

SAS: APUNTS DE CLASSE

1 INTRODUCCIÓ

1.1 Bàsics

1.1.1 Comentaris a SAS

Els comentaris a SAS es fan en amb `/* al principi i */` al final. “Al principi i” seria un comentari. També poden fer-se amb `* text;`

1.1.2 Organització del programa

Tenim quatre finestres principals:

- Editor (Script a R)
- Log (Consola a R): mostra avisos, errors, notes i una còpia de la sintaxi.
- Resultats
- Explorer (≈ Files a RStudio)

Si alguna finestra se'n tanca: Menu → View → Explorer → Windows Docked (acoplar).

- **Libreries a SAS:** `Libname NOMLIB 'ruta'.`

`libname Sesion1 'W:\SAS\Sesion01'; /* Es permanent només durant aquell programa.*/`

`DATA Sesion01.ex1`

Una altra opció és `x 'md W:\SAS\Sesion01'; /*Salida a sistema operativo y crear la estructura de carpetas especificada. Los nombres de las carpetas que especifiquemos no pueden contener blancos en su interior*/`

1.1.3 Normes de sintaxi a SAS

- ✚ No diferencia majúscules/minúscules, **excepte en variables de tipus cadena.**
- ✚ Tots els comandaments han d'**acabar amb punt i coma.**
- ✚ Els missings es marquen amb un punt o un blanc si la variable es de tipus caracter.
- ✚ El separador decimal és el punt.

1.1.4 Execució d'instruccions

Es realitza amb F8 o icona del running man. Si no hi ha cap text seleccionat, s'executa tot el script. **A més, cal tenir les taules tancades. El SAS no sap executar en taules obertes (Log → error).**

1.1.5 Ajuda

Ens interessen principalment dos apartats:

- Ayuda y documentación SAS
- Introducción al software SAS

1.2 Programes a SAS

Els programes s'agrupen en tres blocs:

- DATA: comandes relacionades amb la creació i gestió de dades
- PROC: anàlisi estadístic, llistats, reports, freqüències,...
- Independents: programes que no entren a cap dels dos blocs.
 - `options noxwait; /*evita tener que teclear exit después de ejecutar el siguiente comando*/`

La execució d'un bloc DATA no dona cap sortida (com resultats del PROC) però si al **log**. Els missatges **d'error** - suposen la interrupció de la execució del programa – s'assenyalen de color vermell. Els avisos **'warning'** – que mostren quelcom que no agrada al SAS – es mostren en verd. Aquests warnings poden ser greus o no. En qualsevol cas, sempre s'ha de llegir el missatge i decidir la seva gravetat. Les **notes** (note), es mostren en blau i son comentaris a les execucions. A vegades poden indicar que no ha anat la execució com volíem, no està de més llegir-les.

És convenient anar esborrant la finestra del log, això es fa amb CTRL+E.

Atenció! A la FME s'obre per defecte SAS Enterprise. Nosaltres hem de seleccionar “Obrir amb SAS 9.3”.

1.2.1 Extensions i programes del SAS.

BASE (Estadística Básica y Avanzada)
STAT (Estadística más avanzada)
OR (Operational Research)

ETS (Series Temporales)
QC (Control de calidad)

GRAPH (Gráficos)
SAS IML (Lenguaje matricial)

1.3 Un exemple per començar

DATA ex1; /*Creació de DB temporal*/

INPUT YEAR SEX \$ ALT PES; /*INPUT defineix els noms de las variables, el tipus i com les ha de llegir el SAS*/

/*SAS considera que totes les variables són nombres o caràcters (nomvar \$)*/

/* Els noms necessiten començar per A-Z (no "ç", "ñ") i poden contenir nombres i guions baixos "_". */

/*No pot tenir espais en blanc. */

AGE = 2015-YEAR; /*Nova variable*/

LABEL YEAR = 'Año nacimiento' age = 'Edad' alt = 'Altura'; /*label no canvia el nom de la variable, només la seva etiqueta. */

/*datalines o cards (es el mateix)*/

datalines;

1989 h 1.74 87

1991 d 1.67 58

1992 h 1.74 67

1988 d . 89

1987 d 1.74 75

;

PROC PRINT data = ex1;

RUN; /* S'ha d'acompanyar el PROC d'un tipus de procediment, les opcions i el RUN*/

ODS HTML CLOSE; /*Tanca la sortida en format HTML si està obert*/

ODS LISTING; /*Obre la sortida de resultats a la finestra output*/

PROC PRINT data = ex1;

RUN;

ODS LISTING CLOSE; /*Tanquem la sortida a la finestra output. Ara mateix no hi hauria cap sortida activa*/

ODS HTML

El format de sortida dels resultats es pot controlar des de la sintaxi mitjançant el comandament ODS (Output Delivery System). Possibles formats són: listing, html, pdf, rtf,...

TITLE 'Primer ejemplo con SAS'; /*Introducció d'un títol*/

PROC PRINT data = ex1 label; /*S'imprimeix el label en comptes del nom de la variable*/

ID PES; /*S'organitzen les dades en funció de la variable com si fos la principal*/

RUN;

TITLE; /*Anul·la quelcom definició de títol*/

Footnote 'Realizado con amor por SAS'; /*Igual que el títol, però a peu de pàgina*/

PROC FREQ DATA = ex1;

table sex;

***table v1 v3 v4 v1*v3;** /*Table especifica de PROC FREQ. */

RUN;

/*DESCRIPCIÓ DE NÚMERIQUES*/

PROC MEANS DATA = ex1 maxdec = 2; /*maxdec = n controla el # de decimals a mostrar */ **RUN;**

/*DESCRIPCIÓ DE ALT i AGE*/

PROC MEANS DATA = ex1 maxdec = 2; var age alt; RUN;

PROC MEANS DATA = ex1 maxdec = 2 printalltypes; /*Printalltypes dona el resultat d'anàlisi global, si no es posa tenim l'anàlisi per subgrups.*/

var alt pes;

class sex; /*Especifiquem quines variables defineixen els grups darrera de class*/

RUN;

2 GESTIÓ DE BASES DE DADES

2.1 Dades en format SAS (*.SAS7BDAT)

2.1.1 Des d'un bloc DATA

```
DATA libref.nombre_nueva_base [(opcions)]; /*la/es base/s que es crearà/n*/  
SET libref.nombre_base_existente [(opcions)]; /*La/es base/s que ja estan creades*/  
/*altres comandes;*/  
RUN;  
data Lib.girona_red; set Lib.girona (obs = 20); run;
```

2.1.2 Des d'un bloc PROC

```
PROC nomprocedimiento /* els procs no alteren la base de dades, només fan càlculs */  
DATA = libref.nombre_bdd_existente [(opciones)];  
/*COMANDES DEL PROC*/  
/*per guardar coses del proc, podem fer out amb una extensió que crei una DB nova */  
RUN;  
proc print data = lib.girona (obs =10); run;
```

2.2 Opcions dels datasets

Les opcions són més flexibles i les podem utilitzar en els PROC. Les comandes d'aquest estil només entren al bloc DATA.

- Eliminar o seleccionar variables:
 - **DROP** = listvar → no s'inclouen la llista de variables especificada.
 - **KEEP** = listvar → el fitxer mantindrà la llista de variables especificada.
- Selección de observaciones
 - Sense condició
 - **OBS** = N1 → Llegeix fins la observació
 - **FIRSTOBS** = N2 → Llegeix de N2 fins al final.
 - Amb condició
 - **WHERE** = Condició.
 - Operadors de comparació: < (LT) <= (LE) = (EQ) > (GT) >= (GE) ~= (NE) i també ^= (NE)
 - Operadors lògics: AND (&) OR (||) NOT (^ o ~)
 - Aritmètics: +, -, *, /, **
- Renombrar variables
 - **RENAME** = (nom_ant = nom_nou)
- Prioritat en la utilització de les opcions:
 1. KEEP / DROP
 2. RENAME
 3. WHERE

Totes les opcions excepte FIRSTOBS i OBS es poden utilitzar en fitxers d'entrada com de sortida i en PROCs.

```
data ex1 (drop = weight sex);  
set sashelp.class (rename = (height = altura) where = (sex = 'M' and altura >60));  
run;  
proc contents data = ex1; run; /* Informa de manera ràpida del que hi ha a la DB sense necessitat de incorporar-ho a la memòria*/
```

```
data ex2 (rename = (sex = genero));  
set sashelp.class ; X1 = (SEX= 'M'); run;
```

PROC FREQ DATA = EX2; TABLE X1; RUN; /*Obtenir la distribució de freqüències de X1*/

```
data p1;
input x1-x3; /*de x1 a x3*/
datalines;
1 2 3
1 1 1
2 3 4
1 0 1
;
```

```
data p2;
set p1;
total=x1+x2+x3;
keep total; /*keep i rename es poden
posar fora però where no, s'hauria de
posar if. Es veurà més endavant. */
run;
```

```
data p2 (keep = total);
set p1;
total=x1+x2+x3;
run;
```

Es la mateixa operació amb diferent sintaxi.

2.2.1 Dos exemples més

/*Crear class0 -temporal a partir de SASHELP.CLASS CLASS0 ha de tener renombradas las variables SEX, AGE y HEIGHT por GENERO, EDAD y ALTURA respectivamente y seleccionar solo los casos correspondientes a los hombres*/

```
data class0 (rename = (sex=genero age=edad height=altura));
set sashelp.class (where = (sex ='M'));
run;
```

/*Crear class1 -temporal a partir de SASHELP.CLASS. CLASS1 ha de tener solo las variables SEX, AGE y HEIGHT renombradas por GENERO, EDAD y ALTURA respectivamente pero solo los casos correspondientes a un valor de AGE = 12*/

```
data class1 (drop = name weight rename = (sex=genero age=edad height=altura));
set sashelp.class (where = (AGE=12));
run;

data class2 (keep = sex age height rename = (sex=genero age=edad height=altura));
set sashelp.class (where = (AGE=12));
run;

data class3 (rename = (sex=genero age=edad height=altura));
set sashelp.class (keep = sex age height where = (AGE=12));
run;
```

2.3 Importació i exportació.

2.3.1 Importació de dades d'altres formats

La manera simple: Arxiu → Importar dades → seguir les passes. La manera amb sintaxi:

```
PROC IMPORT OUT= WORK.mond2
DATAFILE= "d:\CURSAS14_15\S02\dades s02\mond2.xls"
DBMS=EXCEL REPLACE; /*TAB REPLACE si està separat per tabulador*/
RANGE="Mon_Dades$"; /*range és perquè és un excel*/
GETNAMES=YES;
DATAROW = 2; /*Fila a partir de la qual es comencen a llegir les dades. Útil amb GETNAMES = NO */
/* No tocar a partir d'aquí
MIXED=NO;
SCANTEXT=YES;
USEDATE=YES;
SCANTIME=YES;
*/
RUN;
```

2.3.2 Exportació de dades de SAS a altres formats

- Assistent d'exportació: **teclejar DEXPORT** des de la casella de comandes (a dalt a l'esquerra) i seguir els passos.
- Arxiu → Exportar dades.

