INTRODUCCION al SAS

Bibliografía básica

L. Delwiche and S. Slaughter. *The Little SAS Book: A Primer*, ISBN: 978-1-59047-333-7

G. Der and B.S. Everitt. *A Handbook of Statistical Analyses using SAS*, ISBN: l-58488-245-X

C. Pérez. *El Sistema Estadístico SAS*, ISBN: 8420531685

Páginas WEB con documentación sobre SAS

Documentación de SAS 9.1.3

http://support.sas.com/91doc/docMainpage.jsp

http://support.sas.com/documentation/onlinedoc/91pdf/index_913.html

SAS Information Guides

http://www.psych.yorku.ca/lab/sas/

Resources to help you learn and use SAS

http://www.ats.ucla.edu/stat/sas/

SAS Coding Tips and Techniques

http://www.sconsig.com/sastip.htm

Statistics Tutorials using SAS Software

http://www.stattutorials.com/SAS/index.html

Introducción a la programación en SAS

Ideas Básicas sobre SAS :

- SAS es un programa *declarativo* que se basa en una serie de programas o *scripts* que realizan las tareas. Existe un interface con ventanas denominada *Enterprise Guide* que facilita el uso de SAS, pero se vende como un programa aparte.
- Toda sentencia de SAS termina con un *punto-y-coma*: (**;**)
- Todas las variables y los nombres de ficheros de datos deben empezar con una letra o con _ y no deben contener signos especiales (raros) como %, # etc.
- Los comandos de SAS son equivalentes en mayúsculas y minúsculas. Por ejemplo, es lo mismo TuBody, tubody, tUboDY y tuBoDy.
- No hay restricciones en cuanto a dónde empiezan los comandos o terminan, o si los comandos se pueden colocar en líneas contiguas, aunque es conveniente evitar los programas confusos.
- Los nombres de variables o conjuntos de datos no pueden tener más de 32 caracteres.

Clasificación de las sentencias SAS

Paso DATA

Control

DO, DO UNTIL, DO WHILE, LINK
GO TO, IF-THEN-ELSE, END
RETURN, SELECT

Manejo

BY, CARDS, DATA, FILE, INFILE, INPUT
MERGE, PUT, SET, UPDATE, WHERE

Acción

ABORT, CALL, DELETE, DISPLAY, SUM
OUTPUT, STOP, LIST, MISSING

Información

ARRAY, ATTRIB, DROP, FORMAT
KEEP, LABEL, LENGTH
RENAME, RETAIN, INFORMAT

Paso PROC

Atributos de variables

ATTRIB, LABEL
FORMAT

Información de procs

BY, CLASS, FREQ, ID

OUTPUT, PROC, QUIT

VAR, WHERE, WEIGHT

Cualquier Paso

Información y control

FILENAME

LIBNAME

FOOTNOTE

OPTIONS

RUN

TITLE

X

COMMENT

Otras

COMMENT

ENDSAS

Se puede encontrar un diccionario de los comandos anteriores en

`http://javeeh.net/sasintro/intro48.html`

Ejemplo: Programa simple con formato *fijo*:

```
data naranjas;  /* defines un fichero sas llamado naranjas */
input estado $ 1-10 tempranas 12-14 tardias 16-18;
datalines;
florida    124 90
california  40 27
texas      1.3 .3
arizona     .4 .5
;

proc print data=naranjas;
run;
```

Los programas de `SAS` se componen de bloques `DATA` y bloques `PROCedure`.

Los bloques `DATA` indican qué hacer con los datos: se definen formatos, nombres de nuevas variables, se crean nuevas variables, se filtran los datos etc.

En los procedimientos `PROC` se indican qué aplicaciones estadísticas se realizan sobre los datos, por ejemplo, tablas de contingencia, análisis de regresión etc.

Lectura de datos

Empezamos con un ejemplo de recogida de datos desde Internet.

Lectura de datos desde Internet

Ejemplo: Bajamos los datos de los *huevos* del repositorio *StatLib* de la universidad de *Carnegie Mellon*. Contiene el número medio de huevos producidos al año por las hembras

de los cangrejos reales en Alaska.

```
filename conecto URL 'http://lib.stat.cmu.edu:80/crab/eggs';

data cangrejos;
infile conecto;
input anno 1-2 numhuevos 4-9;
run;

proc print data=cangrejos;
run;
```

El nombre de *conecto* es un *nickname* usado por SAS para identificar la dirección de Internet. A veces, esto no funciona porque hay problemas con los *firewalls*. La solución sería importar el fichero directamente y luego leerlo como un fichero normal desde el dispositivo (disco o cualquier otro soporte) local

Actualmente, la forma más habitual de trabajar con la WEB es mediante bases de datos *SQL*. Para ello, `SAS` dispone del `PROC SQL`.

De hecho, `SQL` es un lenguaje de programación sencillo que tiene como órdenes básicas:

- `SELECT`: columnas (variables) que se seleccionan.
- `FROM`: tablas (datasets) que se seleccionan.
- `ON` condiciones que se han de cumplir.
- `WHERE` condiciones por filas (observaciones) que se han de cumplir.
- `GROUP BY` agrupaciones según las columnas.
- `HAVING` condiciones que han de cumplir las agrupaciones.

- `ORDER BY` ordenar según las columnas.

Lectura de un fichero tipo Excel

La opción más cómoda es usar el *SAS Import Wizard*, que importa el fichero en formato Excel desde el menú principal.

Se asume que los nombres de las variables están en la primera fila del fichero Excel.

File → Import Data

Elegir `Excel .xls format` (es el que viene por defecto)

Next → Browse → `fichero.xls`

La opción por defecto es leer el nombre de las variables desde la primera fila.

Poner el nombre **alumnos** para el data set.

Pinchar en **Finish**

Este método funciona también para otros formatos como datos separados por comas o por tabulaciones.

Otra alternativa es usando un *script*:

```
/* Importar ficheros Excel */
proc import out=veo
datafile="c:\pasas\alumnos.xls"
     dbms=excel97 replace;
     sheet="hs0";
     getnames=yes;
run;
```

```
/* Vemos como son las 10 primeras observaciones */
proc print data=veo (obs=10);
run;
```

Lectura de ficheros *CSV*

Uno de los formatos más empleados es el formato `ascii` separado por comas (ficheros con extensión `csv`):

```
/* Importar ficheros csv */
data veo;
infile 'c:\pasas\alumnos.csv' delimiter=',' dsd;
input gender id race ses schtyp prgtype $ read write math science
socst;
run;

/* Vemos como son las 10 primeras observaciones */
proc print data=veo (obs=10);
run;
```

NOTAS: El fichero **no** tiene el nombre de las variables en la primera fila:

```
....
0,11,1,2,1,"academic",34,46,45,39,36
0,84,4,2,1,"general",63,57,54,,51
0,48,3,2,1,"academic",57,55,52,50,51
....
```

En algún caso puede haber datos *missing* que se indican, en este caso concreto, por dos comas consecutivas en la fila. Para que se reconozca como *missing* se usa la opción **dsd** en el comando **infile**.

