

Curso Selectivo 2016

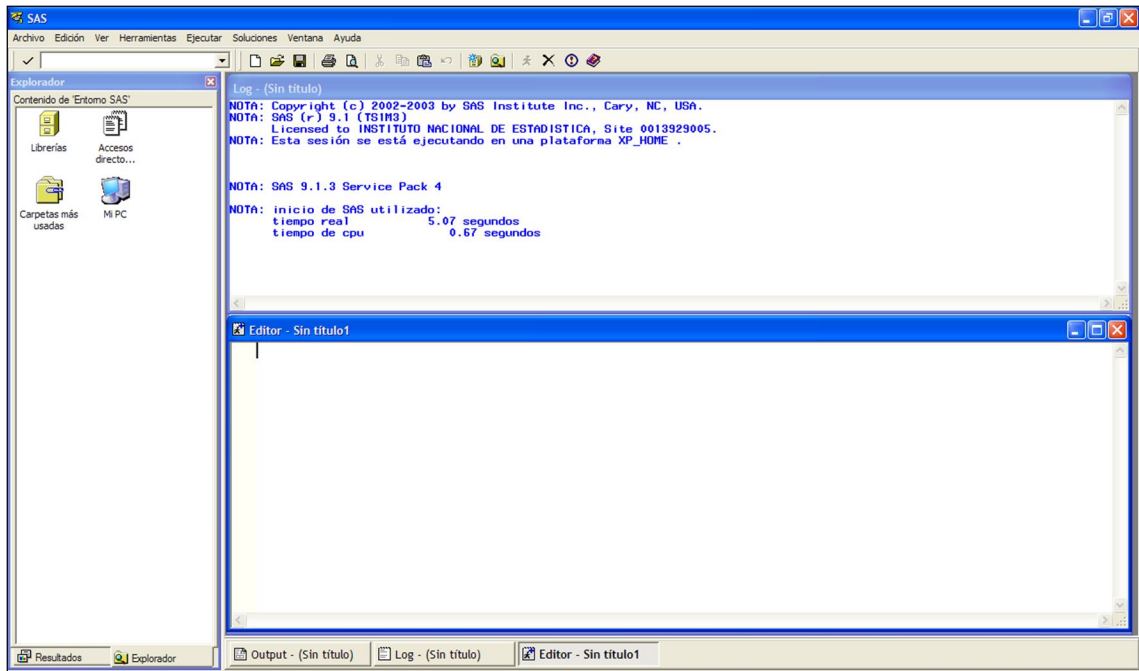
SAS

1	INTRODUCCIÓN SAS	3
1.1	Inicio de una sesión de SAS	3
1.2	Los ficheros SAS	7
1.3	Las librerías SAS	10
2	CONCEPTOS BÁSICOS	14
2.1	Estructura de un programa SAS	14
2.2	Sentencias globales	16
2.3	El paso DATA	17
2.4	Procedimientos SAS o PROC	18
3	DATOS, OPERADORES Y FUNCIONES SAS	19
3.1	Tipos de datos	19
3.2	Operadores	22
3.3	Funciones SAS	23
4	LECTURA DE FICHEROS	28
4.1	Lectura de ficheros planos	28
4.2	Tabla resumen lectura ficheros planos	34
4.3	Lectura de ficheros SAS	35
4.4	Lectura de ficheros EXCEL (xls) usando PROC IMPORT	36
4.5	Lectura de ficheros ACCESS (mdb) usando PROC IMPORT	39
4.6	Lectura de datos desde el mismo programa	41
5	ESCRITURA DE FICHEROS	43
5.1	Escritura de ficheros planos	43
5.2	Tabla resumen escritura ficheros planos	48
5.3	Escritura de ficheros SAS	49
5.4	Escritura de ficheros EXCEL (xls) usando PROC EXPORT	50
5.5	Escritura de ficheros ACCESS (mdb) usando PROC EXPORT	53
6	OPCIONES DE INFILE, SET Y FILE	54
6.1	Opciones INFILE: lectura de ficheros planos (.txt, .dat)	54
6.2	Opciones SET: lectura de ficheros SAS	55
6.3	Opciones FILE: Escritura de ficheros planos (.txt, .dat...)	57
7	EL PASO DATA	58
7.1	Seleccionar y renombrar variables (KEEP, DROP, RENAME)	58
7.2	Cambiar atributos de una variable (ATTRIB)	59
7.3	Retener el valor de una variable (RETAIN)	62
7.4	Sentencias condicionales	63
7.5	Filtrar datos con WHERE	66
7.6	Procesos iterativos (bucle DO)	68
7.7	Concatenar ficheros SAS	70
8	PROC SORT	71
9	UNIÓN DE CONJUNTOS SAS (MERGE)	72
9.1	Sentencia BY. Variables FIRST Y LAST	80
10	OPCIONES GLOBALES. SENTENCIA OPTIONS	84
10.1	Títulos y notas a pie de página	85
11	PROC PRINT	86
12	FORMATOS	87
12.1	Formatos predefinidos por SAS	87
12.2	Formatos definidos por el usuario: PROC FORMAT	89
12.3	Opción notsorted	95
13	PROC FREQ	97
14	PROC SUMMARY / PROC MEANS	100
15	Anexo	104
15.1	Código para generar los datos de los ejemplos	104

1 INTRODUCCIÓN SAS

1.1 Inicio de una sesión de SAS

Pantalla de inicio de una sesión de SAS



Al iniciar una sesión SAS, básicamente, se visualizan dos ventanas:

- Ventana izquierda:
 - Explorador: contiene accesos directos a los ficheros de interés y las librerías propias de SAS.
 - Resultados: ventana de resultados donde aparece la información obtenida de las diferentes ejecuciones.
- Ventana de la derecha: Contiene las ventanas principales; EDITOR, LOG, OUTPUT.


El modo de trabajo en una sesión de SAS se basa prácticamente en estas tres últimas ventanas.

EDITOR: proporciona un editor de texto para desarrollar el programa SAS.

LOG: muestra información sobre la ejecución del programa SAS.

OUTPUT: muestra los listados, tablas y/o resultados de las ejecuciones de pasos PROC y DATA.

La ventana Editor (Avanzado)

Es la ventana en la que se escribe y se procesan los programas SAS. Para poder ejecutar un programa, se debe pulsar el botón: . Para ejecutar una parte de la sintaxis de un programa, primero se selecciona dicha parte y después se pulsa el botón.

Ventana Editor Avanzado

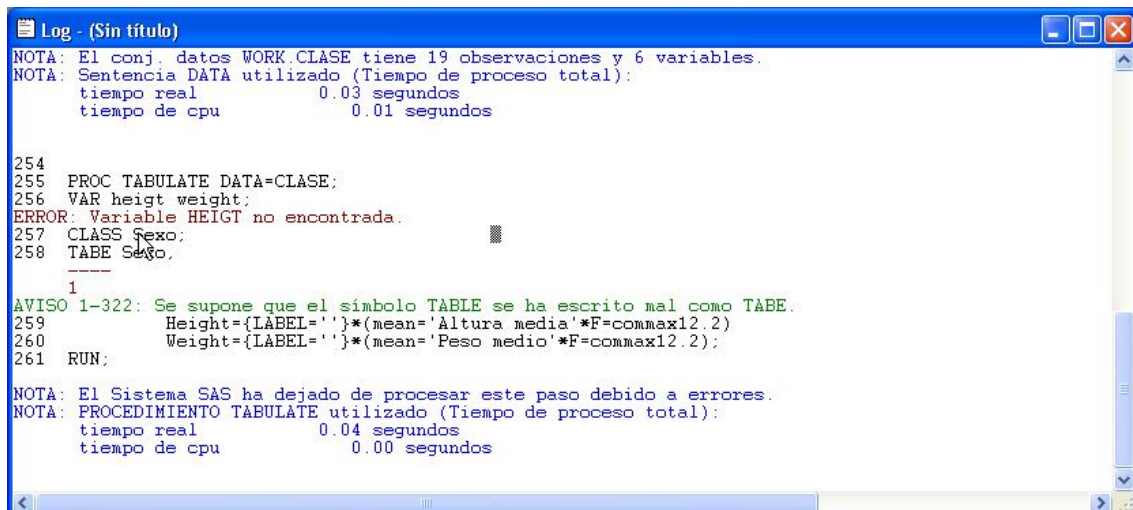


```
Editor - Sin título1 *  
DATA CLASE;  
  SET SASHELP.CLASS;  
  IF Sex='F' THEN Sexo='6';  
  ELSE IF Sex='M' THEN Sexo='1';  
  ELSE Sexo='9';  
RUN;  
  
PROC TABULATE DATA=CLASE;  
  VAR height weight;  
  CLASS Sexo;  
  TABLE Sexo,  
    Height={LABEL="" }*(mean='Altura media'*F=commax12.2)  
    Weight={LABEL="" }*(mean='Peso medio'*F=commax12.2);  
RUN;
```

La ventana LOG

Muestra información sobre la ejecución del programa SAS: ficheros que se leen, número de observaciones y variables que tiene el fichero, mensajes de advertencia y error en caso necesario, así como información sobre el tiempo de ejecución y recursos utilizados. Los errores SAS los señala en color rojo precedidos de la palabra **ERROR:** y los avisos los marca en color verde, precedidos de la palabra **AVISO:**.

Ventana LOG



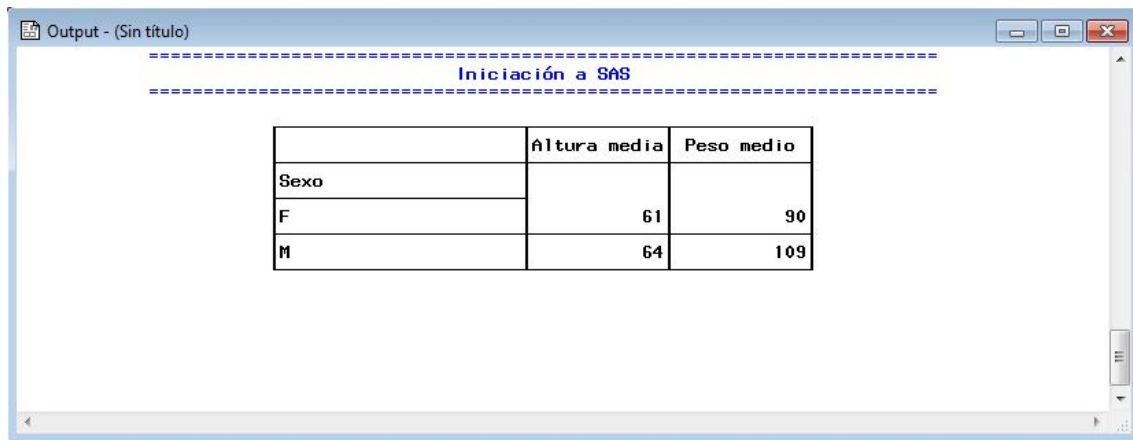
```
Log - (Sin título)  
NOTA: El conj. datos WORK.CLASE tiene 19 observaciones y 6 variables.  
NOTA: Sentencia DATA utilizado (Tiempo de proceso total):  
      tiempo real      0.03 segundos  
      tiempo de cpu    0.01 segundos  
  
254  
255 PROC TABULATE DATA=CLASE;  
256 VAR height weight;  
ERROR: Variable HEIGHT no encontrada.  
257 CLASS Sexo;  
258 TABE Sexo,  
      1  
AVISO 1-322: Se supone que el simbolo TABLE se ha escrito mal como TABE.  
259           Height={LABEL="" }*(mean='Altura media'*F=commax12.2)  
260           Weight={LABEL="" }*(mean='Peso medio'*F=commax12.2);  
261 RUN;  
  
NOTA: El Sistema SAS ha dejado de procesar este paso debido a errores.  
NOTA: PROCEDIMIENTO TABULATE utilizado (Tiempo de proceso total):  
      tiempo real      0.04 segundos  
      tiempo de cpu    0.00 segundos
```

Consejo: Dado que la ventana LOG nunca se vacía por sí sola, sino que va acumulando la información de las sucesivas ejecuciones, es recomendable limpiar dicha ventana antes de cada ejecución. Con esta acción evitaremos mezclar logs de diferentes ocasiones.

La ventana OUTPUT

En la ventana OUTPUT se muestran los listados, tablas y/o resultados de la ejecución de los procedimientos de SAS.

Ventana OUTPUT



	Altura media	Peso medio
Sexo		
F	61	90
M	64	109

En la ventana OUTPUT también se acumulan los datos de las sucesivas ejecuciones por lo que es también conveniente borrarla después de cada ejecución.

Menús

SAS es un programa adaptado para trabajar bajo Windows, de forma que la mayoría de los menús tienen básicamente las mismas opciones que en cualquier otro programa para Windows.

Archivo Edición Ver Herramientas Ejecutar Soluciones Ventana Ayuda

Los menús con las opciones más interesantes son:

- **Archivo:** Nuevo, Abrir, Cerrar, Guardar, Guardar como...
- **Edición:** Deshacer, Cortar, Copiar, Pegar...
- **Ver:** Visualización de ventanas (Log, Output, Editor,...).
- **Herramientas:** Query (se despliega una ventana interactiva que permite explorar bases de datos en formato SAS mediante procedimientos SQL), Editor de tablas SAS, Editor de gráficos...
- **Ejecutar:** Procesar, Recuperar el último proceso (muestra la sintaxis de la última ejecución en el editor de programas).
- **Ayuda:** Ayuda de SAS

Iconos

Como ya es conocido por todos, los iconos representan acciones que se encuentran entre los menús desplegable pero que se utilizan con relativa frecuencia.

Iconos
















De izquierda a derecha representan las siguientes acciones:



Nuevo: para abrir un nuevo programa, log o output en blanco.



Abrir: para abrir un programa o un log o un output guardado anteriormente.

-  Guardar: para guardar el contenido de la ventana activa.
-  Imprimir: para imprimir el contenido de la ventana activa.
-  Vista previa: vista previa del documento a imprimir.
-  Cortar
-  Copiar
-  Pegar
-  Deshacer
-  Librería nueva: crear una nueva librería SAS
-  Explorador de SAS
-  Procesar: para procesar el programa o la líneas seleccionadas del programa.
-  Borrar todo: para borrar el contenido de la ventana activa.
-  Ruptura: para parar el proceso en ejecución.
-  Ayuda: para consultar la ayuda de SAS.

1.2 Los ficheros SAS

Un fichero SAS es un conjunto de información con una característica fundamental: que está dispuesta de una forma especial que SAS puede reconocer. La disposición de los datos en esta forma es muy recomendable cuando se va a trabajar con un paso DATA y absolutamente imprescindible cuando se va a hacer con un paso PROC (las dos estructuras básicas de SAS).

