF.03.11', 'DF.03.12', 'DF.03.13', 'DF.03.14', 'DF.03.15')

 # ------------------------------------------------------------------------
# MODULO 4.1
#
# Puntos del tercer disenyo
# ------------------------------------------------------------------------

# Importante: referenciar los puntos 0.2 (distintos para cada DF) al
# data frame que corresponda, ya que no son unicos.

# ------------------------------------------------------------------------
# MODULO 4.1.01
#
# Del (DF.02.01) al (DF.02.01) + (DF.03.01.ind) ---> DF.03.01
# ------------------------------------------------------------------------
```

```
# (a) Definicion del centro de referencia para la traslacion, O.3:
O.2 <- c(DF.02.01["O.2", "x1"], DF.02.01["O.2", "x2"])
# El salto que corresponde es S6.01.
# Traslacion del centro O.2 al O.3 mediante el salto S6.01:
O.3 <- O.2 + S6.01
# Una vez trasladado el centro, reubico los puntos del cuadrado con las
# mismas formulas usadas en el primer disenyo:
C.3 <- c((O.3[1]-0.5*L),(O.3[2]-0.5*L))
D.3 <- c((O.3[1]+0.5*L),(O.3[2]-0.5*L))
E.3 <- c((O.3[1]+0.5*L),(O.3[2]+0.5*L))
F.3 <- c((O.3[1]-0.5*L),(O.3[2]+0.5*L))
# Atenti 3: hacerlo de esta forma, a partir del centro, mantiene para-
# lelos a los ejes los futuros lados de los nuevos disenyos.
# (b) Valor de la ST en los puntos del primer disenyo, o valores sin
#     ruido
Pi.C.3 <- f.02(C.3[1], C.3[2])
Pi.D.3 <- f.02(D.3[1], D.3[2])
Pi.E.3 <- f.02(E.3[1], E.3[2])
Pi.F.3 <- f.02(F.3[1], F.3[2])
Pi.O.3 <- f.02(O.3[1], O.3[2])
# (c) Simulacion de los numeros de exitos
ex.C.3 <- rbinom(1, M, Pi.C.3)
ex.D.3 <- rbinom(1, M, Pi.D.3)
ex.E.3 <- rbinom(1, M, Pi.E.3)
ex.F.3 <- rbinom(1, M, Pi.F.3)
ex.O.3 <- rbinom(1, M, Pi.O.3)
ex.3 <- matrix(c(ex.C.3, ex.D.3, ex.E.3, ex.F.3, ex.O.3), nc=5, byrow=F)
# (f) Armado del data frame solo con los puntos del segundo disenyo
#     para el DF.03.01 individual, denotado por DF.03.01.ind:
DF.03.01.ind <- rbind(
c(C.3[1], C.3[2], Pi.C.3, M, ex.3[1], ex.3[1]/M),
c(D.3[1], D.3[2], Pi.D.3, M, ex.3[2], ex.3[2]/M),
c(E.3[1], E.3[2], Pi.E.3, M, ex.3[3], ex.3[3]/M),
c(F.3[1], F.3[2], Pi.F.3, M, ex.3[4], ex.3[4]/M),
c(O.3[1], O.3[2], Pi.O.3, M, ex.3[5], ex.3[5]/M) )
rownames(DF.03.01.ind) <- c("C.3", "D.3", "E.3", "F.3", "O.3")
colnames(DF.03.01.ind) <- c("x1", "x2", "Pi", "M", "y", "pr")
# (e) Armado del data frame con todos los puntos del primero, segundo y
#     tercer disenyos:
DF.03.01 <- rbind(DF.02.01,
c(C.3[1], C.3[2], Pi.C.3, M, ex.3[1], ex.3[1]/M),
c(D.3[1], D.3[2], Pi.D.3, M, ex.3[2], ex.3[2]/M),
c(E.3[1], E.3[2], Pi.E.3, M, ex.3[3], ex.3[3]/M),
c(F.3[1], F.3[2], Pi.F.3, M, ex.3[4], ex.3[4]/M),
c(O.3[1], O.3[2], Pi.O.3, M, ex.3[5], ex.3[5]/M) )
```

```
rownames(DF.03.01) <- c("C.1", "D.1", "E.1", "F.1", "O.1", "C.2", "D.2",
"E.2", "F.2", "O.2", "C.3", "D.3", "E.3", "F.3", "O.3")
colnames(DF.03.01) <- c("x1", "x2", "Pi", "M", "y", "pr")
```

Los módulos 4.1.02 al 4.1.15 contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# ------------------------------------------------------------------------
# MODULO 4.2
#
# Ajuste de modelos para todos los puntos utilizados
# ------------------------------------------------------------------------

# En cada data frame de puntos, del DF.03.01 al DF.03.15, ya se ha utili-
# zado el presupuesto previsto de los 1500 puntos. Ahora hay que ajustar
# el modelo "final" para cada data frame, que sera el modelo definitivo
# con que se aproximara la superficie teorica sobre la base de lo apren-
# dido de los puntos experimentales.

# ------------------------------------------------------------------------
# MODULO 4.2.01
#
# Ajuste automatico de modelos para el DF.03.01
# ------------------------------------------------------------------------

rm(uno, x.01, x.02, x.12, x.11, x.22, Pi, m, y, pr)
uno <- rep(1, nrow(DF.03.01))
x.01 <- DF.03.01$x1
x.02 <- DF.03.01$x2
x.12 <- (DF.03.01$x1)*(DF.03.01$x2)
x.11 <- (DF.03.01$x1)*(DF.03.01$x1)
x.22 <- (DF.03.01$x2)*(DF.03.01$x2)
Pi <- DF.03.01$Pi
m <- DF.03.01$M
y <- DF.03.01$y
pr <- DF.03.01$pr
rm(mod.79001A, mod.79001B, mod.79001C, mod.79001D, mod.79001E,
mod.79001F, mod.79001G, mod.79001H)
rm(mod.70000, mod.71000, mod.71100, mod.72100, mod.73100, mod.73400,
mod.73500, mod.74000, mod.74100, mod.74200, mod.79001.1, mod.79001.2)
rm(sum.70000, sum.71000, sum.71100, sum.72100, sum.73100,
sum.74000, sum.74100, sum.74200,             sum.79001.2)
rm(SIG710, SIG720, SIG730, SIG732, SIG733, SIG750, SIG760, SIG770,
SIG780)
```

```
rm(I710, I732, I733, I750, I760)
mod.70000 <- glm(pr ~x.01 + x.02 + x.12 + x.11 + x.22, weights = m,
binomial)
sum.70000 <- summary.glm(mod.70000)$coefficients
p.11.70000 <- sum.70000["x.11", "Pr(>|z|)"]
p.22.70000 <- sum.70000["x.22", "Pr(>|z|)"]
p.70000 <- c("x.11"=p.11.70000, "x.22"=p.22.70000)
xMM.70000 <- get(names(p.70000)[which.max(p.70000)[[1]]])
xmm.70000 <- get(names(p.70000)[which.min(p.70000)[[1]]])
x.MM <- xMM.70000
x.mm <- xmm.70000
mod.71000 <- glm(pr ~x.01 + x.02 + x.12 + x.mm + x.MM, weights = m,
binomial)
sum.71000 <- summary.glm(mod.71000)$coefficients
SIG710 <- sum.71000["x.MM", "Pr(>|z|)"] <= p.valor
SIG720 <- sum.71000["x.mm", "Pr(>|z|)"] <= p.valor
SIG730 <- sum.71000["x.mm", "Pr(>|z|)"] <= p.valor
I710 <- ifelse(SIG710==T, ifelse(SIG720==T, assign("mod.79001A",
glm(pr ~x.01 + x.02 + x.12 + x.11 + x.22, weights = m, binomial)),
assign("mod.71100", glm(pr ~x.01 + x.02 + x.12 + x.MM, weights = m,
binomial))), ifelse(SIG730==T, assign("mod.72100",
glm(pr ~x.01 + x.02 + x.12 + x.mm, weights = m, binomial)),
assign("mod.73100", glm(pr ~x.01 + x.02 + x.12, weights = m,
binomial))))
sum.71100 <- summary.glm(mod.71100)$coefficients
sum.72100 <- summary.glm(mod.72100)$coefficients
sum.73100 <- summary.glm(mod.73100)$coefficients
SIG732 <- sum.71100["x.MM", "Pr(>|z|)"] <= p.valor
SIG733 <- sum.72100["x.mm", "Pr(>|z|)"] <= p.valor
I732 <- ifelse(SIG732==T, assign("mod.79001B",
glm(pr ~x.01 + x.02 + x.12 + x.MM, weights = m, binomial)),
if(SIG732==F) assign("mod.73400", glm(pr ~x.01 + x.02 + x.12,
weights = m, binomial)))
I733 <- ifelse(SIG733==T, assign("mod.79001C",
glm(pr ~x.01 + x.02 + x.12 + x.mm, weights = m, binomial)),
if(SIG733==F) assign("mod.73500", glm(pr ~x.01 + x.02 + x.12,
weights = m, binomial)))
if(SIG732==F) assign("mod.73400", glm(pr ~x.01 + x.02 + x.12,
weights = m, binomial))
if(SIG733==F) assign("mod.73500", glm(pr ~x.01 + x.02 + x.12,
weights = m, binomial))
# El que no de error corresponde al modelo elegido.
# Primero, hay que verificar si alguno de los modelos "finales" ha sido
# elegido o no. Si esto no ocurriera, hay que elegir uno de los 3 mode-
# los: 73100, 73400 y 73500.
```

```
# El que no de error, sera el modelo elegido, y definira el mod.74000:
if(summary.glm(mod.73100)$aic != 0) assign("mod.74000",
glm(pr ~x.01 + x.02 + x.12, weights = m, binomial))
if(summary.glm(mod.73400)$aic != 0) assign("mod.74000",
glm(pr ~x.01 + x.02 + x.12, weights = m, binomial))
if(summary.glm(mod.73500)$aic != 0) assign("mod.74000",
glm(pr ~x.01 + x.02 + x.12, weights = m, binomial))
sum.74000 <- summary.glm(mod.74000)$coefficients
aic.74000 <- summary.glm(mod.74000)$aic
SIG750 <- sum.74000["x.12", "Pr(>|z|)"] <= p.valor
I750 <- ifelse(SIG750==T, assign("mod.79001D",
glm(pr ~x.01 + x.02 + x.12, weights = m, binomial)),
if(SIG750==F & aic.74000 != 0) assign("mod.74100",
glm(pr ~x.01 + x.02, weights = m, binomial)))
sum.74100 <- summary.glm(mod.74100)$coefficients
p.01.410 <- sum.74100["x.01", "Pr(>|z|)"]
p.02.410 <- sum.74100["x.02", "Pr(>|z|)"]
p.74100 <- c("x.01"=p.01.410, "x.02"=p.02.410)
x0M.74100 <- get(names(p.74100)[which.max(p.74100)[[1]]])
x0m.74100 <- get(names(p.74100)[which.min(p.74100)[[1]]])
x.0M <- x0M.74100
x.0m <- x0m.74100
mod.74200 <- glm(pr ~x.0m + x.0M, weights = m, binomial)
sum.74200 <- summary.glm(mod.74200)$coefficients
SIG760 <- sum.74200["x.0M", "Pr(>|z|)"] <= p.valor
SIG770 <- sum.74200["x.0m", "Pr(>|z|)"] <= p.valor
SIG780 <- sum.74200["x.0m", "Pr(>|z|)"] <= p.valor
I760 <- if(SIG760==T) (if(SIG750==F & SIG760==T & SIG770==T)
assign("mod.79001E", glm(pr ~x.0m + x.0M, weights = m, binomial))) else
(if(SIG750==F & SIG760==T & SIG770==F) assign("mod.79001F",
glm(pr ~x.0M, weights = m, binomial)))
I780 <- if(SIG780==T) (if(SIG750==F & SIG760==F & SIG770==T)
assign("mod.79001G", glm(pr ~x.0m, weights = m, binomial))) else
(if(SIG750==F & SIG760==F & SIG770==F) assign("mod.79001H",
glm(pr ~1, weights = m, binomial)))

# --------------------------------
# Desglose del I760 y del I780:

# I760 <- if(SIG760==T) "V760" else "F760"
# V760 <- if(SIG750==F & SIG760==T & SIG770==T)
# assign("mod.79001E", glm(pr ~x.0m + x.0M, weights = m, binomial))

# F760 <- if(SIG750==F & SIG760==T & SIG770==F)
# assign("mod.79001F", glm(pr ~x.0M, weights = m, binomial))
```

```
# I780 <- if(SIG780==T) "V780" else "F780"

# V780 <- if(SIG750==F & SIG760==F & SIG770==T)
# assign("mod.79001G", glm(pr ~x.0m, weights = m, binomial))

# F780 <- if(SIG750==F & SIG760==F & SIG770==F)
# assign("mod.79001H", glm(pr ~1, weights = m, binomial))
# ---------------------------------

# El modelo final correspondiente a DF.02.0X estara guardado en el
# mod.790X.1.
# Para el DF.03.01, sera mod.79001.1
invisible(if(summary.glm(mod.79001A)$aic != 0) assign("mod.79001.1",
mod.79001A))
invisible(if(summary.glm(mod.79001B)$aic != 0) assign("mod.79001.1",
mod.79001B))
invisible(if(summary.glm(mod.79001C)$aic != 0) assign("mod.79001.1",
mod.79001C))
invisible(if(summary.glm(mod.79001D)$aic != 0) assign("mod.79001.1",
mod.79001D))
invisible(if(summary.glm(mod.79001E)$aic != 0) assign("mod.79001.1",
mod.79001E))
invisible(if(summary.glm(mod.79001F)$aic != 0) assign("mod.79001.1",
mod.79001F))
invisible(if(summary.glm(mod.79001G)$aic != 0) assign("mod.79001.1",
mod.79001G))
invisible(if(summary.glm(mod.79001H)$aic != 0) assign("mod.79001.1",
mod.79001H))
summary.glm(mod.79001.1)$coefficients
# Ahora hay que cambiar las variables: de x.mm y x.MM a terminos de x.11
# y x.22.
IN.79001.1 <- attr(mod.79001.1$terms, "intercept")
AT.79001.1 <- attr(mod.79001.1$terms, "term.labels")
# Uno solo de estos "IFX" tiene que resultar verdadero:
IF1 <- if(length(AT.79001.1)==5) "cuadratico en ambos"
IF2 <- if(length(AT.79001.1)==4 & AT.79001.1[4]=="x.MM")
"cuadratico en x.MM"
IF3 <- if(length(AT.79001.1)==4 & AT.79001.1[4]=="x.mm")
"cuadratico en x.mm"
IF4 <- if(length(AT.79001.1)==3) "lineal con interaccion"
IF5 <- if(length(AT.79001.1)==2) "lineal sin interaccion"
IF7 <- if(length(AT.79001.1)==1 & AT.79001.1[1]=="x.0M")
"lineal en x.0M"
IF7 <- if(length(AT.79001.1)==1 & AT.79001.1[1]=="x.0m")
```

```
"lineal en x.0m"
IF8 <- if(IN.79001.1==1 & is.na(AT.79001.1[1]) & is.na(AT.79001.1[2]))
"termino independiente"
eti.79001 <- if(IF1=="cuadratico en ambos") IF1
eti.79001 <- if(IF2=="cuadratico en x.MM") IF2
eti.79001 <- if(IF3=="cuadratico en x.mm") IF3
eti.79001 <- if(IF4=="lineal con interaccion") IF4
eti.79001 <- if(IF5=="lineal sin interaccion") IF5
eti.79001 <- if(IF7=="lineal en x.0M") IF7
eti.79001 <- if(IF7=="lineal en x.0m") IF7
eti.79001 <- if(IF8=="termino independiente") IF8
# El modelo resultante se guarda en un solo modelo, el mod.79001.2:
if(eti.79001=="cuadratico en ambos") assign("mod.79001.2",
glm(pr ~x.01 + x.02 + x.12 + x.11 + x.22, binomial, weights = m))
if(eti.79001=="cuadratico en x.MM" & all(get(AT.79001.1[4])==x.11))
assign("mod.79001.2", glm(pr ~x.01 + x.02 + x.12 + x.11, binomial,
weights = m))
if(eti.79001=="cuadratico en x.MM" & all(get(AT.79001.1[4])==x.22))
assign("mod.79001.2", glm(pr ~x.01 + x.02 + x.12 + x.22, binomial,
weights = m))
if(eti.79001=="cuadratico en x.mm" & all(get(AT.79001.1[4])==x.11))
assign("mod.79001.2", glm(pr ~x.01 + x.02 + x.12 + x.11, binomial,
weights = m))
if(eti.79001=="cuadratico en x.mm" & all(get(AT.79001.1[4])==x.22))
assign("mod.79001.2", glm(pr ~x.01 + x.02 + x.12 + x.22, binomial,
weights = m))
if(eti.79001=="lineal con interaccion") assign("mod.79001.2",
glm(pr ~x.01 + x.02 + x.12, binomial, weights = m))
if(eti.79001=="lineal sin interaccion") assign("mod.79001.2",
glm(pr ~x.01 + x.02, binomial, weights = m))
if(eti.79001=="lineal en x.0M" & all(get(AT.79001.1[1])==x.01))
assign("mod.79001.2", glm(pr ~x.01, binomial, weights = m))
if(eti.79001=="lineal en x.0M" & all(get(AT.79001.1[1])==x.02))
assign("mod.79001.2", glm(pr ~x.02, binomial, weights = m))
if(eti.79001=="lineal en x.0m" & all(get(AT.79001.1[1])==x.01))
assign("mod.79001.2", glm(pr ~x.01, binomial, weights = m))
if(eti.79001=="lineal en x.0m" & all(get(AT.79001.1[1])==x.02))
assign("mod.79001.2", glm(pr ~x.02, binomial, weights = m))
if(eti.79001=="termino independiente") assign("mod.79001.2",
glm(pr ~1, binomial, weights = m))
sum.79001.2 <- summary.glm(mod.79001.2)$coefficients
AT.79001.2 <- attr(mod.79001.2$terms, "term.labels")
# Armado del data frame collage:
```

```
# Modelo candidato: 79001A
# Parametros: b00, x.01, x.02, x.12, x.11, x.22
if(length(AT.79001.2)==5)
assign("DF7.A.01", data.frame(
c(sum.79001.2["(Intercept)","Estimate"],
sum.79001.2["x.01","Estimate"],   sum.79001.2["x.02","Estimate"],
sum.79001.2["x.12","Estimate"],   sum.79001.2["x.11","Estimate"],
sum.79001.2["x.22","Estimate"]),
c(sum.79001.2["(Intercept)","Std. Error"],
sum.79001.2["x.01","Std. Error"], sum.79001.2["x.02","Std. Error"],
sum.79001.2["x.12","Std. Error"], sum.79001.2["x.11","Std. Error"],
sum.79001.2["x.22","Std. Error"]),
c(sum.79001.2["(Intercept)","z value"],
sum.79001.2["x.01","z value"],    sum.79001.2["x.02","z value"],
sum.79001.2["x.12","z value"],    sum.79001.2["x.11","z value"],
sum.79001.2["x.22","z value"]),
c(sum.79001.2["(Intercept)","Pr(>|z|)"],
sum.79001.2["x.01","Pr(>|z|)"],   sum.79001.2["x.02","Pr(>|z|)"],
sum.79001.2["x.12","Pr(>|z|)"],   sum.79001.2["x.11","Pr(>|z|)"],
sum.79001.2["x.22","Pr(>|z|)"])))
colnames(DF7.A.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF7.A.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 79001B
# Parametros: b00, x.01, x.02, x.12, x.11
if(length(AT.79001.2)==4 & AT.79001.2[4]=="x.11")
assign("DF7.B.01", data.frame(
c(sum.79001.2["(Intercept)","Estimate"],
sum.79001.2["x.01","Estimate"],   sum.79001.2["x.02","Estimate"],
sum.79001.2["x.12","Estimate"],   sum.79001.2["x.11","Estimate"],   0),
c(sum.79001.2["(Intercept)","Std. Error"],
sum.79001.2["x.01","Std. Error"], sum.79001.2["x.02","Std. Error"],
sum.79001.2["x.12","Std. Error"], sum.79001.2["x.11","Std. Error"], 0),
c(sum.79001.2["(Intercept)","z value"],
sum.79001.2["x.01","z value"],    sum.79001.2["x.02","z value"],
sum.79001.2["x.12","z value"],    sum.79001.2["x.11","z value"],    0),
c(sum.79001.2["(Intercept)","Pr(>|z|)"],
sum.79001.2["x.01","Pr(>|z|)"],   sum.79001.2["x.02","Pr(>|z|)"],
sum.79001.2["x.12","Pr(>|z|)"],   sum.79001.2["x.11","Pr(>|z|)"],    0)))
colnames(DF7.B.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF7.B.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")
```

```
# Modelo candidato: 79001C
# Parametros: b00, x.01, x.02, x.12, x.22
if(length(AT.79001.2)==4 & AT.79001.2[4]=="x.22")
assign("DF7.C.01", data.frame(
c(sum.79001.2["(Intercept)","Estimate"],
sum.79001.2["x.01","Estimate"],   sum.79001.2["x.02","Estimate"],
sum.79001.2["x.12","Estimate"],   0, sum.79001.2["x.22","Estimate"]),
c(sum.79001.2["(Intercept)","Std. Error"],
sum.79001.2["x.01","Std. Error"], sum.79001.2["x.02","Std. Error"],
sum.79001.2["x.12","Std. Error"], 0, sum.79001.2["x.22","Std. Error"]),
c(sum.79001.2["(Intercept)","z value"],
sum.79001.2["x.01","z value"],    sum.79001.2["x.02","z value"],
sum.79001.2["x.12","z value"],    0, sum.79001.2["x.22","z value"]),
c(sum.79001.2["(Intercept)","Pr(>|z|)"],
sum.79001.2["x.01","Pr(>|z|)"],   sum.79001.2["x.02","Pr(>|z|)"],
sum.79001.2["x.12","Pr(>|z|)"],   0, sum.79001.2["x.22","Pr(>|z|)"])))
colnames(DF7.C.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF7.C.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 79001D
# Parametros: b00, x.01, x.02, x.12
if(length(AT.79001.2)==3)
assign("DF7.D.01", data.frame(
c(sum.79001.2["(Intercept)","Estimate"],
sum.79001.2["x.01","Estimate"],   sum.79001.2["x.02","Estimate"],
sum.79001.2["x.12","Estimate"],   0, 0),
c(sum.79001.2["(Intercept)","Std. Error"],
sum.79001.2["x.01","Std. Error"], sum.79001.2["x.02","Std. Error"],
sum.79001.2["x.12","Std. Error"], 0, 0),
c(sum.79001.2["(Intercept)","z value"],
sum.79001.2["x.01","z value"],    sum.79001.2["x.02","z value"],
sum.79001.2["x.12","z value"],    0, 0),
c(sum.79001.2["(Intercept)","Pr(>|z|)"],
sum.79001.2["x.01","Pr(>|z|)"],   sum.79001.2["x.02","Pr(>|z|)"],
sum.79001.2["x.12","Pr(>|z|)"],   0, 0)))
colnames(DF7.D.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF7.D.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 79001E
# Parametros: b00, x.01, x.02
if(length(AT.79001.2)==2)
assign("DF7.E.01", data.frame(
c(sum.79001.2["(Intercept)","Estimate"],
```

```
sum.79001.2["x.01","Estimate"],   sum.79001.2["x.02","Estimate"],
0, 0, 0),
c(sum.79001.2["(Intercept)","Std. Error"],
sum.79001.2["x.01","Std. Error"], sum.79001.2["x.02","Std. Error"],
0, 0, 0),
c(sum.79001.2["(Intercept)","z value"],
sum.79001.2["x.01","z value"],    sum.79001.2["x.02","z value"],
0, 0, 0),
c(sum.79001.2["(Intercept)","Pr(>|z|)"],
sum.79001.2["x.01","Pr(>|z|)"],   sum.79001.2["x.02","Pr(>|z|)"],
0, 0, 0)))
colnames(DF7.E.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF7.E.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 79001F
# Parametros: b00, x.01
if(length(AT.79001.2)==1 & AT.79001.2[1]=="x.01")
assign("DF7.F.01", data.frame(
c(sum.79001.2["(Intercept)","Estimate"],
sum.79001.2["x.01","Estimate"],   0, 0, 0, 0),
c(sum.79001.2["(Intercept)","Std. Error"],
sum.79001.2["x.01","Std. Error"], 0, 0, 0, 0),
c(sum.79001.2["(Intercept)","z value"],
sum.79001.2["x.01","z value"],    0, 0, 0, 0),
c(sum.79001.2["(Intercept)","Pr(>|z|)"],
sum.79001.2["x.01","Pr(>|z|)"],   0, 0, 0, 0)))
colnames(DF7.F.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF7.F.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")

# Modelo candidato: 79001G
# Parametros: b00, x.02
if(length(AT.79001.2)==1 & AT.79001.2[1]=="x.02")
assign("DF7.G.01", data.frame(
c(sum.79001.2["(Intercept)","Estimate"],
0, sum.79001.2["x.02","Estimate"],   0, 0, 0),
c(sum.79001.2["(Intercept)","Std. Error"],
0, sum.79001.2["x.02","Std. Error"], 0, 0, 0),
c(sum.79001.2["(Intercept)","z value"],
0, sum.79001.2["x.02","z value"],    0, 0, 0),
c(sum.79001.2["(Intercept)","Pr(>|z|)"],
0, sum.79001.2["x.02","Pr(>|z|)"],   0, 0, 0)))
colnames(DF7.G.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF7.G.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
```

```
"x.22")

# Modelo candidato: 79001H
# Parametros: b00
if(length(AT.79001.2)==0)
assign("DF7.H.01", data.frame(
c(sum.79001.2["(Intercept)","Estimate"],   0, 0, 0, 0, 0),
c(sum.79001.2["(Intercept)","Std. Error"], 0, 0, 0, 0, 0),
c(sum.79001.2["(Intercept)","z value"],    0, 0, 0, 0, 0),
c(sum.79001.2["(Intercept)","Pr(>|z|)"],   0, 0, 0, 0, 0)))
colnames(DF7.H.01) <- c("Estimate", "Std. Error", "z value", "Pr(>|z|)")
rownames(DF7.H.01) <- c("(Intercept)", "x.01", "x.02", "x.12", "x.11",
"x.22")
DF7.01 <- DF7.A.01
DF7.01 <- DF7.B.01
DF7.01 <- DF7.C.01
DF7.01 <- DF7.D.01
DF7.01 <- DF7.E.01
DF7.01 <- DF7.F.01
DF7.01 <- DF7.G.01
DF7.01 <- DF7.H.01
# El modelo ajustado automaticamente converge en un solo data frame,
# DF7.01.
```