Lectura de datos con formato fijo ASCII

El formato fijo por columnas requiere una codificación para saber la localización de cada variable.

Por ejemplo, supongamos que los datos son

```
195 094951
26386161941
38780081841
479700 870
56878163690
66487182960
786 069 0
88194193921
98979090781
107868180801
```

La localización de las variables es en este caso:

nombre variable	**número columna**
id	1-2
a1	3-4
t1	5-6
gender	7
a2	8-9
t2	10-11
tgender	12

El programa para leer los datos es

```
data fijos;
infile 'c:\pasas\schdat.fix';
input id 1-2 a1 3-4 t1 5-6 gender 7 a2 8-9 t2 10-11 tgender 12;
run;

proc print data=fijos;
run;
```

A veces, los datos se ponen directamente dentro del programa `SAS`, cuando no tienen un volumen muy grande, o como una práctica de programación

```
data veo;
input id female race ses schtype $ prog
read write math science socst;
datalines;
```

```
147 1 1 3 pub 1 47 62 53 53 61
108 0 1 2 pub 2 34 33 41 36 36
18 0 3 2 pub 3 50 33 49 44 36
153 0 1 2 pub 3 39 31 40 39 51
50 0 2 2 pub 2 50 59 42 53 61
51 1 2 1 pub 2 42 36 42 31 39
102 0 1 1 pub 1 52 41 51 53 56
57 1 1 2 pub 1 71 65 72 66 56
160 1 1 2 pub 1 55 65 55 50 61
136 0 1 2 pub 1 65 59 70 63 51
;
run;

proc print data=veo;
run;
```

Lectura de ficheros con longitud de filas grande

`SAS` puede leer hasta 256 caracteres por línea. Si un fichero tiene filas con un mayor número de caracteres, se ha de usar la opción `LRECL`.

Ejemplo: Consideramos un fichero de datos con 1000 variables numéricas $(x_1, \ldots, x_{1000})$ cada una, con una anchura de 3 posiciones y separadas por un espacio en blanco. Se trata de leer las 1000 variables y calcular la suma para cada individuo.

```
data totales;
infile 'datos.txt' lrecl=3999;
```

```
input x1-x1000;
total=sum(of x1-x1000);
proc print;
run;
```

Modificación básica de datos Se pueden modificar de manera inmediata, mediante el menú principal, los valores de una base de datos que ha sido importada previamente.

Explorer → **Libraries** → **Work** → Pinchar en `alumnos` → **Edit** → **Edit Mode**

Pinchar en la ventana de los datos para modificarlos.

Grabación externa de datos Lo más cómodo es usar el **SAS/Explorer**:

File → **Export Data**

Elegir de **Library**: WORK → **Member**: VEO

Next → tipo de fichero (ejemplo en Excel)

Next → nombre y ruta del nuevo fichero: `veo.xls`

Pinchar en **Finish**

SAS Data sets

Existen distintos tipos de conjuntos de datos en SAS (*data sets*).

En general, un data set de SAS es una representación binaria del conjunto de datos. Se pueden clasificar en:

- *Data sets* **temporales**: permanecen en la memoria de trabajo mientras dura la sesión de SAS pero se borran después. Son adecuados para conjuntos pequeños de datos.
- *Data sets* **permanentes**: se guardan en un fichero para su uso posterior. Son adecuados para conjuntos grandes que requieren numerosos pasos en el análisis.

Ejemplos :

```
DATA mifichero;   /* Crea un fichero SAS temporal */
```

```
DATA _NULL_ ;
```

No crea ningún fichero SAS, se leen los datos desde otro fichero externo.

`LIBNAME`: Identifica en el programa SAS dónde está localizado el fichero SAS creado previamente:

```
libname dondestoy 'c:\pasas';
data dondestoy.mifichero;
```

Cuando se crea un fichero permanente SAS de tipo binario se guarda como `*.sd2`.

Después, basta referirse a él (sin leer los datos) cuando se aplican los distintos procedimientos.

```
libname dondestoy 'c:\pasas';
proc sort data=dondestoy.mifichero;
```

Para crear un fichero SAS permanente:

```
libname colegio 'c:\pasas';
data original;
input depa $ 1-8 cuenta 10-13 clase $ 15-20;
datalines;
Arte     449    dia
```

```
Ciencia  1411   dia
Musica   259  tarde
Lenguaje 759    dia
;
data colegio.lio;
set original;
if clase='tarde' then delete;
run;

proc print;
run;
```

Se obtiene el fichero `lio.SD2` que se guarda en el directorio indicado por `libname` con formato binario.

Cambios de estética en la salida

Para obtener cambios globales, se puede usar el comando `options`:

Las instrucciones se colocan al principio del programa para variar la estética de la salida.

```
/* Ejemplo */
options linesize=80  /* 80 de ancho de linea */
pagesize=54   /* taman~o de la pagina */
nocenter
nodate
nonumber;
```

Para asignar diferentes formatos para facilitar la interpretación de las salidas

Procedimiento FORMAT

proporciona una regla de asignación de formatos.

Ejemplo: aplicamos un formato dado a la variable **schtyp**

```
data veo;
infile 'c:\pasas\alumnos.csv' delimiter=',' dsd;
input gender id race ses schtyp prgtype $ read write math science
socst;
run;

proc format;
value scl 1="publico"
          2="privado";
run;

proc freq data=veo;
/* Se asocia el formato scl a la variable schtyp */
tables schtyp;
format schtyp scl. ;
run;
```

Ponemos una etiqueta a la variable

```
data veo1(label="Educacion Superior, 200 casos");
```

```
set veo;
label schtyp = "Tipo de colegio al que se asiste.";
run;

proc contents data=veo1;
run;
```

Consideramos un ejemplo más elaborado:

```
data veo;
infile 'c:\pasas\alumnos.csv' delimiter=',' dsd;
input gender id race ses schtyp prgtype $ read write math science socst;
run;

proc format;
/* creas etiquetas para schtyp y para grade */
value scl   1="publico"
            2="privado";

value abcdf  0="F"
             1="D"
             2="C"
             3="B"
             4="A";

/* creas etiquetas para female */
value fm    1="mujer"
```

```
            0="hombre";
run;

/* creas el data file hs1 y las etiquetas */
data veo1(label="Educacion Superior, 200 casos");
set veo;

/* con el comando label se etiqueta la variable schtyp */
label schtyp = "Tipo de colegio al que se asiste.";

/* con la sentencia if-then se crea la nueva variable prog,
que es numerica */

if prgtype="academic" then prog=1;
if prgtype="general" then prog=2;
if prgtype="vocational" then prog=3;

/* etiquetas la variable prog */
label prog="El tipo de programa en el que el estudiante se matricula.";

/* creas una nueva variable: female igual que la variable gender
y se elimina gender usando drop */

female=gender;
drop gender;
```

```
/* etiquetas la variable female */
label female="El genero del estudiante.";

/* con el comando if recodificas los valores iguales a 5 en
la variable race a un valor missing (.) */
if race=5 then race=. ;

/* creas la variable total que es la suma de read+write+socst */
total = read + write + socst;

/* creas una una variable, llamada grade, igual al total */
grade=total;

/* recodificas grade */
if (total < 80) then grade=0;
if (80 <= total < 110) then grade=1;
if (110 <= total < 140) then grade=2;
if (140 <= total < 170) then grade=3;
if (total >= 170) then grade=4;

/* etiquetas la variable grade */
label grade = "Estos son los grados combinados de read, write y
socst.";
run;
```

Comprobamos que funciona todo:

```
proc contents data=veo1;
run;

proc print data=veo1 (obs=20);
format schtyp scl. female fm. grade abcdf.;
run;

proc freq data=veo1;
format schtyp scl. female fm.;
tables schtyp female;
run;
```

Arrays

Se pueden simplificar, a veces, las operaciones usando arrays. Estos son útiles para evitar múltiples sentencias `IF`.