INTRODUCCION 1. Ejemplo de fichero SAS

VIEWTABLE: Work.Censo2011				
	CCAA	Total	Varon	Mujer
1	Andalucía	8343655	4138125	4205530
2	Aragón	1331190	665005	666180
3	Asturias, Principado de	1069275	513990	555285
4	Balears, Illes	1096905	551065	545840
5	Canarias	2078280	1037445	1040835
6	Cantabria	589175	289130	300050
7	Castilla y León	2515755	1250500	1265255
8	Castilla - La Mancha	2092395	1057675	1034720
9	Cataluña	7472935	3701250	3771685
10	Comunitat Valenciana	4990345	2481205	2509135
11	Extremadura	1097695	546830	550865
12	Galicia	2759890	1336460	1423430
13	Madrid, Comunidad de	6387250	3084035	3303215
14	Murcia, Región de	1458250	736125	722125
15	Navarra, Comunidad Foral de	635175	317570	317600
16	País Vasco	2173265	1062375	1110890
17	Rioja, La	319460	159855	159605
18	Ceuta	83185	42515	40670
19	Melilla	80655	41545	39110

Un fichero SAS se compone de dos partes:

- **La parte del descriptor** que contiene información atribuida a los datos.
- **La parte de los datos**, que contiene los valores de los datos en forma de tabla rectangular. Las columnas de la tabla reciben el nombre de variables y las filas observaciones.

INTRODUCCION.2 Esquema de las partes de un Fichero SAS

Fichero SAS

Parte del descriptor	<u>Información general del conjunto de datos</u>			
	* Nombre del fichero	* Etiqueta		
	* Fecha de creación	* Información almacenada		
	* Número de observaciones			
	<u>Información para cada variable</u>			
	* Nombre	* Formato	* Longitud	* Etiquetas
	* Tipo	* Formato de lectura	* Posición	
Parte de los datos	CCAA	TOTAL	VARON	MUJER
	Andalucía	8343655	4138125	4205530
	Aragón	1331190	665005	666180
	Asturias, Principado de	1069275	513990	555285
	Balears, Illes	1096905	551065	545840

Cuando los datos se leen en un fichero SAS, cada fila se convierte en una observación, a cada columna se le da un nombre de variable, y los datos adquieren una **estructura rectangular**.

INTRODUCCION.3 Ejemplo de lectura de un fichero Excel en un fichero SAS (estructura rectangular)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	CCAA	Total	Varon	Mujer									
2	Andalucía	8343655	4138125	4205530									
3	Aragón	1331190	665005	666180									
4	Asturias, Principado de	1069275	513990	555285									
5	Balears, Illes	1096905	551065	545840									
6	Canarias	2078280	1037445	1040835									
7	Cantabria	589175	289130	300050									
8	Castilla y León	2515755	1250500	1265255									
9	Castilla - La Mancha	2092395	1057675	1034720									
10	Cataluña	7472935	3701250	3771685									
11	Comunitat Valenciana	4990345	2481205	2509135									
12	Extremadura	1097695	546830	550865									
13	Galicia	2759890	1336460	1423430									
14	Madrid, Comunidad de	6387250	3084035	3303215									
15	Murcia, Región de	1458250	736125	722125									
16	Navarra, Comunidad Foral de	635175	317570	317600									
17	País Vasco	2173265	1062375	1110890									
18	Rioja, La	319460	159855	159605									
19	Ceuta	83185	42515	40670									
20	Melilla	80655	41545	39110									
21													
22													
23													
24													
25													
26													
27													
28													
29													
30													
31													



VIEWTABLE: Work.Censo2011				
	CCAA	Total	Varon	Mujer
1	Andalucía	8343655	4138125	4205530
2	Aragón	1331190	665005	666180
3	Asturias, Principado de	1069275	513990	555285
4	Balears, Illes	1096905	551065	545840
5	Canarias	2078280	1037445	1040835
6	Cantabria	589175	289130	300050
7	Castilla y León	2515755	1250500	1265255
8	Castilla - La Mancha	2092395	1057675	1034720
9	Cataluña	7472935	3701250	3771685
10	Comunitat Valenciana	4990345	2481205	2509135
11	Extremadura	1097695	546830	550865
12	Galicia	2759890	1336460	1423430
13	Madrid, Comunidad de	6387250	3084035	3303215
14	Murcia, Región de	1458250	736125	722125
15	Navarra, Comunidad Foral de	635175	317570	317600
16	País Vasco	2173265	1062375	1110890
17	Rioja, La	319460	159855	159605
18	Ceuta	83185	42515	40670
19	Melilla	80655	41545	39110

En el caso de encontrarse con datos incompletos (es decir, no hay valores para todas las variables en todas las observaciones) SAS los interpreta como valores ausentes o missing y los representa con un punto, cuando la variable es numérica y con un blanco cuando la variables es alfanumérica. Por ejemplo, las líneas siguientes presentan datos que no conforman una estructura rectangular:

nombre	apellido1	apellido2	usuario
Carlos	Paulogorrán		3
Marisa	Márquez		2
Isabel	García	Gallego	
Patricia	Fernández	Royón	8
Lourdes	Aliste		7
Jara	Poch	García	10

Sin embargo, cuando los datos se leen en un fichero SAS, cada fila se convierte en una observación, a cada columna se le da un nombre de variable, y los datos adquieren una estructura rectangular, pues los valores que faltan se leen como valores ausentes:

INTRODUCCION.4 Valores ausentes numéricos y alfanuméricos en SAS

Un valor numérico ausente se representa con un punto.

	nombre	apellido1	apellido2	usuario
1	Carlos	Paulogorrán		3
2	Marisa	Márquez		2
3	Isabel	García	Gallego	.
4	Patricia	Fernández	Royón	8
5	Lourdes	Aliste		7
6	Jara	Poch	García	10

Un valor alfanumérico ausente se representa con un blanco.

Fichero SAS

(Los puntos representan en SAS valores ausentes, cuando la variable es numérica, y los blancos representan en SAS valores ausentes, cuando la variable es alfanumérica).

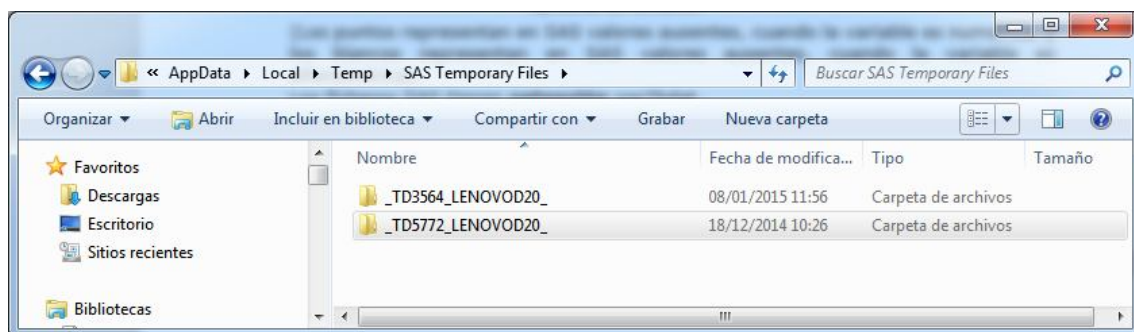
Los ficheros SAS tienen **extensión** sas7bdat .

Los **nombres de los ficheros SAS** deben constar de 1 a 32 caracteres alfanuméricos comenzando por una letra (de la A a la Z) o un subrayado (_) y pudiendo continuar con cualquier combinación de números, letras y subrayados.

Los ficheros SAS por defecto son temporales a no ser que los leamos o escribamos de/en un lugar físico concreto. Los ficheros temporales se guardan en el disco duro de nuestro PC, en general, en el subdirectorio

C:\Users\[Usuario]\AppData\Local\Temp\SAS Temporary Files_XXXX nombre diferente en cada sesión SAS.

INTRODUCCION.5 Pantalla que muestra la ruta y ubicación de los subdirectorios temporales de SAS



NOTA: En ocasiones, como consecuencia de la caída de los servidores, situaciones de reinicio del PC, reseteo, etc., la sesión de SAS no acaba correctamente y los temporales no se borran. Por este motivo, es posible que se vayan almacenando ficheros inservibles en nuestro disco duro que será preciso borrar de manera periódica.

1.3 Las librerías SAS

Una librería SAS es un conjunto de ficheros SAS almacenados en un directorio específico y que SAS reconoce como una unidad.

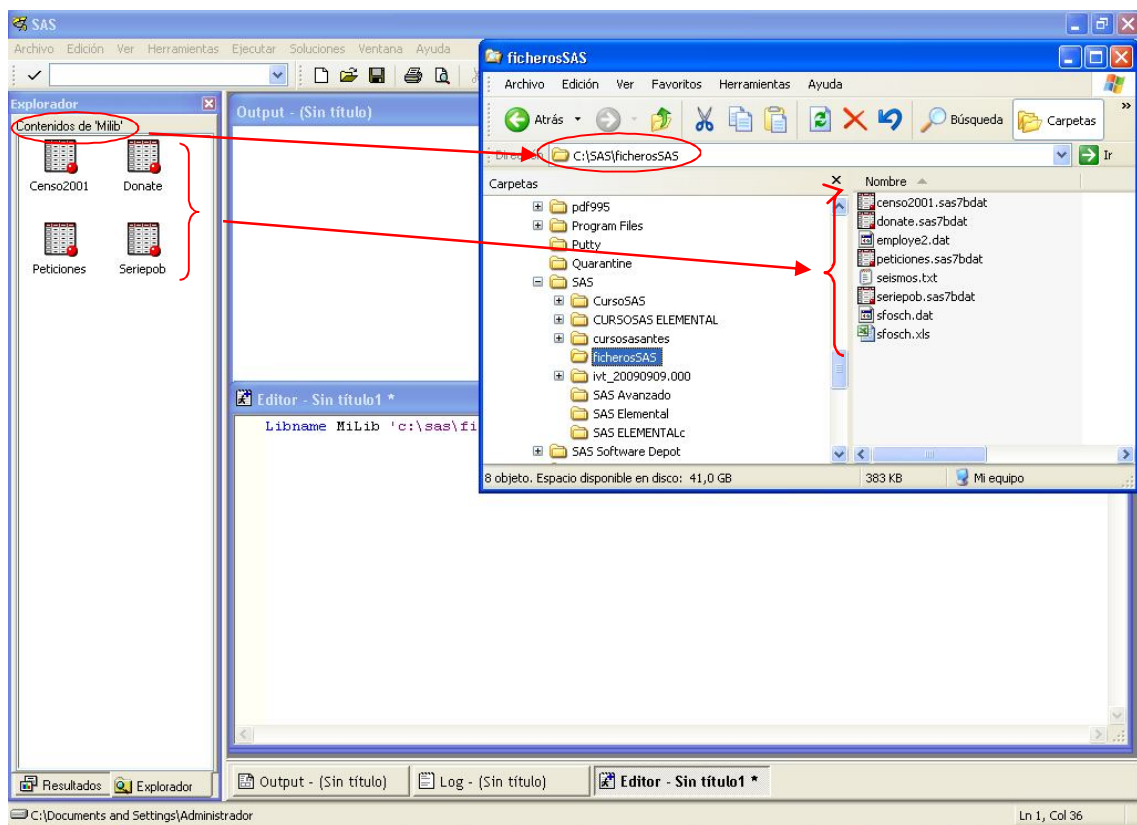
Las Librerías SAS tienen dos nombres: un nombre físico o ruta de ubicación y un nombre lógico llamado Libref.

- ❑ El **nombre físico** debe identificar la localización de los ficheros SAS, es decir la situación de almacenamiento.
- ❑ El nombre lógico o **Libref** es el nombre que el usuario utiliza para indicarle a SAS el directorio donde ha de buscar los ficheros SAS de trabajo. En principio, se trata de un nombre temporal que se asocia a la ruta de ubicación durante una sesión SAS.

MiLib — C:\SAS\ficherosSAS

LIBREF nombre físico o ruta

INTRODUCCION.6 Esquema de asignación librería MiLib y correspondencia con la ruta C:\SAS\ficherosSAS



Crear una librería SAS

Existen dos modos de crear una nueva librería SAS, una mediante la sentencia LIBNAME y otra a través del asistente de SAS.

La sentencia LIBNAME

La sentencia LIBNAME asigna el nombre lógico o libref a la ruta de ubicación de los ficheros SAS.

LIBNAME libref '*nombre físico o ruta*';

Reglas de denominación de librefs:

- debe de tener como máximo 8 caracteres
- debe de empezar con una letra o un subrayado
- el resto de caracteres pueden ser letras, números, o subrayados.

Así, si quisiésemos crear una librería de SAS de nombre CursoSAS y cuya ubicación fuese C:\SAS\CursoSAS. La sentencia LIBNAME correcta sería:

LIBNAME CursoSAS 'C:\SAS\CursoSAS';

IMPORTANTE: A partir de este momento, para crear o hacer referencia a un fichero SAS que pertenezca a esta librería (es decir, a la ubicación C:\SAS\CursoSAS) se utilizará el nombre compuesto: CursoSAS.ficheroSAS.


INTRODUCCION.7 Ejemplo de LIBNAME y referencia a ficheros en librerías



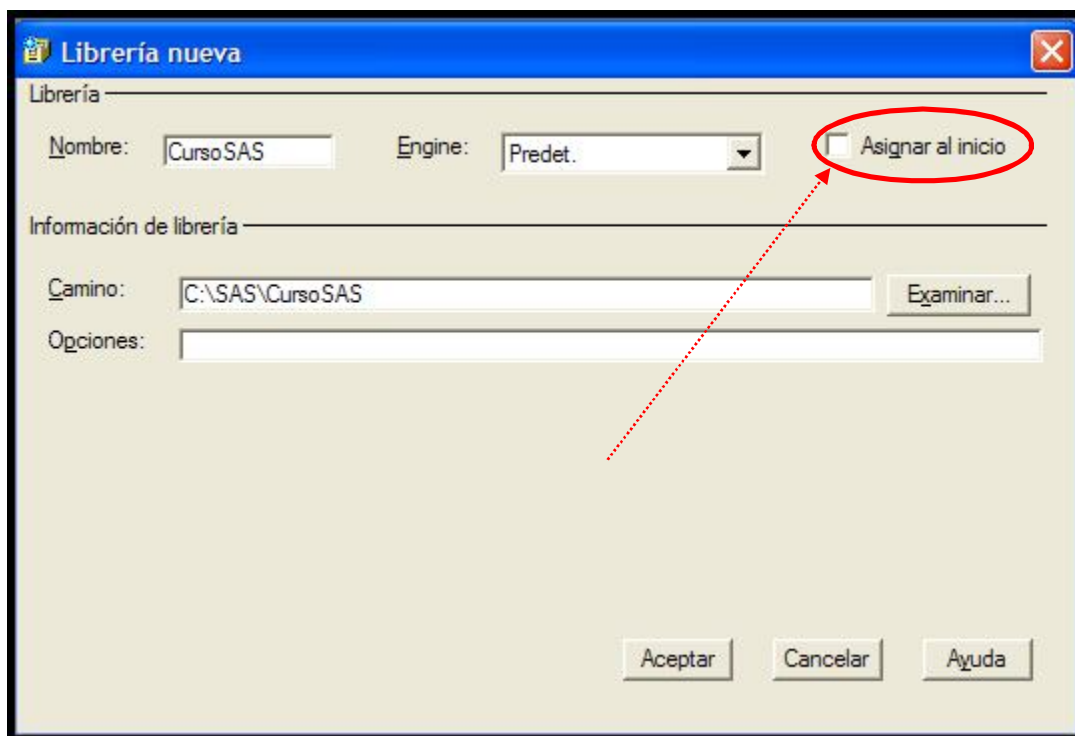
```
LIBNAME CursoSAS 'C:\SAS\CursoSAS';

DATA CursoSAS.CLASE;
  SET SASHELP.CLASS;
  IF Sex='F' THEN Sexo='6';
  ELSE IF Sex='M' THEN Sexo='1';
  ELSE Sexo='9';
RUN;
```

Asistente de SAS para asignar librerías

Pinchando en el sexto icono empezando por la izquierda  aparece el ayudante de SAS para asignar librerías.