Los módulos `4.2.02` al `4.2.15` contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# ------------------------------------------------------------------------
# MODULO 5
#
# Calculo de estadisticos de calidad de ajuste en funcion de los deter-
# minantes de la Matriz Observada de Informacion de Fisher, para cada
# uno de los modelos ajustados.
# ------------------------------------------------------------------------

# INTRODUCCION:
# Habiendo llegado a un modelo final en el que se han utilizado todos los
# puntos presupuestados, se calcula el determinante de la matriz de in-
# formacion observada de Fisher para cada modelo como medida de calidad
# del ajuste.
# Ya que la MIF es la inversa de la matriz de varianzas y covarianzas de
# los # coeficientes del modelo final, se buscaran aquellos modelos para
# los cuales # el determinante de la MIF resulte el mas grande posible.
```

```
# VALORACION DE LA ESTRATEGIA:
# Partiendo del modelo ajustado encontrado al haber utilizado todo el
# presupuesto dado por el numero de puntos disponibles, se trata de valo-
# rar la bondad de la estrategia empleada mediante la definicion de un
# estadistico muestral proveniente de los puntos experimentales y del mo-
# delo ajustado.
# Con el ultimo modelo ajustado, se define el determinante de la matriz
# de informacion de Fisher observada (DEMIFO).
# Se repite el procedimiento mediante la generacion de nuevos vectores de
# numeros de exitos y proporciones, con los que se llega a sus respecti-
# vos determinantes de la MIFO. Se hacen en total 5 corridas, y se calcu-
# lan estadisticos muestrales tipo:
# media(DEMIFO1,...,DEMIFO5), y
# desvtipo(DEMIFO1,...,DEMIFO5),
# con los cuales se pueden determinar intervalos de confianza.

# GENERALIZACION DE LA ESTRATEGIA:
# Lo mismo que se hizo para un par de valores (L,S) se repite para los 24
# valores restantes de la matriz LS, calculandose los mismos estadisticos
# muestrales.
# Finalmente, se grafica sobre un plano LS:
# los valores de la media de los DEMIFO de cada una de las 25 situaciones,
# los valores de la desvtipo de los 25 DEMIFO.
# Se tratara entonces de elegir los valores de L y de S que hagan maxima
# la funcion media(DEMIFO) (es decir, determinante de la matriz de varian-
# zas y covarianzas minima) y los que hagan minima la funcion
# desvtipo(DEMIFO).
# Cada una de estas soluciones definiran condiciones "optimas" para expe-
# rimentar de acuerdo con esta estrategia asi definida.

# DEFINICION DEL DETERMINANTE DE LA MIF:
# Para el modF.01, definiremos el determinante de la matriz de informa-
# cion observada de Fisher (d.1.01) para el modelo completo (p = 6) como:
# d.1.01 = det[I(beta.hat.01)] = det[X' W.1.hat X]
#    dim(X')      = p x n = 6 x 15
#    dim(W.1.hat) = n x n = 15 x 15
#    dim(X)       = n x p = 15 x 6
# => dim(d.1.01)  = p x p = 6 x 6

# ------------------------------------------------------------------------
# MODULO 5.1
#
# Primera variante para el calculo de la matriz W.1.hat
# ------------------------------------------------------------------------
```

```
# Se considerara que W.1.hat = f(pr.obs),
# en donde 'pr.obs' es la proporcion observada para cada numero de exitos.
# Borro todo antecedente:
rm(d.1.01, d.1.02, d.1.03, d.1.04, d.1.05, d.1.06, d.1.07, d.1.08, d.1.09,
d.1.10, d.1.11, d.1.12, d.1.13. d.1.14, d.1 15)

# ------------------------------------------------------------------------
# MODULO 5.1.01
#
# Determinante 1 de la MIF para el conjunto de datos DF.03.01
# ------------------------------------------------------------------------

# (a) Matriz X de puntos de disenyo (n = 15 condiciones experimentales):
#     Atenti: primera columna de unos.
X.nxp.01 <- matrix(
cbind(
rep(1, 15),
DF.03.01$x1,
DF.03.01$x2,
DF.03.01$x1 * DF.03.01$x2,
DF.03.01$x1 * DF.03.01$x1,
DF.03.01$x2 * DF.03.01$x2), nc=6)
# (b) Matriz X', transpuesta de X:
X.pxn.01 <- t(X.nxp.01)
# (c) Matriz diagonal de "pesos" estimados, W.1.hat, a partir del DF del
#     disenyo:
w.1.hat.01 <- numeric(0)
for(i in 1:nrow(DF.03.01))
{
w.1.hat.01[i] <- (DF.03.01[i,"M"])*(DF.03.01[i,"pr"])*(1 -
DF.03.01[i,"pr"])
}
# Con la funcion diag() construyo la matriz diagonal W.1.hat:
W.1.hat.01 <- diag(w.1.hat.01)
# (d) Calculo de la matriz de informacion de Fisher observada en
#     DF.03.01:
mif.1.01 <- X.pxn.01 %*% W.1.hat.01 %*% X.nxp.01
# (e) Determinante de mif.1.01:
d.1.01 <- det(mif.1.01)
# Y esta es una medida de calidad de ajuste del modelo, para valores
# fijos de M, L y s, y para un punto de partida 0.1, que se toma como
# primer centro  de experimentacion.
```

Los módulos 5.1.02 al 5.1.15 contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o

columna de la matriz respectiva.