Su sintaxis es

```
ARRAY nombre (n) lista-variables;
```

donde `n` es el número de variables en el array.

Ejemplo: Supongamos que los datos son

```
data tiendas;
input ciudad $ edad domk wj hwow simbh kt aomm libm tr filp ttr;
array tienda (10) domk wj hwow simbh kt aomm libm tr filp ttr;
** cambias todos los ceros  a valores missing;
do i=1 to 10;
if tienda(i) = 0 then tienda(i) = .;
end;
cards;
Zaragoza  54 4 3 5 0 0 2 1 4 4 0
Valencia  33 5 2 4 3 0 2 0 3 3 3
Oviedo    27 1 3 2 0 0 0 3 4 2 3
Santander 41 4 3 5 5 5 2 0 4 5 5
Sevilla   18 3 4 0 1 4 0 3 0 3 2
;
proc print data = tiendas;
run;
```

Gestión de los ficheros de datos

Comandos que se usan en los ejemplos:

keep	mantiene en memoria las variables nombradas
drop	elimina las variables nombradas
set	Lee el/los fichero(s) dado(s). Si se nombra más de un fichero, se combinan en uno
merge	Mezcla ficheros
where	Condicionales

Ejemplo Supongamos que estamos interesados en analizar a los estudiantes con un índice de lectura mayor que 60. Se toma un subconjunto de la muestra y se define una carpeta donde se guardan los valores de dicho grupo de estudiantes.

```
data veo;
infile 'c:\pasas\alumnos.csv' delimiter=',' dsd;
input gender id race ses schtyp prgtype $ read write math science
socst;
run;

data veo1;
set veo;
  where (read >=60);
run;

proc print;
run;

proc means data=veo1;
var read;
run;
```

Supongamos que el fichero tiene muchas variables, por ejemplo 2000 pero sólo estamos interesados en unas pocas: **id female read** y **write**. Para seleccionarlas:

```
data veo2;
```

```
set veo1;

keep id read write;
run;

proc contents data=veo2;
run;
```

Otra opción sería eliminar del análisis las otras variables. Por ejemplo, eliminamos las variables **ses** y **prog**.

```
data veo3;
set veo1;
drop ses;
run;

proc contents data=veo3;
run;
```

Tenemos ahora una carpeta denominada **masters** y tenemos un fichero para los hombres llamado **hsmale** y otro para las mujeres llamado **hsfemale**. Debemos combinarlos a ambos para analizarlos.

```
proc freq data="c:\pasas\hsmale";
tables female;
run;
```

```
proc freq data="c:\pasas\hsfemale";
tables female;
run;

data hsmasters;
set "c:\pasas\hsmale" "c:\pasas\hsfemale";
run;

proc freq data=hsmasters;
tables female;
run;
```

Consideramos ahora dos ficheros. En el primer fichero tenemos la información demográfica llamado **hsdem** y otro con las puntuaciones llamado **hstest** y los queremos juntar en uno solo.

```
proc print data="c:\pasas\hsdem"(obs=10);
run;

proc print data="c:\pasas\hstest"(obs=10);
run;

proc sort data="c:\pasas\hsdem" out=dem;
by id;
run;

proc sort data="c:\pasas\hstest" out=test;
```

```
by id;
run;

data "c:\pasas\hsdiss";
merge dem(in=one) test(in=two);
fromdem=one;
fromtest=two;
by id;
run;

proc print data="c:\pasas\hsdiss";
run;

proc print data="c:\pasas\hsdiss";
var id fromdem fromtest;
run;

proc freq data="c:\pasas\hsdiss";
tables fromdem*fromtest;
run;
```

Ordenación de ficheros

La ordenación de los conjuntos de datos es una práctica muy habitual. Se realiza mediante el comando `SORT`.

Ejemplo

```
data cuentas;
input Empresa $ 1-22 Debito 25-30 Numero 33-36 Ciudad $ 39-51;
datalines;
Paul Pizza              83.00   1019  Apex
World Wide Electronics  119.95  1122  Garner
Strickland Industries   657.22  1675  Morrisville
Ice Cream Delight       299.98  2310  Holly Springs
Watson Tabor Travel      37.95  3131  Apex
Boyd & Sons Accounting  312.49  4762  Garner
Bob Beds                119.95  4998  Morrisville
Tina   Pet Shop          37.95  5108  Apex
Elway Piano and Organ    65.79  5217  Garner
Tim I s Burger Stand    119.95  6335  Holly Springs
Peter Auto Parts         65.79  7288  Apex
Deluxe Hardware         467.12  8941  Garner
Pauline Antiques        302.05  9112  Morrisville
Apex Catering           37.95   9923  Apex
;
proc sort data=cuentas out=porciudad;
by ciudad empresa;
run;

proc print data=porciudad;
var empresa ciudad debito numero;
title 'Clientes';
title2 'Lista alfabetica de ciudades';
```

```
run;
```

Tratamiento de datos faltantes (*missing*)

Se utiliza la sentencia `MISSING`.

Ejemplo

```
data veo;
/* defino como missing los caracteres A y R */
missing A R;
input id x;
cards;
1   2
2   R
3   1
4   A
5   2
;
proc print;
run;
```

Programación con SAS

Transformaciones de variables

En SAS se pueden usar numerosas funciones matemáticas.

Ejemplo:

```
data getdata;
input name $14. x test1 test2 test3;
datalines;
john smith       4.2 86.5 84.55 81
samuel adams     9.0 70.3 82.37 .
ben johnson     -6.2 82.1 84.81 87
chris adraktas   9.5 94.2 92.64 93
john brown        .  79.7 79.07 72
;
run;

data funct1;
set getdata;
t1int=int(test1); t2int=int(test2);    /* parte entera */
t1rnd=round(test1); t2rnd=round(test2,.1);  /* redondeo */
tave=mean(test1, test2, test3);         /* calculo de la media */
run;

proc print data=funct1;
var test1 test2 test3 t1int t2int t1rnd t2rnd tave;
run;
data funct2;
set getdata;
xsqrt=sqrt(x);   /* raiz cuadrada */
xlog=log(x);     /* log en base e */
xexp=exp(x);     /* exponencial */
```

```
run;

proc print data=funct2;
var x xsqrt xlog xexp;
run;
```