Siguiendo con el ejemplo anterior, la creación de la librería CursoSAS ubicada en el directorio 'C:\SAS\CursoSAS' se realizaría de la manera siguiente:



Nota: marcando la opción "Asignar al inicio" la librería CursoSAS se convierte en una librería permanente y no será necesario asignarla de nuevo en sesiones de SAS posteriores.

Las tres librerías que SAS proporciona por defecto son: SASHELP, SASUSER y WORK. Sólo esas tres librerías mencionadas están asignadas por defecto en todas las sesiones y no hace falta hacerlo explícitamente.

LIBRERIA SASHELP

Presente siempre en todas las instalaciones de SAS, contiene un conjunto de elementos con información que se utiliza para controlar diversos aspectos de la sesión SAS. Todos los valores por defecto almacenados en esta Librería son utilizados por todos los usuarios de la organización. Las personalizaciones de cada uno se almacenan en SASUSER.

Esta Librería contiene todas las ventanas y textos de ayuda correspondientes a todos los módulos que estén instalados, así como las definiciones por defecto de las Teclas de Función, información de Copyright y las características por defecto de las principales ventanas de la sesión de SAS.

LIBRERIA SASUSER

Contiene todos los ficheros y elementos que permiten al usuario personalizar las sesiones de SAS para sus necesidades. Si los valores por defecto que se proporcionan en la Librería SASHELP no satisfacen los requisitos del usuario para sus aplicaciones

y su trabajo, puede modificarlos y almacenar sus personalizaciones en la Librería SASUSER. Por ejemplo, la redefinición de Teclas de Función o las nuevas dimensiones de las ventanas pueden salvarse en SASUSER, para ser utilizadas en sesiones o trabajos subsiguientes.

En la mayoría de puestos de trabajo del INE, la librería apunta al subdirectorio C:\Users\Administrador\Documents\My SAS Files\9.3


LIBRERIA WORK

Se trata de una Librería temporal de trabajo que es asignada automáticamente por SAS al comienzo de cada sesión.

Almacena dos tipos de ficheros: todos los que va generando el usuario y guardando en ella, y los generados internamente por SAS en su trabajo habitual. Normalmente, **la Librería WORK es borrada al final de cada sesión de SAS**. Por tanto, todos los ficheros SAS que el usuario pretenda volver a usar en una sesión siguiente deberán ser almacenados en una Librería distinta de la WORK (por ejemplo, la SASUSER o cualquier otra explícitamente definida por el usuario en la sesión mediante LIBNAME).

Por defecto, la librería WORK de SAS apunta a una carpeta, cuyo nombre comienza por _TDxxxx, ubicada en el subdirectorio C:\Users\Administrador\AppData\Local\Temp\SAS Temporary Files¹.

La ventana de LIBRERÍAS ACTIVAS

Pinchando en el icono  de la ventana de SAS accionaremos la ventana Explorer, en la que se nos mostrarán las librerías activas de SAS, las que hayamos definido con su nombre físico y lógico, y librerías por defecto de SAS (SASUSER, SASHELP, WORK).

Dentro de cada librería podremos acceder a los ficheros SAS existentes.

¹ Ver cuadro INTRODUCCION.5

2 CONCEPTOS BÁSICOS

2.1 Estructura de un programa SAS

Un programa SAS es una secuencia de pasos utilizados por el usuario en el proceso de ejecución

Todo programa SAS se compone de:

- **Sentencias globales**
- **PASOS DATA**
- **PASOS PROC**

Los pasos DATA se utilizan para crear conjuntos de datos SAS.

Los pasos PROC se utilizan para procesar conjuntos de datos SAS (crear informes y gráficos, editar datos y clasificar datos).

Las sentencias globales se mantienen activas hasta que se modifican o se cierra la sesión SAS.

Un paso SAS empieza con

- una sentencia DATA
- una sentencia PROC

SAS detecta el final del paso cuando encuentra

- una sentencia RUN (en la mayoría de los pasos)
- una sentencia QUIT (en algunos procedimientos)
- el comienzo de otro paso (sentencia DATA o sentencia PROC)

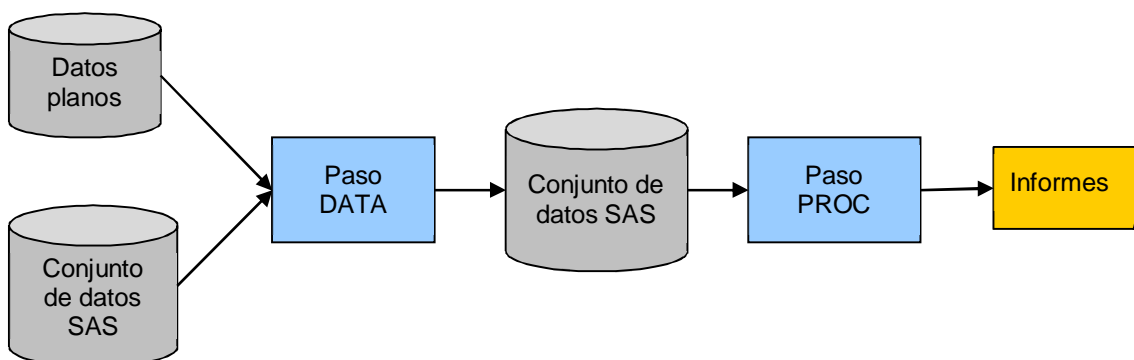
Por supuesto, la terminación de todo programa SAS deberá ser la sentencia RUN o QUIT.

En todos los pasos, en general, existe una entrada y una salida. Un sitio de dónde leer y un sitio en dónde escribir.

Podemos leer ficheros que están físicamente en nuestro disco duro, o en la red, o en un dispositivo externo (lápiz óptico, cd, dvd...), o son ficheros SAS temporales creados durante nuestra sesión SAS. Exactamente igual para la escritura, o salida a la ventana OUTPUT (imprimir algo).

La típica programación SAS consiste en que la salida de un paso sea la entrada del siguiente, o la entrada de unos pasos más adelante.

CONCEPTOS.1: Esquema general programa SAS



CONCEPTOS.2: Programa SAS de ejemplo

```
OPTIONS NOCENTER NODATE NONUMBER LS=132 PS=64 SYSPARM=2001;
TITLE1 '=====';
TITLE2 "Curso SAS ELEMENTAL:
      Datos del Censo de población y viviendas &sysparm";
TITLE3 '=====';

/*ASIGNAMOS librería*/
LIBNAME CursoSAS 'C:\SAS\CursoSAS';

/*IMPORTAMOS datos desde un fichero EXCEL */
PROC IMPORT OUT= WORK.censo2001
      DATAFILE= "C:\SAS\CursoSAS\censo2001_CCAA
      residencia.xls"
      DBMS=EXCEL REPLACE;
      SHEET="Personas$"; *Hoja del libro;
      GETNAMES=YES;
RUN;
```

Opciones globales

Paso PROC

```
/*ETIQUETAMOS y damos FORMATO a las variables*/
DATA CENSO2001;
      SET CENSO2001;
      FORMAT      TOTAL COMMAX12. VARON COMMAX12. MUJER COMMAX12.;
      LABEL      CCAA='CC.AA. de Residencia'
      PROVINCIA='PROVINCIA de Residencia'
      TOTAL='Total' VARON='Varón' MUJER='Mujer';
RUN;
```

Paso DATA

```
/*PROC TABULATE: agregamos el número de personas por CCAA*/
PROC TABULATE DATA=CENSO2001 ORDER=DATA FORMCHAR(1,2,7)='| -+'
OUT=CURSOSAS.CENSO2001;
CLASS CCAA;
VAR TOTAL VARON MUJER;
TABLE CCAA='CC.AA. de Residencia',
      (SUM*TOTAL*F=COMMAX12.)(SUM*VARON*F=COMMAX12.)(SUM*MUJER*F=COMMAX12.
      );
KEYLABEL SUM='SUMA';
RUN;
```

Paso PROC

A la vista de este ejemplo podemos determinar que los comentarios comienzan por `*` y finalizan en `;`

Si en un momento determinado queremos poner un grupo de sentencias como comentario, basta con iniciarlo con `/*` y finalizarlo con `*/`.

2.2 Sentencias globales

Las sentencias globales son sentencias independientes de los pasos DATA y PROC. No corresponden a ninguno de ellos aunque son utilizadas por ellos.

Se utilizan para determinar librerías SAS, asignar nombres lógicos a físicos de ficheros, asignar títulos a salidas de la ventana OUTPUT, determinar las opciones globales (GLOBAL OPTIONS), etc.

Se mantienen activas durante toda la sesión SAS hasta que se modifiquen o anulen, y no es necesario volver a ejecutarlas si no se desea.

Las que veremos a lo largo del curso son:

- FILENAME
- LIBNAME
- OPTIONS
- TITLE
- FOOTNOTE

2.3 El paso DATA

El paso DATA es el método principal para la generación de ficheros.

Un paso DATA es un conjunto de sentencias SAS que comienza con una sentencia DATA. A la sentencia DATA le siguen sentencias de programación que realizan las manipulaciones necesarias para construir el fichero SAS. El paso DATA puede utilizarse para estos propósitos:

- Recuperación, es decir, recopilación de datos de entrada en un fichero SAS.
- Edición, es decir, búsqueda y corrección de errores en unos datos de entrada.
- Cálculo de valores de nuevas variables.
- Generación de informes y creación de ficheros.
- Creación de nuevos ficheros SAS a partir de otros ya existentes mediante mezclas, subconjuntos o actualizaciones de los viejos ficheros.
- Manipulación de datos, es decir, reestructuración de los datos de un fichero SAS para su uso por un PROC.

Ejemplo:

```
DATA salidaSAS;
SET entradaSAS;
LENGTH CCAA $30;
  IF COD_CCAA='01' THEN CCAA='Andalucía';
  IF COD_CCAA='02' THEN CCAA='Aragón';
  IF COD_CCAA='03' THEN CCAA='Asturias (Principado de)';
  IF COD_CCAA='04' THEN CCAA='Balears (Illes)';
  IF COD_CCAA='05' THEN CCAA='Canarias';
  IF COD_CCAA='06' THEN CCAA='Cantabria';
  IF COD_CCAA='07' THEN CCAA='Castilla y León';
  IF COD_CCAA='08' THEN CCAA='Castilla-La Mancha';
  IF COD_CCAA='09' THEN CCAA='Cataluña';
  IF COD_CCAA='10' THEN CCAA='Comunidad Valenciana';
  IF COD_CCAA='11' THEN CCAA='Extremadura';
  IF COD_CCAA='12' THEN CCAA='Galicia';
  IF COD_CCAA='13' THEN CCAA='Madrid (Comunidad de)';
  IF COD_CCAA='14' THEN CCAA='Murcia (Región de)';
  IF COD_CCAA='15' THEN CCAA='Navarra (Comunidad Foral de)';
  IF COD_CCAA='16' THEN CCAA='País Vasco';
  IF COD_CCAA='17' THEN CCAA='Rioja (La)';
  IF COD_CCAA='18' THEN CCAA='Ceuta';
  IF COD_CCAA='19' THEN CCAA='Melilla';
RUN;
```

NOTA: durante el curso veremos un grupo de sentencias que son exclusivas del paso DATA y sólo podrán ser utilizadas en dicho paso.

2.4 Procedimientos SAS o PROC

Los Procedimientos SAS o pasos PROC son subprogramas compilados que están almacenados en SAS. Se invocan utilizando el paso PROC.

Estos subprogramas están diseñados para ejecutar una tarea o conjunto de tareas, y pueden aceptar modificaciones impuestas por el usuario para controlar diversas opciones de ejecución (tales como realizar los cálculos separadamente en subgrupos de observaciones, o bien guardar la salida del programa o mandarla a impresora).

Hay una serie de Procedimientos SAS, los más populares, que vienen incluidos en el módulo BASE de SAS. Vienen, por tanto, documentados en los manuales relativos a este módulo. El resto de los módulos tienen también sus PROC específicos, que están documentados en los manuales respectivos de dichos módulos.

Ejemplo:

```
PROC TABULATE DATA=CENSO2001;  
  CLASS CCAA;  
  VAR TOTAL VARON MUJER;  
  TABLE CCAA='CC.AA. de Residencia',  
    (SUM*TOTAL*F=COMMAX12.)(SUM*VARON*F=COMMAX12.)  
    (SUM*MUJER*F=COMMAX12.);  
  KEYLABEL SUM='SUMA';  
RUN;
```

NOTA: Durante el curso veremos un grupo de sentencias que son exclusivas del paso PROC y sólo podrán ser utilizadas en dicho paso. En general cada PROC tiene las suyas exclusivas aunque pueden ser comunes entre ellos.

3 DATOS, OPERADORES Y FUNCIONES SAS

3.1 Tipos de datos

Para SAS sólo existen dos tipos de datos o variables: **numéricos** y **alfanuméricos** o carácter.

Al ser SAS un paquete estadístico distingue entre valores nulos y valores ausentes.

- En el caso de numéricos los nulos son 0, mientras que los ausentes o missing se representan con .
- En el caso de valores alfanuméricos o carácter, los nulos y ausentes son iguales, equivalen al blanco.

Variables numéricas

- Son las variables que representan números, son medidas de cada observación de nuestro conjunto de datos SAS. Dentro de las numéricas se incluyen las variables de fecha y hora.
- De forma predeterminada se almacenan como números en coma flotante en 8 bytes. Esto proporciona un espacio para 16 o 17 dígitos significativos.
- Los missing numéricos tienen un comportamiento especial y debemos tener especial cuidado al trabajar con variables numéricas que puedan contener valores perdidos:
 - Un valor missing que opera con un número da como resultado un valor missing.
 - Si empleamos una función de SAS, como es min() para determinar el mínimo entre dos valores o sum() para determinar la suma, el valor missing no es tenido en cuenta.