```
# ------------------------------------------------------------------------
# MODULO 5.1.16
#
# Medidas resumidas de los determinantes d.1.01 al d.1.15
# ------------------------------------------------------------------------

# 1. Determinante 1, W = f(pr.obs):
v.d.1 <- c(d.1.01, d.1.02, d.1.03, d.1.04, d.1.05, d.1.06, d.1.07,
d.1.08, d.1.09, d.1.10, d.1.11, d.1.12, d.1.13, d.1.14, d.1.15)
min.v.d.1 <- min(v.d.1)
max.v.d.1 <- max(v.d.1)
rng.v.d.1 <- max.v.d.1 - min.v.d.1
med.v.d.1 <- mean(v.d.1)
std.v.d.1 <- sd(v.d.1)
cvr.v.d.1 <- std.v.d.1/med.v.d.1
# 2. Transformacion logaritmica del determinante 1:
log.v.d.1 <- log(v.d.1)
min.log.v.d.1 <- min(log.v.d.1)
max.log.v.d.1 <- max(log.v.d.1)
rng.log.v.d.1 <- max.log.v.d.1 - min.log.v.d.1
med.log.v.d.1 <- mean(log.v.d.1)
std.log.v.d.1 <- sd(log.v.d.1)
cvr.log.v.d.1 <- std.log.v.d.1/med.log.v.d.1

# ------------------------------------------------------------------------
# MODULO 5.2
#
# Segunda variante para el calculo de la matriz W.2.hat
# ------------------------------------------------------------------------

# Se considerara que W.2.hat = f(Pi.hat),
# en donde 'Pi.hat' es la probabilidad de exito calculada con el modelo
# ajustado.
# Borro todo antecedente:
rm(d.2.01, d.2.02, d.2.03, d.2.04, d.2.05, d.2.06, d.2.07, d.2.08, d.2.09,
d.2.10, d.2.11, d.2.12, d.2.13. d.2.14, d.2.15)

# ------------------------------------------------------------------------
# MODULO 5.2.01
#
# Determinante 2 de la MIF para el conjunto de datos DF.03.01
# ------------------------------------------------------------------------
```

```
# (a) Matriz X de puntos de disenyo (n = 15 condiciones experimentales):
#     Es la misma que en el caso del determinante 1, X.nxp.01
# (b) Matriz X', transpuesta de X:
#     Idem anterior: t(X.nxp.01)
# (c) Matriz diagonal de "pesos" estimados, W.2.hat, a partir del DF del
#     disenyo:
#     Se debe construir una funcion que calcule la probabilidad de exito
#     ajustada utilizando los coeficientes del modelo ajustado final,
#     DF7.01:
b.00.hat7.01 <- DF7.01["(Intercept)", "Estimate"]
b.01.hat7.01 <- DF7.01["x.01", "Estimate"]
b.02.hat7.01 <- DF7.01["x.02", "Estimate"]
b.12.hat7.01 <- DF7.01["x.12", "Estimate"]
b.11.hat7.01 <- DF7.01["x.11", "Estimate"]
b.22.hat7.01 <- DF7.01["x.22", "Estimate"]
# Primero calculo la probabilidad construyendo una funcion:
#     Pi.hat.01 = exp(eta.hat.01) / [1+exp(eta.hat.01)]
#     eta.hat.01 = f(betas.hat del DF7.01)
eta.hat.01 <- numeric(0)
Pi.hat.01 <- numeric(0)
for(i in 1:nrow(DF.03.01))
{
eta.hat.01[i] <- b.00.hat7.01+
b.01.hat7.01*DF.03.01[i,"x1"]+
b.02.hat7.01*DF.03.01[i,"x2"]+
b.11.hat7.01*(DF.03.01[i,"x1"])*(DF.03.01[i,"x1"])+
b.22.hat7.01*(DF.03.01[i,"x2"])*(DF.03.01[i,"x2"])+
b.12.hat7.01*(DF.03.01[i,"x1"])*(DF.03.01[i,"x2"])
Pi.hat.01[i] <- exp(eta.hat.01[i])/(1+exp(eta.hat.01[i]))
}
# Luego, calculo la matriz de pesos del data frame, con una funcion
# tambien:
w.2.hat.01 <- numeric(0)
for(i in 1:nrow(DF.03.01))
{
w.2.hat.01[i] <- M * Pi.hat.01[i] * (1 - Pi.hat.01[i])
}
# Con la funcion diag() construyo la matriz diagonal W.2.hat:
W.2.hat.01 <- round(diag(w.2.hat.01), 3)
# (d) Calculo de la matriz de informacion de Fisher observada en
#     DF.03.01:
mif.2.01 <- X.pxn.01 %*% W.2.hat.01 %*% X.nxp.01
# (e) Determinante de mif.2.01:
d.2.01 <- det(mif.2.01)
```

Los módulos `5.2.02` al `5.2.15` contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# ------------------------------------------------------------------------
# MODULO 5.2.16
#
# Medidas resumidas de los determinantes d.2.01 al d.2.15
# ------------------------------------------------------------------------

# 1. Determinante 2, W = f(Pi.hat):
v.d.2 <- c(d.2.01, d.2.02, d.2.03, d.2.04, d.2.05, d.2.06, d.2.07,
d.2.08, d.2.09, d.2.10, d.2.11, d.2.12, d.2.13, d.2.14, d.2.15)
min.v.d.2 <- min(v.d.2)
max.v.d.2 <- max(v.d.2)
rng.v.d.2 <- max.v.d.2 - min.v.d.2
med.v.d.2 <- mean(v.d.2)
std.v.d.2 <- sd(v.d.2)
cvr.v.d.2 <- std.v.d.2/med.v.d.2
# 2. Transformacion logaritmica del determinante 2:
log.v.d.2 <- log(v.d.2)
min.log.v.d.2 <- min(log.v.d.2)
max.log.v.d.2 <- max(log.v.d.2)
rng.log.v.d.2 <- max.log.v.d.2 - min.log.v.d.2
med.log.v.d.2 <- mean(log.v.d.2)
std.log.v.d.2 <- sd(log.v.d.2)
cvr.log.v.d.2 <- std.log.v.d.2/med.log.v.d.2

# ------------------------------------------------------------------------
# MODULO 6

# Calculo de estadisticos de calidad de ajuste en funcion de la altura
# denominada 'Pi.max' sobre la superficie teorica, para el punto (x1max,
# x2max) en que el modelo ajustado alcanza el maximo, para cada uno de
# los modelos ajustados.
# ------------------------------------------------------------------------

# INTRODUCCION:
# Habiendo llegado a un modelo final en el que se han utilizado todos
# los puntos presupuestados, se calcula cuanto vale en la funcion teori-
# ca aquel punto sobre x1x2 que hace maximo el valor del modelo ajustado.
# Se toma este estadistico para cada modelo como medida de calidad del
# ajuste.
# Ya que lo que se pretende es tratar de acercarse lo maximo posible al
# maximo teorico, se tratara de maximizar esta funcion.
```

```
# VALORACION DE LA ESTRATEGIA:
# Partiendo del modelo ajustado encontrado al haber utilizado todo el
# presupuesto dado por el numero de puntos disponibles, se trata de valo-
# rar la bondad de la estrategia empleada mediante la definicion de un
# estadistico muestral proveniente de los puntos experimentales y del
# modelo ajustado.

# Definimos este estadistico como:
# h = Pi(x.hat.max)
# que es el valor de la funcion teorica evaluado en el maximo calculado
# mediante el modelo ajustado. La determinacion del punto x.DF.hat.max
# depende basicamente de la forma que tome el modelo ajustado:
# a) En el caso en que el modelo ajustado sea un maximo, este quedara de-
#    finido de manera puntual, sin lugar a dudas. En este caso, el argu-
#    mento del estadistico "h" se define como:
#    x.hat.max = arg.x {max[Pi.hat(x)]}
# b) En el caso en que tuviera un minimo, un punto de silla o un plano
#    como modelos ajustados, se sigue un criterio que tiene sentido con
#    la practica: tomar como argumento el maximo valor de la proporcion
#    observada:
#    x.hat.max = arg.x {max[Pi.hat.01(x),...,Pi.hat.15(x)]}
# Luego, con el argumento ya calculado, se lo evalua en la superficie
# teorica. Cuanto mayor sea el valor de este estadistico, tanto mejor
# aproximara el modelo ajustado al teorico.

# ------------------------------------------------------------------------
# MODULO 6.1
#
# Clasificacion de los modelos ajustados segun si estos contienen maxi-
# mos o no maximos (minimos, puntos de silla o planos).
# ------------------------------------------------------------------------

# El criterio que se sigue es el de evaluar la matriz de terminos cuadra-
# ticos del modelo completo y calcular sus autovalores. Si el resultado
# es que ambos autovalores tienen SIGNO NEGATIVO, entonces la superficie
# contiene un maximo. En el resto de los casos, diremos que la superficie
# no contiene un maximo, simplemente.

# ------------------------------------------------------------------------
# MODULO 6.1.01
#
# Clasificacion de la superficie que ajusta al DF.03.01
# ------------------------------------------------------------------------
```

```
# Resulta de ajustar un glm() con todo el DF.03.01 y tomar todos sus
# terminos.
uno <- rep(1, nrow(DF.03.01))
x.01 <- DF.03.01$x1
x.02 <- DF.03.01$x2
x.12 <- (DF.03.01$x1)*(DF.03.01$x2)
x.11 <- (DF.03.01$x1)*(DF.03.01$x1)
x.22 <- (DF.03.01$x2)*(DF.03.01$x2)
Pi <- DF.03.01$Pi
m <- DF.03.01$M
y <- DF.03.01$y   # El caso estudiado.
pr <- y/m
mod.F.01 <- glm(pr ~x.01 + x.02 + x.12 + x.11 + x.22, weights = m,
binomial)
sum.mod.F.01 <- summary.glm(mod.F.01)$coefficients
# Construccion de la matriz B, de coeficientes de segundo orden del mode-
# lo ajustado:
#      [b.hat.11      0.5*b.hat.12]
# BB = [                           ] =
#      [0.5*b.hat.12      b.hat.22]
#   [ DF.03.01["x.11","Estimate"]       0.5*DF.03.01["x.12","Estimate"] ]
# = []
#   [ 0.5*DF.03.01["x.12","Estimate"]       DF.03.01["x.22","Estimate"] ]
# Construyo la matriz:
BB.01 <- matrix(c(
sum.mod.F.01["x.11","Estimate"], 0.5*sum.mod.F.01["x.12","Estimate"],
0.5*sum.mod.F.01["x.12","Estimate"], sum.mod.F.01["x.22","Estimate"]),
nc=2, byrow=T)
# Calculo los autovalores: haciendo eigen(BB.01)[[1]]
# Nota: al indicar el [[1]], me devuelve solamente los autovalores, ya
#       que la funcion eigen() tambien tiene el [[2]], que son los au-
#       tovectores.
# Compruebo si tienen igual signo:
prueba.01 <- ifelse(sign(eigen(BB.01)[[1]][1])==(-1) &
sign(eigen(BB.01)[[1]][2])==(-1), "Maximo", "No Maximo")

# Criterio 1:
# ----------
rm(X.hat.01, Pi.X.hat.01, h.1.01)
# Si la superficie ajustada tiene un maximo, entonces se elige el punto
# sobre el plano de los factores que corresponda a ese maximo.
# Para calcular el maximo se utiliza la formula:
# Xs=(x1s,x2s)'= -0.5 * B^-1 * b,
# Esta expresion surge de d/dX[Pi.hat(x1,x2,beta.hat)] = 0.
# Si se cumple que prueba.01 da un maximo, entonces le asigno el valor
```

```
# X.hat.01:
if(prueba.01=="Maximo") assign("X.hat.01",
(-0.5)*solve(BB.01) %*% c(sum.mod.F.01["x.01","Estimate"],
sum.mod.F.01["x.02","Estimate"]))
# Notese que las cantidades para el calculo de Xs se refieren todas a
# los resultados del modelo ajustado, expresados en sum.mod.F.01.
# Finalmente, si existe este punto maximo, nuestro valor buscado sera
# el evaluado en la funcion teorica f.03.01:
f.03.01 <- function(x1, x2)
{
eta.3.01 <- M7[k.2, "B.00"]+
M7[k.2, "B.01"]*x1+
M7[k.2, "B.02"]*x2+
M7[k.2, "B.12"]*x1*x2+
M7[k.2, "B.11"]*x1*x1+
M7[k.2, "B.22"]*x2*x2
Pi.3.01 <- exp(eta.3.01)/(1+exp(eta.3.01))
round(Pi.3.01, 6)
}
# Nuevamente le pongo una condicional para encontrar el valor buscado:
if(prueba.01=="Maximo") assign("Pi.X.hat.01", f.03.01(X.hat.01[1],
X.hat.01[2]))
# Si la superficie tiene un maximo, las variables recogen sus valores y
# los guardan. Esto no afecta a los resultados del criterio 2.

# Criterio 2:
# ----------
# Si la superficie ajustada no contiene un maximo, se toma como x.hat.max
# a aquel valor que corresponda con la 'pr' maxima del DF.03.01.
# Paso a paso:
# a. Valor maximo de 'pr' en el DF.03.01:
max(DF.03.01$pr)
# b. Comprobacion de existencia de maximos multiples:
length(which(DF.03.01$pr==max(DF.03.01$pr)))
# [1] 1 ----> 1 solo maximo
# c. Indice absoluto del maximo dentro de la matriz:
which.max(DF.03.01$pr)
# Devuelve la posicion numero en la columna 'pr', de arriba hacia abajo.
# d. Numero de fila correspondiente al maximo:
#    Debo representarla como "as.matrix".
row.max.01 <- row(as.matrix(DF.03.01$pr))[which.max(DF.03.01$pr)]
# e. Valor en la columna x1 correspondiente al maximo en f.03.01:
x1.max.01 <- DF.03.01[row.max.01, 1]
# f. Valor en la columna x2 correspondiente al maximo en f.03.01:
x2.max.01 <- DF.03.01[row.max.01, 2]
```

```
# g. Valor final del punto que corresponde al maximo del modelo ajustado:
# Tengo que ponerle un 'if' por si el modelo tiene maximo:
if(prueba.01=="No Maximo") assign("X.hat.01", c(x1.max.01,x2.max.01))
# Finalmente, nuestro valor buscado sera el evaluado en la funcion teorica
# f.03.01:
Pi.X.hat.01 <- f.03.01(X.hat.01[1], X.hat.01[2])
# Si es un 'Maximo', las variables se actualizan: X.hat.01 y Pi.X.hat.01.
# Y si es un 'No Maximo', entonces se conserva el valor calculado en el
# criterio 1.
# Guardo el resultado en la variable h:
h.1.01 <- Pi.X.hat.01
```

Los módulos `6.1.02` al `6.1.15` contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# ------------------------------------------------------------------------
# MODULO 6.1.16
#
# Medidas resumidas de las funciones de altura Pi.hat
# ------------------------------------------------------------------------

# 1. La funcion P.hat:
v.h.1 <- c(h.1.01, h.1.02, h.1.03, h.1.04, h.1.05, h.1.06, h.1.07,
h.1.08, h.1.09, h.1.10, h.1.11, h.1.12, h.1.13, h.1.14, h.1.15)
min.v.h.1 <- min(v.h.1)
max.v.h.1 <- max(v.h.1)
rng.v.h.1 <- max.v.h.1 - min.v.h.1
med.v.h.1 <- mean(v.h.1)
std.v.h.1 <- sd(v.h.1)
cvr.v.h.1 <- std.v.h.1/med.v.h.1
# 2. Transformacion logit de la altura Pi.hat:
log.v.h.1 <- log(v.h.1/(1-v.h.1))
min.log.v.h.1 <- min(log.v.h.1)
max.log.v.h.1 <- max(log.v.h.1)
rng.log.v.h.1 <- max.log.v.h.1 - min.log.v.h.1
med.log.v.h.1 <- mean(log.v.h.1)
std.log.v.h.1 <- sd(log.v.h.1)
cvr.log.v.h.1 <- std.log.v.h.1/med.log.v.h.1
# 3. Transformacion del arco seno de Pi.hat:
# Fuente: Bisgaard & Fuller (1994)
arc.v.h.1 <- asin(sqrt(v.h.1))
min.arc.v.h.1 <- min(arc.v.h.1)
max.arc.v.h.1 <- max(arc.v.h.1)
rng.arc.v.h.1 <- max.arc.v.h.1 - min.arc.v.h.1
```

```
med.arc.v.h.1 <- mean(arc.v.h.1)
std.arc.v.h.1 <- sd(arc.v.h.1)
cvr.arc.v.h.1 <- std.arc.v.h.1/med.arc.v.h.1
# 4. Transformacion de Freeman & Tukey:
# Fuente: Bisgaard & Fuller (1994)
fyt.v.h.1 <- 0.5*(
asin(sqrt((M*v.h.1)/(M+1))) +
asin(sqrt(((M*v.h.1)+1)/(M+1))) )
min.fyt.v.h.1 <- min(fyt.v.h.1)
max.fyt.v.h.1 <- max(fyt.v.h.1)
rng.fyt.v.h.1 <- max.fyt.v.h.1 - min.fyt.v.h.1
med.fyt.v.h.1 <- mean(fyt.v.h.1)
std.fyt.v.h.1 <- sd(fyt.v.h.1)
cvr.fyt.v.h.1 <- std.fyt.v.h.1/med.fyt.v.h.1

# ------------------------------------------------------------------------
# MODULO 7

# Calculo de otros estadisticos de calidad de ajuste generales para cada
# uno de los modelos ajustados: log-likelihood, devianza residual y aic.
# ------------------------------------------------------------------------

# Hay que utilizar los modelos ajustados de cada uno de los DF.03.XX,
# pero no de los obtenidos por el criterio jerarquico sino de los comple-
# tos. Tomo los modelos del MODULO 6.1, que son de la forma mod.F.XX

# ------------------------------------------------------------------------
# MODULO 7.1
#
# Determinacion de las log-verosimilitudes de cada modelo ajustado
# ------------------------------------------------------------------------