3 LECTURA DE DADES EN TEXT O ASCII

3.1 Comanda INPUT

```
1 INPUT nv1 nv2 .... nvk $ .....;
2 INPUT nv1 p1-p2   nvk $ pk-pk2   ....; pk-pk2 → num col inicial - final on es troba el valor a llegir.
3 INPUT  @p1 V1 informat1w. ....@pk Vk informatkw.;
```

És obligatori aquest input si la informació no és estandar. Informació estandar: Vbles numèriques, decimal amb punt i cadenes curtes.

3.2 Input list o lectura en format lliure

Requisits:

- ✓ Tenim dades de tipus estandar.
- ✓ Les variables de text han de ser curtes (8 caràcters màxim).
- ✓ No existeixen dades missing representades per espais en blanc.
- ✓ Els valors de les variables estan separats per un o més blancs.

```
DATA EX_INPUT1;
  INFILE 'w:\ruta\nom_archivo.ext'
opcions; /*opcions de infile*/
  INPUT V1 V2 V3 $;
RUN;
```

```
DATA BAS1;
  INPUT V1-V3 $;
  datalines;
  1 3 h
  5 6 d
  4 8 d
  ;
```

```
DATA BAS2;
  input x1 $ x2 X3 $;
  DATALINES;
  PEDRO 1 H
  ANA 0 M
  GUILLEM 1 H
  JOAN 1 H
  ;
```

```
DATA alt;
  infile 'e:\SAS\DadesT3\alcades.dat';
  input alti altf;
  if alti<=0 then alti=.; /*Passem a missing*/ /*Corretgim els valors alti fora de rang que hem vist amb un PROC MEANS bàsic*/
  altm=(alti+altf)/2; /*altura promedio*/
run;
PROC MEANS DATA = ALT; RUN;
```

3.3 Column input o format fixe

INPUT var1 p11-p1j var2 p21-p2j vark pk1-pkj;

Amb:

- VARK Simboliza el nombre de la variable k-ésima
- PK1 → número columna a la que ha de situar-se el punter de lectura per iniciar la lectura del valor de la VARK
- PKj → número columna a la que ha de situar-se el punter de lectura per finalitzar la lectura del valor de la VARK
- IMPORTANT → Les dades a llegir deuen estar alineades.

Característiques de les dades per poder utilitzar el column input:

- a. La informació ha de ser el que s'anomena estandar a SAS: números sense caràcters especials (excepte el punt pels decimals), cadena o alfanumèrica.
- b. Els valors de les variables no han d'estar necessàriament separats per espais en blanc
- c. Les variables poden presentar valors missing representats per espais en blanc
- d. Podem llegir només informació relativa a algunes variables
- e. Les variables cadena poden tenir una longitud superior a 8 caràcters i/o blancs en el seu interior.

```
data ex4;
input edad peso 4-8
genero $;
datalines;
23 87.70 H
28 54 D
21 D
32 59 H
54 62 D
;
```

```
DATA ex5;
infile
'e:\SAS\DadesT3\notes.txt';
input nombre $ nota1 13-15
nota2 17-19;
notaf=(nota1+nota2)/2;
run;



proc print data = ex5; run;
```

```
DATA EX6;
infile 'e:\SAS\DadesT3\notes_m.txt';
INPUT nombre $ sexo $ edad nota1 13-15 nota2 17-19;
notaf1=(nota1+nota2)/2; /*l'operador i si hi ha un missing no el
suma, posa missing.*/
notaf2=mean(of nota1 nota2); /*Fa el càlcul hi hagi o no
missings*/
run;

proc print data = ex6; run;
```

3.4 Input amb punter de columnes

INPUT @p1 V1 informat1w.@pk Vk informatkw.; /*Amb @pj es un número que indica al programa a quina columna s'ha de dirigir el punter per llegir la informació amb un format de lectura que ve donat pel informatjw*/

-  **INFORMATS**: son formats de lectura predefinits pel SAS
-  **FORMATS**: son formats d'escriptura predefinits pel SAS

AQUESTA ESPECIFICACIÓ DE INPUT ÉS OBLIGATÒRIA QUAN LA INFORMACIÓ NO ÉS ESTÀNDAR.

3.4.1 Informat data

- 190813 informat ddmmyy6. el 6 es la amplada numèrica de la variable. **El punt és essencial!!**
- 19-8-13 informat ddmmyy8.
- 19/08/2013 informat ddmmyy10.
- 081913 informat mmddyy6.
- 8-19-13 informat mmddyy8.
- 08/19/2013 informat mmddyy10.

Hem de tenir en compte la opció *year cut off* al treballar amb dades de dates.

OPTIONS YEARCUTOFF = YEAR especifica el primer any de la finestra de 100 anys que agafa SAS de referencia. El valor per defecte és 1920.

3.4.2 Exemples

INPUT nom \$ 1-22 edat curs \$ @33 data_i ddmmyy8.;
/*informat de la data en el format lliure talla a 8 digits. **Atenció!**
El SAS guarda les dates en base a dies transcorreguts des del 1/1/1960. Per això pot aparèixer -1 si no apliquem la instrucció del format. */

FORMAT data_i ddmmyy10.; /*Aspecte de la data de sortida */

DATALINES;

Rodríguez Alvaro, José 23 A 15/10/96

Carbó Durró, Maria 20 B 31/12/59

Santos Abella, Jaume 21 B 11/09/95

;

RUN;

PROC PRINT DATA = ex7;

RUN;

DATA Ex8;

***INPUT** Nom \$ Cog \$ @14 ddn ddmmyy8. @24 ddi ddmmyy6. sex \$; /*No funciona perquè no estan encolumnats!*/

INPUT Cog \$ Nom \$ ddn :ddmmyy8. ddi :ddmmyy6. sex \$;
/*Al canviar l'arroba posició per l'informat amb dos punts davant li diem al SAS que vagi al següent punt que no estigui en blanc!*/

format ddn ddi ddmmyy10.;

DATALINES;

BACH JORDI 13/10/05 110711 H

CASES EVA 01/09/50 070801 D

TORT JOSEP 07/01/44 230899 H

SALA JOAN 24/11/83 090906 H

;

PROC PRINT DATA = Ex8;RUN;

4 PROCEDIMENTS (I)

4.1 PROC CONTENTS

Dóna informació sobre l'arxiu com el número de variables, les observacions, les etiquetes, formats,...

La opció **varnum** permet que deixi les variables en l'ordre de la DB i no en ordre alfabètic.

4.2 PROC SORT

Permet la ordenació dels casos d'una DB segons els valors d'una o més variables. No té sortida a la finestra de resultats.

4.2.1 Opcions del PROC SORT

- DATA=** [libref.]nombre_base_sas Base a ordenar.
- OUT=** SAS-data-set Nombre base donde se guardaran los valores ordenados.
- NODUP|NODUPRECS** Elimina del archivo ya ordenado las duplicidades (casos idénticos en todas las variables).
- NODUPKEY** Elimina del archivo ya ordenado los casos con valor coincidente en la/las variable/s clave/es especificadas.

4.3 PROC PRINT

Llista la informació demanada a la finestra de resultats.

4.3.1 Sintaxis

```
PROC PRINT <option(s)>;
    ID variable(s);  IDENTIFICA OBSERVACIONES CON VALORES DE LA VARIABLE INDICADA
    BY variable(s);  ORGANIZA LISTADO POR SUBGRUPOS
    SUM variable(s); IMPRIME LAS SUMAS DE VARIABLES
    VAR variable(s); VARIABLES A LISTAR
```

4.3.2 Opcions del PROC PRINT

- DATA= [libref.]nomb_base_sas
- NOOBS No se visualizará el número de la observación.
- L | LABEL Se visualizan las etiquetas de las variables.
- N Informa del número de observaciones listadas al final del output.

4.3.3 Exemples generals

```
PROC CONTENTS DATA=BASE1 VARNUM;
RUN;
TITLE 'Ej1: Listado bdd original';
PROC PRINT DATA = Base1 ; RUN;
TITLE "Ej1: Listado bdd original id codi";
PROC PRINT DATA = Base1 ; id codi; RUN;
```

```
PROC SORT DATA = Base1 OUT = B1_ord2 (label = 'Sin
duplicidades') NODUP ;
BY codi x1 x2 x3;
RUN;
/*NoDUP elimina del arxiu endreçat els casos idèntics i
consecutius.*/
```

```
TITLE 'Ej1: Listado de la base ordenada';
PROC PRINT DATA = B1_ord n noobs ; /* No Obs*/
RUN;
```

```
PROC SORT DATA = Base1 OUT = B1_ord (label ="fichero
ordenado en función de codi");
BY codi;
RUN;
```

S'endrecen els registres de la base1 segons els valor de la variable CODI i la base endreçada es guarda en una nova DB temporal (B1_ORD). A més, el label afegeix una etiqueta al arxiu, explicant el que conté. També es poden especificar etiquetes just després del DATA:
DATA NomArxiu (label =) i assigna l'etiqueta a l'arxiu creat.