Variables alfanuméricas

- Las variables alfanuméricas son aquellas que almacenan caracteres. Pueden contener cualquier valor: letras, números, caracteres especiales y blancos. Los valores alfanuméricos se almacenan con una longitud de 1 a 32.767 bytes. Un byte equivale a un carácter.
- Siempre que en un programa SAS aparezca una constante de tipo alfanumérica o carácter deberá ir entre comillas, bien simples o dobles. Si se abre con comilla simple deberá cerrarse con simple, e igualmente para las comillas dobles.
- Si en la compilación de un paso, SAS encuentra unas comillas impares (por ejemplo se abren y no se cierran), piensa que todo el programa es una constante alfanumérica y si llegase hasta la sentencia RUN final de programa queda en suspenso esperando la finalización de la comilla.
- El operador fundamental para las variables alfanuméricas es la concatenación, para ello SAS emplea ||².

² Está en la tecla 1.

Formatos ³

Antes de pasar a ver un tipo especial de datos numéricos en SAS, las variables fecha, daremos una noción básica de lo que es un formato.

Un formato es una característica o atributo de la variable que se utiliza para visualizar sus valores de una manera determinada.

Por ejemplo, si nuestra variable recoge la población por Provincia, podremos aplicarle el formato definido por SAS *commax12.* para visualizar las cantidades con separación de miles por puntos.

Datos.1: Visualización de una variable con formato

	cod_prov	PROVINCIA	TOTAL	Hombres	Mujeres
1	04	Almería	536.731	272023	264708
2	11	Cádiz	1.116.491	552463	564028
3	14	Córdoba	761.657	372464	389193
4	18	Notar las diferencias de visualización entre la población Total y población Hombres	821.660	401638	420022
5	21		462.579	229013	233566
6	23		643.820	317343	326477
7	29	Málaga	1.287.017	630902	656115

Los formatos, en general, se especifican en SAS de acuerdo a la siguiente estructura:

Formatos de tipo carácter: **\$FORMAT**w.

Formatos de tipo numérico: **FORMAT**w.d

- El símbolo \$ indica que el formato es de tipo carácter.
- FORMATO se refiere a un nombre de formato a veces opcional para SAS.
- w indica la anchura de la variable.
- d indica el número de decimales

Todos los formatos deben terminar con un punto (.) para que SAS pueda diferenciar un formato de una variable.

Hay formatos definidos por SAS pero también un usuario podrá definir sus propios formatos utilizando el procedimiento PROC FORMAT.

Variables fecha

- Un tipo muy importante de **variable numérica** es la variable que representa una **fecha u hora en SAS**.
- La variable fecha se codifica internamente como la diferencia en días entre la fecha en cuestión y el 1 de enero de 1960 de modo que las fechas posteriores serán un número positivo y las anteriores a 1960 serán un número negativo. Igualmente las variables hora serán la diferencia en horas con el 1 de enero de 1960 a las 00:00.
- El formato es uno de los atributos más importantes de las variables y en función de él podremos ver correctamente los resultados. Los formatos fecha/hora más empleados son:

³ En un apartado posterior de este se da una explicación detallada del concepto de formato y el PROC FORMAT.

Datos.2: Cuadro de formatos fechas más utilizados

DATEw.	DATE9.	ddMESyyyy
	DATE7.	ddMESyy
(MES=JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC)		
DDMMYYw.	DDMMYY6.	ddmmyy
	DDMMYY8.	dd/mm/yy
	DDMMYY10.	dd/mm/yyyy
MMDDYYw.	MMDDYY6.	mmddyy
	MMDDYY8.	mm/dd/yy
	MMDDYY10.	mm/dd/yyyy
DATETIMEw. Formato para fecha-hora		

Analicemos con algunos ejemplos cómo codifica SAS las variables fecha:

Ejemplo1: Supongamos que partimos del conjunto de datos SAS work.fechas con tres variables; una alfanumérica y dos numéricas que recogen los mismos valores de fecha.

Fichero SAS WORK.fechas

	id	fecha	fechaFMT
1	id1	0	0
2	id2	1	1
3	id3	2	2
4	id4	3	3
5	id5	17861	17861

Si a una de esas dos variables numéricas, le asignamos un formato fecha **ddmmyy10.** y a la otra no mediante el siguiente código:

```
DATA formatoFecha;
SET work.fechas;
FORMAT fechaFMT ddmmyy10.;
RUN;
```

El resultado sería un fichero como el siguiente:

Fichero SAS WORK.formatoFecha

	id	fecha	fechaFMT
1	id1	0	01/01/1960
2	id2	1	02/01/1960
3	id3	2	03/01/1960
4	id4	3	04/01/1960
5	id5	17861	25/11/2008

3.2 Operadores

Los operadores matemáticos fundamentales que se pueden emplear con las variables numéricas son:

**	Exponencial
*	Multipliación
/	División
+	Suma
-	Resta

Los operadores lógicos fundamentales que se pueden emplear son:

AND	Y
OR	O
NOT	NO

Para comparar valores de variables numéricas o alfanuméricas se utilizan los siguientes operadores:

= , EQ	Igual
< , LT	Menor
> , GT	Mayor (Greater Than
<= , LE	Menor o igual
>= , GE	Mayor o igual
^= , NE	Distinto

LT: Less Than – **GT:** Greater Than
LE: Less or Equal - **GE:** Greater or Equal

Operadores especiales

IN, NOT IN – si una variable se encuentra dentro de una lista de valores.

Ejemplo: CCAA IN ('Andalucía', 'Aragón', 'Asturias (Principado de)');

|| - concatenación

Cualquier operación entre valores numéricos donde uno de ellos es missing da como resultado un valor missing y siempre es menor que cualquier cosa.

SAS admite sintaxis del tipo `valor <= variable <= valor` sin necesidad de utilizar los operadores lógicos.

3.3 Funciones SAS

Una función SAS es una rutina que, tras recibir uno o más argumentos de entrada, devuelve un valor determinado.

Forma general de una función SAS:

funcion(argumento, argumento...)

Cada argumento, en el caso de existir más de uno, está separado de los otros por una coma.

La mayoría de las funciones aceptan argumentos que sean:

- Constantes
- Variables
- Expresiones
- Funciones

Las funciones pueden llevar;

- Muchos argumentos en cualquier orden
- Un número específico de argumentos en un orden fijo.

En las funciones que lleven muchos argumentos, en cualquier orden, se puede acortar la lista de argumentos utilizando la palabra clave OF seguida de la lista de variables. Ejemplo: SUM(x1,of x10-x20, x32); o SUM(of x:);

Funciones de variables numéricas

ABS(argumento) Valor Absoluto

SQRT(argumento) Raíz Cuadrada

ROUND(argumento) Redondear //

ROUND(argumento, precisión) Redondear con cierta precisión (la precisión es una potencia de 10, por ejemplo: 10, 1, 0.1, 0.01, etc.)

Ej: ROUND (22.456 , 1) = 22
ROUND (22.456) = 22
ROUND (22.456 , .1) = 22.5
ROUND (22.456 , .01) = 22.46
ROUND (22.456 , 10) = 20
ROUND (4.5)=5

INT(argumento) Devuelve la parte entera del argumento

EXP(argumento) Exponencial

LOG(argumento) Logaritmo

SUM(argumento1,argumento2,...,argumentoN)* Suma de argumentos.

Se puede acortar la lista de argumentos utilizando la palabra clave OF seguida de la lista de variables. Ejemplo: SUM(x1,of x10-x20, x32); o SUM(of x:); o SUM(of z--x); si se quiere sumar todas las variables comprendidas entre x1 e y3.

MEAN(argumento1,argumento2,...,argumentoN) * Media de argumentos

MIN(argumento1,argumento2,...,argumentoN)* Valor mínimo

MAX(argumento1,argumento2,...,argumentoN) * Valor máximo

VAR(argumento1,argumento2,...,argumentoN) * Varianza de los argumentos

STD(argumento1,argumento2,...,argumentoN) * Desviación estándar de los argumentos

** Las funciones seleccionadas admiten muchos argumentos en cualquier orden e ignoran los valores missing.*

Funciones de variables carácter

SAS dispone de un gran número de funciones para el tratamiento de variables carácter, de todas ellas destacan:

UPCASE(argumento) Convierte todas las letras del argumento a mayúsculas

Ejemplo: **UPCASE('MARia peREZ')**='MARIA PEREZ'

LOWCASE(argumento) Convierte todas las letras del argumento a minúsculas

Ejemplo: **LOWCASE('Maria peREZ')**='maria perez'

PROPCASE(argumento) Convierte en mayúscula la primera letra de cada palabra del argumento.

Ejemplo: **PROPCASE('Maria peREZ')**='Maria Perez'

INDEX(argumento, cadena) Localiza en un argumento una cadena y devuelve su localización. Si no localiza la cadena devuelve un 0.

Ejemplo:

INDEX('María Pérez Galván','í')=4

INDEX('María Pérez Galván','áéíóú')=0

INDEX('María Pérez Galván','Pérez')=7

INDEXC(argumento, cadena1, cadena2,...) o **INDEXC(argumento, lista de valores)**

Localiza, en el primer argumento, alguna de las cadenas especificadas y devuelve la primera localización. Si no localiza ninguna de las cadenas devuelve un 0.

Ejemplo:

INDEXC('María Pérez Galván','áéíóú')=4

INDEXC('María Pérez Galván','á','é','í','ó','ú')=4

COMPRESS(argumento) Elimina caracteres específicos de una cadena.

Ejemplo: **COMPRESS(' (908) 777-1234', ' (-)')**='9087771234'

Si se escribe como tercer argumento de COMPRESS una k, la lista de caracteres especificada como segundo argumento corresponde a la lista de caracteres permitidos eliminándose todos aquellos caracteres no contemplados en dicha lista. Así: **COMPRESS(' (908) 777-1234', '1234567890', 'k')**='9087771234'

COMPBL(argumento) Reemplaza todas las ocurrencias de dos o más blancos con un único blanco. Esta función es particularmente útil para homogeneizar direcciones, nombres...

Ejemplo: **COMPBL('Paseo de la Castellana')**='Paseo de la Castellana'

TRIM(argumento) Elimina los blancos finales del argumento.

Ejemplo: TRIM('Paseo de la Castellana')= 'Paseo de la Castellana'

LEFT(argumento) Alinea a la izquierda una argumento de caracteres

Ejemplo: LEFT(' María Perez')= 'Maria Perez '

SCAN(cadena, n, delimitadores) Extrae la palabra n-ésima de la *cadena*, considerando que las palabras de la cadena está separadas por él o los delimitadores especificados en el tercer argumento.

- Si no se especifican delimitadores, se toma por defecto la lista siguiente:
- Blanco . < (+ & !...\$...*...)...;...-.../...,...% |
- Cualquier carácter o conjunto de caracteres puede utilizarse de delimitador.
- La longitud de la variable a la que se iguala la función SCAN toma la longitud 200, a no ser que previamente se haya definido mediante una sentencia LENGTH o ATTRIB.
- Los delimitadores anteriores a la primera palabra no tienen efecto.
- Si dos o más delimitadores están contiguos se toman como uno solo.
- Si hay menos de n palabras en el argumento, SCAN devuelve un valor missing.
- Si n es negativo (n<0) la palabra a extraer se busca de derecha a izquierda.
- Si n es cero, aparece un error en el log de SAS.

Ejemplos:

SCAN('María Pérez Galván', 2, '=')= 'Pérez'

SCAN('http://www.ine.es/ine/organigramaine01.htm', 2, '/')= 'www.ine.es'

SCAN('Maria/Juan/Paco\Manuel\Sara/Patricia', 2, '\') = 'Juan'

SCAN('Maria/Juan/Paco\Manuel\Sara/Patricia', 5, '\') = 'Sara'

SUBSTR(cadena, n, l) Extrae una subcadena de longitud l, empezando en la posición n de la cadena origen. Si se omite la longitud, l, de la subcadena, la función SUBSTR devuelve todos los caracteres desde la posición inicio, n.

Ejemplo: SUBSTR('28002000101', 1, 5)= '28002'

TRANSLATE(cadena, carácter-final, carácter-origen) Reemplaza un carácter por otro dentro de una cadena. El segundo y tercer argumento, carácter-final y carácter-origen, pueden ser un único carácter o una lista de caracteres.

Ejemplo: TRANSLATE('maría perez galván', 'aeiou', 'áéíóú')='maria perez galvan'

TRANWRD(cadena, cadena-inicial, cadena-final) Reemplaza todas las ocurrencias de una palabra en una cadena.

Ejemplo: TRANSLATE('Mª José Rodríguez', 'Mª', 'María')='María José Rodríguez'

VERIFY(argumento, cadena de caracteres) Devuelve la posición del primer carácter que no esté incluido en la cadena de caracteres permitido. Útil para depurar textos.

Ejemplo: VERIFY('91-5830221', '0123456789')=3

CAT, CATS, CATT, CATX (argumento₁,..., argumento_n) son cuatro funciones utilizadas para concatenar que producen resultados equivalentes a la combinación del operador de concatenación || con otras funciones vistas.

Funciones.1: Funciones CAT, CATS, CATT y CATX y su código equivalente

Función	Código equivalente
CAT(X1,X2,X3,X4)	X1 X2 X3 X4
CATS(X1,X2,X3,X4)	trim(left(X1)) trim(left(X2)) trim(left(X3)) trim(left(X4))
CATT(X1,X2,X3,X4)	trim(X1) trim(X2) trim(X3) trim(X4)
CATX("-",X1,X2,X3,X4)	trim(left(X1)) "-" trim(left(X2)) "-" trim(left(X3)) "-" trim(left(X4))

Funciones aleatorias

RANBIN(SEMILLA, n, p) Binomial de parámetros "n" y "p" generada a partir de cierta semilla ⁴.

RANNOR(SEMILLA) Normal de media "0" y desviación estándar "1" generada a partir de cierta semilla.

RANPOI(SEMILLA, a) Poisson de parámetro "a" generada a partir de cierta semilla.

RANUNI(SEMILLA) Uniforme con parámetros "0" y "1" generada a partir de cierta semilla.

Cuando la semilla es cero, este valor se toma en realidad aleatoriamente.

Funciones de variables fecha

Para operar con variables fecha podemos emplear los mismos operadores que hemos visto para las variables numéricas pero cabe destacar que SAS dispone de numerosas funciones para el trabajo con variables de este tipo. Las funciones más destacadas son:

DATE() o TODAY() fecha del sistema, no lleva argumento

DATETIME() fecha y hora del sistema, no lleva argumento

TIME() hora del sistema, no lleva argumento.

DAY(argumento) día correspondiente al valor numérico marcado por el argumento

MONTH(argumento) mes correspondiente al valor numérico marcado por el argumento

⁴ Los algoritmos que generan números pseudoaleatorios necesitan de un valor inicial para generarlos. A dicho valor inicial se le conoce con el nombre de **semilla**.

YEAR(argumento) año correspondiente al valor numérico marcado por el argumento

WEEKDAY(argumento) día de la semana correspondiente al argumento, teniendo en cuenta que Domingo=1, Lunes=2,..., Sábado=7

MDY(mes,día,año) transforma a fecha SAS (valor numérico) el día, mes y año marcado.