# Atenti: cargar previamente el paquete 'stats'
j.1.01 <- logLik(mod.F.01)
j.1.02 <- logLik(mod.F.02)
j.1.03 <- logLik(mod.F.03)
j.1.04 <- logLik(mod.F.04)
j.1.05 <- logLik(mod.F.05)
j.1.06 <- logLik(mod.F.06)
j.1.07 <- logLik(mod.F.07)
j.1.08 <- logLik(mod.F.08)
j.1.09 <- logLik(mod.F.09)
j.1.10 <- logLik(mod.F.10)
j.1.11 <- logLik(mod.F.11)
j.1.12 <- logLik(mod.F.12)
```

```
j.1.13 <- logLik(mod.F.13)
j.1.14 <- logLik(mod.F.14)
j.1.15 <- logLik(mod.F.15)
v.j.1 <- c(j.1.01, j.1.02, j.1.03, j.1.04, j.1.05, j.1.06, j.1.07,
j.1.08, j.1.09, j.1.10, j.1.11, j.1.12, j.1.13, j.1.14, j.1.15)
min.v.j.1 <- min(v.j.1)
max.v.j.1 <- max(v.j.1)
rng.v.j.1 <- max.v.j.1 - min.v.j.1
med.v.j.1 <- mean(v.j.1)
std.v.j.1 <- sd(v.j.1)
cvr.v.j.1 <- std.v.j.1/med.v.j.1

# ------------------------------------------------------------------------
# MODULO 7.2
#
# Determinacion de las devianzas residuales de cada modelo ajustado
# ------------------------------------------------------------------------

j.2.01 <- summary.glm(mod.F.01)$deviance
j.2.02 <- summary.glm(mod.F.02)$deviance
j.2.03 <- summary.glm(mod.F.03)$deviance
j.2.04 <- summary.glm(mod.F.04)$deviance
j.2.05 <- summary.glm(mod.F.05)$deviance
j.2.06 <- summary.glm(mod.F.06)$deviance
j.2.07 <- summary.glm(mod.F.07)$deviance
j.2.08 <- summary.glm(mod.F.08)$deviance
j.2.09 <- summary.glm(mod.F.09)$deviance
j.2.10 <- summary.glm(mod.F.10)$deviance
j.2.11 <- summary.glm(mod.F.11)$deviance
j.2.12 <- summary.glm(mod.F.12)$deviance
j.2.13 <- summary.glm(mod.F.13)$deviance
j.2.14 <- summary.glm(mod.F.14)$deviance
j.2.15 <- summary.glm(mod.F.15)$deviance
v.j.2 <- c(j.2.01, j.2.02, j.2.03, j.2.04, j.2.05, j.2.06, j.2.07,
j.2.08, j.2.09, j.2.10, j.2.11, j.2.12, j.2.13, j.2.14, j.2.15)
min.v.j.2 <- min(v.j.2)
max.v.j.2 <- max(v.j.2)
rng.v.j.2 <- max.v.j.2 - min.v.j.2
med.v.j.2 <- mean(v.j.2)
std.v.j.2 <- sd(v.j.2)
cvr.v.j.2 <- std.v.j.2/med.v.j.2
```

```
# ------------------------------------------------------------------
# MODULO 7.3
#
# Determinacion de los valores 'aic' de cada modelo ajustado
# ------------------------------------------------------------------

j.3.01 <- summary.glm(mod.F.01)$aic
j.3.02 <- summary.glm(mod.F.02)$aic
j.3.03 <- summary.glm(mod.F.03)$aic
j.3.04 <- summary.glm(mod.F.04)$aic
j.3.05 <- summary.glm(mod.F.05)$aic
j.3.06 <- summary.glm(mod.F.06)$aic
j.3.07 <- summary.glm(mod.F.07)$aic
j.3.08 <- summary.glm(mod.F.08)$aic
j.3.09 <- summary.glm(mod.F.09)$aic
j.3.10 <- summary.glm(mod.F.10)$aic
j.3.11 <- summary.glm(mod.F.11)$aic
j.3.12 <- summary.glm(mod.F.12)$aic
j.3.13 <- summary.glm(mod.F.13)$aic
j.3.14 <- summary.glm(mod.F.14)$aic
j.3.15 <- summary.glm(mod.F.15)$aic
v.j.3 <- c(j.3.01, j.3.02, j.3.03, j.3.04, j.3.05, j.3.06, j.3.07,
j.3.08, j.3.09, j.3.10, j.3.11, j.3.12, j.3.13, j.3.14, j.3.15)
min.v.j.3 <- min(v.j.3)
max.v.j.3 <- max(v.j.3)
rng.v.j.3 <- max.v.j.3 - min.v.j.3
med.v.j.3 <- mean(v.j.3)
std.v.j.3 <- sd(v.j.3)
cvr.v.j.3 <- std.v.j.3/med.v.j.3

# ------------------------------------------------------------------
# MODULO 8
#
# Resumen de puntos experimentales, modelos y graficos
# ------------------------------------------------------------------

# Para cada uno de los 15 data frames de puntos generados aleatoriamente,
# se tienen sus respectivos modelos ajustados:
# Puntos del DF.03.01 -----> Modelo ajustado DF7.01
# Puntos del DF.03.02 -----> Modelo ajustado DF7.02
# ...
# Puntos del DF.03.15 -----> Modelo ajustado DF7.15
```

```
# ------------------------------------------------------------------------
# MODULO 8.1
#
# Puntos experimentales, angulos y saltos
# ------------------------------------------------------------------------

# Un resumen en forma de lista para cada uno de los data frame de puntos.

# ------------------------------------------------------------------------
# MODULO 8.1.01
#
# Puntos experimentales, angulos y saltos para el DF.03.01
# ------------------------------------------------------------------------

list.P.01 <- list(
"Primer factorial: DF.01.01" = DF.03.01[1:5,],
"Primer salto: S5.01" = S5.01,
"Primer angulo, en grados" = phi5.g.01,
"Segundo factorial: DF.02.01" = DF.03.01[6:10,],
"Segundo salto: S6.01" = S6.01,
"Segundo angulo, en grados" = phi6.g.01,
"Los tres factoriales: DF.03.01" = DF.03.01,
"Modelo final:"= DF7.01,
"Naturaleza del modelo"= paste("", prueba.01),
"Autovalores"= paste("", round(eigen(BB.01)[[1]], 4)))
```

Los módulos `8.1.02` al `8.1.15` contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

```
# ------------------------------------------------------------------------
# MODULO 8.2
#
# Resumen general de los indicadores para cada modelo ajustado, con sus
# indices de bondad de ajuste.
# ------------------------------------------------------------------------

# Resumen general de todos los modelos:
# Columna  1: Det.1 (determinante de la MIF, criterio 1)
# Columna  2: Log.1 (logaritmo de Det.1)
# Columna  3: Det.2 (determinante de la MIF, criterio 2)
# Columna  4: Log.2 (logaritmo de Det.2)
# Columna  5: Pi.mx (funcion teorica evaluada en el maximo ajustado)
# Columna  6: Logit (transformacion logit de Pi.mx)
```

```
# Columna  7: Arc.s (transformacion angular de Pi.mx)
# Columna  8: Fr-Tk (transformacion de Freeman y Tukey para Pi.mx)
# Columna  9: L.Lik (log-verosimilitud del modelo ajustado)
# Columna 10: Dev.R (devianza residual del modelo ajustado)
# Columna 11: E.AIC (estadistico AIC de Akaike)
# Fila  1: datos del DF.03.01
# Fila  2: datos del DF.03.02
# Fila  3: datos del DF.03.03
# Fila  4: datos del DF.03.04
# Fila  5: datos del DF.03.05
# Fila  6: datos del DF.03.06
# Fila  7: datos del DF.03.07
# Fila  8: datos del DF.03.08
# Fila  9: datos del DF.03.09
# Fila 10: datos del DF.03.10
# Fila 11: datos del DF.03.11
# Fila 12: datos del DF.03.12
# Fila 13: datos del DF.03.13
# Fila 14: datos del DF.03.14
# Fila 15: datos del DF.03.15
# Fila 16: Valor maximo
# Fila 17: Valor minimo
# Fila 18: Rango (maximo - minim0)
# Fila 19: Promedio
# Fila 20: Desviacion estandar
# Fila 21: Coeficiente de variacion
# Columnas:
col.01 <- round(c(v.d.1, max.v.d.1, min.v.d.1, rng.v.d.1, med.v.d.1,
std.v.d.1, cvr.v.d.1), 6)
col.02 <- round(c(log.v.d.1, max.log.v.d.1, min.log.v.d.1, rng.log.v.d.1,
med.log.v.d.1, std.log.v.d.1, cvr.log.v.d.1), 6)
col.03 <- round(c(v.d.2, max.v.d.2, min.v.d.2, rng.v.d.2, med.v.d.2,
std.v.d.2, cvr.v.d.2), 6)
col.04 <- round(c(log.v.d.2, max.log.v.d.2, min.log.v.d.2, rng.log.v.d.2,
med.log.v.d.2, std.log.v.d.2, cvr.log.v.d.2), 6)
col.05 <- round(c(v.h.1, max.v.h.1, min.v.h.1, rng.v.h.1, med.v.h.1,
std.v.h.1, cvr.v.h.1), 6)
col.06 <- round(c(log.v.h.1, max.log.v.h.1, min.log.v.h.1, rng.log.v.h.1,
med.log.v.h.1, std.log.v.h.1, cvr.log.v.h.1), 6)
col.07 <- round(c(arc.v.h.1, max.arc.v.h.1, min.arc.v.h.1, rng.arc.v.h.1,
med.arc.v.h.1, std.arc.v.h.1, cvr.arc.v.h.1), 6)
col.08 <- round(c(fyt.v.h.1, max.fyt.v.h.1, min.fyt.v.h.1, rng.fyt.v.h.1,
med.fyt.v.h.1, std.fyt.v.h.1, cvr.fyt.v.h.1), 6)
col.09 <- round(c(v.j.1, max.v.j.1, min.v.j.1, rng.v.j.1, med.v.j.1,
std.v.j.1, cvr.v.j.1), 6)
```

```
col.10 <- round(c(v.j.2, max.v.j.2, min.v.j.2, rng.v.j.2, med.v.j.2,
std.v.j.2, cvr.v.j.2), 6)
col.11 <- round(c(v.j.3, max.v.j.3, min.v.j.3, rng.v.j.3, med.v.j.3,
std.v.j.3, cvr.v.j.3), 6)
# Filas para los criterios que dependen de los determinantes de la MIF:
fil.A1 <- c("det(pr.obs)", "log[det(pr.obs)]", "det(pi.hat)",
"log[det(pi.hat)]")
# Filas para los criterios que dependen de la Pi.mx:
fil.B1 <- c("Pi.Max", "Logit(Pi.Max)", "Arc.s(Pi.Max)", "Fr-Tk(Pi.Max)")
# Filas para los criterios generales del modelo ajustado:
fil.C1 <- c("Log.Lik", "Dev.Res.", "Est.AIC")
# Filas comunes para todos los criterios:
fil.2 <- c(
"DF.03.01","DF.03.02","DF.03.03","DF.03.04","DF.03.05",
"DF.03.06","DF.03.07","DF.03.08","DF.03.09","DF.03.10",
"DF.03.11","DF.03.12","DF.03.13","DF.03.14","DF.03.15",
"Maximo", "Minimo", "Rango", "Media", "Stdev.", "Coef.V.")
ref.a <- c(round(d.OM, 6))
ref.b <- M
ref.c <- round(LS[fila.LS, 1], 6)  # lado L
ref.d <- round(LS[fila.LS, 2], 6)  # salto S
ref.h <- matrix(c(col.01, col.02, col.03, col.04, col.05, col.06, col.07,
col.08, col.09, col.10, col.11), nc=11, byrow=F)
# Matriz de resumen de los determinantes:
mat.det <- matrix(ref.h[, 1:4], nc=4, byrow=F)
rownames(mat.det) <- fil.2
colnames(mat.det) <- fil.A1
# Matriz de resumen de Pi.mx:
mat.pim <- matrix(ref.h[, 5:8], nc=4, byrow=F)
rownames(mat.pim) <- fil.2
colnames(mat.pim) <- fil.B1
# Matriz de resumen del modelo ajustado:
mat.mod <- matrix(ref.h[, 9:11], nc=3, byrow=F)
rownames(mat.mod) <- fil.2
colnames(mat.mod) <- fil.C1
# En esta lista se guarda toda la informacion de resumen de la calidad
# de los ajustes.
list.W005.F01 <- list(
"Valor de w% elegido" = w.00,
"Niveles de L:" = LL,
"Niveles de S:" = SS,
"Matriz LS para el w elegido:" = LS,
"Fila estudiada de la matriz LS:" = fila.LS,
"Distancia entre el maximo y el primer centro (d.OM):" = ref.a,
"No. de observaciones por punto de dise\U{f1}o (M):" = ref.b,
```

```
"Valor del lado del factorial (L):" = ref.c,
"Valor del modulo del salto entre centros (S):" = ref.d,
"Matriz-resumen de los determinantes de las MIF" = mat.det,
"Matriz-resumen de las probabilidades maximas" = mat.pim,
"Matriz-resumen de los indicadores del modelo ajustado" = mat.mod,
"Ajustes para DF.03.01:" = list.P.01,
"Ajustes para DF.03.02:" = list.P.02,
"Ajustes para DF.03.03:" = list.P.03,
"Ajustes para DF.03.04:" = list.P.04,
"Ajustes para DF.03.05:" = list.P.05,
"Ajustes para DF.03.06:" = list.P.06,
"Ajustes para DF.03.07:" = list.P.07,
"Ajustes para DF.03.08:" = list.P.08,
"Ajustes para DF.03.09:" = list.P.09,
"Ajustes para DF.03.10:" = list.P.10,
"Ajustes para DF.03.11:" = list.P.11,
"Ajustes para DF.03.12:" = list.P.12,
"Ajustes para DF.03.13:" = list.P.13,
"Ajustes para DF.03.14:" = list.P.14,
"Ajustes para DF.03.15:" = list.P.15)

# ------------------------------------------------------------------------
# MODULO 8.3
#
# Actualizacion de los ficheros-resumen con la informacion de los data
# frames de modelos ajustados.
# ------------------------------------------------------------------------

dump('list.W005.F01', file = 'list.W005.F01.txt', append = F)

# ------------------------------------------------------------------------
# MODULO 9
#
# Graficas de la estrategia secuencial para cada data frame
# ------------------------------------------------------------------------

# Partiendo de la curva de nivel de referencia, se grafica la trayectoria
# de experimentacion que van siguiendo los factoriales en su aproximacion
# al punto maximo de la superficie teorica.
# Defino las escalas de los ejes X1 y X2 para el grafico de persp(): tomo
# un entorno de amplitud arbitraria, "escala", alrededor de cada coorde-
# nada del punto estacionario de ambos ejes.
escala <- 2.5
X1.inf <- -(abs(b.k.2[,"X1.S"])+escala)
X1.sup <- -X1.inf
```

```
X1 <- seq(from=X1.inf, to=X1.sup, length=50)
X2.inf <- -(abs(b.k.2[,"X2.S"])+escala)
X2.sup <- -X2.inf
X2 <- seq(from=X2.inf, to=X2.sup, length=50)
# Queda definida una grilla X1,X2 de valores, para los cuales se calcula-
# ran los valores de la funcion logistica.
# Defino la funcion que me permita calcular el valor de la funcion logis-
# tica para cada uno de los pares X1,X2 de la grilla de valores defini-
# dos anteriormente y de acuerdo al modelo definido por k.2.
f.01 <- function(X1, X2)
{
eta.k.2 <- M7[k.2,"B.00"]+
M7[k.2,"B.01"]*X1+
M7[k.2,"B.02"]*X2+
M7[k.2,"B.11"]*X1*X1+
M7[k.2,"B.22"]*X2*X2+
M7[k.2,"B.12"]*X1*X2
Pi.k.2 <- exp(eta.k.2)/(1+exp(eta.k.2))
round(Pi.k.2, 5)
}
# Con la funcion 'outer()' preparo los puntos para graficarlos:
out.f.01 <- outer(X1, X2, f.01)

# ------------------------------------------------------------------------
# MODULO 9.01
#
# Grafico de los factoriales sucesivos para el DF.03.01
# ------------------------------------------------------------------------

windows()
con.f.01 <- contour(x = X1, y = X2, z = out.f.01
, xlab="X1", ylab="X2",
, cex.lab=0.75,
, levels = c(w.00),
xlim=c(-2.25, 2.25), ylim=c(-2.25, 2.50))
title(main = "Exploracion secuencial mediante factoriales
Conjunto de datos: DF.03.01", cex.main=1, line=2)
title(main = paste("Parametros de la matriz LS: fila", fila.LS),
cex.main=0.85, line=1)
# El origen de coordenadas:
abline(h=0, lty='dashed')
abline(v=0, lty='dashed')
# El maximo:
points(x=M7[k.2, "X1.S"], y=M7[k.2, "X2.S"], lwd=3.0)
text(x=M7[k.2, "X1.S"], y=M7[k.2, "X2.S"]+0.2, "X.Max", cex=0.75,
```