També existeix la opció **NODUPKEY**, que elimina del arxiu resultant els casos amb un valor coincident a les variables clau.

```
PROC SORT DATA = Base1 OUT = B1_ord3 nodupkey;
BY codi ; RUN;
```

```
TITLE 'Ej1: NODUPKEY Sin duplicados var clave';
PROC PRINT DATA = B1_ord3 n noobs; RUN;
```

4.3.4 Exemples d'ordenació de menor a major en variables caràcter (les majúscules primer)

```
DATA BASE3;
INPUT X1 $;
DATALINES;
banda
ZAMORANO
CAÑERIA
CANARIO
CANÁMO
caña
CAÑA
RUN;
```

```
PROC SORT DATA = BASE4 OUT = B4_ORD;
BY DESCENDING X1 DESCENDING X2;
RUN;
```

```
TITLE 'Ej4: Bdd original';
PROC PRINT DATA = BASE4;
RUN;
```

```
TITLE 'Ej4: Bdd ordenada';
PROC PRINT DATA = B4_ORD;
RUN;
```

```
PROC SORT DATA = BASE3 OUT = B3_OR EBCDIC; TITLE;
*EBCDIC: mayus después de minusculas;
BY X1; RUN;
```

4.4 PROC FORMAT

Procediment per definir etiquetes de valors i/o agrupacions. Com a conseqüència, es crearà un arxiu amb la definició d'aquestes com a FORMAT.SAS7BCAT

4.4.1 Definició d'etiquetes

El nom dels formats és de màxim 8 caràcters i no pot acabar en número.

```
PROC FORMAT ;
VALUE nom1f v1 = etiqueta .... vk = etiqueta;
VALUE nom2f v1-v2 = etiqueta .... vj-vk = etiqueta;
RUN;
```

4.4.2 Especificació dels valors

valor1 = etiq valor2 = etiq 1 = 'Afroamericano' 2 = 'Caucasiano'

Valor1-valorn Un rango de valores, incluye Valor1 y Valorn 13-20 = 'Entre 13 y 20'

valor1, valor2, valork Una llista de valores 1,5,7,8 = 0

Una llista de valores y un rango 1,5,7,8, 9-15 = 'Nivel alto'

valor1<-valorn Todos los valores del rango, excepto el valor1 13<-20 = 1 20<-40 = 2

valor1-<valorn Todos los valores del rango, excepto el valorn 13-<20 = 1 20-40 = 2

Poden utilitzar-se les paraules: low, high, other

4.4.3 Creació d'arxius FORMAT.SAS7BCAT permanents

```
LIBNAME alias 'ruta';
PROC FORMAT LIBRARY = alias;
value ....;
value ....;
RUN;
```

No podem veure com estan definits els formats clicant en l'arxiu FORMAT, haurem d'utilitzar la opció FMTLIB per poder averiguar-ho.

NOTA: Cuando queremos decirle a SAS que asigne un formato que tenemos definido en un catálogo de formatos a una determinada variable, SAS lo buscará en la librería WORK o en la librería que tenga asignado el libname LIBRARY. Si el formato en cuestión está en otra librería que no es ni WORK ni LIBRARY se lo tenemos que indicar en OPTIONS FMTSEARCH = (nomlibrería).

4.4.4 Exemples

```
proc format /*fmtlib page*/; /*crea catálogo de formatos
temporal, lo guarda en WORK*/
value $gene 'm'='Mujer' 'h'='Hombre';
value resp 0 = 'NO' 1 = 'SI';
value age low -< 18 = 'menor de edad' 18-<45 = 'adulto
menor de 45' 45-high = 'adulto con 45 o más';
run;
```

```
proc format library = library /*fmtlib page*/; /*crea catálogo
de formatos permanentes*/
value $gene 'm'='Mujer' 'h'='Hombre';
value resp 0 = 'NO' 1 = 'SI';
value age low -< 18 = 'menor de edad' 18-<45 = 'adulto
menor de 45' 45-high = 'adulto con 45 o más';
run;
```

Nota: obrir un arxiu que tingui formats definits per l'usuari requereix tenir el catàleg de formats i indicar a SAS on és.

Per exemple, creem la DB permanent FORM.Ex3 que té formats definits al catàleg FORMATS que hi ha a la carpeta FORM.

```
proc print data=form.ejemplo3;
run; /*Error perquè no troba la carpeta de FORMATS*/
options fmtsearch = (form);
proc print data=form.ejemplo3;
run;
```

Puc dir-li a SAS que obri la DB sense format també. Això és útil si no tenim el catàleg de formats i volem obrir la DB. Aleshores fem:

```
options nofmtterr;
```


5 FUSIÓ D'ARXIU

5.1 Unió d'arxius en vertical.

Correspon a l'addició d'observacions. És necessari utilitzar el **SET**.

5.1.1 Sintaxis

```
LIBNAME libref 'ruta';  
DATA libref.UV (DataSet options);  
SET b1 (DSoptions) b2 (DSoptions) ... bK (DSoptions);  
RUN;
```

Como DSoptions es especialmente útil utilizar:

IN=nomvar que nos permitirá crear una variable identificadora de la procedencia de la observación;

5.1.2 Exemple

```
data b1_2;  
set b1 (in=origen) b2; /*1 a los que vienen de b1 y un 0 a los que vengan de otras*/  
varb1=origen; /*para guardar la vble temporal*/  
run;  
proc print data = b1;  
proc print data = b2;  
proc print data = b1_2; run;
```

5.2 Unió d'arxius en paral·lel

Aquí afegim variables. Necessari utilitzar **MERGE**.

5.2.1 Sintaxi

```
LIBNAME libref 'ruta';  
PROC CONTENTS DATA = libref.B1; RUN;  
PROC CONTENTS DATA = libref.B2; RUN; /*Hem de saber si la DB està endreçada abans de fer el BY*/  
PROC SORT DATA = libref.B1 OUT = b1; BY variable_comuna; RUN; /*Variable comuna = Individu per garantir una unió correcte*/  
PROC SORT DATA = libref.B2 OUT = b2; BY variable_comuna; RUN;  
DATA libref.UH (DSopcions);  
MERGE b1 (DSopcions) b2 (DSopcions);  
BY variable_comun;  
RUN;
```

5.2.2 Detalls

Els valors de la segona base s'emporten els valors de la primera amb variables del mateix nom ja que considera que conceptualment són les mateixes, tot i no tenir la mateixa amplada.

Per omisió, el MERGE combina totes les observacions de l'arxiu d'entrada. Si només volem seleccionar les observacions que siguin coincidents en tots o alguns dels arxius d'entrada aleshores s'ha d'utilitzar la opció de DB: **IN = Var_auxiliar** i alguna combinació amb comandes **IF**. Aquesta opció es pot utilitzar en unions verticals i horitzontals.

5.3 Notes importants!

1. La utilización del comando BY supone que los registros de las bases de datos que intervienen en la unión están ordenados según los valores de la variable/s especificada/as detrás de este comando, antes de realizar la unión. El PROC CONTENTS proporciona información relativa a si las bases de datos que intervienen en la unión están o no ordenadas.
2. La variable/s común/es utilizadas para realizar este tipo de unión deben tener el mismo nombre, ser del mismo tipo y anchura. Para el resto de variables hay que tener en cuenta que si algunas son comunes en los ficheros de entrada especificados detrás de MERGE, y los valores no son necesariamente coincidentes: los valores que se escribirán para estas variables en el archivo de salida – el especificado detrás de DATA – serán los que encontrará en el último archivo de entrada.

3. Por otra parte hay que tener en cuenta que si en los archivos de entrada hay variables con el mismo nombre y diferente anchura, la anchura de la variable que se escribirá en el fichero unión vendrá determinada por la anchura que tenga la primera variable que encuentre al ejecutar el comando MERGE. Esto siempre puede tener consecuencias no deseadas pero especialmente cuando se trate de variables alfanuméricas.

5.3.1 Exemple d'unió d'una mateixa variable amb noms diferents a cada fitxer.

DATA treb;	DATA salaris;
INPUT Nom \$13. edat;	INPUT treballador \$13. salar;
DATALINES;	DATALINES;
Elena 23	Santiago 900
Joan 21	Jorge Enrique 1250
Santiago 19	Elena 2300
Jorge Enrique 25	Maria Teresa 1690
;	Vanessa 1200
	;

```
proc sort data = treb;  
by nom; run;  
proc sort data = salaris (rename = (treballador=nom));  
by nom; run;
```

```
data empresa; merge treb salaris; by nom; run;  
proc print data=treb; id nom;  
proc print data=salaris; id nom;  
proc print data=empresa; id nom; run;
```

6 PROCEDIMENTS (II)

6.1 PROC FREQ

Permet la construcció de taules de freqüències i de contingència, contrast de la χ^2 de independència, etc. S'utilitza per variables categòriques o contínues agrupades.

6.1.1 Sintaxi bàsica

```
PROC FREQ DATA = NOMARCHIVO OPCIONES1;  
TABLE = VAR1 VAR2*VAR3 / OPCIONES2;  
RUN;
```

6.1.2 Opcions

- ❖ OPCIONES1: Las más utilizadas
 - NOPRINT: suprime cualquier impresión de tablas.
 - ORDER: FREQ/FORMATED/INTERNAL/... Categorías ordenadas por frecuencia/valor formateado/valor original de la variable.
 - Las numericas sin formatos aparecen en orden ascendente.
- ❖ OPCIONES2:
 - CHISQ: para hacer el contraste Chi2 de independència
- ❖ También se puede controlar la información que aparece en las casillas de las tablas de contingencia con las opciones:
 - NOROW
 - NOCOL
 - NOPERCENT
 - NOFREQ
 - EXPECTED
 - DEVIATION
- ❖ MISSING: trata el grupo missing como un grupo más y lo añade como una categoría más de la tabla
- ❖ OUT: permite guardar las frecuencias calculadas

6.2 PROC MEANS

Fa l'anàlisi descriptiva per variables contínues.