DATDIF(d1,d2,base) Número de días entre d1 y d2 considerando meses homogéneos de 30 días y años de 360 días, si se especifica base='30/360', o considerando los meses naturales y el año natural, si se especifica base='ACT/ACT'.

INTCK('day',d1,d2) Número de días entre d1 y d2.

INTCK('week',d1,d2) Número de semanas entre d1 y d2.

INTCK('month',d1,d2) Número de meses entre d1 y d2.

INTCK('year',d1,d2) Número de años entre d1 y d2.

INTNX('year',d1,n) Suma n años a la fecha d1.

4 LECTURA DE FICHEROS

Para poder trabajar con SAS lo primero que tendremos que tener es una entrada de datos. Estos datos se podrán leer de ficheros externos o introducir manualmente con código SAS.

Los ficheros que usualmente se leen son:

- Ficheros planos (*.DAT, *.TXT, *.ASC, etc).
- Datos introducidos en el mismo programa
- XLS (*.XLS)
- ACCESS (*.MDB)
- SAS (*.Sas7bdat)

4.1 Lectura de ficheros planos

Para leer un fichero plano precisaremos de un paso DATA, bien escrito de esta forma:

```
DATA salidaSAS;  
INFILE "directorio\nombre fichero físico";  
INPUT variables ...;  
RUN;
```

O de esta otra

```
FILENAME logico "directorio\nombre fichero físico";  
  
DATA salidaSAS;  
INFILE logico;  
INPUT variables ...;  
RUN;
```

- El conjunto de datos **salidaSAS** es el fichero SAS donde se escriben los registros del fichero plano de entrada.
- La sentencia **INFILE** se utiliza para la lectura de datos externos y en ella se menciona la ruta donde se encuentra el fichero que contiene los datos. SAS permite referenciar mediante nombres lógicos los nombres físicos de los ficheros mediante la sentencia FILENAME⁵. El nombre físico siempre se indicará entre comillas simples o dobles.
- En la sentencia **INPUT** se declaran las variables que se van a leer.

Los ficheros planos son ficheros sin ningún tipo de información, por esta razón es preciso decirle a SAS el nombre de cada una de las variables, qué posiciones ocupan (o qué longitud tienen) así como de qué tipo son (numéricas o alfanuméricas) y el formato específico de lectura en caso de ser necesario.

INPUT modo columna

Se indica el nombre de las variables así como su posición inicial y final dentro del fichero de entrada. El símbolo \$ se utiliza para variables alfanuméricas. Si no se especifica \$ la variable es numérica.

⁵ El nombre lógico sigue las mismas reglas que los nombres lógicos de las librerías SAS o librefs. Es decir, debe tener una longitud inferior o igual a 8 caracteres.

Esquema:

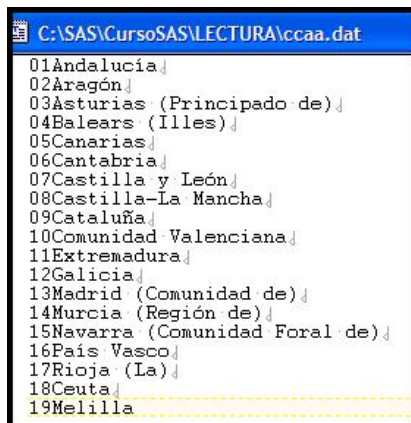
infile 'directorio\nombre fichero fisico' ;

input $variable_1$ Tipo $col_{inicio} - col_{fin}$
 $variable_2$ Tipo $col_{inicio} - col_{fin}$
 ...
 $variable_k$ Tipo $col_{inicio} - col_{fin}$;

Recomendable para leer fichero con las siguientes características:

- Rectangular (registros de longitud fija)
- Inicio y fin de cada variable bien delimitado
- No es necesario formato especial de lectura para las variables

Ejemplo:



```
FILENAME lccaa "C:\SAS\CursoSAS\LECTURA\ccaa.dat";
```

```
DATA lccaa;
  INFILE lccaa PAD;
  INPUT cod_ccaa $ 1-2 ccaa $ 3-30;
  RUN;
```

La opción **PAD** ordena a SAS que todos los registros tengan la misma longitud, para ello añade espacios al final de los registros más cortos.

INPUT con formatos

En ocasiones es preciso utilizar formatos para leer datos que no son estándares. A estos formatos de lectura se les llama formatos de lectura o *informatos* y se agrupan en 3 categorías: formatos de tipo **carácter**, formatos de tipo **numérico** y formatos de tipo **fecha/hora**.

Con **@n** se indica la posición de inicio de cada variable. A continuación irá el nombre de la variable y seguidamente el formato aplicado a la misma.

Los formatos se especifican en SAS de acuerdo a la siguiente estructura:

Formatos de tipo carácter: **\$FORMATw.**

Formatos de tipo numérico: **FORMATw.d**

Formatos de tipo fecha/hora: **FORMATw.**

- * El símbolo **\$** indica que el formato es de tipo carácter.
- * **FORMATO** se refiere a un nombre de formato a veces opcional para SAS.
- * **w** indica la anchura de la variable.
- * **d** indica el número de decimales

Todos los formatos deben terminar con un punto (.) para que SAS pueda diferenciar un formato de una variable.

Lectura 2. Esquema lectura INPUT con formatos

Esquema:

```
infile 'directorio\nombre fichero físico' ;

input    @ colinicio variable1 formato de lectura
          @ colinicio variable2 formato de lectura
          ...
          @ colinicio variablek formato de lectura ;
```

Recomendable para leer fichero con las siguientes características:

- Rectangular (registros de longitud fija)
- Inicio y fin de cada variable bien delimitado
- Es necesario formato especial de lectura para algunas variables (p.ej: fechas, variables numéricas con decimales separados por comas etc...)

Ejemplo1: Superficie media de las viviendas en cada CCAA (censo 2001)

Andalucía	93.48
Aragón	90.90
Asturias (Principado de)	82.17
Balears (Illes)	106.10
Canarias	93.66
Cantabria	91.25
Castilla y León	94.07
Castilla-La Mancha	103.99
Cataluña	89.00
Comunidad Valenciana	99.86
Extremadura	97.35
Galicia	98.03
Madrid (Comunidad de)	88.04
Murcia (Región de)	99.25
Navarra (Comunidad Foral de)	98.97
País Vasco	84.90
Rioja (La)	93.25
Ceuta	74.93
Melilla	82.70

DATA superficie;

INFILE 'C:\SAS\CursoSAS\LECTURA\SuperficieMedia.txt';

INPUT @1 ccaa \$30.

@31 s_media commax6.2;

RUN;

Ejemplo 2: Fechas de alta y baja de los trabajadores de una empresa



FILENAME emplea 'C:\SAS\CursoSAS\LECTURA\empleados.txt';

DATA emplea;

INFILE emplea **MISSOVER**;

INPUT @1 ident \$5.

 @7 salario 4.

 @12 f_alta ddmmyy10.

 @23 f_baja ddmmyy10.;

RUN;

Nota: la opción MISSOVER evita el salto de línea cuando SAS encuentra el final de la misma al existir valores missing, ya que intenta completar todas las variables (estructura rectangular).

INPUT modo lista

El modo lista indica a SAS que los campos están separados por delimitadores. Por ello, SAS lee desde un delimitador a otro en vez de leer desde un lugar específico en el registro de datos planos.

Los delimitadores más comunes son los blancos, las comas y los tabuladores. El delimitador predeterminado es un espacio en blanco, sin embargo cualquier carácter que se pueda escribir con el teclado puede ser un delimitador. La opción **DELIMITER=** o **DLM=** en la sentencia **INFILE** permite especificar el carácter que se va a considerar como delimitador en la lectura del fichero.

Para el modo lista, en la sentencia input las variables se deben especificar en el mismo orden en el que aparecen en el fichero de datos plano. Si la variable es alfanumérica, se debe especificar un símbolo \$ después del nombre de ésta. Si no aparece este símbolo la variable es numérica. Del mismo modo que se ha mencionado antes, si estamos ante datos no estándares se deberán utilizar los formatos de lectura para leerlos correctamente.

Cuando se usa el modo lista, la longitud predeterminada para variables alfanuméricas y numéricas es de 8 bytes.⁶

La longitud de las variables alfanuméricas se puede establecer con un formato de lectura.

⁶ En el caso de las variables alfanuméricas cada byte es un carácter.

Lectura 3. Esquema INPUT modo lista

Esquema:

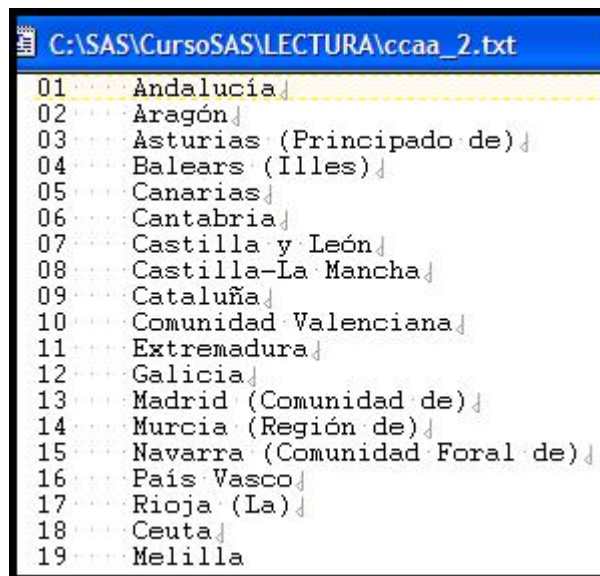
```
INFILE 'directorio\nombre fichero físico' DLM='delimitador' ;
```

```
INPUT  variable1  Tipo o formato de lectura  
      variable2  Tipo o formato de lectura  
      ...  
      variablen  Tipo o formato de lectura ;
```

Recomendable para leer fichero con la siguiente característica:

- Variables separadas por delimitador

Ejemplo 1:



```
FILENAME lccaa2 "C:\SAS\CursoSAS\LECTURA\ccaa_2.txt";
```

```
DATA ccaa2;
```

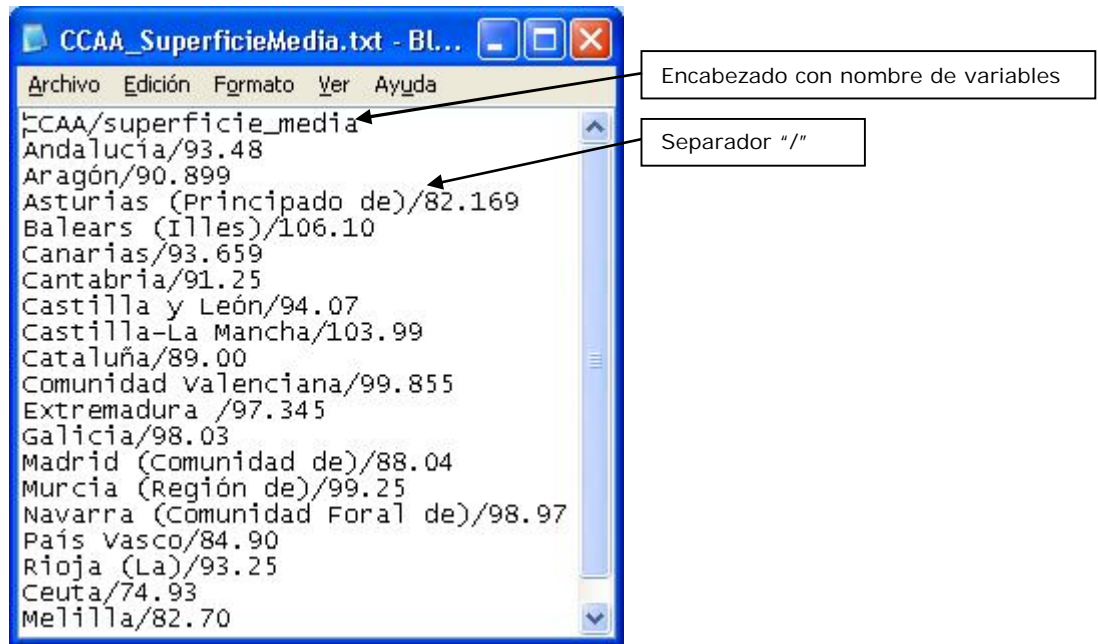
```
INFILE lccaa2 DLM='09'x PAD;
```

```
INPUT cod_ccaa $ ccaa $30.;
```

```
RUN;
```

Nota: DLM='09'X los datos están separados por TABULADOR. '09' es el código ASCII del TABULADOR en hexadecimal y X indica que está en hexadecimal.

Ejemplo 2:



```
FILENAME CCAA "C:\SAS\CursoSAS\LECTURA\CCAA_SuperficieMedia.txt";
```

```
DATA CCAA_SMedia;
```

```
INFILE CCAA DLM='/' firstobs=2;
```

```
INPUT ccaa :$30. supmedia ;
```

```
RUN;
```

Nota: firstobs=2 indica que la primera línea de datos es la segunda. Y los dos puntos (:) antes del formato de lectura de la variable ccaa (\$30.), indica a SAS que ha de leer hasta el próximo delimitador o hasta que complete todo el número de caracteres identificados en el formato de lectura. Si no se pusiesen los dos puntos, SAS interpretaría "/" como parte del valor de la variable CCAA.

4.2 Tabla resumen lectura ficheros planos

MODO DE LECTURA	(Recomendable) PARA LEER TIPO DE FICHERO ...	SENTENCIA INPUT	EJEMPLO
INPUT modo Columna	<ul style="list-style-type: none"> Rectangular (registros de longitud fija) Inicio y fin de cada variable bien delimitado No es necesario formato especial de lectura para las variables 	infile 'directorio\nombre fichero físico' ; input variable ₁ Tipo col _{inicio} - col _{fin} variable ₂ Tipo col _{inicio} - col _{fin} ... variable _k Tipo col _{inicio} - col _{fin} ;	DATA nacidosPAD; INFILE "C:\SAS\nacidosPAD.txt"; INPUT codnac \$ 1-10 edadm 11-12 edadp 13-14 ; RUN ;
INPUT con formatos	<ul style="list-style-type: none"> Rectangular (registros de longitud fija) Inicio y fin de cada variable bien delimitado Es necesario formato especial de lectura para algunas variables (p.ej. variables fecha, numéricas con decimales separados por comas o delimitador de miles) 	infile 'directorio\nombre fichero físico' ; input @ col _{inicio} variable ₁ formato. @ col _{inicio} variable ₂ formato. ... @ col _{inicio} variable _k formato. ;	DATA nacidosDAT; INFILE "C:\SAS\nacidosDAT.txt"; INPUT @1 codnac \$10. @15 fecha date9. @24 sexo \$1. @29 peso 5.3 @37 altura 2.; RUN ;
INPUT modo Lista	<ul style="list-style-type: none"> Variables separadas por delimitador 	infile 'directorio\nombre fichero físico' DLM=' delimitador ' ; input variable ₁ Tipo variable ₂ Tipo ... variable _k Tipo ;	DATA nacidosHOSP; INFILE "C:\SAS\nacidosHOSP.txt" dlm=" "; INPUT codnac \$ hospital \$ municipio provincia ; RUN ;

4.3 Lectura de ficheros SAS

Supongamos que queremos leer un fichero SAS que está en una determinada ubicación, por ejemplo, en C:\sas\CursoSAS\ccaa.Sas7bdat.