```
adj=0.5)
# Los puntos de disenyo:
points(DF.03.01["C.1", "x1"], DF.03.01["C.1", "x2"], lwd=1.0)
points(DF.03.01["D.1", "x1"], DF.03.01["D.1", "x2"], lwd=1.0)
points(DF.03.01["E.1", "x1"], DF.03.01["E.1", "x2"], lwd=1.0)
points(DF.03.01["F.1", "x1"], DF.03.01["F.1", "x2"], lwd=1.0)
points(DF.03.01["O.1", "x1"], DF.03.01["O.1", "x2"], lwd=2.5)
points(DF.03.01["C.2", "x1"], DF.03.01["C.2", "x2"], lwd=1.0)
points(DF.03.01["D.2", "x1"], DF.03.01["D.2", "x2"], lwd=1.0)
points(DF.03.01["E.2", "x1"], DF.03.01["E.2", "x2"], lwd=1.0)
points(DF.03.01["F.2", "x1"], DF.03.01["F.2", "x2"], lwd=1.0)
points(DF.03.01["O.2", "x1"], DF.03.01["O.2", "x2"], lwd=2.5)
points(DF.03.01["C.3", "x1"], DF.03.01["C.3", "x2"], lwd=1.0)
points(DF.03.01["D.3", "x1"], DF.03.01["D.3", "x2"], lwd=1.0)
points(DF.03.01["E.3", "x1"], DF.03.01["E.3", "x2"], lwd=1.0)
points(DF.03.01["F.3", "x1"], DF.03.01["F.3", "x2"], lwd=1.0)
points(DF.03.01["O.3", "x1"], DF.03.01["O.3", "x2"], lwd=2.5)
# Segmentos que unen los puntos del factorial:
# Primer disenyo:
segments(x0=DF.03.01["C.1", "x1"], y0=DF.03.01["C.1", "x2"],
x1=DF.03.01["D.1", "x1"], y1=DF.03.01["D.1", "x2"], lty='solid')
segments(x0=DF.03.01["D.1", "x1"], y0=DF.03.01["D.1", "x2"],
x1=DF.03.01["E.1", "x1"], y1=DF.03.01["E.1", "x2"], lty='solid')
segments(x0=DF.03.01["E.1", "x1"], y0=DF.03.01["E.1", "x2"],
x1=DF.03.01["F.1", "x1"], y1=DF.03.01["F.1", "x2"], lty='solid')
segments(x0=DF.03.01["F.1", "x1"], y0=DF.03.01["F.1", "x2"],
x1=DF.03.01["C.1", "x1"], y1=DF.03.01["C.1", "x2"], lty='solid')
# Segundo disenyo:
segments(x0=DF.03.01["C.2", "x1"], y0=DF.03.01["C.2", "x2"],
x1=DF.03.01["D.2", "x1"], y1=DF.03.01["D.2", "x2"], lty='solid')
segments(x0=DF.03.01["D.2", "x1"], y0=DF.03.01["D.2", "x2"],
x1=DF.03.01["E.2", "x1"], y1=DF.03.01["E.2", "x2"], lty='solid')
segments(x0=DF.03.01["E.2", "x1"], y0=DF.03.01["E.2", "x2"],
x1=DF.03.01["F.2", "x1"], y1=DF.03.01["F.2", "x2"], lty='solid')
segments(x0=DF.03.01["F.2", "x1"], y0=DF.03.01["F.2", "x2"],
x1=DF.03.01["C.2", "x1"], y1=DF.03.01["C.2", "x2"], lty='solid')
# Tercer disenyo:
segments(x0=DF.03.01["C.3", "x1"], y0=DF.03.01["C.3", "x2"],
x1=DF.03.01["D.3", "x1"], y1=DF.03.01["D.3", "x2"], lty='solid')
segments(x0=DF.03.01["D.3", "x1"], y0=DF.03.01["D.3", "x2"],
x1=DF.03.01["E.3", "x1"], y1=DF.03.01["E.3", "x2"], lty='solid')
segments(x0=DF.03.01["E.3", "x1"], y0=DF.03.01["E.3", "x2"],
x1=DF.03.01["F.3", "x1"], y1=DF.03.01["F.3", "x2"], lty='solid')
segments(x0=DF.03.01["F.3", "x1"], y0=DF.03.01["F.3", "x2"],
x1=DF.03.01["C.3", "x1"], y1=DF.03.01["C.3", "x2"], lty='solid')
```

```
# Flechas que unen los centros:
arrows(DF.03.01["0.1", "x1"], DF.03.01["0.1", "x2"],
DF.03.01["0.2", "x1"], DF.03.01["0.2", "x2"], length = 0.10, lwd = 1.75)
arrows(DF.03.01["0.2", "x1"], DF.03.01["0.2", "x2"],
DF.03.01["0.3", "x1"], DF.03.01["0.3", "x2"], length = 0.10, lwd = 1.75)
# Centros de cada disenyo
text(x=DF.03.01["0.1", "x1"]+0.25, y=DF.03.01["0.1", "x2"], "0.1",
cex=0.5, adj=0.5)
text(x=DF.03.01["0.2", "x1"]+0.25, y=DF.03.01["0.2", "x2"], "0.2",
cex=0.5, adj=0.5)
text(x=DF.03.01["0.3", "x1"]+0.25, y=DF.03.01["0.3", "x2"], "0.3",
cex=0.5, adj=0.5)
# Miscelanea:
text(x=0.25, y=2.50, paste("Angulo del vector c(0.1, 0.2) =",
round(phi5.g.01, 2)), cex=0.65, adj=0)
text(x=0.25, y=2.25, paste("Angulo del vector c(0.2, 0.3) =",
round(phi6.g.01, 2)), cex=0.65, adj=0)
text(x=-2.3, y=2.50, paste("Valor del lado utilizado:   L =",
round(L, 3)), cex=0.65, adj=0)
text(x=-2.3, y=2.25, paste("Valor del salto utilizado:  S =",
round(S, 3)), cex=0.65, adj=0)
text(x=-2.3, y=2.00, paste("No. observaciones por punto: M =", M),
cex=0.65, adj=0)
text(x=-2.3, y=1.75, paste("Valor de w utilizado:", w.00),
cex=0.65, adj=0)
```

Los módulos `9.02` al `9.15` contienen la misma estructura que el anterior, solamente que se adaptan al cuadro de puntos que corresponde y utilizan la fila o columna de la matriz respectiva.

C.4. Serie de programas `MEDCA`

C.4.1. Descripción general

En esta serie de programas se compilan todos los valores promedio para cada uno de los estadisticos de medida de calidad de ajuste, para cada uno de los modelos ajustados con los 1500 puntos "éxito-fracaso" ó 15 puntos de diseño. Hay una serie para cada valor de w. Asimismo, se definen algunos gráficos de utilidad para valorar mejor la calidad del ajuste conseguida. Del mismo modo que en los programas de la serie `CRJER`, aquí solamente mostraremos las líneas de programa correspondientes al tratamiento de un solo cuadro de puntos.

C.4.2. Resumen de entradas y salidas

Inputs o entradas principales

- Ejecución completa del programa correspondiente de la serie `BETAS`. En el caso de las líneas de programa que comentamos para la serie `CRJER`, el programa de la serie `BETAS` correspondiente es el `BETAS-W005-UNI-v3.txt`.
- No es necesario ejecutar todos los programas de la serie `CRJER` correspondientes a un mismo valor de w.

Outputs o salidas principales

- Las mismas que para los programas de la serie `BETAS`.
- Generación de gráficas relacionadas con la superficie teórica.
- Gráfica de las curvas de nivel superficie de respuesta teórica.
- Gráfica de la curva de nivel con el primer centro de experimentación.
- Cálculo de todos los estadísticos del criterio I para todas filas de la matriz **LS**, evaluados en `pr.obs` y en `pi.hat`. Representación del gráfico de "rejilla" para este criterio. Proyecciones de los estadísticos sobre los planos $L = 0$ y $S = 0$.
- Cálculo de todos los estadísticos del criterio II para todas filas de la matriz **LS**. Representación del gráfico de "rejilla" para este criterio. Proyecciones de los estadísticos sobre los planos $L = 0$ y $S = 0$.
- Cálculo de otras medidas de calidad de ajuste: log-verosimilitud, devianza residual y AIC. Representación del gráfico de "rejilla" para este criterio. Proyecciones de los estadísticos sobre los planos $L = 0$ y $S = 0$.

C.4.3. Detalle de entradas y salidas por módulos

PROGRAMAS DE LA SERIE "MEDCA"		
INPUTS	MÓDULOS	OUTPUTS
• Programa correspondiente de la serie BETAS.	1	• Gráficas de la exploración secuencial: superficie teórica, curvas de nivel.
• Todos los programas de la serie CRJER (25 programas).	2	• Resumen tabular de todas las cantidades que intervienen en los diseños para un valor de w.
• Módulo 2.	3, 4, 5 y 6	• Evaluación de los estadísticos del criterio I para la matriz **LS**, para un valor de w. • Rejilla de valores para el criterio I, para un valor de w y proyecciones sobre los planos L=0 y S=0.
• Módulo 2.	7, 8, 9 y 10	• Evaluación de los estadísticos del criterio II para la matriz **LS**, para un valor de w. • Rejilla de valores para el criterio II, para un valor de w y proyecciones sobre los planos L=0 y S=0.
• Módulo 2.	11, 12 y 13	• Evaluación de otros estadísticos de ajuste para la matriz **LS**, para un valor de w. • Rejilla de valores para los otros estadísticos de ajuste, para un valor de w y proyecciones sobre los planos L=0 y S=0.
• Módulos 2 a 13	14	• Representaciones tabulares de todos los estadísticos definidos para la calidad del ajuste.
• Módulo 14.	15	• Representación gráfica detallada de la definición del centro de experimentación y lados del factorial sobre la curva de nivel de la ST cuando se toma w=5%.

Figura C.3: Características principales de los programas de la serie `MEDCA`.

C.4.4. Líneas de programa en R

```
# PROGRAMA:                                          MEDCA-W005-UNI-v2.txt

# FECHA REVISION:                                              03 ene 2006

# GENERALIDADES:
# En este programa se compilan todos los valores promedio para cada uno
# de los estadisticos de medida de calidad de ajuste para cada uno de los
# modelos ajustados con los 1500 puntos.
# Luego se definen algunos graficos de utilidad para valorar mejor la ca-
# lidad del ajuste conseguida.
```

```
# WORKING DIRECTORY:
setwd('C:/Documents and Settings/Arturo/Mis documentos/Tesis/Programas/W005')

# REQUISITOS:
# Se debe ejecutar previamente el programa BETAS-UNI-v2.txt:

source('BETAS-W005-UNI-v2.txt')

# ------------------------------------------------------------------------
# MODULO 1
#
# Generacion de graficas relacionadas con la superficie teorica
# ------------------------------------------------------------------------

# Distintas formas de visualizar la superficie generadora de los datos
# del proceso.
# En los comandos que se ejecutan en este modulo, solamente hace falta
# la informacion contenida en el programa BETAS correspondiente, y no
# es necesario correr todos los CRJER.

# ------------------------------------------------------------------------
# MODULO 1.1
#
# Grafica de la superficie de respuesta teorica
# ------------------------------------------------------------------------

windows()
per.f.01 <- persp(
x = X1,
y = X2,
z = out.f.01,
xlab="X1",
ylab="X2",
zlab="PI(X1,X2)",
theta=30,
phi=30,
cex=0.75,
xlim=1.0*c(X1.inf, X1.sup),
ylim=1.0*c(X2.inf, X2.sup),
zlim=c(0, M7[k.2,"Pi.Max"]),
scale=TRUE,
expand=1,
ticktype='detailed')
title(main = c("Superficie de respuesta teorica",
expression(pi == e^eta * (1+e^eta)^-1)), cex.main=1.25, line=3)
```

```
title(main = expression(pi == over(e^{beta[0]+beta[1]*x[1]+beta[2]*x[2]+
beta[12]*x[1]*x[2]+beta[11]*x[1]^2+beta[22]*x[2]^2},
1+e^{beta[0]+beta[1]*x[1]+beta[2]*x[2]+beta[12]*x[1]*x[2]+
beta[11]*x[1]^2+beta[22]*x[2]^2})), cex.main=0.95, line=1)
expr.1 <- c(
M7[k.2, "B.00"],
M7[k.2, "B.01"],
M7[k.2, "B.02"],
M7[k.2, "B.12"],
M7[k.2, "B.11"],
M7[k.2, "B.22"])
title(sub = paste("(b00, b01, b02, b12, b11, b22) = ", deparse(expr.1)),
cex.sub=0.75, line=1)

# ------------------------------------------------------------------------
# MODULO 1.2
#
# Grafica de las curvas de nivel superficie de respuesta teorica
# ------------------------------------------------------------------------

windows()
con.f.01 <- contour(
x = X1,
y = X2,
z = out.f.01,
xlab="X1",
ylab="X2",
xlim=1.0*c(X1.inf, X1.sup),
ylim=1.0*c(X2.inf, X2.sup),
levels = c(0.01, 0.05, 0.25, 0.50, 0.75, 0.90),
main="Curvas de nivel de la superficie teorica",
cex.main=1)
# Agrego algunas etiquetas y puntos a las curvas de nivel:
points(
x=M7[k.2, "X1.S"],
y=M7[k.2, "X2.S"],
lwd=3.0)
text(
x=M7[k.2, "X1.S"]+0.10,
y=M7[k.2, "X2.S"]+0.20,
"X.Max",
cex=0.70,
adj=0)
segments(
x0=M7[k.2, "X1.S"],
```

```
y0=M7[k.2, "X2.S"]-4,
x1=M7[k.2, "X1.S"],
y1=M7[k.2, "X2.S"]+0.3,
lty='dashed')
segments(
x0=M7[k.2, "X1.S"]-4,
y0=M7[k.2, "X2.S"],
x1=M7[k.2, "X1.S"]+0.3,
y1=M7[k.2, "X2.S"],
lty='dashed')
text(
x=-2.75,
y=2.75,
paste("Valor de x1 en X.Max = ", M7[k.2, "X1.S"]),
cex=0.65,
adj=0)
text(
x=-2.75,
y=2.50,
paste("Valor de x2 en X.Max = ", M7[k.2, "X2.S"]),
cex=0.65,
adj=0)
text(
x=-2.75,
y=2.25,
paste("Valor de Pi en X.Max = ", round(M7[k.2, "Pi.Max"], 5)),
cex=0.65,
adj=0)

# ------------------------------------------------------------------------
# MODULO 1.3
#
# Grafico de la curva de nivel con el primer centro de experimentacion
# ------------------------------------------------------------------------

windows()
con.f.02 <- contour(
x = X1,
y = X2,
z = out.f.01,
xlab="X1",
ylab="X2",
xlim=c(-3, 3),
ylim=c(-3, 3),
levels = c(w.00),
```

```
main="Curva de nivel de la superficie teorica
y primer centro de experimentacion para w = 5%",
cex.main=1)
# El origen de coordenadas:
abline(h=0, lty='solid')
abline(v=0, lty='solid')
points(
x=0,
y=0,
lwd=3.0)
text(
x=0+0.10,
y=0+0.25,
"(0, 0)",
cex=0.75,
adj=0)
# El maximo:
points(
x=M7[k.2, "X1.S"],
y=M7[k.2, "X2.S"],
lwd=3.0)
text(
x=M7[k.2, "X1.S"]+0.15,
y=M7[k.2, "X2.S"]+0.20,
"X.Max",
cex=0.75,
adj=0)
segments(
x0=M7[k.2, "X1.S"],
y0=M7[k.2, "X2.S"]-4,
x1=M7[k.2, "X1.S"],
y1=M7[k.2, "X2.S"]+0.3,
lty='dashed')
segments(
x0=M7[k.2, "X1.S"]-4,
y0=M7[k.2, "X2.S"],
x1=M7[k.2, "X1.S"]+0.3,
y1=M7[k.2, "X2.S"],
lty='dashed')
# La recta que une el origen con el maximo:
segments(
x0=0,
y0=0,
x1=0.1[1]+2,
y1=0.1[2]-2,
```