6.2.1 Sintaxi

```
PROC MEANS DATA = Bdd [OPCIONES];
```

```
VAR nomV1 nomV2.. ;
```

```
[CLASS/BY];
```

```
[OUTPUT OUT=BddOUTPUT]
```

```
RUN;
```

6.2.2 Opcions

- ❖ CLASS define subgrupos de individuos para realizar el analisis por separado (no requiere que el archivo esté ordenado previamente)
- ❖ BY define subgrupos de individuos para realizar el analisis por separado (requiere que el archivo esté ordenado previamente)
- ❖ OUTPUT OUT indica una base de datos que SAS creara con los valores de los estadísticos calculados
- ❖ OPCIONES detrás del DATA:
 - ALPHA: indica nivel de confianza para intervalos de conf
 - CLM: intervalo de confianza
 - MEAN: calcula media
 - RANGE: max - min
 - CV: coef. variación
 - SKEWNESS: asimetria
 - KURTOSIS
 - MAX
 - MIN
 - MODE: moda
 - VAR: varianza
 - STD: desv standard
 - MEDIAN: mediana
 - SUM; suma
 - P1, P5.. (percentiles)
 - Q1, Q3 cuartiles
 - Por defecto calcula N MEAN STD MIN MAX
 - MAXDEC numero maximo de decimales para mostrar calculos
 - PRINTALLTYPES: hace el analisis para los grupos indicados detras de CLASS y también para el total de individuos
 - NOPRINT: no hace la salida de resultados
- ❖ HYPOTHESIS TESTING KEYWORDS:
 - T: Prueba T con H0 media = 0
 - PROBT: P-valor del contraste

6.2.3 Exemples

proc means data=sashelp.class; var height; run;	proc means data=sashelp.class; var height; output out=descrip1; run;	proc means data=sashelp.class alpha=.1 clm mean maxdec=2; var height; run;
proc means data=sashelp.class noprint; var height; class sex; output out=descrip2 mean=media q1=cuart1 q3=cuart3 kurt=curt skew=asimet; run;	proc means data=sashelp.class printalltypes mean median var skew kurt p5 q3; var height; CLASS sex; run;	proc means data=sashelp.class t probt; var height; run;

6.3 PROC CORR

Crea una matriu de correlacions i covariàncies. Contrasta si els coeficients de correlació son o no diferents de zero.

6.3.1 Sintaxi

```
PROC CORR DATA = Bdd [OPCIONES];
```

```
VAR NomV1
```

```
[CLASS/BY];
```

```
[WITH]; /* WITH hará el calculo del coef de cada variable que ponemos detras de VAR con la que indicamos detras de WITH */
```

```
RUN;
```

6.3.2 Opcions

- PEARSON: coef correl de Pearson
- SPEARMAN: correl por rangos de Spearman
- KENDALL: Tau de Kendall
- COV: Matriz de covarianzas
- OUTP: Creará base de datos con los coeficientes de corr

6.3.3 Exemples

```
proc corr data=sashelp.class;
var height weight age;
run;
```

```
proc corr data=sashelp.class cov
outp=corrclass;
var height weight age; run;
```

```
proc corr data=sashelp.class;
var height weight;
with age; run;
```

6.4 PROC UNIVARIATE

Fa un anàlisi univariant complet.

6.4.1 Sintaxi

```
PROC UNIVARIATE DATA = Bdd [OPCIONES];
```

```
VAR NomV1 NomV2..;
```

```
[HISTOGRAM];
```

```
[PROBPLOT/QQPLOT];
```

```
[INSET];
```

```
[OUTPUT];
```

```
[CLASS];
```

```
RUN;
```

6.4.2 Opcions

- ALPHA Nivel de confianza para intervalos
- CIBASIC Intervalo de confianza asumiendo normalidad
- PLOT Box plot, steam and leaf plots
- NORMAL Test de normalidad
- MU0 Valor para contrastar H0: $\mu_0 = \text{valor}$
- I també:
 - HISTOGRAM Hace histogramas
 - PROBPLOT
 - QQPLOT

6.4.3 Exemples

```
proc univariate data=W.megaz;
var age;
class gender;
output out=perce p5=p5 p95=p95;
run;
```

```
proc univariate data=W.megaz;
var age;
histogram / midpoints=20 to 70 by 5 grid
midpercents;
inset mean='Media' min='Minimo'
max='Maximo' std = 'Desv. tipica'
/header='Estadisticos' position=ne
format=5.2;
run;
```

```
proc univariate data=W.megaz;
var age;
class gender dept;
histogram / midpoints=20 to 70 by 5 grid
midpercents nrow=2 ncol=4 normal;
inset mean='Media' min='Minimo'
max='Maximo' std = 'Desv. tipica'
/header='Estadisticos' position=ne
format=5.2;
label dept='Dept';
run;
```

6.5 PROC TABULATE

Permet una presentació de resultats endreçada, amb taules de freqüències i descriptius.

6.5.1 Sintaxi

```
PROC TABULATE DATA = Bdd [OPCIONES];
VAR NomV1 NomV2...;
[CLASS];
TABLES NomV1*NomV2,...; /*La llista de variables que posem aquí ha d'apareixer a VAR o CLASS. */ /*PRIMER files, coma,
columnes*/
RUN;
```

6.5.2 Exemples

<pre>proc tabulate data=W.megaz out=tabla; class gender store; var v4 v5; tables gender*store,(v4*mean v5*mean); run;</pre>	<pre>proc tabulate data=W.megaz; class gender store; var v4 v5; tables gender*store,(v4*mean='Media de V4' v5*mean='Media de V5'); run;</pre>	<pre>proc tabulate data=W.megaz; class gender; var age income; tables (age*(mean min max std) income*(mean min max std)),gender; run;</pre>
---	---	---

7 CREACIÓ DE VARIABLES

```
DATA MEGAZ;
SET W.MEGAZ;
vsum1=v1+v2+v3+v4+v5+v6;
vsum2=sum(of v1 -- v6); /* es lo mismo que antes, pero versión reducida, incluidas las subpuntos. Importante el of!!!! */
varit=exp(abs(v1-v2)); /*cualquier operacion que queramos*/
vmean=mean(of v1 -- v6); /*media*/
vstd=std(of v1 -- v6); /*dev estandar*/
vvar=var(of v1 -- v6); /*varianza*/
vmean_i=int(vmean); /*devuelve la parte entera*/
vmiss=nmiss(of v1 -- v6); /*cuenta el numero de missings*/
RUN;
```

7.1 Transformació condicional de variables

<pre>data megaz; set megaz; if (gender=1 & age < 35) then var4 = 1; else if (gender=1 & age >= 35) then var4 = 2; else var4 = 3; run;</pre>	<pre>data megaz; set megaz; if (gender=1 & age < 35) then var2 = 1; else var2=2; run; proc freq data = megaz; tables var1*var2 / missing; run;</pre>	<pre>data megaz; set megaz; if (gender=1 & age < 35) then var3 = 1; if (gender=1 & age >= 35) then var3 = 2; if (gender=2) then var3 = 3; /* else no pq solo afecta al ultimo if*/ run;</pre>
---	---	---

```
data megaz;
set megaz;
var5 = (gender=1 & age < 35)*1 + (gender=1 & age >= 35)*2 + (gender = 2)*3;
run;
proc freq data = megaz; tables var3*var4*var5; run;
```

7.2 Creació de més d'una variable a partir del mateix bloc IF

```
data megaz;
set megaz;
if (gender=1 & age < 35) then do;
    var6 = 1;
    var7=sum(of v1 -- v6);
    var8=mean(of v1 -- v6); /* media para cada registro, por filas*/
    var9=std(of v1 -- v6);
end;
run;
```

7.2.1 Selecció d'observacions

```
data megaz2;
set megaz;
if (gender=1 & age < 35);
run;
```

7.3 Creació de variables a través d'arrays

```
DATA MEGAZ (DROP = COUNT);
SET MEGAZ;
ARRAY VAL(6) V1 - V6; /* array que integra les sis variables*/
ARRAY XX(6) X1 - X6;
DO COUNT = 1 TO 6;
    IF (VAL(COUNT)>=5) THEN XX(COUNT)=1;
    ELSE IF (VAL(COUNT)>=0) THEN XX(COUNT)=0;
    ELSE XX(COUNT)=.;
END;
RUN;
```

7.3.1 Arrays amb dimensió variable

```
data mis2 /*(drop = i)*/;
set mis1;
/*todas las numericas de mi base
tiene los miss como -99*/
ARRAY X(*) _numeric_; /*el contará por
nosotros (*), dimensionará todo. Con el
_numeric, dimensiona solo teniendo en
cuanta las vbles numericas.*/
DO i = 1 TO dim(x);
    IF X(i) = -99 THEN X(i) =.;
END;
run;
```

```
data mis4 /*(drop = i)*/;
set mis3; /*busca solo en las cadena */
ARRAY X(*) _character_; /* las buscará
por nosotros*/
DO i = 1 TO dim(x);
    IF X(i) = "p" THEN X(i) = ' ';
END;
run;
```

```
data mis6 /*(drop = i)*/;
set mis5; /*todas son numericas*/
ARRAY X(*) _all_; /*el contará por
nosotros*/
DO i = 1 TO dim(x);
    IF X(i) = -99 THEN X(i) =.;
END;
run;
```

7.4 Definició de valors missing del usuari

Dues opcions:

- DATA EJEM1; SET EJEM1; IF X3=99999 THEN X3=.; /*EL VALOR A SERÁ MISSING DE SISTEMA, PUNTO*/
 - RUN;
 - PROC MEANS DATA=EJEM1; RUN;
- DATA EJEM1; SET EJEM1; IF X3=99999 THEN X3=.A; /*EL VALOR A SERÁ MISSING DE USUARIO, VALOR A*/
 - RUN;
 - PROC MEANS DATA=EJEM1; RUN;







7.4.1 Definir el missing d'usuari dins del bloc DATA

```
DATA EJEM1;
INPUT X1 X2 X3;
DATALINES;
1 2 .A
3 4 7
5 1 .A
2 3 8
2 . 7
3 4 3
3 7 7
;
RUN;

PROC MEANS DATA=EJEM1; RUN;
```

7.5 Creació de variables i DB per simulació

A partir de random process. **RANnnn(seed)**. Exemples:

-  RANBIN(SEED,N,P) → Binomial
-  RANEXP(SEED) → Exponencial
-  RANGAM(SEED,ALPHA) → Gamma
-  RANNOR(SEED) → Normal
-  RANPOI(SEED,M) → Poisson
-  RANUNI(SEED) → Uniforme

També es pot fer una rutina CALL, alguns exemples:

- CALL RANBIN(SEED,N,P,X)
- CALL RANEXP(SEED,X)
- CALL RANGAM(SEED,ALPHA,X)
- CALL RANNOR(SEED,X)
- CALL RANPOI(SEED,M,X)
- CALL RANUNI(SEED,X)

7.5.1 Exemples

```
DATA u1 (drop = i);
do i = 1 to 100;
  x=rannor(635985);
  output;
end;
run;

proc print data= u1; run;
```

```
data u2(keep = x);
  seed = 898647;
  do i = 1 to 100;
    call ranuni(seed, x);
    output;
  end;
run;
```

7.6 Funcions data





```
DATA MEGAZ;
SET MEGAZ;
DIA=CEIL(28*RANUNI(346895));
MES=CEIL(12*RANUNI(859647));
ANY=2013;
DATAE=MDY(MES,DIA,ANY);
FORMAT DATAE DDMMYY10.;
RUN;
```