En este caso, aunque pretendemos leer un fichero que no es de nuestra sesión SAS (no es temporal), no se trata de un fichero externo a SAS por lo se utilizará la sentencia **LIBNAME** y un paso **DATA** con un **SET**.

```
LIBNAME lectura "C:\SAS\CursoSAS";  
DATA ccaa;  
SET lectura.ccaa;  
RUN;
```

Mediante la sentencia **LIBNAME** estamos asignando una librería SAS y dando un nombre lógico al subdirectorío donde se encuentra nuestro fichero SAS.

Mediante la sentencia **SET** realizamos la lectura del fichero, pero debemos decirle dónde se encuentra. En caso de no indicar la librería en la sentencia SET, SAS lo buscará donde tiene todos los ficheros por defecto, es decir en la librería WORK.

Así, en el ejemplo anterior lo que se hace es leer el fichero SAS **ccaa** guardado en la librería **lectura** y hacer una copia del mismo, con el mismo nombre, en la librería **work**.

NOTA: se debe tener muy presente que si los nombres de los ficheros SAS que especifiquemos en las sentencias DATA y SET son iguales, se sustituirá uno por otro, el de lectura se pierde. En ocasiones esto es lo deseable con el fin de economizar espacio, pero en otros casos puede llevarnos a errores.

Continuando con el ejemplo anterior sería:

```
DATA ccaa;  
SET ccaa;  
RUN;
```

4.4 Lectura de ficheros EXCEL (xls) usando PROC IMPORT

Una de las opciones para importar un fichero SAS a un fichero Excel es utilizar el procedimiento PROC IMPORT.

Su sintaxis es la siguiente:

```
PROC IMPORT OUT=<libname>.ficheroSAS
    DATAFILE= "directorio\nombre_fichero.xls"
    DBMS=EXCEL REPLACE;
    SHEET="Hoja1$";
    GETNAMES=YES;
    MIXED=NO;
    SCANTEXT=YES;
    USEDATE=YES;
    SCANTIME=YES;
RUN;
```

Ejemplo:

```
PROC IMPORT OUT= WORK.ccaa
    DATAFILE= "C:\sas\cursosas\LECTURA\ccaa.xls"
    DBMS=EXCEL REPLACE;
    SHEET="Hoja1$";
    GETNAMES=YES;
    MIXED=NO;
    SCANTEXT=YES;
    USEDATE=YES;
    SCANTIME=YES;
RUN;
```

Existe, además, otra opción para importar un fichero Excel a SAS sin tener que escribir código por nuestra parte. Dicha opción consiste en activar el asistente de SAS para importar ficheros, que se encuentra en Archivo>Importar datos..., y seguir los pasos indicados por él.

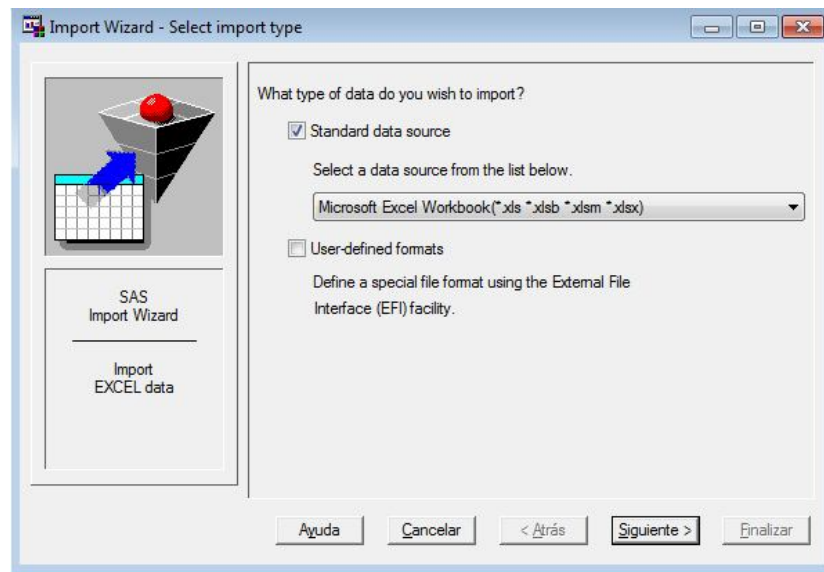
Siguiendo con el ejemplo anterior, podemos ver en los siguientes gráficos, cuáles son las pantallas a través de las que nos guiará el asistente para conseguir importar el fichero "C:\sas\cursosas\LECTURA\ccaa.xls" a un conjunto de datos SAS.

Lectura 4. Activar asistente para importar datos EXCEL



Seleccionamos como tipo Excel los datos a importar:

Lectura 5. Asistente para importar datos EXCEL. Seleccionar tipo de datos EXCEL.

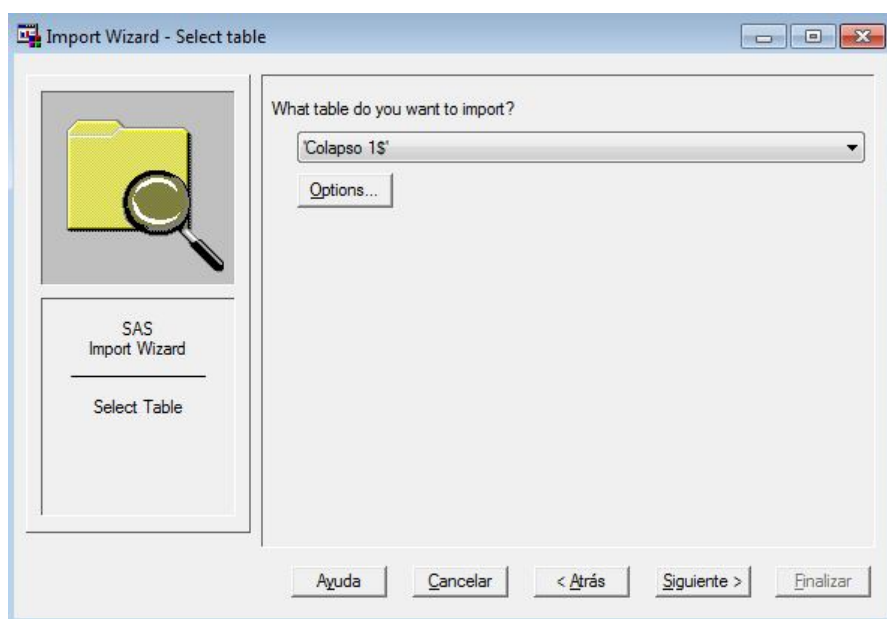


A continuación especificamos el libro y la hoja concreta del mismo donde están nuestros datos, así como las opciones:

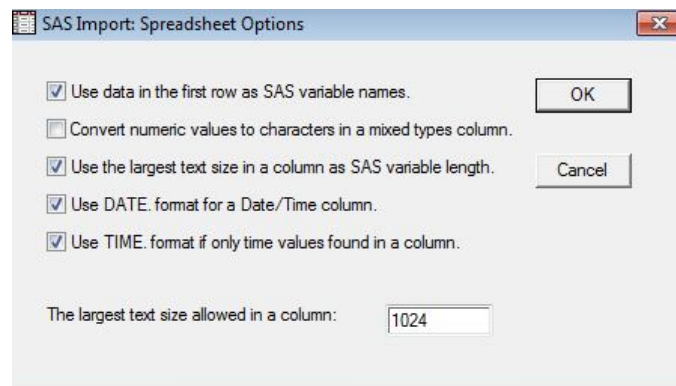
Lectura 6. Asistente para importar datos EXCEL. Especificación de libro



Lectura 7 Asistente para importar datos EXCEL. Especificación de hoja.

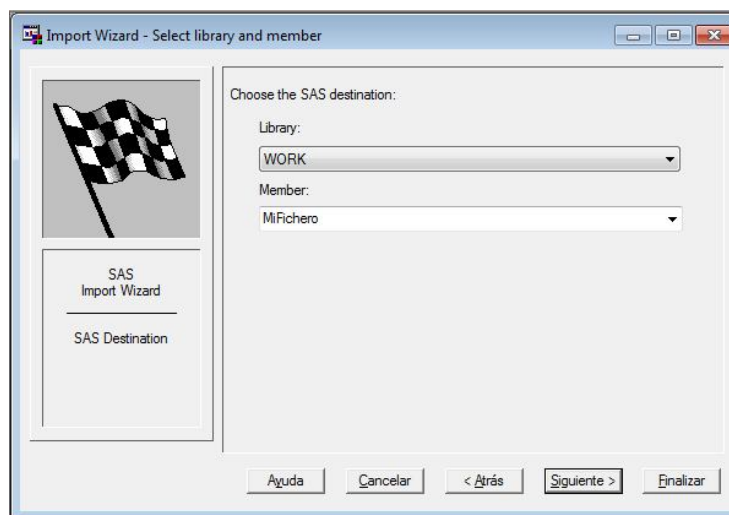


Lectura 8. Asistente para importar datos EXCEL. Especificación de opciones



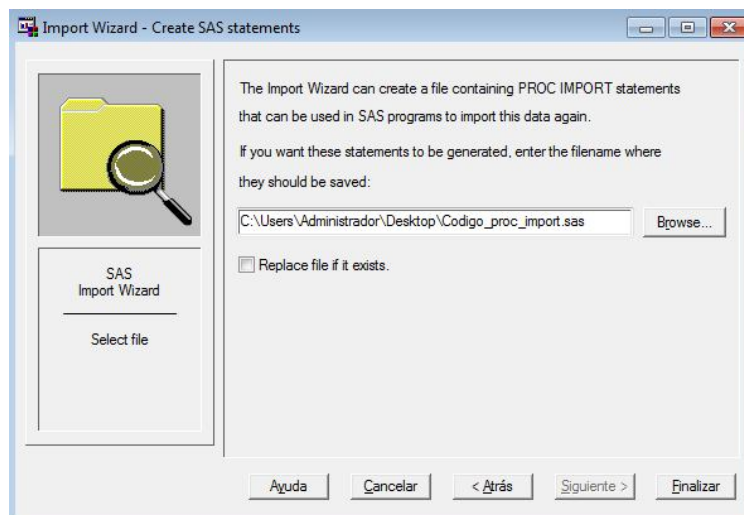
Indicamos la librería y el nombre del conjunto de datos SAS resultado de la importación:

Lectura 9. Asistente para importar datos EXCEL. Especificación de librería y nombre fichero SAS.



Y por último SAS, en esta pantalla, da la opción de guardar el código del PROC IMPORT para poder utilizarlo en otras ocasiones sin tener que recurrir al asistente.

Lectura 10. Asistente para importar datos EXCEL. Posibilidad de guardar código SAS.



4.5 Lectura de ficheros ACCESS (mdb) usando PROC IMPORT

Con el procedimiento SAS visto en el apartado anterior se pueden también importar ficheros ACCESS. La sintaxis es la siguiente:

```
PROC IMPORT OUT= <libname> .ficheroSAS  
    DATATABLE= "nombre_tabla"  
    DBMS=ACCESS REPLACE;  
    DATABASE="directorio\nombre_fichero.mdb";  
    SCANMEMO=YES;  
    USEDATE=NO;  
    SCANTIME=YES;  
RUN;
```

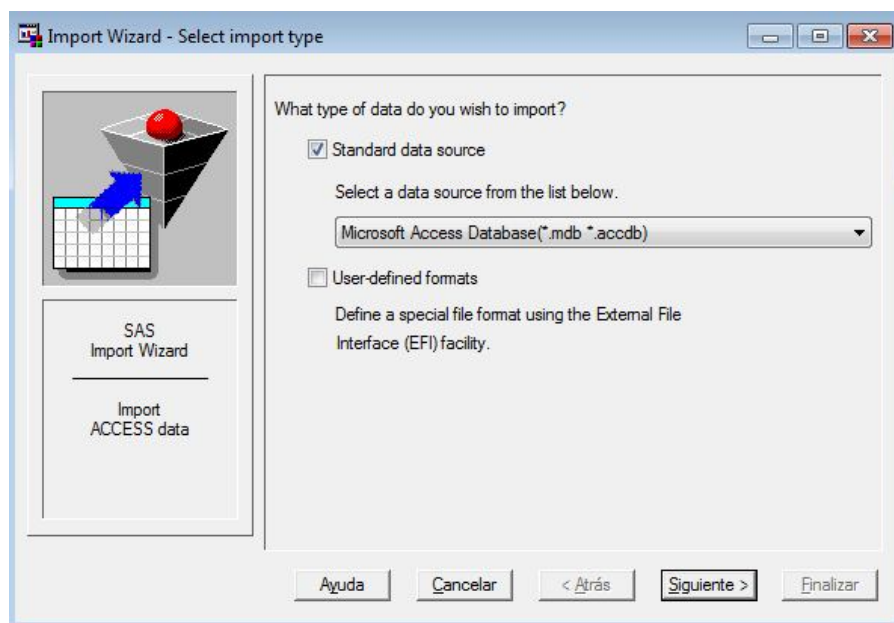
Ejemplo:

```
PROC IMPORT OUT= WORK.ccaa  
    DATATABLE= "caa"  
    DBMS=ACCESS REPLACE;  
    DATABASE="C:\sas\cursosas\LECTURA\Censo2001.mdb";  
    SCANMEMO=YES;  
    USEDATE=NO;  
    SCANTIME=YES;  
RUN;
```

De nuevo existe la opción de importar ficheros ACCESS a SAS sin tener que escribir el código, a través del asistente proporcionado por SAS. El procedimiento será el mismo que en el caso de importar un fichero Excel a SAS teniendo en cuenta que, en el momento en que aparezca la pantalla de selección del tipo de fichero a importar, deberemos especificar "Microsoft Access Database (*.mdb, *.accdb)".

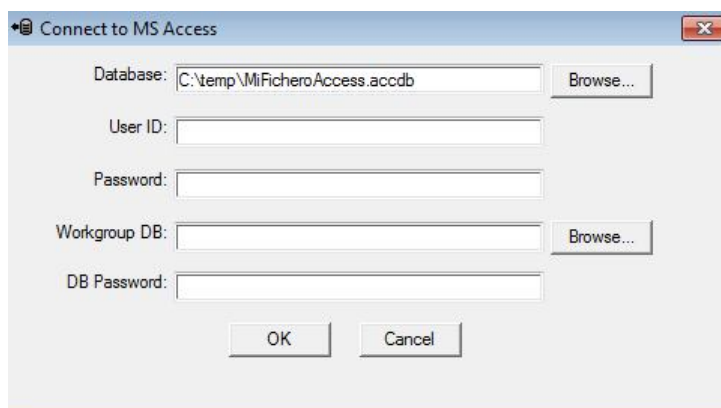
Seleccionamos como tipo Access los datos a importar:

Lectura 11. Asistente para importar datos ACCESS.