Se pueden usar también funciones para *literales*:

```
data funct3;
set getdata;
c1=upcase(name);       /* convierte a mayusculas */
c2=substr(name,3,8);   /* subcadenas */
len=length(name);      /* longitud de una cadena */
ind=index(name,' ');   /* posicion en una cadena */
fname=substr(name,1,index(name,' '));
lname=substr(name,index(name,' '));
run;

proc print data=funct3;
var name c1 c2 len ind fname lname;
run;
```

Se pueden generar y usar numeros aleatorios. Si se usa una semilla igual a -1, se obtiene cada vez un número aleatorio diferente.

```
data random1;
    x = uniform(-1);
```

```
    y = 50 + 3*normal(-1);
    if x>.5 then coin = 'heads';
      else coin = 'tails';
run;

data random2;
    x = uniform(-1);
    y = 50 + 3*normal(-1);
    if x>.5 then coin = 'heads';
      else coin = 'tails';
run;

proc print data=random1;
  var x y coin;
run;

proc print data=random2;
  var x y coin;
run;
```

A veces, interesa reproducir exactamente la misma secuencia de números aleatorios:

```
data random3;
    x = uniform(123456);
    y = 50 + 3*normal(123456);
    if x>.5 then coin = 'heads';
      else coin = 'tails';
```

```
run;

data random4;
    x = uniform(123456);
    y = 50 + 3*normal(123456);
    if x>.5 then coin = 'heads';
      else coin = 'tails';
run;

proc print data=random3;
  var x y coin;
run;
proc print data=random4;
  var x y coin;
run;
```

Como ejemplo consideramos la generación de la tirada de una moneda 100 veces:

```
data random5;
  do i=1 to 100;
    x = uniform(123456);
    if x>.5 then monea = 'caras';
      else monea = 'cruces';
    output;
  end;
run;
```

```
proc freq data=random5;
  table monea;
run;
```

Macros

El método más eficiente de optimizar recursos en SAS es mediante la escritura de programas, que reciben el nombre de *MACROS*.

Una macro es un programa con sentencias y comandos de SAS que realiza una tarea que se puede repetir varias veces. Para escribir macros es necesario un conocimiento *experto* de SAS.

Ejemplo

```
data credit;
input name $ card_no debt;
datalines;
Stephens 67382910 516.32
Hill 74839222 43.99
Halford 74283848 24.12
Myatt 82843054 88.65
Guerrant 37759831 29.54
;
run;

/* se repite el mismo proceso tres veces
donde cambia la variable de agrupacion */
```

```
proc sort data=credit;
/* por la variable name */
by name;
run;
proc print data=credit;
format debt dollar7.2;
run;

proc sort data=credit;
/* por la variable credit */
by debt;
run;
proc print data=credit;
format debt dollar7.2;
run;

proc sort data=credit;
/* por la variable card_no */
by card_no;
run;
proc print data=credit;
format debt dollar7.2;
run;
```

Pero esto es, cuanto menos, poco *elegante*...

Mejor se podría escribir una *macro*:

```
/* empieza el programa por el comando %macro
y los argumentos de la funcion */

%macro arrange(v=,dat=);

/* Los argumentos de la funci\'{o}n se usan
por referencia poniendo & delante: */

proc sort data=&dat;
by &v;
run;

proc print data=&dat;
format debt dollar7.2;
run;
%mend arrange;
```

El programa *raíz* sería el siguiente:

```
data credit;
input name $ card_no debt;
datalines;
Stephens 67382910 516.32
Hill 74839222 43.99
```

```
Halford 74283848 24.12
Myatt 82843054 88.65
Guerrant 37759831 29.54
;
run;
```

Y las llamadas a la macro serían:

```
%arrange(v=name,dat=credit);
%arrange(v=debt,dat=credit);
%arrange(v=card_no,dat=credit);
```

Gráficos en SAS

Básicos: Scatter plots

Los diagramas de dispersión son el procedimiento gráfico más simple para visualizar las relaciones entre dos variables.

Busca en

`http://en.wikipedia.org/wiki/Scatterplot`

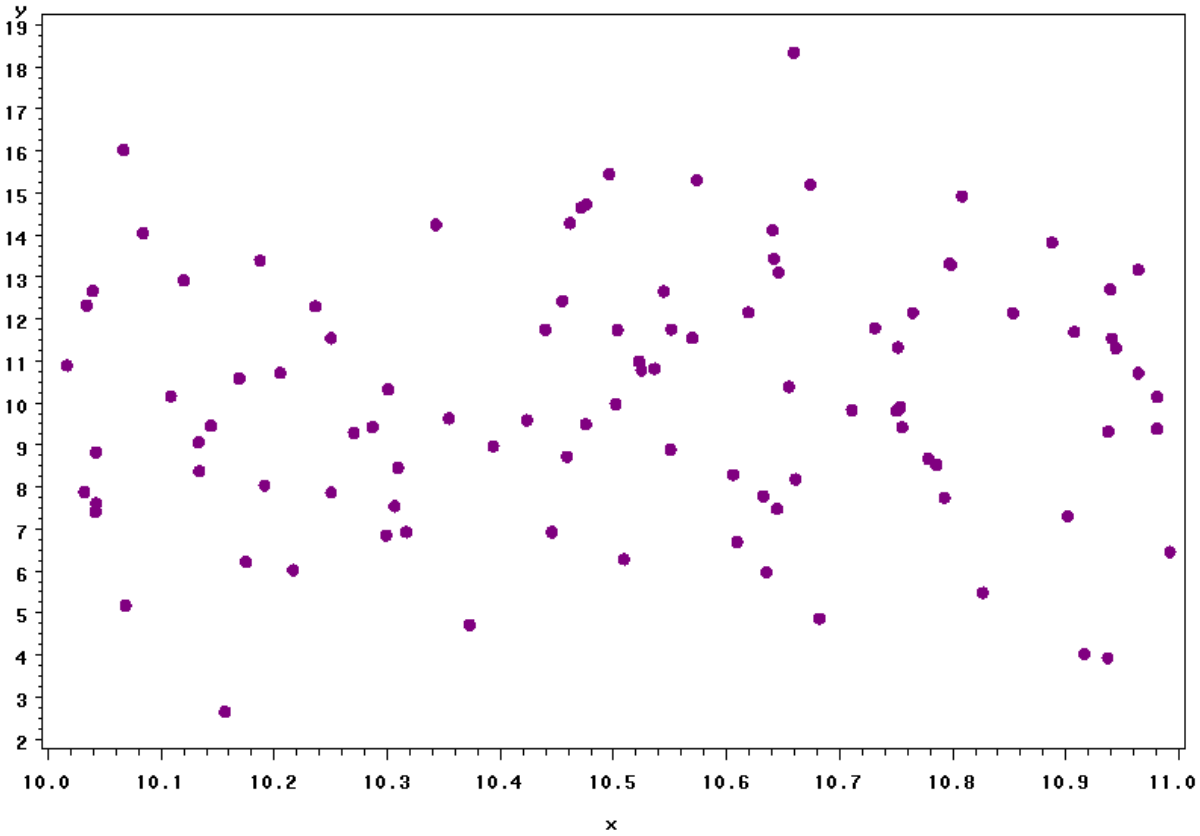
El modo más simple es usar el procedimiento **gplot** para dibujar un gráfico de dispersión de dos variables, x e y por ejemplo. El símbolo por defecto es un signo más, aunque se pueden modificar los parámetros básicos de un gráfico.