Creem ara una variable SS1 que és 1 si la entrevista es va fer en febrer i 0 si no. Utilitzar només la variable DATAE.

```
DATA MEGAZ;
SET MEGAZ;
IF MONTH(DATAE)=2 THEN SS1=1;
ELSE SS1=0;
RUN;
```

```
proc print; var dia mes any datae ss1; run;
```

7.7 Funcions cadena

-  **Cat(char1,char2,...):** concatena
-  **Lowcase(char):** passa a minúscules
-  **Uppcase(char):** passa a majúscules
-  **Substr(char,start,length):** extreu una cadena

```
DATA EJEM2;
SET EJEM2;
CODI1=cat(ID,PROV);
CODI2=cat(ID,UPCASE(PROV));
/* per espais, com a R: (ID, " ", UPCASE (PROV)*/
CODI3=SUBSTR(ID,3,2);
RUN;
```

7.8 Conversió de variables cadena a numèriques o data, i viceversa.

```
data char;
input string $8. date $6. numer fecha ddmmyy8.; /* variable carácter - variable fecha - variable numerica */
numeric=input(string,8.); /* Convertir carácter a numérica */
numeric2=string+0; /* Convertir carácter a numérica: si sólo contiene números la carácter pasa a numérica mediante cualquier operación aritmética */
chara=put(numer,5.2); /* Convertir numérica a carácter */
sasdate=input(date,mmddy6.); /* Convertir carácter a fecha */
fechcar=put(fecha,ddmmyy8.); /* Convertir fecha a carácter */
format sasdate ddmmyy10. chara $5. fecha ddmmyy8.;
datalines;
1234.56 031704 121.2 12/11/15
3920 123104 132.2 20/11/25
;
proc contents varnum; /*ver las de entrada y las nuevas*/
run;
proc print; run;
```

8 MACROS & IML (NO VAIG ANAR, APUNTS DEL SCRIPT)

```
proc contents data=sashelp.orsales;run;
proc freq data=sashelp.orsales; table year; run;
```

8.1 /* Código base */

```
DATA carp1.year_1999;
set sashelp.orsales;
if year=1999;
run;
proc print data=carp1.year_1999 (obs=3);
run;
```

```
/* Queremos repetir el código base para todos los años, de 1999 a 2002 */
/* Este caso es fácil de modificar y replicar cuatro veces pero no siempre será así */
/* Utilizando VARIABLES MACRO podemos eliminar ese problema */
```

8.1.1 /* 1. Definición de la variable Macro %LET */

%LET el_any = 1999; * No hacen falta comillas;

8.1.2 /* 2. Llamada a la variable Macro & */

&el_any

/* Programación SAS es la conocida hasta ahora: DATA y PROCs */

/* El lenguaje de MACROS de SAS tiene sus propios comandos y sintaxis */

/* Los caracteres % & son los que informan al SAS que usamos lenguaje de MACROS */

/* % precede a los comandos Macro mientras que & indican las variables macro */

/* Las variables Macro se almacenan en una área de memoria llamada "Global symbol table".

Quando iniciamos SAS se crea la "Global symbol table" y se inicializa con variables macro.

(Las variables MACRO se almacenan como texto)

Variables Macro Automáticas

SYSBUFFR text entered in response to %INPUT

SYSCMD last non-SAS command entered

SYSDATE current date in DATE6. or DATE7. format

SYSDAY current day of the week

SYSDEVIC current graphics device

SYSDSN last SAS dataset built (i.e., WORK.SOFTSALE)

SYSENV SAS environment (FORE or BACK)

SYSERR return code set by SAS procedures

SYSFILRC whether last FILENAME executed correctly

SYSINDEX number of macros started in job

SYSINFO system information given by some PROCs

SYSJOBID name of executing job or user

SYSLAST last SAS dataset built (i.e., WORK.SOFTSALE)

SYSLIBRC return code from last LIBNAME statement

SYSLOCKRC whether most recent lock was successful

SYSMENV macro execution environment

SYSMSG message displayed with %DISPLAY

SYS Parm value passed from SYS Parm in JCL

SYSPROD indicates whether a SAS product is licensed

SYSBUFF all macro parameters passed

SYSRC return code from macro processor

SYS SCP operating system where SAS is running

SYS TIME starting time of job

SYSVER SAS version */

%PUT _ALL_; *Permite ver todas las variables macro. Aparece en el LOG;

TITLE "Este ejemplo en el día de hoy: &SYSDAY, &SYS DATE"; *Deben ponerse dobles comillas;

proc print data=carp1.year_1999 (obs=3);

run;

TITLE;

/* Volvemos a nuestro Código base */

/* Definimos una variable macro para indicar el año */

%LET el_any=1999;

```
DATA carp1.year_&el_any;  
set sashelp.orsales;  
if year=&el_any;  
run;  
proc print data=carp1.year_&el_any (obs=3);  
run;
```

```
OPTIONS SYMBOLGEN; * Escribe los valores de las variables macro en el LOG ;  
%LET el_any=1999;  
DATA carp1.year_&el_any;  
set sashelp.orsales;  
if year=&el_any;  
run;  
proc print data=carp1.year_&el_any (obs=3);  
run;
```

```
OPTIONS NOSYMBOLGEN;  
%PUT &el_any;  
%PUT El any que estoy considerando es &el_any;
```

8.1.3 /* Ejemplo 1 Texto*/

```
%let office=Sydney;  
proc print data=orion.Employee_Addresses;  
where City="&office";  
var Employee_Name;  
title "&office Employees";  
run;
```

8.1.4 /* Ejemplo 2 Número */

```
%let units=4;  
proc print data=orion.Order_Fact;  
where Quantity > &units;  
var Order_Date Product_ID Quantity;  
title "Orders exceeding &units units";  
run;
```

8.1.5 /* Ejemplo 3 Fecha */

```
%let date1=25may2007;  
%let date2=15jun2007;  
proc print data=orion.Order_Fact;  
where Order_Date between "&date1"d and "&date2"d;  
var Order_Date Product_ID Quantity;  
title "Orders between &date1 and &date2";  
run;
```

8.2 /* Combinamos variables macro con texto */

8.2.1 /* Caso 1: texto + variable macro */

```
proc print data=carp1.year_&el_any (obs=3);  
run;
```

8.2.2 /* Caso 2: variable macro + texto */

```
%LET el_any=1999;
DATA carp1.year_&el_any_ventas;
set sashelp.orsales;
if year=&el_any;
run;
proc print data=carp1.year_&el_any_ventas (obs=3);
run;
```

** Sin errores - Debemos usar el punto para separar la variable macro del texto;*

```
DATA carp1.year_&el_any._ventas;
set sashelp.orsales;
if year=&el_any;
run;
proc print data=carp1.year_&el_any._ventas (obs=3); run;
```

8.2.3 /* Caso 3: variable macro + texto con el punto añadido*/

```
%LET libre=carp1;
DATA &libre..year_&el_any._ventas;
set sashelp.orsales;
if year=&el_any;
run;
proc print data=&libre..year_&el_any._ventas (obs=3); run;
```

8.3 /* 3. Programas Macro */

/* Se trata ahora de programar la introducción de todos los años */

```
%MACRO ventas_anuales;
%DO j= 1999 %TO 2002;
data carp1.any_&j;
set sashelp.orsales;
if year=&j;
run;
%END;
%MEND ventas_anuales;
```

/* Para ejecutar la macro */

```
%ventas_anuales;
```

/* Podemos definir macros con parámetros */

```
%MACRO ventas_anuales(INICIO=, FIN=);
%DO j= &INICIO %TO &FIN;
data carp1.any_&j;
set sashelp.orsales;
if year=&j;
run;
%END;
%MEND ventas_anuales;
```

/* Para ejecutar la macro */

```
%ventas_anuales(INICIO=1999, FIN=2002);
```

8.3.1 /* EJEMPLOS DE MACROS - FUNCIONES */

```
libname carp1 'ruta';
```

```
DATA treb;  
INPUT Nom $13. edat;  
DATALINES;  
Elena      23  
Joan       21  
Santiago   19  
Jorge Enrique 25  
;
```

```
DATA salaris;  
INPUT Nom $13. salar;  
DATALINES;  
Santiago    900  
Jorge Enrique 1250  
Elena       2300  
Maria Teresa 1690  
Vanessa     1200  
;
```

8.3.2 /*Ejemplo de macro temporal: ordena dos bases de datos por var comun y las fusiona*/

```
%macro ejemplo1(datos1, datos2, varc, datosf);  
proc sort data = &datos1;  
by &varc;  
run;  
proc sort data = &datos2;  
by &varc;  
run;  
data &datosf;  
merge &datos1 &datos2;  
by &varc;  
run;  
%mend ejemplo1;
```

```
%ejemplo1(treb,salaris,nom,trebsal);
```

8.3.3 /*Ejemplo de macro permanente*/

```
OPTIONS MSTORED SASMSTORE=carp1; /*aqui le digo en qué carpeta guardo el catálogo de macros*/
```

```
%macro ejemplo2(datos1, datos2, varc, datosf) / store;  
proc sort data = &datos1;  
by &varc;  
run;  
proc sort data = &datos2;  
by &varc;  
run;  
data &datosf;  
merge &datos1 &datos2;  
by &varc;  
run;
```

```
proc print data=&datosf;
run;
%mend ejemplo2;

%ejemplo2(treb,salaris,nom,trebsal);
```

/*en una nueva sesión de SAS si quiero usar esta macro, tendré que volver a decirle donde está guardada con OPTIONS MSTORED SASMSTORE=carp1;*/

8.4 /*Macros con instrucciones condicionales*/

```
%macro ejemplo3(info,datos) / store;
%if &info=print %then %do;
  proc print data=&datos;
  run;
%end;
%else %if &info=contents %then %do;
  proc contents data=&datos;
  run;
%end;
%else %put "Error: Esta macro sólo se aplica a PRINT o CONTENTS!!!";
%mend ejemplo3;
```

```
OPTIONS FMTSEARCH = (carp1);
```

```
%ejemplo3(contents,carp1.megaz);
```

8.5 /*Macro con bloques data*/

```
%macro ejemplo4(datos, nom, nvar) / store;
data &datos&nvar;
set &datos;
%do i = 1 %to &nvar;
  &nom&i = uniform(12345);
%end;
run;
%mend ejemplo4;
```

```
%ejemplo4(carp1.megaz, x, 10);
```