```
lty='solid')
text(
x=O.1[1]+0.75,
y=O.1[2]-0.50,
"R1",
cex=0.75,
adj=0)
segments(
x0=M7[k.2, "X1.S"],
y0=M7[k.2, "X2.S"],
x1=O.1[1],
y1=O.1[2],
lty='solid',
lwd=2.0)
text(
x=0.5*(M7[k.2, "X1.S"]+O.1[1]+0.50),
y=0.5*(M7[k.2, "X2.S"]+O.1[2]+0.50),
"d.OM",
cex=0.75,
adj=0.5)
# El punto sobre la curva de nivel:
points(
x=O.1[1],
y=O.1[2],
lwd=3.0)
text(
x=O.1[1]+0.20,
y=O.1[2]+0.15,
"O.1",
cex=0.75,
adj=0)
segments(
x0=O.1[1],
y0=O.1[2]-6,
x1=O.1[1],
y1=O.1[2]+0.3,
lty='dashed')
segments(
x0=O.1[1]-6,
y0=O.1[2],
x1=O.1[1]+0.3,
y1=O.1[2],
lty='dashed')
# Miscelanea:
text(
```

```
x=-3.00,
y=3.00,
paste("Valor de x1 en X.Max = ", M7[k.2, "X1.S"]),
cex=0.65,
adj=0)
text(
x=-3.00,
y=2.75,
paste("Valor de x2 en X.Max = ", M7[k.2, "X2.S"]),
cex=0.65,
adj=0)
text(
x=-3.00,
y=2.50,
paste("Valor de Pi en X.Max = ", M7[k.2, "Pi.Max"]),
cex=0.65,
adj=0)
text(
x=0.25,
y=3.00,
paste("Valor de x1 en O.1 =", round(O.1[1], 4)),
cex=0.65,
adj=0)
text(
x=0.25,
y=2.75,
paste("Valor de x2 en O.1 =", round(O.1[2], 4)),
cex=0.65,
adj=0)
text(
x=0.25,
y=2.50,
paste("Valor de Pi en O.1 = ", round((w.00 * M7[k.2, "Pi.Max"]), 5)),
cex=0.65,
adj=0)
text(
x=-3.00,
y=-2.75,
paste("w consdierada  =", w.00),
cex=0.65,
adj=0)
text(
x=-3.00,
y=-3.00,
paste("d.OM calculado =", round(d.OM, 4)),
```

```
cex=0.65,
adj=0)

# ------------------------------------------------------------------------
# MODULO 2
#
# Carga de los ficheros-resumen que contienen los datos utilizados en
# este programa
# ------------------------------------------------------------------------

# Esto es para prescindir de la ejecucion de los programas de la serie
# CRJER uno por uno antes de ejecutar este programa.
# Deben estar disponibles los ficheros-resumen de los 25 casos de la ma-
# triz LS en la carpeta fijada como working directory:
source('list.W005.F01.txt')
source('list.W005.F02.txt')
source('list.W005.F03.txt')
source('list.W005.F04.txt')
source('list.W005.F05.txt')
source('list.W005.F06.txt')
source('list.W005.F07.txt')
source('list.W005.F08.txt')
source('list.W005.F09.txt')
source('list.W005.F10.txt')
source('list.W005.F11.txt')
source('list.W005.F12.txt')
source('list.W005.F13.txt')
source('list.W005.F14.txt')
source('list.W005.F15.txt')
source('list.W005.F16.txt')
source('list.W005.F17.txt')
source('list.W005.F18.txt')
source('list.W005.F19.txt')
source('list.W005.F20.txt')
source('list.W005.F21.txt')
source('list.W005.F22.txt')
source('list.W005.F23.txt')
source('list.W005.F24.txt')
source('list.W005.F25.txt')
# Una vez ejecutados estos comandos, se los ejecuta en la misma linea de
# comandos escribiendo:
# list.W00.FXX
# correspondiendo FXX a la fila que corresponda.
```

```
# ------------------------------------------------------------------------
# MODULO 3
#
# Estadistico det(pr.obs) para todas las filas de la matriz LS
# ------------------------------------------------------------------------

# Este estadistico es el determinante de la MIF, cuya matriz de pesos
# estimada se calcula como W = f(pr.obs).
# En los comandos que se ejecutan en este modulo y en todos los que se
# refieran a los otros estadisticos de medida de calidad de ajuste, se
# requiere tener todos los ficheros tipo 'resumen'. Para ello, previamen-
# te es necesario haber ejecutado todos los programas de la serie CRJER
# pertenecientes a una misma familia de W, como por ejemplo todos los 25
# archivos que contengan CRJER y W005 en su nombre.

# ------------------------------------------------------------------------
# MODULO 3.1
#
# Matriz-resumen de los promedios del estadistico det(pr.obs) evaluados
# en cada una de las filas de la matriz LS
# ------------------------------------------------------------------------

# Matriz de origen de los datos: list.W005.F01[[10]] a list.W005.F25[[10]],
# fila 19, columna 1.
# Primera columna: (barrido de los 5 valores de L, para S = S1)
c1.det.1 <- c(
list.W005.F01[[10]][19,1],
list.W005.F02[[10]][19,1],
list.W005.F03[[10]][19,1],
list.W005.F04[[10]][19,1],
list.W005.F05[[10]][19,1])
# Primera columna: (barrido de los 5 valores de L, para S = S1)
c2.det.1 <- c(
list.W005.F06[[10]][19,1],
list.W005.F07[[10]][19,1],
list.W005.F08[[10]][19,1],
list.W005.F09[[10]][19,1],
list.W005.F10[[10]][19,1])
c3.det.1 <- c(
list.W005.F11[[10]][19,1],
list.W005.F12[[10]][19,1],
list.W005.F13[[10]][19,1],
list.W005.F14[[10]][19,1],
list.W005.F15[[10]][19,1])
c4.det.1 <- c(
```

```
list.W005.F16[[10]][19,1],
list.W005.F17[[10]][19,1],
list.W005.F18[[10]][19,1],
list.W005.F19[[10]][19,1],
list.W005.F20[[10]][19,1])
c5.det.1 <- c(
list.W005.F21[[10]][19,1],
list.W005.F22[[10]][19,1],
list.W005.F23[[10]][19,1],
list.W005.F24[[10]][19,1],
list.W005.F25[[10]][19,1])
# Configuro todo en una matriz, con SS columnas y LL filas:
MM.01 <- cbind(c1.det.1, c2.det.1, c3.det.1, c4.det.1, c5.det.1)
colnames(MM.01) <- c('S1', 'S2', 'S3', 'S4', 'S5')
rownames(MM.01) <- c('L1', 'L2', 'L3', 'L4', 'L5')

# ------------------------------------------------------------------
# MODULO 3.2
#
# Caracterizacion de la matriz MM.01
# ------------------------------------------------------------------

# a. Valor maximo:
max(MM.01)
# c. Indice absoluto del maximo dentro de la matriz:
which.max(MM.01)
# d. Numero de fila correspondiente al maximo:
r.max.MM.01 <- row(as.matrix(MM.01))[which.max(MM.01)]
# e. Numero de columna correspondiente al maximo:
c.max.MM.01 <- col(as.matrix(MM.01))[which.max(MM.01)]
# f. Coordenadas del maximo:
pos.max.MM.01 <- c(r.max.MM.01, c.max.MM.01)

# ------------------------------------------------------------------
# MODULO 3.3
#
# Rejilla de valores para la matriz MM.01
# ------------------------------------------------------------------

windows()
plot(x="", y="", xlab="Niveles de S", ylab="Niveles de L",
xlim=c(min(list.W005.F01[[3]]), max(list.W005.F01[[3]])+0.250),
ylim=c(min(list.W005.F01[[2]]), max(list.W005.F01[[2]])+0.015))
abline(h=list.W005.F01[[2]], lty='dashed')
abline(v=list.W005.F01[[3]], lty='dashed')
```

```
title(main="Valores promedio de los determinantes de la MIF",
cex.main=0.9, line=3)
title(main="para todos los niveles considerados de L y S, w = 5%",
cex.main=0.9, line=2)
title(main="Matriz de pesos estimada:  W = f(pr.obs)",
cex.main=0.85, line=1)
# Puntos de la primera columna de la rejilla:
points(x=list.W005.F01[[3]][1], y=list.W005.F01[[2]][1], lwd=2.0)
points(x=list.W005.F01[[3]][1], y=list.W005.F01[[2]][2], lwd=2.0)
points(x=list.W005.F01[[3]][1], y=list.W005.F01[[2]][3], lwd=2.0)
points(x=list.W005.F01[[3]][1], y=list.W005.F01[[2]][4], lwd=2.0)
points(x=list.W005.F01[[3]][1], y=list.W005.F01[[2]][5], lwd=2.0)
text(x=list.W005.F01[[3]][1]+0.015, y=list.W005.F01[[2]][1]+0.015,
round(MM.01[1,1], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][1]+0.015, y=list.W005.F01[[2]][2]+0.015,
round(MM.01[2,1], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][1]+0.015, y=list.W005.F01[[2]][3]+0.015,
round(MM.01[3,1], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][1]+0.015, y=list.W005.F01[[2]][4]+0.015,
round(MM.01[4,1], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][1]+0.015, y=list.W005.F01[[2]][5]+0.015,
round(MM.01[5,1], 2), cex=0.65, adj=0)
# Puntos de la segunda columna de la rejilla:
points(x=list.W005.F01[[3]][2], y=list.W005.F01[[2]][1], lwd=2.0)
points(x=list.W005.F01[[3]][2], y=list.W005.F01[[2]][2], lwd=2.0)
points(x=list.W005.F01[[3]][2], y=list.W005.F01[[2]][3], lwd=2.0)
points(x=list.W005.F01[[3]][2], y=list.W005.F01[[2]][4], lwd=2.0)
points(x=list.W005.F01[[3]][2], y=list.W005.F01[[2]][5], lwd=2.0)
text(x=list.W005.F01[[3]][2]+0.015, y=list.W005.F01[[2]][1]+0.015,
round(MM.01[1,2], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][2]+0.015, y=list.W005.F01[[2]][2]+0.015,
round(MM.01[2,2], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][2]+0.015, y=list.W005.F01[[2]][3]+0.015,
round(MM.01[3,2], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][2]+0.015, y=list.W005.F01[[2]][4]+0.015,
round(MM.01[4,2], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][2]+0.015, y=list.W005.F01[[2]][5]+0.015,
round(MM.01[5,2], 2), cex=0.65, adj=0)
# Puntos de la tercera columna de la rejilla:
points(x=list.W005.F01[[3]][3], y=list.W005.F01[[2]][1], lwd=2.0)
points(x=list.W005.F01[[3]][3], y=list.W005.F01[[2]][2], lwd=2.0)
points(x=list.W005.F01[[3]][3], y=list.W005.F01[[2]][3], lwd=2.0)
points(x=list.W005.F01[[3]][3], y=list.W005.F01[[2]][4], lwd=2.0)
points(x=list.W005.F01[[3]][3], y=list.W005.F01[[2]][5], lwd=2.0)
text(x=list.W005.F01[[3]][3]+0.015, y=list.W005.F01[[2]][1]+0.015,
```

```
round(MM.01[1,3], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][3]+0.015, y=list.W005.F01[[2]][2]+0.015,
round(MM.01[2,3], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][3]+0.015, y=list.W005.F01[[2]][3]+0.015,
round(MM.01[3,3], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][3]+0.015, y=list.W005.F01[[2]][4]+0.015,
round(MM.01[4,3], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][3]+0.015, y=list.W005.F01[[2]][5]+0.015,
round(MM.01[5,3], 2), cex=0.65, adj=0)
# Puntos de la cuarta columna de la rejilla:
points(x=list.W005.F01[[3]][4], y=list.W005.F01[[2]][1], lwd=2.0)
points(x=list.W005.F01[[3]][4], y=list.W005.F01[[2]][2], lwd=2.0)
points(x=list.W005.F01[[3]][4], y=list.W005.F01[[2]][3], lwd=2.0)
points(x=list.W005.F01[[3]][4], y=list.W005.F01[[2]][4], lwd=2.0)
points(x=list.W005.F01[[3]][4], y=list.W005.F01[[2]][5], lwd=2.0)
text(x=list.W005.F01[[3]][4]+0.015, y=list.W005.F01[[2]][1]+0.015,
round(MM.01[1,4], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][4]+0.015, y=list.W005.F01[[2]][2]+0.015,
round(MM.01[2,4], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][4]+0.015, y=list.W005.F01[[2]][3]+0.015,
round(MM.01[3,4], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][4]+0.015, y=list.W005.F01[[2]][4]+0.015,
round(MM.01[4,4], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][4]+0.015, y=list.W005.F01[[2]][5]+0.015,
round(MM.01[5,4], 2), cex=0.65, adj=0)
# Puntos de la quinta columna de la rejilla:
points(x=list.W005.F01[[3]][5], y=list.W005.F01[[2]][1], lwd=2.0)
points(x=list.W005.F01[[3]][5], y=list.W005.F01[[2]][2], lwd=2.0)
points(x=list.W005.F01[[3]][5], y=list.W005.F01[[2]][3], lwd=2.0)
points(x=list.W005.F01[[3]][5], y=list.W005.F01[[2]][4], lwd=2.0)
points(x=list.W005.F01[[3]][5], y=list.W005.F01[[2]][5], lwd=2.0)
text(x=list.W005.F01[[3]][5]+0.015, y=list.W005.F01[[2]][1]+0.015,
round(MM.01[1,5], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][5]+0.015, y=list.W005.F01[[2]][2]+0.015,
round(MM.01[2,5], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][5]+0.015, y=list.W005.F01[[2]][3]+0.015,
round(MM.01[3,5], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][5]+0.015, y=list.W005.F01[[2]][4]+0.015,
round(MM.01[4,5], 2), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][5]+0.015, y=list.W005.F01[[2]][5]+0.015,
round(MM.01[5,5], 2), cex=0.65, adj=0)
# Etiqueta que identifica el maximo:
points(x=list.W005.F01[[3]][c.max.MM.01],
y=list.W005.F01[[2]][r.max.MM.01], lwd=3, pch=21, cex=2.5)
text(x=list.W005.F01[[3]][c.max.MM.01]+0.015,
```

```
y=list.W005.F01[[2]][r.max.MM.01]-0.015, "Maximo", lwd=3, cex=0.75, adj=0)

# ------------------------------------------------------------------------
# MODULO 3.4
#
# Traza de los puntos en el plano (Pi, S)
# ------------------------------------------------------------------------

# Es necesario construir una tabla cuyas columnas sean L, S y
# det.MIF.pr.obs:
col.MM.01 <- as.vector(c(MM.01[,1], MM.01[,2], MM.01[,3], MM.01[,4],
MM.01[,5]))
LSM.01 <- cbind(list.W005.F01[[4]], col.MM.01)
windows()
plot(x=LSM.01[,2], y=LSM.01[,3],
xlab="Niveles de S", ylab="det.MIF(pr.obs)",
xlim=c(min(LSM.01[,2]), max(LSM.01[,2])+0.25),
ylim=c(min(LSM.01[,3]), max(LSM.01[,3])),
lwd=2, cex=1.25)
abline(v=list.W005.F01[[3]], lty='dashed')
title(main="Valores promedio de los determinantes de la MIF",
cex.main=0.9, line=3)
title(main="Matriz de pesos estimada:  W = f(pr.obs)",
cex.main=0.9, line=2)
title(main="Proyeccion sobre el plano L = 0 para w = 5%",
cex.main=0.85, line=1)
points(x=list.W005.F01[[3]][c.max.MM.01], y=max(LSM.01[,3]), lwd=3, pch=21,
cex=2.5)
text(x=list.W005.F01[[3]][c.max.MM.01]+0.05,
y=max(LSM.01[,3]), round(max(MM.01),2), lwd=3, cex=0.7, adj=0)

# ------------------------------------------------------------------------
# MODULO 3.5
#
# Traza de los puntos en el plano (Pi, L)
# ------------------------------------------------------------------------

windows()
plot(x=LSM.01[,1], y=LSM.01[,3],
xlab="Niveles de L", ylab="det.MIF(pr.obs)",
xlim=c(min(LSM.01[,1]), max(LSM.01[,1])+0.060),
ylim=c(min(LSM.01[,3]), max(LSM.01[,3])),
lwd=2, cex=1.25)
abline(v=list.W005.F01[[2]], lty='dashed')
title(main="Valores promedio de los determinantes de la MIF",
```

```
cex.main=0.9, line=3)
title(main="Matriz de pesos estimada:  W = f(pr.obs)",
cex.main=0.9, line=2)
title(main="Proyeccion sobre el plano S = 0 para w = 5%",
cex.main=0.85, line=1)
points(x=list.W005.F01[[2]][r.max.MM.01], y=max(LSM.01[,3]), lwd=3, pch=21,
cex=2.5)
text(x=list.W005.F01[[2]][r.max.MM.01]+0.01,
y=max(LSM.01[,3]), round(max(MM.01),2), lwd=3, cex=0.7, adj=0)
```

Los módulos 4 al 6 tienen la misma estructura que los módulos 3.XX, solamente que se refieren a los respectivos estadísticos de medida de calidad de ajuste para el criterio I.