El comando `goption` aparece antes de cada gráfico para *resetear* todas las opciones que hay por defecto.

```
data veo;
/* genero 100 numeros aleatorios */
do i=1 to 100;
    x = 10 + uniform(-1);
    y = 10 + 3*normal(-1);
output;
end;

proc print;
run;

/* pongo las opciones graficas */
```

```
goptions reset=all;
/* puntos morados */
symbol value=dot color=purple;

/* goptions device=gif gsfname=output gsfmode=replace;
filename output "c:\pasas"; */
proc gplot data=veo;
plot y*x;
run;
```

Se obtiene



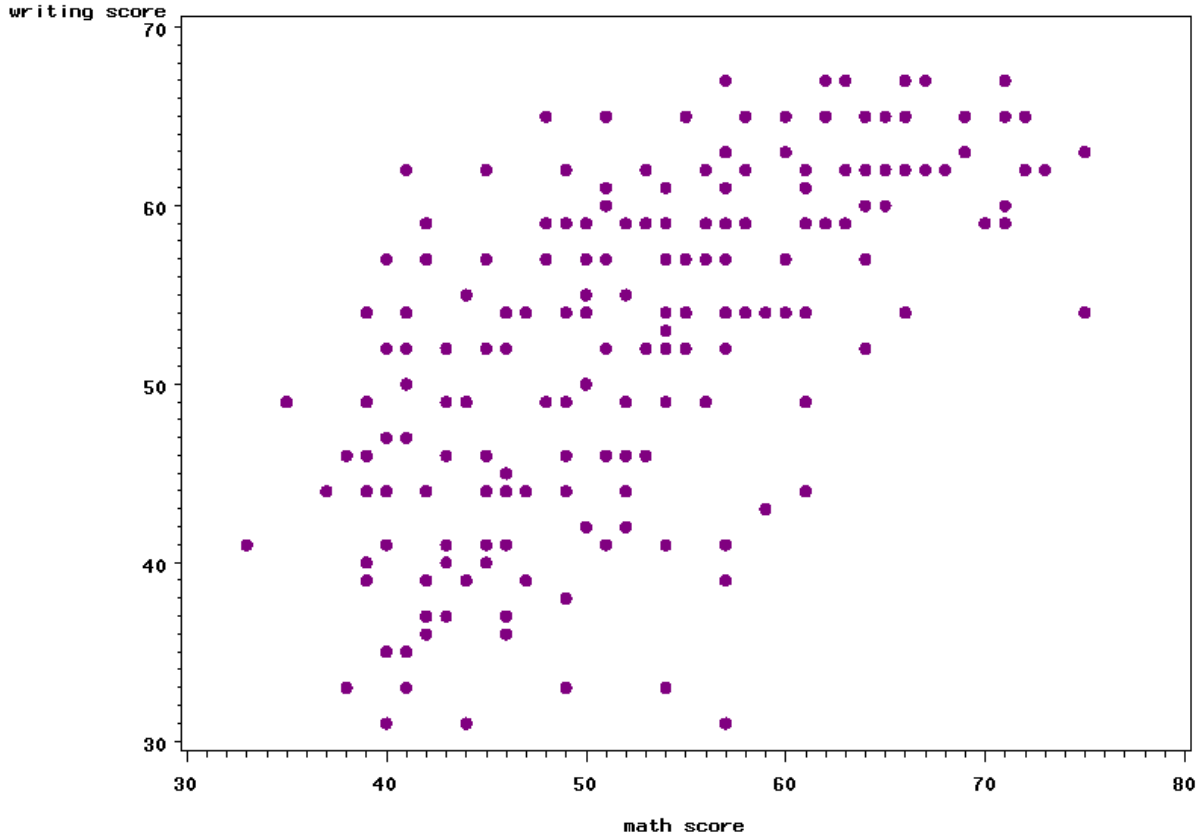
Si se quiere dibujar una línea en lugar de un diagrama de puntos se usa `join` con la opción `i` de *interpolación*.

```
symbol1 i=join;
proc gplot;
plot y * x;
run;
```

Otras posibles variaciones son: `x`, `star`, `square`, `diamond`, `triangle`, `hash`, `dot` y `circle`.

```
options nocenter nodate nonumber;
data hsb2;
  set 'f:\pasas\hsb2';
run;

/* goptions reset=all device=gif gsfname=output gsfmode=replace;
filename output "c:\pasas"; */
goptions reset=all;
symbol value=dot color=purple;
proc gplot data=hsb2;
  plot write*math;
run;
quit;
```