8.6 /*Macro que crea base de datos a partir de un PROC*/

```
%macro ejemplo5(indatos, outdatos , vnum, vcat) / store;
proc means data = &indatos noprint;
var &vnum;
class &vcat;
output out = &outdatos
  mean = media
  std = desvt
  max = maximo
  min = minimo
  median = mediana
  skew = asim
```

```
kurt = curt  
q1 = cuart1  
q3 = cuart3;  
run;  
%mend ejemplo5;
```

```
%ejemplo5(carp1.megaz, megdes, age, store);
```

8.7 /* PROC IML */

```
proc iml; /*calculo matrical con IML*/
```

```
a={4 2 2 3 2,  
 3 3 1 2 1,  
 2 1 0 2 1,  
 5 4 4 3 4};
```

```
print a;
```

```
b={4 2 2 3,  
 2 1 4 0};
```

```
c=b*a;
```

```
print c;
```

```
a14=a[1,4];
```

```
print a14;
```

```
a1=a[1,];
```

```
print a1;
```

```
d={5 6 1 0,  
 2 3 5 1};
```

```
f=b#d;
```

```
print f;
```

```
g=b//d;
```

```
h=b||d;
```

```
print g h;
```

```
i=h`; /*i=t(h);*/
```

```
print i;
```

```
j=a[,1:4];
```

```
print j;
```

```
k=inv(j);
```

```
print k;
```

```
l=j(10,2,1);
```

```
print l;
```

```
quit;
```

```
proc iml; /*uso de funciones en IML*/
```

```
a={1 5 2 9};
```

```
b = max(a);
```

```
c = min(a);
```

```
d=cusum(a);
```

```
print b c d;
```

```
f={2 3 2,  
 1 2 1,
```

```
 0 2 1};
```

```
g=diag(f);  
print g;
```



```
h=do(3,18,3);  
i=do(3,-1,-1);  
print h i;  
j={2 3 2,  
   1 2 1};  
k=ncol(j);  
l=nrow(j);  
print k l;  
m=j[+,];  
n=j[+,];  
print m n;  
quit;
```

```
proc iml; /*definición de funciones dentro de IML*/  
  start corr;  
    n=nrow(x);  
    sum=x[+,];  
    print sum;  
    xpx=t(x)*x-t(sum)*sum/n;  
    s=diag(1/sqrt(vecdiag(xpx)));  
    corr=s*xpx*s;  
    print corr[rowname=nm colname=nm] ;  
  finish corr;  
x = { 1 2 3,  
      3 2 1,  
      4 2 1,  
      0 4 1,  
      24 1 0,  
      1 3 8};  
nm={x1 x2 x3};  
run corr;  
quit;
```

```
proc iml;  
  start corr(x, nm);  
    n=nrow(x);  
    sum=x[+,];  
    print sum;  
    xpx=t(x)*x-t(sum)*sum/n;  
    s=diag(1/sqrt(vecdiag(xpx)));  
    corr=s*xpx*s;  
    print corr[rowname=nm colname=nm] ;  
  finish corr;  
datos = { 1 2 3,  
          3 2 1,  
          4 2 1,  
          0 4 1,  
          24 1 0,  
          1 3 8};  
titulo={x1 x2 x3};  
run corr(datos,titulo);
```

```
quit;
```

8.7.1 /*Estimación por MCO de un modelo de regresión con PROC IML*/

```
proc iml;
use carp1.megaz;
read all var{age} into age;
read all var{income} into income;
unos=j(nrow(age),1,1);
x=unos||age;
beta=inv(t(x)*x)*(t(x)*income);
print beta;
pred=x*beta;
print pred;
solu = age||income||pred;
nomvar={age,income,pred};
create carp1.solucion from solu[colname=nomvar];
append from solu;
close carp1.solucion;
quit;
```

8.8 /*Modelo de regresión lineal con PROC REG*/

```
proc reg data=carp1.megaz;
model income=age;
run;
```

9 GRÀFICS AMB SAS

- ✚ PROC GCHART (BARRAS (HORIZONTAL, VERTICALES, APILADAS, 3D, PIE Y BLOCK Y STARS)
- ✚ PROC GPLOT (xy SERIES TEMPORALES O GRAFICOS BIVARIANTES (DISPERSION)
- ✚ TYPE
 - CONTROLA SI REPRESENTAMOS FRECUENCIAS / PORCENTAJES, ACUMULADOS / NO ACUMULADOS.
 - POR OMISIÓN, REPRESENTA FRECUENCIAS
 - = PCT
 - = CFREQ
 - = CPCT

9.1 Diagrama de sectores

PROC GCHART DATA = SASHELP.CARS; PIE TYPE; RUN;	PROC GCHART DATA = SASHELP.CARS; PIE TYPE / TYPE= PCT; RUN;
PROC GCHART DATA = SASHELP.CARS; PIE3D TYPE / TYPE= PCT; RUN;	PROC GCHART DATA = SASHELP.CARS; BLOCK ORIGIN / TYPE= PCT; RUN;

9.2 Diagrama de barras

9.2.1 Opció vertical

```
PROC GCHART DATA = SASHELP.CARS;
VBAR LENGTH / TYPE= PCT;
RUN;
```

9.2.2 Opció horitzontal

```
PROC GCHART DATA = SASHELP.CARS;  
HBAR LENGTH / TYPE= PCT;  
RUN;
```

```
PROC GCHART DATA = SASHELP.CARS;  
HBAR LENGTH / TYPE= PCT MIDPOINTS = 140 TO 240 BY 20;  
RUN;
```

9.2.3 Vertical amb valors discrets

```
PROC GCHART DATA = SASHELP.CLASS;  
VBAR AGE / DISCRETE TYPE= PCT ;  
RUN;
```

```
PROC GCHART DATA = SASHELP.CLASS;  
VBAR AGE/DISCRETE TYPE = PCT space = 4; /*SPACE CONTROLA SEPARACIÓN ENTRE BARRAS*/  
RUN;
```

Si volem fer un gràfic del total o de la mitjana d'una variable continua segons els valors d'una discreta, s'ho podem indicar amb **TYPE = MEAN, SUM; SUMVAR = NomVbleContinua;**

9.2.4 Exemple

9.2.4.1 Barres verticals

```
PROC GCHART DATA = SASHELP.CLASS;  
VBAR sex/DISCRETE TYPE = mean space = 4 /*mean*/ sumvar = height;  
RUN;
```

9.2.4.2 Barres horitzontals

```
PROC GCHART DATA = SASHELP.CLASS;  
HBAR sex/DISCRETE TYPE = mean space = 4 sumvar = height;  
RUN;
```

```
PROC GCHART DATA = SASHELP.CLASS;  
HBAR sex/DISCRETE NOSTATS TYPE = mean space = 4 sumvar = height;  
RUN;
```

```
PROC GCHART DATA = SASHELP.CLASS;  
VBAR AGE /DISCRETE TYPE = mean space = 4 mean  
sumvar = height GROUP = SEX; En funció del gènere i després per edat dins de cada sexe.  
RUN;
```

9.2.5 Diagrama de barres agrupades en tres dimensions

```
PROC GCHART DATA = SASHELP.CARS;  
HBAR3D TYPE/ TYPE = PCT G100 GROUP = ORIGIN; /*g100=total grupo = 100%*/  
RUN;
```

```
PROC GCHART DATA = SASHELP.CLASS;  
VBAR AGE/ DISCRETE TYPE = PCT SUBGROUP = SEX; Barres apilades amb colors per cada sexe.  
RUN;
```

9.3 Format

9.3.1 Canvi de color

PATTERN VALUE= L1 COLOR=RED; /*left y width 1*/

```
PROC GCHART DATA = SASHELP.CLASS;
VBAR AGE/ DISCRETE TYPE = PCT SUBGROUP = SEX;
RUN;
```

9.3.2 Altres característiques

goptions reset=global /*reset de los ejes, footnotes, leyendas, patterns, symbol, title*/ gunit=pct /*tamaño de lo que aparece en los gráficos. pct = % */

ftext=swiss /*fuentes del texto*/ ftitle=swissb /*fuente titulo. b de bold*/ htitle=6 htext=4 /*height de texto y titulo.*/

border /*borde por omisión en la barra de graficos*/;

9.3.3 Títols i notes de peu

```
title1 'Dow Jones Yearly Highs and Lows';
footnote1 h=3 j=l ' Source: 1997 World Almanac'
j=r 'GR21N06 ';
```

9.3.4 Símbols

```
symbol1 color=red interpol=join value=dot height=3;
symbol2 color=blue interpol=join value=star height=2;
```

9.3.5 Eixos i llegenda

axis1 order=(1955 to 1995 by 5) offset=(2,2) /*eje x*/ /*offset=intersección ejes y blancos a cada lado. posiciones que dejamos a la izq y derecha*/

```
label=none
major=(height=2)
minor=(height=1)
width=3;
```

```
axis2 order=(0 to 6000 by 1500) offset=(0,0) /*eje y*/
label=none
major=(height=2) /*vbles mayor y menor eje, separación!*/
minor=(height=1)
width=3; /*anchura eje*/
```

legend1 label=none shape=symbol(4,2) /*4 = controla separación simbolos de la leyenda, y el 2 se refiere a la altura de los simbolos de la leyenda*/
position=(top center);

9.4 Gràfics de dispersió

```
proc gplot DATA = SASHELP.CLASS;
plot height*weight;
run;
```

```
symbol1 c = Red v= square;
symbol2 c = green v= triangle;
proc gplot DATA = SASHELP.CLASS;
plot height*weight=SEX;
run;
```

```
proc gplot DATA = SASHELP.CLASS;
plot height*weight=SEX;
run;
```

```
symbol1 i=join v=dot c=black;
proc gplot data=CC.seriesg;
title 'International Airline Passengers';
plot x * date / haxis= '1jan49'd to '1jan61'd by year;
run;
```

9.5 Inserció de més d'un gràfic amb un sol proc

```
proc gplot data=CC.stocks;
  plot (high low)*year ; run; /*salen dos plots, uno de low y otro de high*/
  plot (high low)*year / overlay legend=legend1 /*overlay los superpone en el mismo*/
    vref=0 to 6000 by 750 lvref=2 /*lvref=tipo de linea!!*/
    haxis=axis1 hminor=4 /*horizontal axis, cuatro marcas de graduacion menor!!*/
    vaxis=axis2 vminor=1;

run;
quit;
```