```
# ------------------------------------------------------------------------
# MODULO 7
#
# Estadistico Pi.max para todas las filas de la matriz LS
# ------------------------------------------------------------------------

# Este estadistico es el valor de la probabilidad Pi.Max.

# ------------------------------------------------------------------------
# MODULO 7.1
#
# Matriz-resumen de los promedios del estadistico Pi.max evaluados en
# cada una de las filas de la matriz LS
# ------------------------------------------------------------------------

# Matriz de origen de los datos: list.W005.F01[[11]] a list.W005.F25[[11]],
# fila 19, columna 1.
 Primera columna: (barrido de los 5 valores de L, para S = S1)
c1.pim.1 <- c(
list.W005.F01[[11]][19,1],
list.W005.F02[[11]][19,1],
list.W005.F03[[11]][19,1],
list.W005.F04[[11]][19,1],
list.W005.F05[[11]][19,1])
# Primera columna: (barrido de los 5 valores de L, para S = S1)
c2.pim.1 <- c(
list.W005.F06[[11]][19,1],
list.W005.F07[[11]][19,1],
list.W005.F08[[11]][19,1],
list.W005.F09[[11]][19,1],
```

```
list.W005.F10[[11]][19,1])
c3.pim.1 <- c(
list.W005.F11[[11]][19,1],
list.W005.F12[[11]][19,1],
list.W005.F13[[11]][19,1],
list.W005.F14[[11]][19,1],
list.W005.F15[[11]][19,1])
c4.pim.1 <- c(
list.W005.F16[[11]][19,1],
list.W005.F17[[11]][19,1],
list.W005.F18[[11]][19,1],
list.W005.F19[[11]][19,1],
list.W005.F20[[11]][19,1])
c5.pim.1 <- c(
list.W005.F21[[11]][19,1],
list.W005.F22[[11]][19,1],
list.W005.F23[[11]][19,1],
list.W005.F24[[11]][19,1],
list.W005.F25[[11]][19,1])
# Configuro todo en una matriz, con SS columnas y LL filas:
MM.05 <- cbind(c1.pim.1, c2.pim.1, c3.pim.1, c4.pim.1, c5.pim.1)
colnames(MM.05) <- c('S1', 'S2', 'S3', 'S4', 'S5')
rownames(MM.05) <- c('L1', 'L2', 'L3', 'L4', 'L5')

# ------------------------------------------------------------------------
# MODULO 7.2
#
# Caracterizacion de la matriz MM.05
# ------------------------------------------------------------------------

# a. Valor maximo:
max(MM.05)
# c. Indice absoluto del maximo dentro de la matriz:
which.max(MM.05)
# d. Numero de fila correspondiente al maximo:
r.max.MM.05 <- row(as.matrix(MM.05))[which.max(MM.05)]
# e. Numero de columna correspondiente al maximo:
c.max.MM.05 <- col(as.matrix(MM.05))[which.max(MM.05)]
# f. Coordenadas del maximo:
pos.max.MM.05 <- c(r.max.MM.05, c.max.MM.05)
```

```
# ------------------------------------------------------------------------
# MODULO 7.3
#
# Rejilla de valores para la matriz MM.05
# ------------------------------------------------------------------------

windows()
plot(x="", y="", xlab="Niveles de S", ylab="Niveles de L",
xlim=c(min(list.W005.F01[[3]]), max(list.W005.F01[[3]])+0.250),
ylim=c(min(list.W005.F01[[2]]), max(list.W005.F01[[2]])+0.015))
abline(h=list.W005.F01[[2]], lty='dashed')
abline(v=list.W005.F01[[3]], lty='dashed')
title(main="Valores promedio de las probabilidades Pi.Max",
cex.main=0.9, line=2)
title(main="para todos los niveles considerados de L y S, w = 5%",
cex.main=0.9, line=1)
# Puntos de la primera columna de la rejilla:
points(x=list.W005.F01[[3]][1], y=list.W005.F01[[2]][1], lwd=2.0)
points(x=list.W005.F01[[3]][1], y=list.W005.F01[[2]][2], lwd=2.0)
points(x=list.W005.F01[[3]][1], y=list.W005.F01[[2]][3], lwd=2.0)
points(x=list.W005.F01[[3]][1], y=list.W005.F01[[2]][4], lwd=2.0)
points(x=list.W005.F01[[3]][1], y=list.W005.F01[[2]][5], lwd=2.0)
text(x=list.W005.F01[[3]][1]+0.015, y=list.W005.F01[[2]][1]+0.015,
round(MM.05[1,1], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][1]+0.015, y=list.W005.F01[[2]][2]+0.015,
round(MM.05[2,1], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][1]+0.015, y=list.W005.F01[[2]][3]+0.015,
round(MM.05[3,1], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][1]+0.015, y=list.W005.F01[[2]][4]+0.015,
round(MM.05[4,1], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][1]+0.015, y=list.W005.F01[[2]][5]+0.015,
round(MM.05[5,1], 6), cex=0.65, adj=0)
# Puntos de la segunda columna de la rejilla:
points(x=list.W005.F01[[3]][2], y=list.W005.F01[[2]][1], lwd=2.0)
points(x=list.W005.F01[[3]][2], y=list.W005.F01[[2]][2], lwd=2.0)
points(x=list.W005.F01[[3]][2], y=list.W005.F01[[2]][3], lwd=2.0)
points(x=list.W005.F01[[3]][2], y=list.W005.F01[[2]][4], lwd=2.0)
points(x=list.W005.F01[[3]][2], y=list.W005.F01[[2]][5], lwd=2.0)
text(x=list.W005.F01[[3]][2]+0.015, y=list.W005.F01[[2]][1]+0.015,
round(MM.05[1,2], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][2]+0.015, y=list.W005.F01[[2]][2]+0.015,
round(MM.05[2,2], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][2]+0.015, y=list.W005.F01[[2]][3]+0.015,
round(MM.05[3,2], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][2]+0.015, y=list.W005.F01[[2]][4]+0.015,
```

```
round(MM.05[4,2], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][2]+0.015, y=list.W005.F01[[2]][5]+0.015,
round(MM.05[5,2], 6), cex=0.65, adj=0)
# Puntos de la tercera columna de la rejilla:
points(x=list.W005.F01[[3]][3], y=list.W005.F01[[2]][1], lwd=2.0)
points(x=list.W005.F01[[3]][3], y=list.W005.F01[[2]][2], lwd=2.0)
points(x=list.W005.F01[[3]][3], y=list.W005.F01[[2]][3], lwd=2.0)
points(x=list.W005.F01[[3]][3], y=list.W005.F01[[2]][4], lwd=2.0)
points(x=list.W005.F01[[3]][3], y=list.W005.F01[[2]][5], lwd=2.0)
text(x=list.W005.F01[[3]][3]+0.015, y=list.W005.F01[[2]][1]+0.015,
round(MM.05[1,3], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][3]+0.015, y=list.W005.F01[[2]][2]+0.015,
round(MM.05[2,3], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][3]+0.015, y=list.W005.F01[[2]][3]+0.015,
round(MM.05[3,3], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][3]+0.015, y=list.W005.F01[[2]][4]+0.015,
round(MM.05[4,3], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][3]+0.015, y=list.W005.F01[[2]][5]+0.015,
round(MM.05[5,3], 6), cex=0.65, adj=0)
# Puntos de la cuarta columna de la rejilla:
points(x=list.W005.F01[[3]][4], y=list.W005.F01[[2]][1], lwd=2.0)
points(x=list.W005.F01[[3]][4], y=list.W005.F01[[2]][2], lwd=2.0)
points(x=list.W005.F01[[3]][4], y=list.W005.F01[[2]][3], lwd=2.0)
points(x=list.W005.F01[[3]][4], y=list.W005.F01[[2]][4], lwd=2.0)
points(x=list.W005.F01[[3]][4], y=list.W005.F01[[2]][5], lwd=2.0)
text(x=list.W005.F01[[3]][4]+0.015, y=list.W005.F01[[2]][1]+0.015,
round(MM.05[1,4], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][4]+0.015, y=list.W005.F01[[2]][2]+0.015,
round(MM.05[2,4], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][4]+0.015, y=list.W005.F01[[2]][3]+0.015,
round(MM.05[3,4], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][4]+0.015, y=list.W005.F01[[2]][4]+0.015,
round(MM.05[4,4], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][4]+0.015, y=list.W005.F01[[2]][5]+0.015,
round(MM.05[5,4], 6), cex=0.65, adj=0)
# Puntos de la quinta columna de la rejilla:
points(x=list.W005.F01[[3]][5], y=list.W005.F01[[2]][1], lwd=2.0)
points(x=list.W005.F01[[3]][5], y=list.W005.F01[[2]][2], lwd=2.0)
points(x=list.W005.F01[[3]][5], y=list.W005.F01[[2]][3], lwd=2.0)
points(x=list.W005.F01[[3]][5], y=list.W005.F01[[2]][4], lwd=2.0)
points(x=list.W005.F01[[3]][5], y=list.W005.F01[[2]][5], lwd=2.0)
text(x=list.W005.F01[[3]][5]+0.015, y=list.W005.F01[[2]][1]+0.015,
round(MM.05[1,5], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][5]+0.015, y=list.W005.F01[[2]][2]+0.015,
round(MM.05[2,5], 6), cex=0.65, adj=0)
```

```
text(x=list.W005.F01[[3]][5]+0.015, y=list.W005.F01[[2]][3]+0.015,
round(MM.05[3,5], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][5]+0.015, y=list.W005.F01[[2]][4]+0.015,
round(MM.05[4,5], 6), cex=0.65, adj=0)
text(x=list.W005.F01[[3]][5]+0.015, y=list.W005.F01[[2]][5]+0.015,
round(MM.05[5,5], 6), cex=0.65, adj=0)
# Etiqueta que identifica el maximo:
points(x=list.W005.F01[[3]][c.max.MM.05],
y=list.W005.F01[[2]][r.max.MM.05], lwd=3, pch=21, cex=2.5)
text(x=list.W005.F01[[3]][c.max.MM.05]+0.015,
y=list.W005.F01[[2]][r.max.MM.05]-0.015, "Maximo", lwd=3, cex=0.75, adj=0)

# ------------------------------------------------------------------------
# MODULO 7.4
#
# Traza de los puntos en el plano (Pi, S)
# ------------------------------------------------------------------------

# Es necesario construir una tabla cuyas columnas sean L, S y
# det.MIF.Pi.max:
col.MM.05 <- as.vector(c(MM.05[,1], MM.05[,2], MM.05[,3], MM.05[,4],
MM.05[,5]))
LSM.05 <- cbind(list.W005.F01[[4]], col.MM.05)
windows()
plot(x=LSM.05[,2], y=LSM.05[,3],
xlab="Niveles de S", ylab="Pi.Max",
xlim=c(min(LSM.05[,2]), max(LSM.05[,2])+0.25),
ylim=c(min(LSM.05[,3]), max(LSM.05[,3])),
lwd=2, cex=1.25)
abline(v=list.W005.F01[[3]], lty='dashed')
title(main="Valores promedio de las probabilidades Pi.Max",
cex.main=0.9, line=2)
title(main="Proyeccion sobre el plano L = 0 para w = 5%",
cex.main=0.85, line=1)
points(x=list.W005.F01[[3]][c.max.MM.05], y=max(LSM.05[,3]), lwd=3, pch=21,
cex=2.5)
text(x=list.W005.F01[[3]][c.max.MM.05]+0.05,
y=max(LSM.05[,3]), round(max(MM.05),6), lwd=3, cex=0.7, adj=0)

# ------------------------------------------------------------------------
# MODULO 7.5
#
# Traza de los puntos en el plano (Pi, L)
# ------------------------------------------------------------------------
```

```
windows()
plot(x=LSM.05[,1], y=LSM.05[,3],
xlab="Niveles de L", ylab="Pi.max",
xlim=c(min(LSM.05[,1]), max(LSM.05[,1])+0.060),
ylim=c(min(LSM.05[,3]), max(LSM.05[,3])),
lwd=2, cex=1.25)
abline(v=list.W005.F01[[2]], lty='dashed')
title(main="Valores promedio de las probabilidades Pi.Max",
cex.main=0.9, line=2)
title(main="Proyeccion sobre el plano S = 0 para w = 5%",
cex.main=0.85, line=1)
points(x=list.W005.F01[[2]][r.max.MM.05], y=max(LSM.05[,3]), lwd=3, pch=21,
cex=2.5)
text(x=list.W005.F01[[2]][r.max.MM.05]+0.01,
y=max(LSM.05[,3]), round(max(MM.05),6), lwd=3, cex=0.7, adj=0)
```

Los módulos 8 al 10 tienen la misma estructura que los módulos 3.XX, solamente que se refieren a los respectivos estadísticos de medida de calidad de ajuste para el criterio II. Para los módulos 11 al 13, los mismos siguen la misma marcha de cálculos anterior pero para medidas complementarias de calidad de ajuste: log-verosimilitud, devianza residual y AIC.

```
# ------------------------------------------------------------------------
# MODULO 14
#
# Resumen de algunos estadisticos
# ------------------------------------------------------------------------

setwd('C:/Documents and Settings/Arturo/Mis documentos/Tesis/Programas')
list.max.W005 <- c(
max(MM.01), max(MM.02), max(MM.03), max(MM.04), max(MM.05),
max(MM.06), max(MM.07), max(MM.08), max(MM.09), max(MM.10), max(MM.11))
dump('list.max.W005', file = 'list.max.txt', append = T)


# ------------------------------------------------------------------------
# MODULO 15
#
# Un grafico ad-hoc: el primer disenyo sobre la curva de w = 5%
# ------------------------------------------------------------------------

source('DF.03.01.txt')
windows()
con.f.02 <- contour(
```

```
x = X1,
y = X2,
z = out.f.01,
xlab="X1",
ylab="X2",
xlim=c(1.70, -0.1),
ylim=c(-1.70, 0.1),
levels = c(w.00),
main="Primer centro y factorial para w = 5%",
cex.main=1,
lwd=2.0)
# El origen de coordenadas:
abline(h=0, lty='solid')
abline(v=0, lty='solid')
points(
x=0,
y=0,
lwd=3.0)
text(
x=0.15,
y=0.10,
"(0,0)",
cex=0.75,
adj=0)
# El maximo:
points(
x=M7[k.2, "X1.S"],
y=M7[k.2, "X2.S"],
lwd=3.0)
text(
x=M7[k.2, "X1.S"]+0.20,
y=M7[k.2, "X2.S"]+0.10,
"X.Max",
cex=0.75,
adj=0)
segments(
x0=M7[k.2, "X1.S"],
y0=M7[k.2, "X2.S"]-4,
x1=M7[k.2, "X1.S"],
y1=M7[k.2, "X2.S"]+0.075,
lty='dashed')
segments(
x0=M7[k.2, "X1.S"]+4,
y0=M7[k.2, "X2.S"],
x1=M7[k.2, "X1.S"]-0.075,
```

```
y1=M7[k.2, "X2.S"],
lty='dashed')
# La recta que une el origen con el maximo:
segments(
x0=0,
y0=0,
x1=0.1[1]+2,
y1=0.1[2]-2,
lty='solid')
text(
x=0.1[1]-0.35,
y=0.1[2]+0.55,
"R1",
cex=0.75,
adj=0)
# El punto sobre la curva de nivel:
points(
x=0.1[1],
y=0.1[2],
lwd=3.0)
segments(
x0=0.1[1],
y0=0.1[2]-6,
x1=0.1[1],
y1=0.1[2]+0.075,
lty='dashed')
segments(
x0=0.1[1]+6,
y0=0.1[2],
x1=0.1[1]-0.075,
y1=0.1[2],
lty='dashed')
# Puntos del primer factorial:
points(
x=DF.03.01[[1]][1,1],
y=DF.03.01[[1]][1,2],
lwd=3.0)
points(
x=DF.03.01[[1]][2,1],
y=DF.03.01[[1]][2,2],
lwd=3.0)
points(
x=DF.03.01[[1]][3,1],
y=DF.03.01[[1]][3,2],
lwd=3.0)
```