Se obtiene

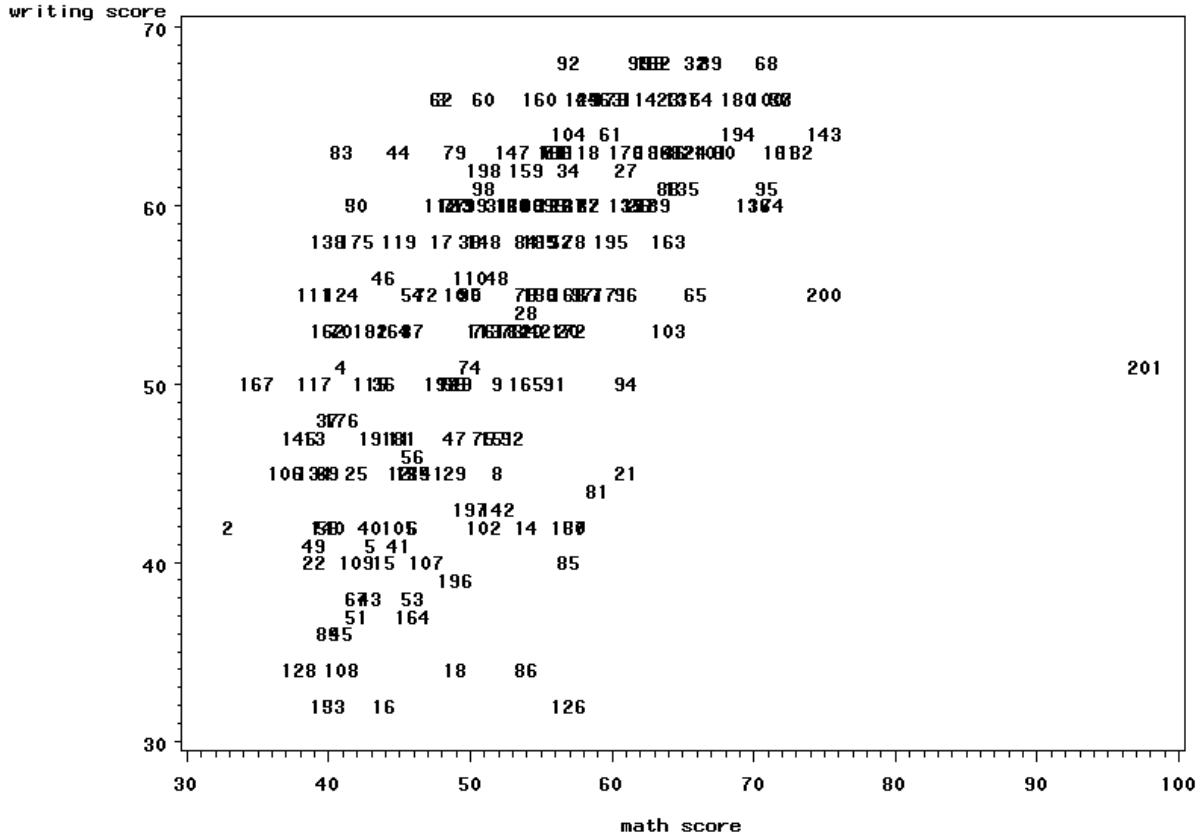


Se pueden modificar los símbolos para usar la variable **id** en lugar de los puntos. Así se pueden detectar outliers.

```
/* simulamos un outlier */
data outlier;
/* lo pongo en la primera fila */
  if _n_ = 1 then do;
  write = 50;
  math = 99;
  id = 201;
```

```
  end;
  output;
  set hsb2;
run;

proc print;
run;
/* uso etiquetas para identificar los otuliers */
goptions reset=all device=gif gsfname=output gsfmode=replace;
filename output "c:\pasas";
** goptions reset=all;
symbol1 pointlabel=("#id") value=none;
proc gplot data=outlier;
  plot write*math;
run;
```

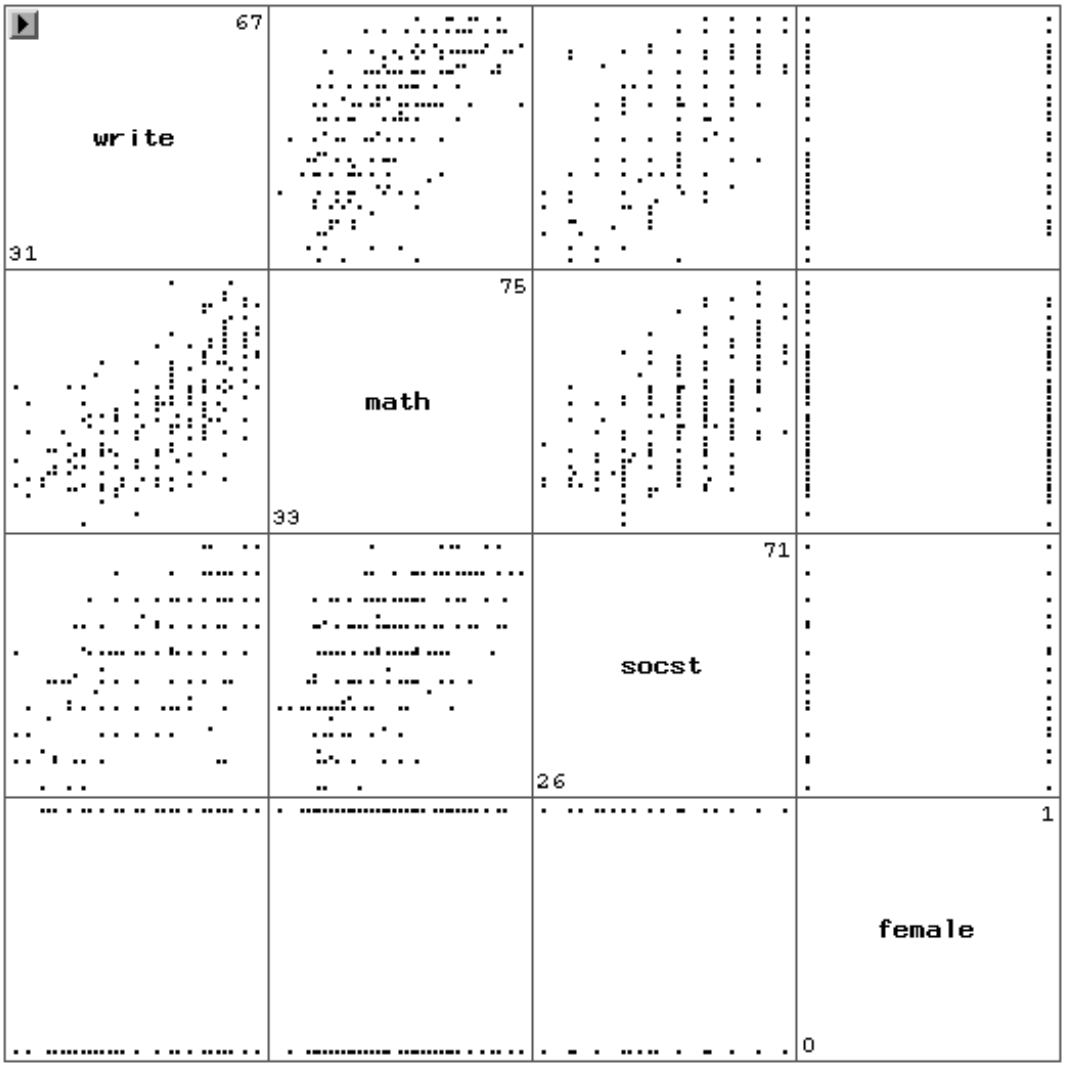
Se obtiene



Cuando se trata de observar las relaciones entre varias variables al mismo tiempo, se hace mediante una matriz de gráficos de dispersión mediante el procedimiento `insight` (si se tiene licencia para el procedimiento).

```
proc insight data=hsb2;
  scatter write math socst female * write math socst female ;
run;
```