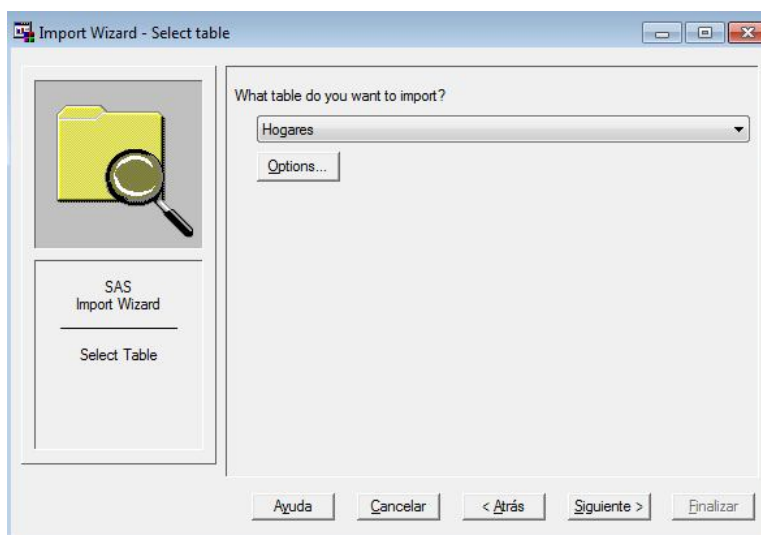
Seleccionamos la base de datos concreta de donde se desean importar los datos:

Lectura 12. Asistente para importar datos ACCESS.



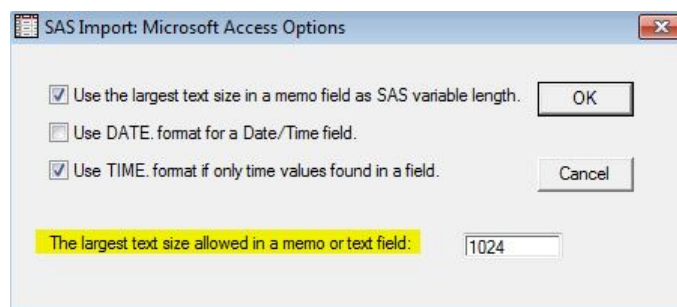
Seleccionamos dentro de la base de datos anterior la tabla a importar:

Lectura 13. Asistente para importar datos ACCESS.



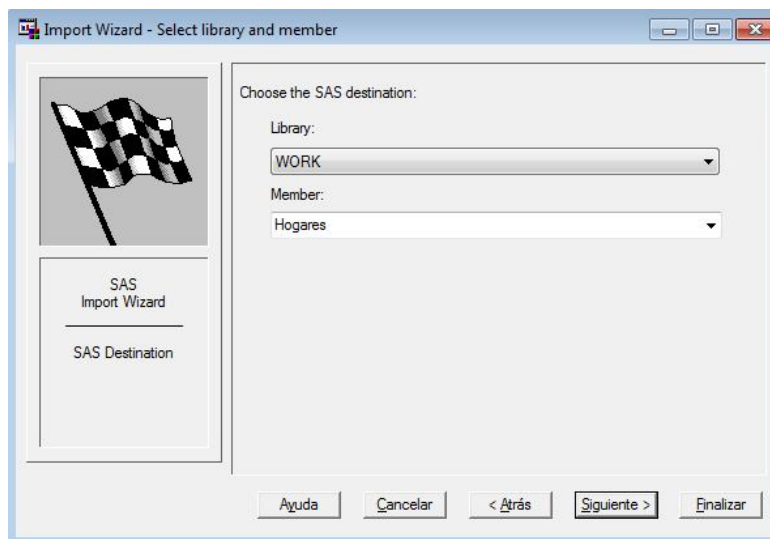
En "Options", dejamos la longitud dada por defecto para los campos importados:

Lectura 14. Asistente para importar datos ACCESS.



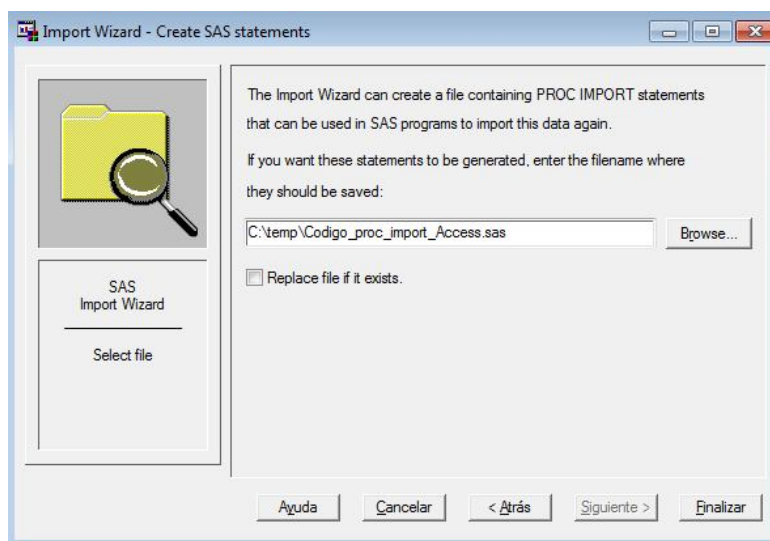
Indicamos la librería y el nombre del conjunto de datos SAS resultado de la importación:

Lectura 15. Asistente para importar datos ACCESS.



Se guarda el código del PROC IMPORT para poder utilizarlo en otras ocasiones sin tener que recurrir al asistente:

Lectura 16. Asistente para importar datos ACCESS.



4.6 Lectura de datos desde el mismo programa

En algunas ocasiones, en lugar de leer datos de un fichero externo los datos se podrán incluir dentro del propio paso **DATA**. La sentencia que se debe emplear en este caso es **DATALINES** la cual sustituirá a la sentencia **INFILE** e irá siempre después de la sentencia **INPUT**.

La sentencia sería la siguiente:

```
DATA salidaSAS;  
INPUT variables ...;  
DATALINES;  
datos ...  
;  
RUN;
```

Ejemplo:

DATA profesion;

INPUT idioma \$1-2

cod_profesion \$4

des_profesion \$6-80;

DATALINES;

ES 1 Profesionales, Técnicos y similares

ES 2 Directivos y Gerentes de la Administración Pública y las Empresas

ES 3 Personal Administrativo

ES 4 Comercio

ES 5 Trabajadores de la hostelería y resto de servicios

ES 6 Agricultura y ganadería

ES 7 Trabajadores de la construcción y la industria

ES 8 Peones y trabajadores no especializados

ES 9 Profesionales de las fuerzas armadas

;

RUN;

5 ESCRITURA DE FICHEROS

De manera casi análoga a la lectura de ficheros, podemos desde un conjunto de datos SAS crear un fichero ASCII (.dat, .txt, etc..), un fichero Excel (.xls, .xlsx), un fichero Access (.mdb, .accdb), o guardar físicamente nuestro fichero SAS (.sas7bdat).

5.1 Escritura de ficheros planos

Como en la lectura, para escribir un fichero plano precisamos de un paso DATA bien escrito de esta forma:

```
DATA _NULL_;  
  SET entradaSAS;  
  FILE 'directorio\nombre fichero';  
  PUT variables....;  
RUN;
```

O bien escrito de esta otra

```
FILENAME lógico 'directorio\nombre fichero';  
DATA _NULL_;  
  SET entradaSAS;  
  FILE lógico;  
  PUT variables....;  
RUN;
```

- En la sentencia SET se especifica el conjunto de datos SAS, entradaSAS, que leemos y queremos exportar a un fichero ASCII.
- La sentencia FILE se utiliza para la escritura de datos externos y en ella se menciona la ruta donde queremos guardar el fichero que contendrá los datos. SAS permite asociar un nombre físico o ruta a un nombre lógico mediante la sentencia FILENAME. La ruta siempre se indicará entre comillas simples o dobles.
- Con la sentencia PUT le indicamos a SAS cómo escribir el fichero plano, qué variables escribirá, en qué columna, con qué longitud, y qué formato (si queremos especificar alguno).

Los ficheros ASCII son ficheros planos, sin ningún tipo de información, es por esto que es preciso decirle a SAS el nombre de cada una de las variables, qué posiciones ocupan (o qué longitud tienen) así como de qué tipo son (numéricas o alfanuméricas).

PUT modo columna

Se indica el nombre de las variables así como su posición inicial y final dentro del fichero de salida. El símbolo \$ se utiliza para variables alfanuméricas. Si no se especifica \$ la variable es numérica.

Escritura 1. Esquema PUT modo columna

Esquema:

```
PUT variable1 Tipo col1i – col1f  
    variable2 Tipo col2i – col2f  
    ...  
    variablen Tipo colni - colnf ;
```

Recomendable para crear fichero ASCII con las siguientes características:

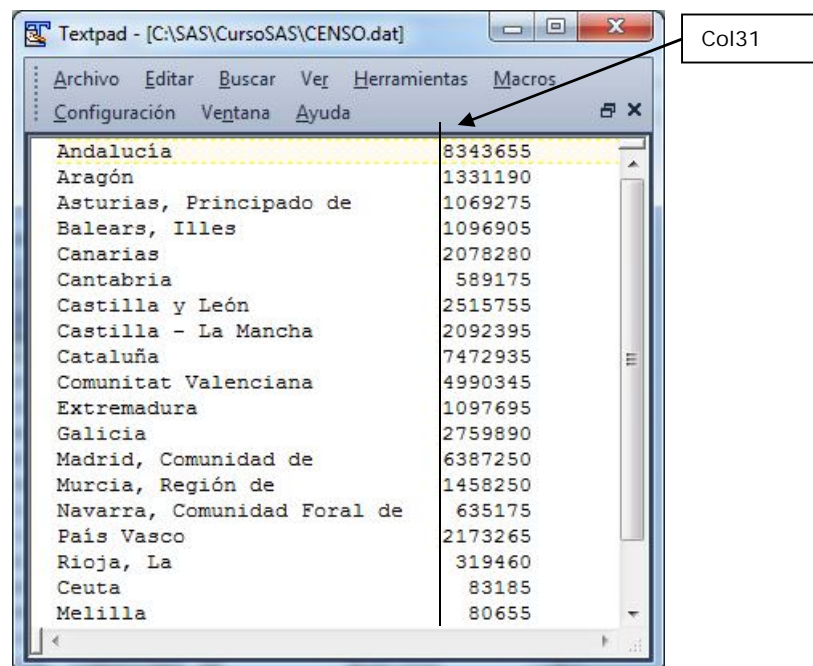
- Rectangular (registros de longitud fija)
- Inicio y fin de cada variable bien delimitado
- Las variables no requieren formato especial de escritura

Ejemplo: Partiendo del fichero de datos SAS, CursoSAS.Censo2011, que contiene las personas residentes en España por CCAA según el Censo de Población y Viviendas de 2001, generaremos el fichero plano C:\SAS\CursoSAS\Censo.dat utilizando el PUT en modo columna.

```
FILENAME censo "C:\SAS\CursoSAS\CENSO.dat";
```

```
DATA _NULL_;  
  SET CursoSAS.censo2011;  
  FILE censo;  
  PUT CCAA $ 1-30  
    TOTAL 31-37;
```

```
RUN;
```



Andalucía	8343655
Aragón	1331190
Asturias, Principado de	1069275
Baleares, Illes	1096905
Canarias	2078280
Cantabria	589175
Castilla y León	2515755
Castilla - La Mancha	2092395
Cataluña	7472935
Comunitat Valenciana	4990345
Extremadura	1097695
Galicia	2759890
Madrid, Comunidad de	6387250
Murcia, Región de	1458250
Navarra, Comunidad Foral de	635175
País Vasco	2173265
Rioja, La	319460
Ceuta	83185
Melilla	80655

PUT con formatos

En ocasiones es preciso utilizar formatos para escribir datos. Los formatos se agrupan en 3 categorías: formatos de tipo **carácter**, formatos de tipo **numérico** y formatos de tipo **fecha/hora**.

Con **@n** se indica la posición de inicio de cada variable. A continuación irá el nombre de la variable y seguido del formato aplicado a la misma.

Los formatos, al igual que los formatos de lectura, se especifican en SAS de acuerdo a la siguiente estructura:

Formatos de tipo carácter: **\$FORMATw**.
Formatos de tipo numérico: **FORMATw.d**
Formatos de tipo fecha/hora: **FORMATw**.

- * El símbolo **\$** indica que el formato es de tipo carácter.
- * **FORMATO** se refiere a un nombre de formato a veces opcional para SAS.
- * **w** indica la anchura de la variable.
- * **d** indica el número de decimales

Todos los formatos deben terminar con un punto (.) para que SAS pueda diferenciar un formato de una variable.

Escritura 2. Esquema PUT con formatos

Esquema:

```
file 'directorio\nombre fichero físico' ;

put  @ colinicio  variable1  formato.
      @ colinicio  variable2  formato.
      ...
      @ colinicio  variablek  formato. ;
```

Recomendable para crear fichero planos con las siguientes características:

- Rectangular (registros de longitud fija)
- Inicio y fin de cada variable bien delimitado
- Algunas variables SÍ requieren un formato especial de escritura (p.ej: fechas, variables numéricas con decimales separados por comas, separadores de miles)

Ejemplo 1

Partiendo del fichero SAS, cursoSAS.Censo2011, que contiene los datos de personas residentes en España por CCAA (censo 2011) se genera el fichero plano "C:\SAS\CursoSAS\Censo.dat" utilizando PUT con formatos.

```
DATA _NULL_;
SET CursoSAS.censo2011;
FILE "C:\SAS\CursoSAS\CENSO.dat";
PUT  @1  CCAA $30.
      @31 TOTAL 7.;
RUN;
```

Textpad - [C:\SAS\CursoSAS\CENSO.dat]

Archivo Editar Buscar Ver Herramientas Macros
Configuración Ventana Ayuda

Andalucía	8343655
Aragón	1331190
Asturias, Principado de	1069275
Balears, Illes	1096905
Canarias	2078280
Cantabria	589175
Castilla y León	2515755
Castilla - La Mancha	2092395
Cataluña	7472935
Comunitat Valenciana	4990345
Extremadura	1097695
Galicia	2759890
Madrid, Comunidad de	6387250
Murcia, Región de	1458250
Navarra, Comunidad Foral de	635175
País Vasco	2173265
Rioja, La	319460
Ceuta	83185
Melilla	80655

Col31

Ejemplo 2:

Partiendo del fichero SAS, cursoSAS.Censo2011, que contiene los datos de personas residentes en España por CCAA (censo 2001) se genera el fichero plano "C:\SAS\CursoSAS\Censo.txt" utilizando PUT con formatos y el formato COMMAX9. (separador de miles, 9 indica la longitud del mayor valor de TOTAL, es decir, 7 dígitos más 2 puntos del formato) para la variable TOTAL.

```
DATA _NULL_;
  SET CursoSAS.censo2011;
  FILE "C:\SAS\CursoSAS\CENSO.txt";
  PUT    @1 CCAA $30.
        @31 TOTAL commax9.;
RUN;
```

Textpad - [C:\SAS\CursoSAS\CENSO.txt]

Archivo Editar Buscar Ver Herramientas Macros
Configuración Ventana Ayuda

Andalucía	8.343.655
Aragón	1.331.190
Asturias, Principado de	1.069.275
Balears, Illes	1.096.905
Canarias	2.078.280
Cantabria	589.175
Castilla y León	2.515.755
Castilla - La Mancha	2.092.395
Cataluña	7.472.935
Comunitat Valenciana	4.990.345
Extremadura	1.097.695
Galicia	2.759.890
Madrid, Comunidad de	6.387.250
Murcia, Región de	1.458.250
Navarra, Comunidad Foral de	635.175
País Vasco	2.173.265
Rioja, La	319.460
Ceuta	83.185
Melilla	80.655

PUT modo lista

El modo lista indica a SAS que en el fichero plano de salida los campos deben estar separados por delimitadores, en lugar de en un lugar específico, como ocurría en los casos anteriores.