```
points(
x=DF.03.01[[1]][4,1],
y=DF.03.01[[1]][4,2],
lwd=3.0)
segments(
x0=DF.03.01[[1]][1,1],
y0=DF.03.01[[1]][1,2],
x1=DF.03.01[[1]][2,1],
y1=DF.03.01[[1]][2,2],
lwd=1.0, lty='solid')
segments(
x0=DF.03.01[[1]][2,1],
y0=DF.03.01[[1]][2,2],
x1=DF.03.01[[1]][3,1],
y1=DF.03.01[[1]][3,2],
lwd=1.0, lty='solid')
segments(
x0=DF.03.01[[1]][3,1],
y0=DF.03.01[[1]][3,2],
x1=DF.03.01[[1]][4,1],
y1=DF.03.01[[1]][4,2],
lwd=1.0, lty='solid')
segments(
x0=DF.03.01[[1]][4,1],
y0=DF.03.01[[1]][4,2],
x1=DF.03.01[[1]][1,1],
y1=DF.03.01[[1]][1,2],
lwd=1.0, lty='solid')
text(
x=O.1[1],
y=O.1[2]+0.20,
"L", cex=0.80)
text(
x=O.1[1]-0.20,
y=O.1[2],
"L", cex=0.80)
```

C.5. Programa GRAPH-UNI

C.5.1. Descripción general

En este programa se compilan todos los valores máximos de los estadísticos calculados para todos los niveles considerados de w, y se hacen representaciones gráficas para visualizar sus comportamientos. A través del mismo, y no solamente con sus resultados analíticos sino también con los gráficos, es posible hacer la "compilación" y evaluación de todos casos estudiados: 3 variables de estudio (w, L y S), a 5 niveles

cada una, y seleccionar qué niveles de las mismas son los que hacen máximos los dos criterios de evaluación que hemos propuesto: el de cantidad de información (criterio I) y el de proximidad al máximo (criterio II).

C.5.2. Resumen de entradas y salidas

Inputs o entradas principales

Los programas de la serie `CRJER` generan listas automáticamente con los valores máximos de ambos estadísticos para cada uno de los casos estudiados. Al comenzar la ejecución del programa `GRAPH-UNI` todas esas listas se resumen de forma automática en una, la `list.max.txt`, que es el input principal de este programa.

Outputs o salidas principales

- Representación gráfica para los valores promedio máximos de los estadísticos del criterio I, para todos los niveles de las 3 variables de estudio.
- Representación gráfica para los valores promedio máximos de los estadísticos del criterio II, para todos los niveles de las 3 variables de estudio.

C.5.3. Detalle de entradas y salidas por módulos

PROGRAMAS DE LA SERIE "GRAPH-UNI"		
INPUTS	MÓDULOS	OUTPUTS
• Lista-resumen `list.max.txt` para todos las matrices **LS**, contando todos los niveles de w.	1, 2, 3 y 4	• Gráficas para los estadísticos del criterio I, para todos los niveles de las 3 variables de estudio.
• Lista-resumen `list.max.txt` para todos las matrices **LS**, contando todos los niveles de w.	5, 6, 7 y 8	• Gráficas para los estadísticos del criterio II, para todos los niveles de las 3 variables de estudio.

Figura C.4: Características principales del programa `GRAPH-UNI`.

C.5.4. Líneas de programa en R

```
# PROGRAMA:                                              GRAPH-UNI-v1.txt
# FECHA REVISION:                                             14 ene 2006

# GENERALIDADES:
# En este programa se compilan todos los valores maximos de los estadis-
# ticos calculados para los niveles considerados de w, y se hacen repre-
# sentaciones graficas para visualizar sus comportamientos.
```

```
# WORKING DIRECTORY:
setwd('C:/Documents and Settings/Arturo/Mis documentos/Tesis/Programas')
source('list.max.txt')

# ------------------------------------------------------------------------
# MODULO 1
#
# Grafico para los determinantes de la MIF, W = f(pr.obs)
# ------------------------------------------------------------------------

mat.01 <- matrix(c(
0.05, 0.10, 0.15, 0.20, 0.25,
list.max.W005[1], list.max.W010[1], list.max.W015[1], list.max.W020[1],
list.max.W025[1]), nc=2, byrow=F)
colnames(mat.01) <- c("w%", "det(pr.obs)")
windows()
plot(
x=c(mat.01[,1]),
y=c(mat.01[,2]),
xlab= "Niveles de w",
xlim=c(0.05, 0.25),
ylab="det.MIF(pr.obs)",
cex = 1.5, lwd = 2,
type="h")
points(x=mat.01[1,1], y=mat.01[1,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.01[2,1], y=mat.01[2,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.01[3,1], y=mat.01[3,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.01[4,1], y=mat.01[4,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.01[5,1], y=mat.01[5,2], pch=21, lwd=2, cex=1.25, bg='white')
# El valor de w% que corresponde al maximo del estadistico:
ff.01 <- mat.01[row(mat.01)[which.max(mat.01)],1]
# Un punto gordo alrededor del maximo:
points(x=ff.01, y=max(mat.01), pch=21, lwd=3, cex=2.5)
# Una etiqueta con el valor del maximo:
text(x=ff.01+0.008, y=max(mat.01), round(max(mat.01), 0), cex=0.65, adj=0)
# Titulos:
title(main="Valores promedio de los determinantes de la MIF",
cex.main=0.9, line=3)
title(main="para todos los niveles considerados de w",
cex.main=0.9, line=2)
title(main="Matriz de pesos estimada:  W = f(pr.obs)",
cex.main=0.85, line=1)
```

```
# ------------------------------------------------------------------------
# MODULO 2
#
# Grafico para los logaritmos de los determinantes, W = f(pr.obs)
# ------------------------------------------------------------------------

mat.02 <- matrix(c(
0.05, 0.10, 0.15, 0.20, 0.25,
list.max.W005[2], list.max.W010[2], list.max.W015[2], list.max.W020[2],
list.max.W025[2]), nc=2, byrow=F)
colnames(mat.02) <- c("w%", "ln[det(pr.obs)]")
windows()
plot(
x=c(mat.02[,1]),
y=c(mat.02[,2]),
xlab= "Niveles de w",
xlim=c(0.05, 0.25),
ylab="ln[det.MIF(pr.obs)]",
cex = 1.5, lwd = 2,
type="h")
points(x=mat.02[1,1], y=mat.02[1,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.02[2,1], y=mat.02[2,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.02[3,1], y=mat.02[3,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.02[4,1], y=mat.02[4,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.02[5,1], y=mat.02[5,2], pch=21, lwd=2, cex=1.25, bg='white')
# El valor de w% que corresponde al maximo del estadistico:
ff.02 <- mat.02[row(mat.02)[which.max(mat.02)],1]
# Un punto gordo alrededor del maximo:
points(x=ff.02, y=max(mat.02), pch=21, lwd=3, cex=2.5)
# Una etiqueta con el valor del maximo:
text(x=ff.02+0.008, y=max(mat.02), round(max(mat.02), 5), cex=0.65, adj=0)
# Titulos:
title(main="Valores promedio de los logaritmos de los determinantes de la
MIF", cex.main=0.9, line=3)
title(main="para todos los niveles considerados de w",
cex.main=0.9, line=2)
title(main="Matriz de pesos estimada:  W = f(pr.obs)",
cex.main=0.85, line=1)

# ------------------------------------------------------------------------
# MODULO 3
#
# Grafico para los determinantes de la MIF, W = f(Pi.hat)
# ------------------------------------------------------------------------
```

```
mat.03 <- matrix(c(
0.05, 0.10, 0.15, 0.20, 0.25,
list.max.W005[3], list.max.W010[3], list.max.W015[3], list.max.W020[3],
list.max.W025[3]), nc=2, byrow=F)
colnames(mat.03) <- c("w%", "det(Pi.hat)")
windows()
plot(
x=c(mat.03[,1]),
y=c(mat.03[,2]),
xlab= "Niveles de w",
xlim=c(0.05, 0.25),
ylab="det.MIF(Pi.hat)",
cex = 1.5, lwd = 2,
type="h")
points(x=mat.03[1,1], y=mat.03[1,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.03[2,1], y=mat.03[2,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.03[3,1], y=mat.03[3,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.03[4,1], y=mat.03[4,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.03[5,1], y=mat.03[5,2], pch=21, lwd=2, cex=1.25, bg='white')
# El valor de w% que corresponde al maximo del estadistico:
ff.03 <- mat.03[row(mat.03)[which.max(mat.03)],1]
# Un punto gordo alrededor del maximo:
points(x=ff.03, y=max(mat.03), pch=21, lwd=3, cex=2.5)
# Una etiqueta con el valor del maximo:
text(x=ff.03+0.008, y=max(mat.03), round(max(mat.03), 0), cex=0.65, adj=0)
# Titulos:
title(main="Valores promedio de los determinantes de la MIF",
cex.main=0.9, line=3)
title(main="para todos los niveles considerados de w",
cex.main=0.9, line=2)
title(main="Matriz de pesos estimada:  W = f(Pi.hat)",
cex.main=0.85, line=1)

# ------------------------------------------------------------------
# MODULO 4
#
# Grafico para los logaritmos de los determinantes, W = f(Pi.hat)
# ------------------------------------------------------------------

mat.04 <- matrix(c(
0.05, 0.10, 0.15, 0.20, 0.25,
list.max.W005[4], list.max.W010[4], list.max.W015[4], list.max.W020[4],
list.max.W025[4]), nc=2, byrow=F)
colnames(mat.04) <- c("w%", "ln[det(Pi.hat)]")
windows()
```

```
plot(
x=c(mat.04[,1]),
y=c(mat.04[,2]),
xlab= "Niveles de w",
xlim=c(0.05, 0.25),
ylab="ln[det.MIF(Pi.hat)]",
cex = 1.5, lwd = 2,
type="h")
points(x=mat.04[1,1], y=mat.04[1,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.04[2,1], y=mat.04[2,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.04[3,1], y=mat.04[3,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.04[4,1], y=mat.04[4,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.04[5,1], y=mat.04[5,2], pch=21, lwd=2, cex=1.25, bg='white')
# El valor de w% que corresponde al maximo del estadistico:
ff.04 <- mat.04[row(mat.04)[which.max(mat.04)],1]
# Un punto gordo alrededor del maximo:
points(x=ff.04, y=max(mat.04), pch=21, lwd=3, cex=2.5)
# Una etiqueta con el valor del maximo:
text(x=ff.04+0.008, y=max(mat.04), round(max(mat.04), 5), cex=0.65, adj=0)
# Titulos:
title(main="Valores promedio de los logaritmos de los determinantes de la
MIF", cex.main=0.9, line=3)
title(main="para todos los niveles considerados de w",
cex.main=0.9, line=2)
title(main="Matriz de pesos estimada:  W = f(Pi.hat)",
cex.main=0.85, line=1)

# ------------------------------------------------------------------------
# MODULO 5
#
# Grafico para los valores de Pi.Max
# ------------------------------------------------------------------------

mat.05 <- matrix(c(
0.05, 0.10, 0.15, 0.20, 0.25,
list.max.W005[5], list.max.W010[5], list.max.W015[5], list.max.W020[5],
list.max.W025[5]), nc=2, byrow=F)
colnames(mat.05) <- c("w%", "Pi.Max")
windows()
plot(
x=c(mat.05[,1]),
y=c(mat.05[,2]),
xlab= "Niveles de w",
xlim=c(0.05, 0.25),
ylab="Pi.Max",
```

```
cex = 1.5, lwd = 2,
type="h")
points(x=mat.05[1,1], y=mat.05[1,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.05[2,1], y=mat.05[2,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.05[3,1], y=mat.05[3,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.05[4,1], y=mat.05[4,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.05[5,1], y=mat.05[5,2], pch=21, lwd=2, cex=1.25, bg='white')
# El valor de w% que corresponde al maximo del estadistico:
ff.05 <- mat.05[row(mat.05)[which.max(mat.05)],1]
# Un punto gordo alrededor del maximo:
points(x=ff.05, y=max(mat.05), pch=21, lwd=3, cex=2.5)
# Una etiqueta con el valor del maximo:
text(x=ff.05+0.008, y=max(mat.05), round(max(mat.05), 6), cex=0.65, adj=0)
# Titulos:
title(main="Valores promedio de las probabilidades Pi.Max",
cex.main=0.9, line=2)
title(main="para todos los niveles considerados de w",
cex.main=0.9, line=1)

# ------------------------------------------------------------------------
# MODULO 6
#
# Grafico para los valores del logit de Pi.Max
# ------------------------------------------------------------------------

mat.06 <- matrix(c(
0.05, 0.10, 0.15, 0.20, 0.25,
list.max.W005[6], list.max.W010[6], list.max.W015[6], list.max.W020[6],
list.max.W025[6]), nc=2, byrow=F)
colnames(mat.06) <- c("w%", "logit(Pi.Max)")
windows()
plot(
x=c(mat.06[,1]),
y=c(mat.06[,2]),
xlab= "Niveles de w",
xlim=c(0.05, 0.25),
ylab="logit(Pi.Max)",
cex = 1.5, lwd = 2,
type="h")
points(x=mat.06[1,1], y=mat.06[1,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.06[2,1], y=mat.06[2,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.06[3,1], y=mat.06[3,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.06[4,1], y=mat.06[4,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.06[5,1], y=mat.06[5,2], pch=21, lwd=2, cex=1.25, bg='white')
# El valor de w% que corresponde al maximo del estadistico:
```

```
ff.06 <- mat.06[row(mat.06)[which.max(mat.06)],1]
# Un punto gordo alrededor del maximo:
points(x=ff.06, y=max(mat.06), pch=21, lwd=3, cex=2.5)
# Una etiqueta con el valor del maximo:
text(x=ff.06+0.008, y=max(mat.06), round(max(mat.06), 6), cex=0.65, adj=0)
# Titulos:
title(main="Valores promedio de los logit de Pi.Max",
cex.main=0.9, line=2)
title(main="para todos los niveles considerados de w",
cex.main=0.9, line=1)

# ------------------------------------------------------------------------
# MODULO 7
#
# Grafico para los valores de la transformacion angular de Pi.Max
# ------------------------------------------------------------------------

mat.07 <- matrix(c(
0.05, 0.10, 0.15, 0.20, 0.25,
list.max.W005[7], list.max.W010[7], list.max.W015[7], list.max.W020[7],
list.max.W025[7]), nc=2, byrow=F)
colnames(mat.07) <- c("w%", "asin(Pi.Max)")
windows()
plot(
x=c(mat.07[,1]),
y=c(mat.07[,2]),
xlab= "Niveles de w",
xlim=c(0.05, 0.25),
ylab="asin(Pi.Max)",
cex = 1.5, lwd = 2,
type="h")
points(x=mat.07[1,1], y=mat.07[1,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.07[2,1], y=mat.07[2,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.07[3,1], y=mat.07[3,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.07[4,1], y=mat.07[4,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.07[5,1], y=mat.07[5,2], pch=21, lwd=2, cex=1.25, bg='white')
# El valor de w% que corresponde al maximo del estadistico:
ff.07 <- mat.07[row(mat.07)[which.max(mat.07)],1]
# Un punto gordo alrededor del maximo:
points(x=ff.07, y=max(mat.07), pch=21, lwd=3, cex=2.5)
# Una etiqueta con el valor del maximo:
text(x=ff.07+0.008, y=max(mat.07), round(max(mat.07), 6), cex=0.65, adj=0)
# Titulos:
title(main="Valores promedio de las transformaciones angulares para Pi.Max",
cex.main=0.9, line=2)
```

```
title(main="para todos los niveles considerados de w",
cex.main=0.9, line=1)

# ------------------------------------------------------------------------
# MODULO 8
#
# Grafico para los valores de la transformacion Freeman-Tukey de Pi.Max
# ------------------------------------------------------------------------

mat.08 <- matrix(c(
0.05, 0.10, 0.15, 0.20, 0.25,
list.max.W005[8], list.max.W010[8], list.max.W015[8], list.max.W020[8],
list.max.W025[8]), nc=2, byrow=F)
colnames(mat.08) <- c("w%", "F-T(Pi.Max)")
windows()
plot(
x=c(mat.08[,1]),
y=c(mat.08[,2]),
xlab= "Niveles de w",
xlim=c(0.05, 0.25),
ylab="Freeman-Tukey(Pi.Max)",
cex = 1.5, lwd = 2,
type="h")
points(x=mat.08[1,1], y=mat.08[1,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.08[2,1], y=mat.08[2,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.08[3,1], y=mat.08[3,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.08[4,1], y=mat.08[4,2], pch=21, lwd=2, cex=1.25, bg='white')
points(x=mat.08[5,1], y=mat.08[5,2], pch=21, lwd=2, cex=1.25, bg='white')
# El valor de w% que corresponde al maximo del estadistico:
ff.08 <- mat.08[row(mat.08)[which.max(mat.08)],1]
# Un punto gordo alrededor del maximo:
points(x=ff.08, y=max(mat.08), pch=21, lwd=3, cex=2.5)
# Una etiqueta con el valor del maximo:
text(x=ff.08+0.008, y=max(mat.08), round(max(mat.08), 6), cex=0.65, adj=0)
# Titulos:
title(main="Valores promedio de las transformaciones Freeman-Tukey para
Pi.Max", cex.main=0.9, line=2)
title(main="para todos los niveles considerados de w",
cex.main=0.9, line=1)
```