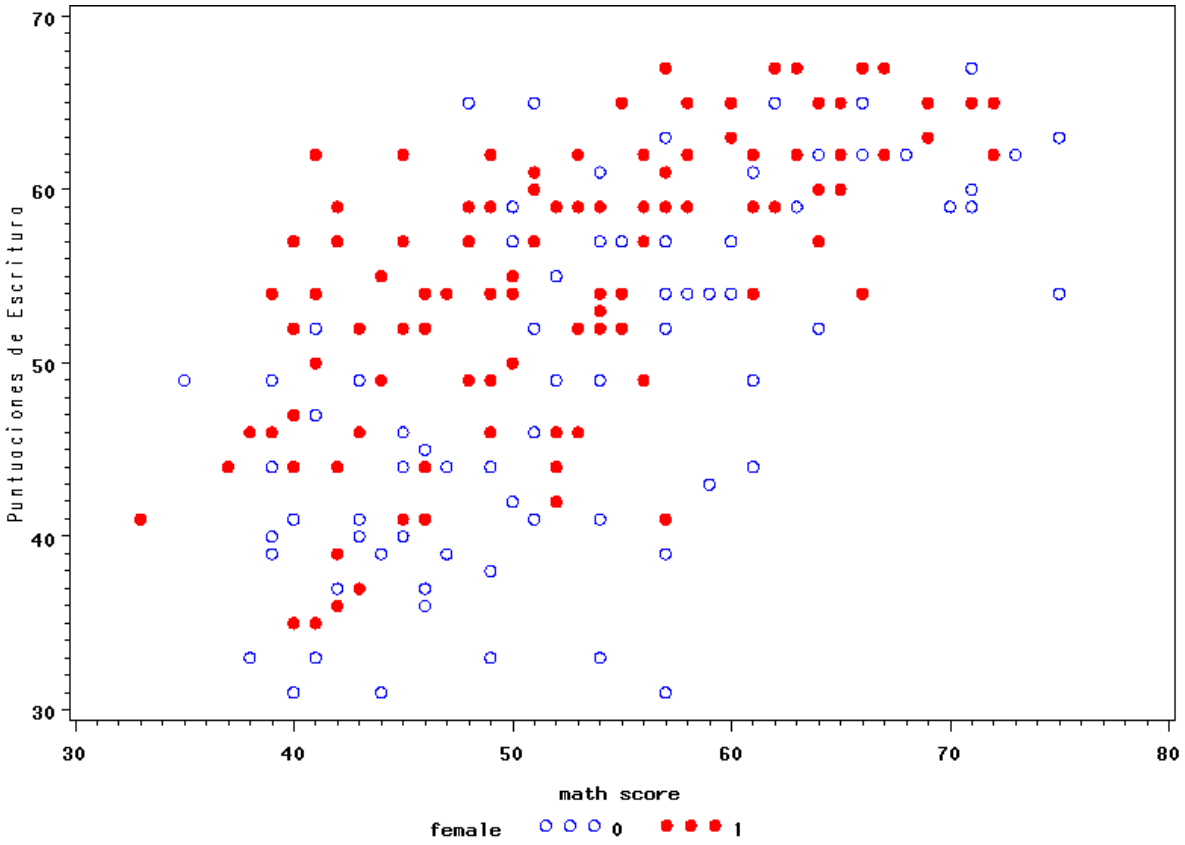
Se obtiene



Se puede definir, también, un diagrama de puntos donde se tienen diferentes símbolos dependiendo de cuál es el sexo de los sujetos. Así se podría comprobar, por ejemplo, si la relación entre `write` y `math` es lineal para cada grupo.

```
options nocenter nodate nonumber;
data hsb2;
```

```
  set 'c:\pasas\hsb2';
run;

**goptions reset=all;
goptions reset=all device=gif gsfname=output gsfmode=replace;
filename output "c:\pasas";
/* hay dos sentencias symbol para cada grupo */
symbol1 value=circle color=blue;
symbol2 value=dot  color=red;

/* se coloca la etiqueta con 90 grados */
axis label = (a=90 'Puntuaciones de Escritura');
proc gplot data=hsb2;
/* la opcion vaxis especifica las opciones segun el
comando axis anterior */

plot write*math=female / vaxis=axis;
run;
quit;
```

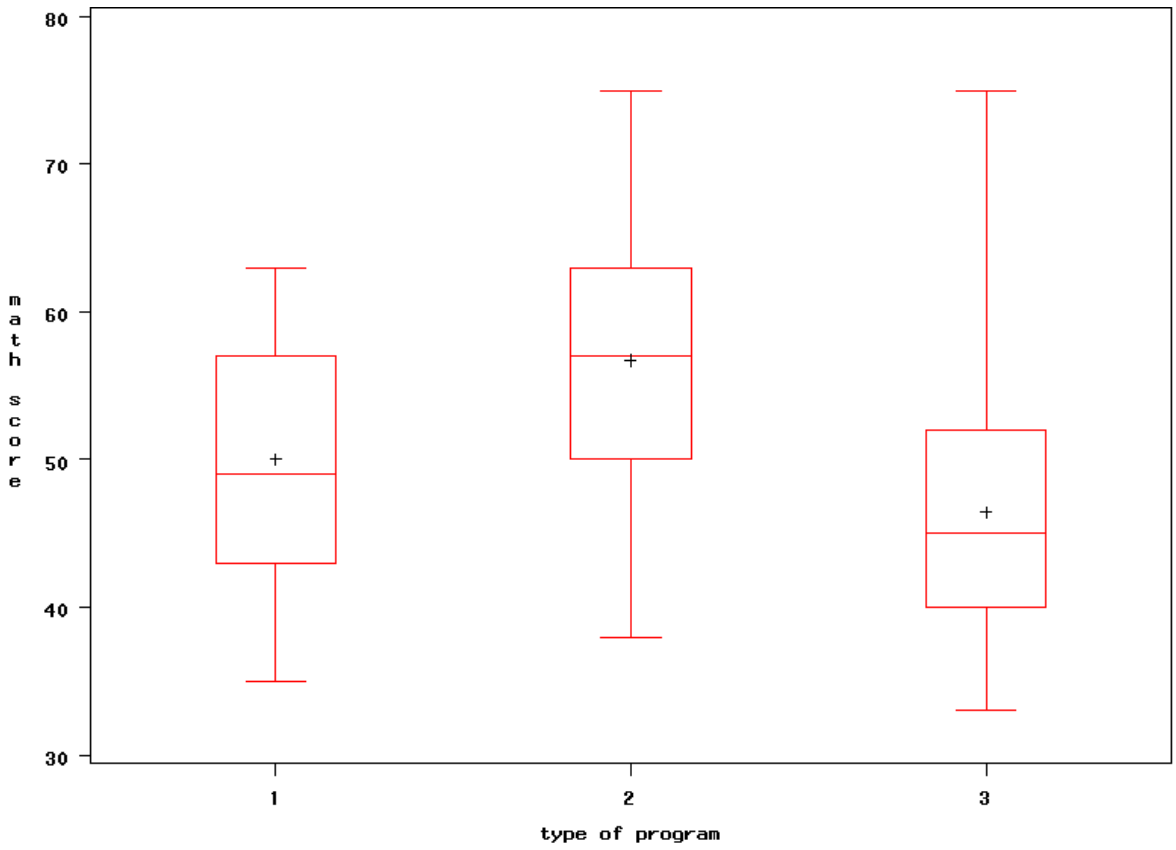


Diagramas de cajas e histogramas

Una manera gráfica de comparar la distribución de distintos grupos es dibujar mediante los diagramas de cajas