PARA RESTAURAR OPCIONES POR OMISIÓN DE LOS GRAFICOS: **GOPTIONS RESET = ALL;**

NO CONFUNDIR TYPE CON SUMVAR, OJOOOO

10 REGRESSIÓ

10.1 Model de regressió clàssic

```
PROC REG DATA=glm.fullcoverage2;
MODEL Y = men urban private age antig;
RUN;
```

10.2 Modelo LOGIT o de regresión LOGÍSTICA

PROC LOGISTIC DATA=GLM.FULLCOVERAGE2 DESCENDING; /* si no ponemos descending, calcula la proba de 0 en lugar de la de 1*/

```
CLASS MARITAL(PARAM=REF REF='S'); /* vbles cualitativas, y la cat de referencia que quiero usar*/
MODEL Y = men urban private marital age antig /* link = probit */;
RUN;
```

*/*AIC = termino independiente y covariable. buscamos siempre el más pequeño. */*

```
PROC LOGISTIC DATA=GLM.FULLCOVERAGE2 DESCENDING;
CLASS MARITAL(PARAM=REF REF='S');
MODEL Y = men urban private marital age antig / ctble; /*sensitivity and others */
RUN;
```

```
PROC LOGISTIC DATA=GLM.FULLCOVERAGE2 OUTMODEL= glm.m4;
CLASS MARITAL(PARAM=REF REF='S');
MODEL Y(REF='0') = men urban private marital age antig;
RUN;
```

/*
al final, valores de los parametros todo codificado para aplicarlo a otros individuos (predicción)
*/

```
PROC CONTENTS DATA=glm.autocollision varnum; RUN;
PROC FREQ DATA=glm.autocollision; TABLES AGE VEHICLE_USE; RUN; /*severity en unidades monetarias*/
```

```
PROC UNIVARIATE DATA=glm.autocollision;
HISTOGRAM severity / midpoints=0 to 1000 by 50 ;
run;
```

10.3 GLM distribución continua

```
PROC GENMOD DATA=GLM.AUTOCOLLISION;  
  CLASS age(PARAM=REF REF='60+') vehicle_use(PARAM=REF REF='Pleasure');  
  MODEL severity = age vehicle_use / DIST=GAMMA;  
  WEIGHT claim_count;  
  RUN;
```

```
PROC CONTENTS DATA=glm.claims varnum; RUN;
```

```
PROC MEANS DATA=glm.claims; RUN;
```

10.3.1 GLM distribución discreta - COUNT DATA

```
PROC GENMOD DATA=GLM.CLAIMS;  
  MODEL Y = mujer urbana exp_med exp_max joven cob_adic cob_tori potente / DIST=poisson;  
  RUN;  
  
PROC GENMOD DATA=GLM.CLAIMS;  
  MODEL Y = mujer urbana exp_med exp_max joven cob_adic cob_tori potente / DIST=NB; /*binomial negativo*/  
  RUN;
```

/*un signo negativo implica menor numero esperado las mujeres tienen menos accidentes que los Hombres */
/*es mejor el NB por menor AIC. Matemáticamente se explica porque la varianza no tiene que ser igual a la media como en el caso de la Poisson*/

```
PROC CONTENTS DATA=glm.ejs1 varnum; RUN;  
PROC MEANS DATA=glm.ejs1 ; RUN;
```

10.4 Funcions de supervivència

```
proc lifetest data=glm.ejs1 plots=(survival) notable /* METHOD=KM*/; /*notable = no table de sup con todos los individuos*/  
time reslife*cens(1);  
run;
```

10.4.1 Gráficos de funciones de supervivencia y de riesgo (hazard)

```
proc lifetest data=glm.ejs1 method=act  
plots=(survival(name=Actsurv), hazard(name=Acthaz)) notable;  
time reslife*cens(1);  
run;
```

10.4.2 Análisis de la supervivencia: modelo de Weibull

```
proc lifereg data=glm.ejs1;  
model reslife*cens(1) = complain age /dist=weibull;  
run;
```

10.4.3 Análisis de la supervivencia: modelo de COX

```
proc phreg data=glm.ejs1;  
model reslife*cens(1) = complain age ;  
run;
```






11 INVESTIGACIÓ OPERATIVA (APUNTS DE CLASSE SENSE COMENTARIS, NO VAIG ANAR)

/* SAS/OR incluye una completa generación de nuevos procedimientos que permiten resolver una amplia tipología de problemas de optimización que incluyen:

- Optimización matemática, con y sin restricciones, lineales y no lineales.
- Tratamiento de variables discretas.
- Programación per metas o multiobjetivo.
- Redes y gestión de proyectos.
- etc...

*/

11.1 PROCS DE SAS OR

-  PROC LP
-  PROC NLP
-  PROC OPTMODEL
-  PROC PM
-  PROC CPM

El procedimiento PROC OPTMODEL también permite solucionar modelos de PL. PROC OPTMODEL es un procedimiento que funciona con un lenguaje de programación.

11.1.1 Ejemplo 1 PROGRAMACIÓN LINEAL

Una compañía manufacturera fabrica tres tipos de papel: de 60 g/m², de 80 g/m² y de 120 g/m². Para la fabricación del papel la compañía dispone de dos máquinas distintas, cada una de ellas puede producir los tres tipos de papel de forma independiente. Cada máquina puede producir las siguientes cantidades en toneladas por hora:

	Máquina 1	Máquina 2	
Papel de 60 g/m ²	53	52	
Papel de 80 g/m ²	51	49	
Papel de 120 g/m ²	52	45	
	Coste Máq 1	Coste Máq 2	Precio venta (euros por tonelada)
Papel de 60 g/m ²	76	75	77
Papel de 80 g/m ²	82	80	81
Papel de 120 g/m ²	96	95	99

La compañía tiene que servir 3 pedidos mensuales: 10000 toneladas de papel de 60 g/m², 5000 toneladas de papel de 80 g/m² y 4000 toneladas papel de 120 g/m². Además, debido al mantenimiento de las máquinas la disponibilidad de horas de trabajo efectives son de 670 horas mensuales para la máquina 1 y de 600 horas mensuales para la máquina 2.

Suponiendo que se vende todo lo que se produce, siendo X_{ij} el número de horas de producción del papel i en la máquina j , el modelo de programación lineal que se plantea resolver para determinar la producción que maximiza el beneficio es:

$$\text{Max } z = 53 X_{11} + 104 X_{12} - 51 X_{21} + 49 X_{22} + 156 X_{31} + 180 X_{32}$$

Sujeto a:

$$\begin{aligned}
 53 X_{11} + 52 X_{12} &\geq 10000 \\
 51 X_{21} + 49 X_{22} &\geq 5000 \\
 52 X_{31} + 45 X_{32} &\geq 4000 \\
 X_{11} + X_{21} + X_{31} &\leq 670 \\
 X_{12} + X_{22} + X_{32} &\leq 600 \\
 X_{ij} &\geq 0 \quad i=1,2,3; j=1,2.
 \end{aligned}$$

11.2 BASE DE DATOS

Values for the TYPE variable in an observation:

- ✚ MIN: contains the price coefficients of an objective row to be minimized.
- ✚ MAX: contains the price coefficients of an objective row to be maximized.
- ✚ EQ: contains coefficients of an equality constrained row.
- ✚ LE: contains coefficients of an inequality, less than or equal to, constrained row.
- ✚ GE: contains coefficients of an inequality, greater than or equal to, constrained row.

11.2.1 Example

```
data p1_14;
input _row_$9. x11 x12 x21 x22 x31 x32 _type_ $ _rhs_;
datalines;
beneficio 53 104 -51 49 156 180 MAX .
papel1 53 52 0 0 0 0 GE 10000
papel2 0 0 51 49 0 0 GE 5000
papel3 0 0 0 0 52 45 GE 4000
maquina1 1 0 1 0 1 0 LE 672
maquina2 0 1 0 1 0 1 LE 600
;
run;
```

```
proc print data=p1_14; run;
proc lp data=p1_14 rangerhs rangeprice; run;
```

11.2.2 Example 2: Programació sencera

Una empresa produce tres tipos distintos de productos en una planta que abre 40 horas a la semana. Cada producto requiere un tiempo de procesamiento (en horas) en cada una de tres máquinas:

	Producto 1	Producto 2	Producto 3
Máquina tipo I	2	2	1
Máquina tipo II	3	4	6
Máquina tipo III	4	6	5

Cada máquina debe ser utilizada por uno de los 20 trabajadores capacitados para ello, los cuales trabajan 40 horas a la semana cada uno. La planta tiene 12 máquinas de tipo I, 8 de tipo II y 10 de tipo III. Además la empresa tiene el compromiso de servir un pedido semanal de 10 unidades de producto 2. Los beneficios unitarios asociados a cada productos son, respectivamente, 90€, 120€ y 150€.

El modelo de programación lineal que nos permite determinar las unidades producidas de cada tipo de producto, con el objetivo de maximizar el beneficio semanal, teniendo en cuenta la capacidad de producción de la planta es:

$$\text{Max } z = 90 X_1 + 120 X_2 + 150 X_3$$

Sujeto a:

$$2 X_1 + 2 X_2 + 1 X_3 \leq 480$$

$$3 X_1 + 4 X_2 + 6 X_3 \leq 320$$

$$4 X_1 + 6 X_2 + 5 X_3 \leq 400$$

$$X_2 \geq 10$$

$$X_1, X_2, X_3 \geq 0$$

$$X_1, X_2, X_3 \text{ ENTEROS}$$

Values for the TYPE variable in an observation:

- ✚ INTEGER: identifies variables that are integer-constrained. In a feasible solution, these variables must have integer values. A missing value in a row with 'INTEGER' type keyword indicates that the variable is not integer-constrained. The value of variables in the 'INTEGER' row gives an ordering to the integer-constrained variables that is used when the VARSELECT= option equals PRIOR.
 - Note: Every integer-constrained variable must have an upper bound defined in a row with type 'UPPERBD'.

- ✚ UPPERBD: identifies upper bounds on the structural variables. For each structural variable that is to have an upper bound +INF, the observation must contain a missing value or the current value of INFINITY. All other values are interpreted as upper bounds, including 0.
- ✚ LOWERBD: identifies lower bounds on the structural variables. If all structural variables are to be nonnegative, that is, low=0, then you do not need to include an observation with the 'LOWERBD' keyword in a variable specified in the TYPE statement. Missing values for variables in a lower-bound row indicate that the variable has lower bound equal to zero.
 - Note: A variable with lower or upper bounds cannot be identified as unrestricted.

```
data p2;
input _row_ $9. x1 x2 X3 _type_ $ _rhs_;
datalines;
benefici 90 120 150 MAX .
maquina1 2 2 1 LE 480
maquina2 3 4 6 LE 320
maquina3 4 6 5 LE 400
pedido 0 1 0 GE 10
sup 1000 1000 1000 UPPERBD .
enteras 1 2 3 INTEGER .
;
```

```
proc lp data=p2 imaxit=500;
run;
```

11.2.3 Exemple: Programació sencera binària

Incorporad en el model de PLE anterior la existència de unos costes fijos de producción, que son 100€, 60€ y 90€ para los productos 1, 2 y 3 respectivamente: $\text{Max } z = 90 X_1 + 120 X_2 + 150 X_3 - 100 Y_1 - 60 Y_2 - 90 Y_3$

Sujeto a:

$$2 X_1 + 2 X_2 + 1 X_3 \leq 480$$

$$3 X_1 + 4 X_2 + 6 X_3 \leq 320$$

$$4 X_1 + 6 X_2 + 5 X_3 \leq 400$$

$$X_2 \geq 10$$

$$X_1 - M Y_1 \leq 0$$

$$X_2 - M Y_2 \leq 0$$

$$X_3 - M Y_3 \leq 0$$

$$X_1, X_2, X_3 \text{ ENTEROS}$$

$$Y_1, Y_2, Y_3 \in \{0, 1\} \text{ BINARIAS}$$

$$X_1, X_2, X_3 \geq 0$$

Values for the TYPE variable in an observation:

- ✚ BINARY: identifies variables that are constrained to be either 0 or 1. This is equivalent to specifying that the variable is an integer variable and has a lower bound of 0 and an upper bound of 1. A missing value in a row with 'BINARY' type keyword indicates that the variable is not constrained to be 0 or 1. The value of variables in the 'BINARY' row gives an ordering to the integer-constrained variables that is used when the VARSELECT= option equals PRIOR.

```
data p2;
input _row_ $9. x1 x2 x3 y1 y2 y3 _type_ $ _rhs_;
datalines;
benefici 90 120 150 -100 -60 -90 MAX .
maquina1 2 2 1 0 0 0 LE 480
maquina2 3 4 6 0 0 0 LE 320
```

```

maquina3  4  6  5  0  0  0 LE  400
pedido    0  1  0  0  0  0 GE  10
cf1       1  0  0 -1000  0  0 LE  0
cf1       0  1  0  0 -1000  0 LE  0
cf1       0  0  1  0  0 -1000 LE  0
limsup    1000 1000 1000 . . . UPPERBD .
enteras   1  2  3 . . . INTEGER .
binarias . . . 1  2  3 BINARY .
;
run;

```

```

proc print data=p2; run;
proc lp data=p2 imaxit=500; run;

```

11.2.4 Ejemplo: PROGRAMACIÓN ENTERA BINARIA

Utilizando variables binarias, incorporad en el modelo anterior el hecho de que la empresa se plantea alargar el tiempo de apertura, actualmente de 40 horas semanales. Por motivos de mantenimiento sólo dos de las tres máquinas podrían estar funcionando durante más horas de las que se marcan en la restricción correspondiente. $\text{Max } z = 90 X_1 + 120 X_2 + 150 X_3 - 100 Y_1 - 60 Y_2 - 90 Y_3$

```

Sujeto a: 2 X1 + 2 X2 + 1 X3 <= 480 + M2(1-Y4)
          3 X1 + 4 X2 + 6 X3 <= 320 + M2(1-Y5)
          4 X1 + 6 X2 + 5 X3 <= 400 + M2(1-Y6)
          Y4 + Y5 + Y6 = 1
          X2 >= 10
          X1 - M1 Y1 <= 0
          X2 - M1 Y2 <= 0
          X3 - M1 Y3 <= 0
          X1,X2,X3 ENTEROS
          Y1,Y2,Y3,Y4,Y5,Y6 {0,1} BINARIAS
          X1,X2,X3 >= 0

```

```

data p2;
input _row_ $9. x1 x2 x3 y1 y2 y3 y4 y5 y6 _type_ $ _rhs_;
datalines;
benefici  90 120 150 -100 -60 -90  0  0  0 MAX .
maquina1  2  2  1  0  0  0 10000 0  0 LE 10480
maquina2  3  4  6  0  0  0  0 10000 0 LE 10320
maquina3  4  6  5  0  0  0  0  0 10000 LE 10400
un        0  0  0  0  0  0  1  1  1 EQ 1
pedido    0  1  0  0  0  0  0  0  0 GE 10
cf1       1  0  0 -1000  0  0  0  0  0 LE 0
cf1       0  1  0  0 -1000  0  0  0  0 LE 0
cf1       0  0  1  0  0 -1000  0  0  0 LE 0
limsup    1000 1000 1000 . . . . . UPPERBD .
enteras   1  2  3 . . . . . INTEGER .
binarias . . . 1  2  3  4  5  6 BINARY .
;
run;

```

```

proc lp data=p2 maxit=5000; run;

```

11.2.5 Exemple: programació per metes

Replantead el modelo inicial como un modelo de programación con las siguientes metes. El gerente de la empresa quiere:

1. conseguir un beneficio de al menos 10.000€.
2. no subutilizar la capacidad productiva de la empresa.
3. conseguir que todas las máquinas estén funcionando durante todo el tiempo en que la empresa está abierta.
4. que se permita hacer horas extras, pero que éstas no sean superiores a 1 hora extra semanal por trabajador.
5. que se sirva el pedido semanal al que se ha comprometido la empresa.

$$\text{Min } z = 1/10000 \, d1(-) + 1/800 \, d2(-) + 1/480 \, d3(-) + 1/320 \, d4(-) + 1/400 \, d5(-) + 1/20 \, d6(+) + 1/10 \, d7(-)$$

Sujeto a:

Meta I: Beneficio de al menos 10000

$$90 \, X1 + 120 \, X2 + 150 \, X3 + d1(-) - d1(+) = 10000$$

Meta II: La capacidad productiva semanal se liga a los 20 trabajadores

$$9 \, X1 + 12 \, X2 + 12 \, X3 + d2(-) - d2(+) = 800$$

Meta III: Horas de funcionamiento de las máquinas

$$2 \, X1 + 2 \, X2 + 1 \, X3 + d3(-) = 480$$

$$3 \, X1 + 4 \, X2 + 6 \, X3 + d4(-) = 320$$

$$4 \, X1 + 6 \, X2 + 5 \, X3 + d5(-) = 400$$

Meta IV:

$$d2(+) + d6(-) - d6(+) = 20$$

Meta V:

$$X2 + d7(-) - d7(+) = 10$$

$X1, X2, X3$ ENTEROS

$d_i(+), d_i(-)$ ENTEROS $i=1, \dots, 7$

$$d_i(+) \times d_i(-) = 0, \quad i=1, \dots, 7$$

$$X1, X2, X3 \geq 0$$

$$d_i(+), d_i(-) \geq 0, \quad i=1, \dots, 7$$

data p2;

```
input _row_ $12. x1 x2 x3 d1me d1ma d2me d2ma d3me d4me d5me d6me d6ma d7me d7ma _type_ $
_rhs_;
```

datalines;

```
Min          0  0  0  0.0001  0  0.00125  0  0.0021  0.0031  0.0025  0  0.05  0.1  0  MIN
```

```
Beneficio    90 120 150      1 -1      0  0      0  0      0  0      0  0      EQ 10000
```

```
Capacidad    9  12 12      0  0      1 -1      0  0      0  0      0  0      EQ   800
```

```
Maq_tipo_1    2  2  1      0  0      0  0      1  0      0  0      0  0      EQ   480
```

```
Maq_tipo_2    3  4  6      0  0      0  0      0  1      0  0      0  0      EQ   320
```

```
Maq_tipo_3    4  6  5      0  0      0  0      0  0      1  0      0  0      EQ   400
```

```
H_extra_1     0  0  0      0  0      0  1      0  0      0  1     -1  0      EQ    20
```

```
Pedido        0  1  0      0  0      0  0      0  0      0  0      0  1     -1  0      EQ    10
```

```
limsup       1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 UPPERBD
```

```
enteras       1  2  3      4  5      6  7      8  9     10 11     12 13     14  INTEGER
```

;

run;

proc lp data=p2 maxit=500; run;

11.3 Programació no lineal

 TECH (Algoritmo de optimización)

a) Quadratic Programming

QUADAS
LICOMP

b) General Nonlinear Optimization

*Nonlinear Constraints

Small Problems: NMSIMP

Not suitable for highly nonlinear problems or for problems with .

Medium Problems: QUANEW

*Only Linear Constraints

Small Problems: TRUREG (NEWRAP, NRRIDG)

(n < 40) where the Hessian matrix is not expensive to compute.

Medium Problems: QUANEW (DBLDOG)

(n < 200) where the objective function and the gradient are much faster to evaluate than the Hessian.

Large Problems: CONGRA

(n > 200) where the objective function and the gradient can be computed much faster than the Hessian and where too much memory is needed to store the (approximate) Hessian.

No Derivatives: NMSIMP

(n < 20) where derivatives are not continuous or are very difficult to compute.

c) Least Squares Minimization

Small Problems: LEVMAR (HYQUAN)

(n < 60) where the crossproduct Jacobian matrix is inexpensive to compute.

Medium Problems: QUANEW (DBLDOG)

(n < 200) where the objective function and the gradient are much faster to evaluate than the crossproduct Jacobian.

Large Problems: CONGRA

No Derivatives: NMSIMP

*/

```
proc nlp tech=CONGRA OUTEST=pp;
max z;
parms x=3, y=3;
bounds x>=0 ,y>=0;
lincon 2*x+3*y<=90;
z=(x*y);
run;
```

```
proc print data=pp;
run;
proc nlp tech=TRUREG OUTEST=pp;;
max z;
parms x=3, y=3;
bounds x>=0 ,y>=0;
lincon 2*x+3*y<=90 , y-x>=0;
z=(x*y);
run;
```

```
proc print data=pp;
run;
```