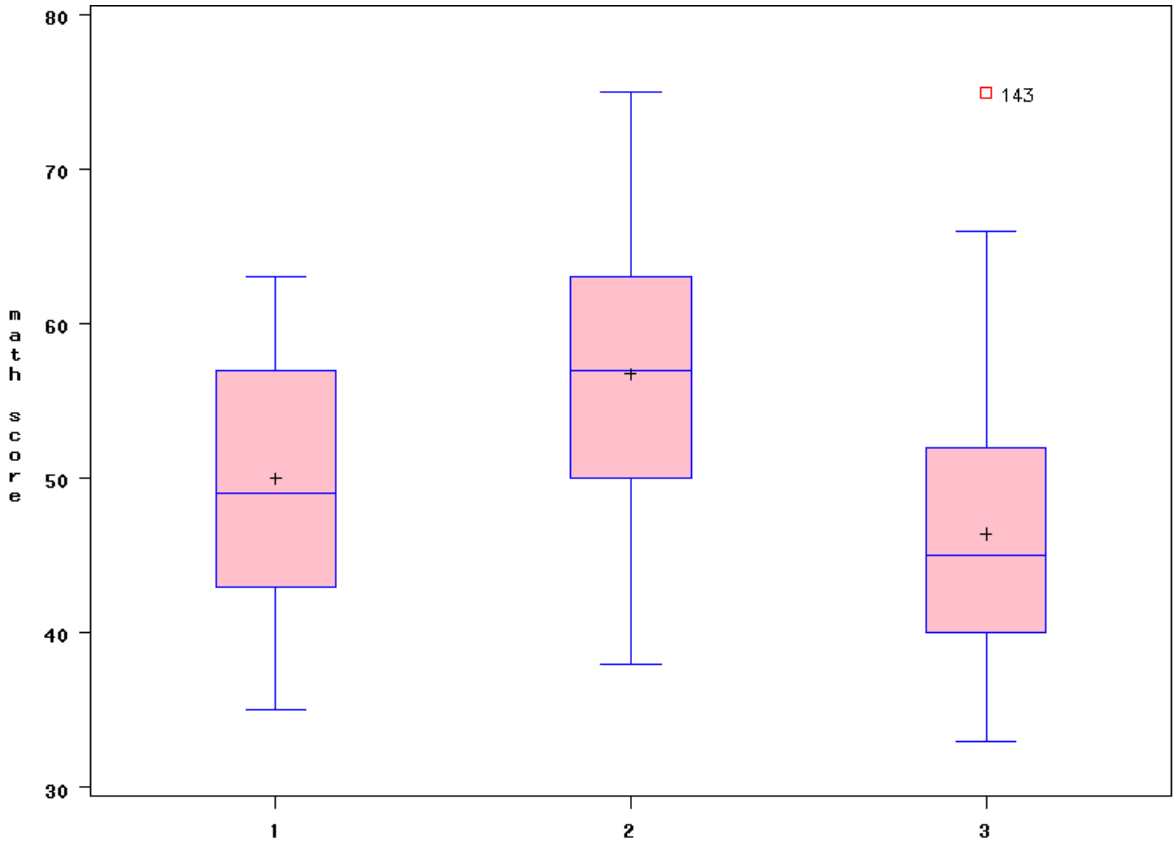
```
options nocenter nodate nonumber;
data hsb2;
  set 'c:\pasas\hsb2';
run;

/* ordenamos previamente los datos segun la variable prog */
```

```
proc sort data=hsb2 out=sort;
 by prog;
run;

**goptions reset=all;
goptions reset=all device=gif gsfname=output gsfmode=replace;
filename output "c:\pasas";

proc boxplot data=sort;
 plot (math socst)*prog / boxwidth=10;
run;
```



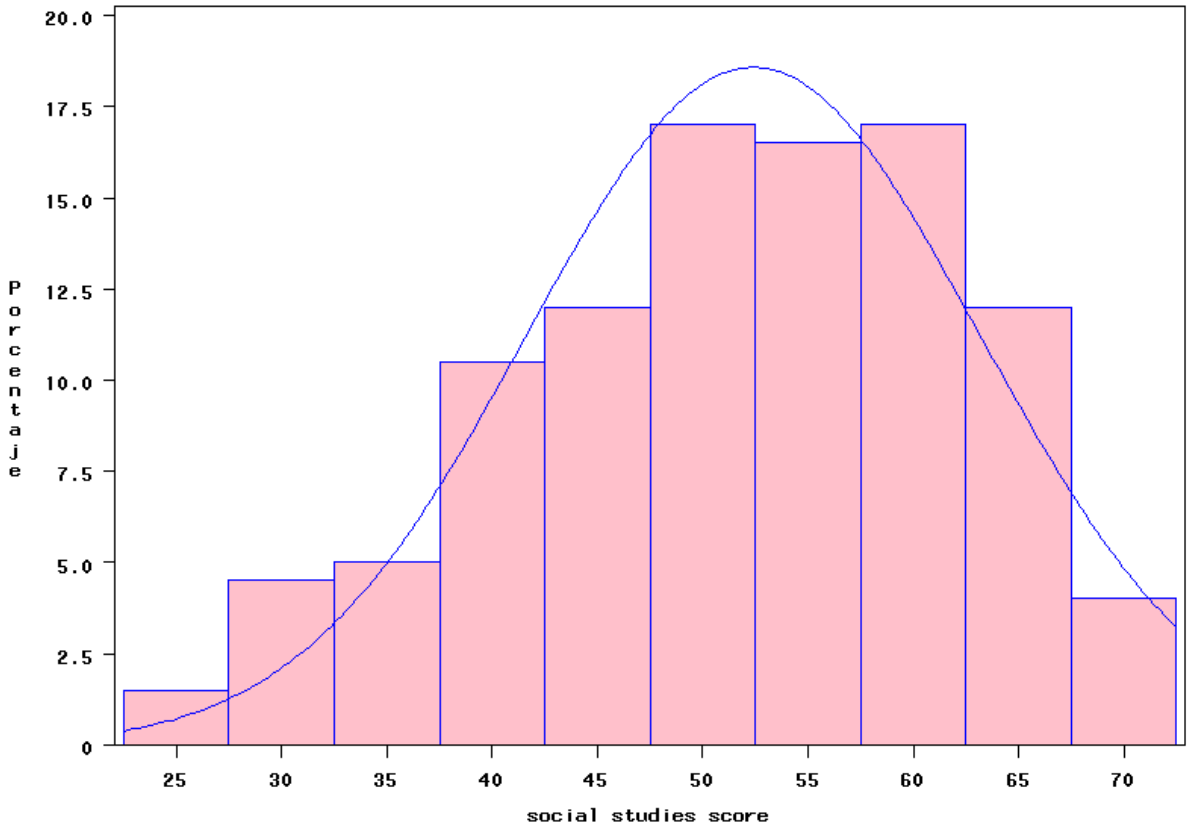
Se puede mejorar la estética de los diagramas de cajas:

```
options nocenter nodate nonumber;
data hsb2;
  set 'c:\pasas\hsb2';
run;

/* ordenamos previamente los datos segun la variable prog */
proc sort data=hsb2 out=sort;
 by prog;
run;

proc boxplot data=sort;

goptions reset=all device=gif gsfname=output gsfmode=replace;
filename output "c:\pasas";
plot (math socst)*prog / boxstyle=schematicid nohlabel
cboxes=blue cboxfill=yellow idcolor=red boxwidth=10;

/* id especifica la variable que se usa para identificar
los outliers */
 id id;
run;
```



Un gráfico que se usa habitualmente en variables continuas son los *histogramas*.

En SAS se obtienen como una opción del procedimiento general `univariate`.

```
options nocenter nodate nonumber;
data hsb2;
  set 'c:\pasas\hsb2';
run;

/* ordenamos previamente los datos segun la variable prog */
proc sort data=hsb2 out=sort;
```

```
 by prog;
run;

goptions reset=all device=gif gsfname=output gsfmode=replace;
filename output "c:\pasas";
proc univariate data=hsb2 noprint;
 histogram socst/ cfill=yellow normal cbarline=blue;
run;
```