El delimitador predeterminado es un espacio en blanco, sin embargo cualquier carácter que se pueda escribir con el teclado puede ser un delimitador. La opción **DLM=** en la sentencia **FILE** permite especificar el carácter que se va a considerar como delimitador en la escritura del fichero.

En la sentencia PUT se especificarán las variables que se quieran exportar del fichero de datos SAS. Si la variable es alfanumérica, se debe escribir un símbolo \$ después del nombre de ésta. Si no aparece este símbolo la variable es numérica.

Escritura 3. Esquema PUT modo lista

Esquema:

```
file 'directorio\nombre fichero físico' DLM=' delimitador';
```

```
put  variable1 Tipo  
     variable2 Tipo  
     ...  
     variablek Tipo ;
```

Recomendable para crear fichero con la siguiente característica:

- Variables separadas por delimitador

Ejemplo: Partiendo del fichero WORK.fechas, generamos el fichero ASCII C:\SAS\CursoSAS\fechas.txt.

```
DATA _NULL_;  
  SET fechas;  
  FILE "C:\SAS\CursoSAS\fechas.txt" DLM=';';  
  PUT ID $ FECHA;  
RUN;
```

Esquema del ejemplo anterior. A la izquierda fichero SAS work.fechas y a la derecha el fichero plano, fechas.txt, generado.

	ID	FECHA
1	01	24/11/2008
2	02	25/11/2008
3	03	26/11/2008
4	04	27/11/2008
5	05	28/11/2008
6	06	29/11/2008
7	07	30/11/2008
8	08	01/12/2008
9	09	02/12/2008
10	10	03/12/2008
11	11	04/12/2008
12	12	05/12/2008

Archivo	Edición	Formato	Ver	Ayuda
01; 24/11/2008				
02; 25/11/2008				
03; 26/11/2008				
04; 27/11/2008				
05; 28/11/2008				
06; 29/11/2008				
07; 30/11/2008				
08; 01/12/2008				
09; 02/12/2008				
10; 03/12/2008				
11; 04/12/2008				
12; 05/12/2008				

5.2 Tabla resumen escritura ficheros planos

MODO DE ESCRITURA	(Recomendable) PARA CREAR TIPO DE FICHERO ...	SENTENCIA PUT	EJEMPLO
PUT modo Columna	<ul style="list-style-type: none"> Rectangular (registros de longitud fija) Inicio y fin de cada variable bien delimitado No es necesario formato especial de escritura para las variables 	file 'directorio\nombre fichero físico'; put variable ₁ Tipo col _{inicio} - col _{fin} variable ₂ Tipo col _{inicio} - col _{fin} ... variable _k Tipo col _{inicio} - col _{fin} ;	DATA _NULL_; SET work.matrimPAIS; FILE "C:\SAS\matrimoniosPAIS.txt"; PUT codmat \$ 1-10 pais1 12-16 pais2 18-22 ; RUN ;
PUT con Formatos	<ul style="list-style-type: none"> Rectangular (registros de longitud fija) Inicio y fin de cada variable bien delimitado Es necesario formato especial de escritura para algunas variables (p.ej. variables fecha, numéricas con decimales o delimitador de miles) 	file 'directorio\nombre fichero físico'; put @ col _{inicio} variable ₁ formato. @ col _{inicio} variable ₂ formato. ... @ col _{inicio} variable _k formato. ;	DATA _NULL_; SET work.matrimEDAD; FILE "C:\SAS\matrimoniosEDAD.txt"; PUT @1 codmat \$10. @11 fecha date9. @20 edad1 3. @23 edad2 3. ; RUN ;
PUT modo Lista	<ul style="list-style-type: none"> Variables separadas por delimitador 	file 'directorio\nombre fichero físico' DLM='delimitador'; put variable ₁ Tipo variable ₂ Tipo ... variable _k Tipo ;	DATA _NULL_; SET work.matrimTIPO; FILE "C:\SAS\matrimoniosPAIS.txt" dlm=" "; PUT codmat \$ tipo \$ municipio provincia ; RUN ;

5.3 Escritura de ficheros SAS

En primer lugar deberemos asignar una librería, es decir, asociar, mediante la sentencia LIBNAME, un nombre lógico a la ruta o ubicación del directorio donde deseemos guardar físicamente el fichero SAS.

```
LIBNAME libreríaSAS 'directorio';  
DATA libreríaSAS.fichero;  
    SET fichero;  
RUN;
```

Ejemplo:

```
LIBNAME CursoSAS 'C:\SAS\CursoSAS';  
  
DATA CursoSAS.CENSO2011;  
    SET CENSO2011;  
RUN;
```

Así, en este ejemplo lo que se hace es leer el fichero SAS **Censo2011** guardado en la librería **WORK** y hacer una copia del mismo en la librería **CURSOSAS**.

5.4 Escritura de ficheros EXCEL (xls) usando PROC EXPORT

Una de las opciones para exportar un fichero SAS a un fichero Excel es utilizar el procedimiento PROC EXPORT. Su sintaxis es la siguiente:

```
FILENAME nombre 'directorio\nombre_fichero.xls';
```

```
PROC EXPORT DATA=<libname>ficheroSAS  
OUTFILE=nombre DBMS=excel REPLACE ;  
SHEET='Hoja1';  
RUN;
```

Ó

```
PROC EXPORT DATA=<libname> ficheroSAS  
OUTFILE='directorio\nombre_fichero.xls'  
DBMS=excel REPLACE ;  
SHEET='Hoja1';  
RUN;
```

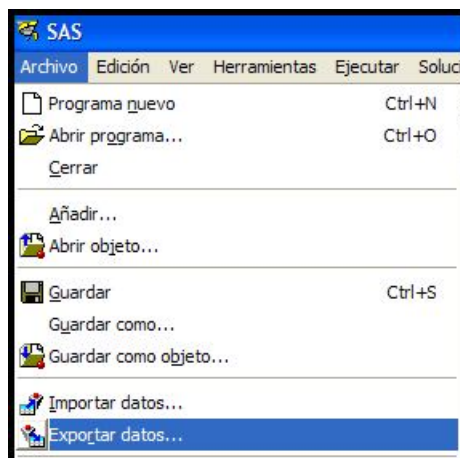
Ejemplo: Este ejemplo exporta el fichero SAS (WORK.CENSO2011) a un libro de EXCEL (Censo2011_cursoSAS.xls) ubicado en 'C:\SAS\CursoSAS' y en él crea una hoja nueva ("CENSOS").

```
PROC EXPORT DATA= WORK.CENSO2011  
OUTFILE= "C:\SAS\CursoSAS\Censo2001_cursoSAS.xls"  
DBMS=EXCEL REPLACE;  
SHEET="CENSO2011";  
RUN;
```

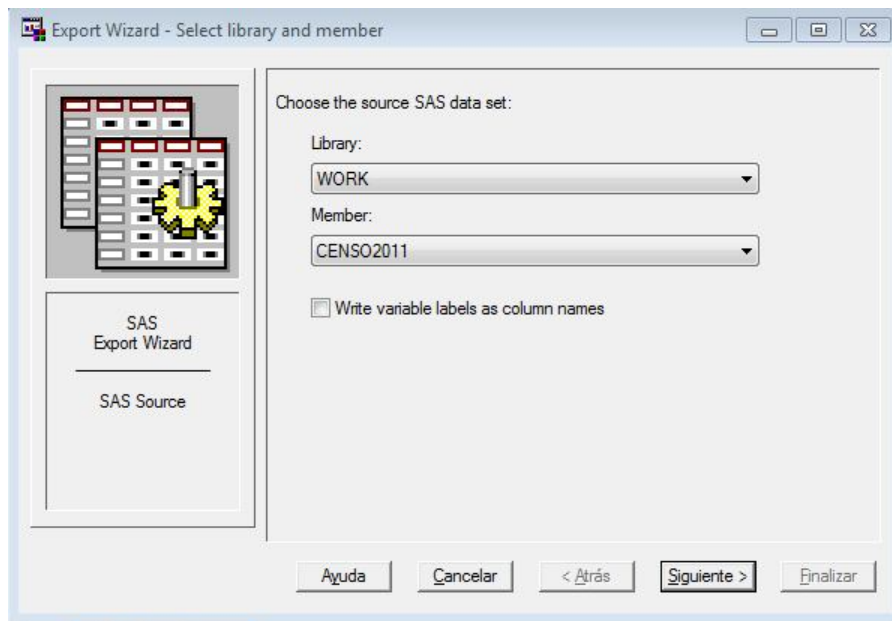
Existe, además, otra opción para exportar un fichero SAS a Excel sin tener que escribir código por nuestra parte. Dicha opción consiste en activar el asistente de SAS para exportar ficheros, que se encuentra en Archivo>Exportar datos..., y seguir los pasos indicados por él.

Siguiendo con el ejemplo anterior, podemos ver, en los siguientes gráficos, las pantallas del asistente para exportar el fichero SAS Censo2011 a C:\SAS\CursoSAS\Censo2011_cursoSAS.xls.

Escritura 4. Activar asistente para exportar a EXCEL.

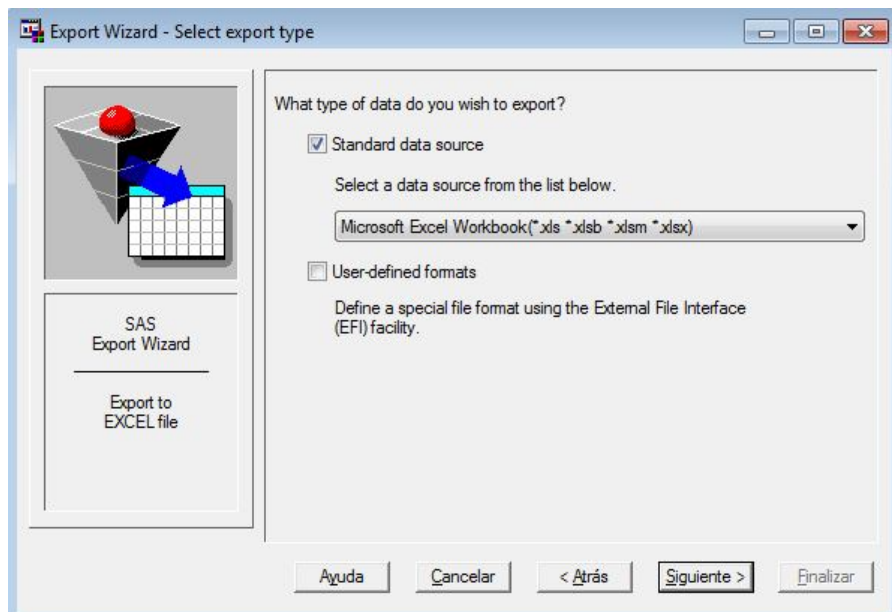


Escritura 5. Asistente para exportar a EXCEL. Seleccionar el **fichero SAS** a exportar.

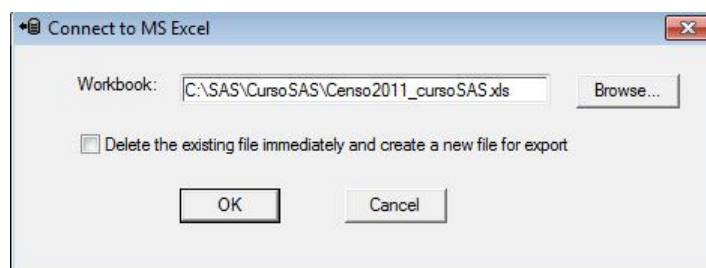


Elegimos el tipo de fichero que queremos crear en la exportación. En nuestro caso Excel.

Escritura 6. Asistente para exportar a EXCEL. Especificar **tipo de fichero** EXCEL.

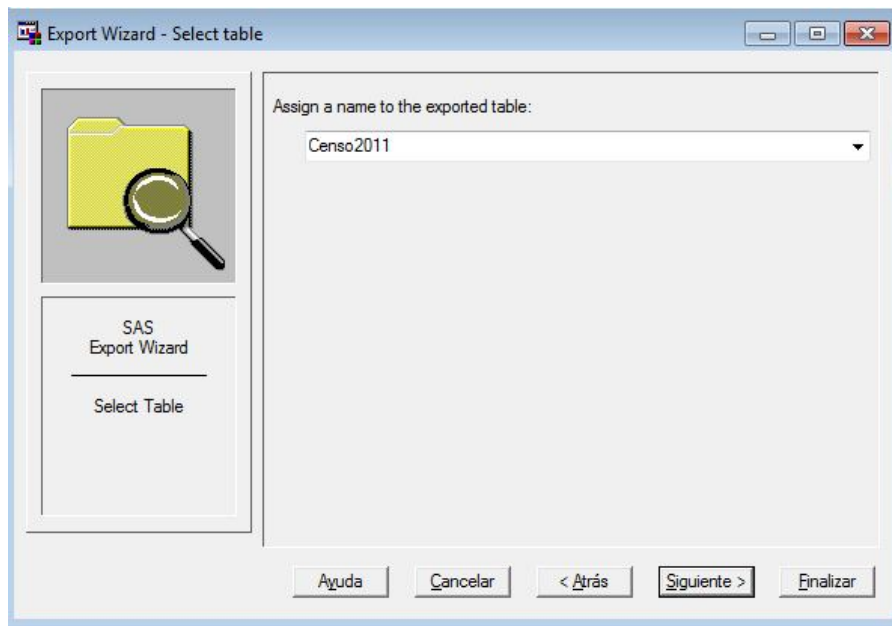


Escritura 7. Asistente para exportar a EXCEL. Especificar **ubicación** fichero EXCEL de salida.



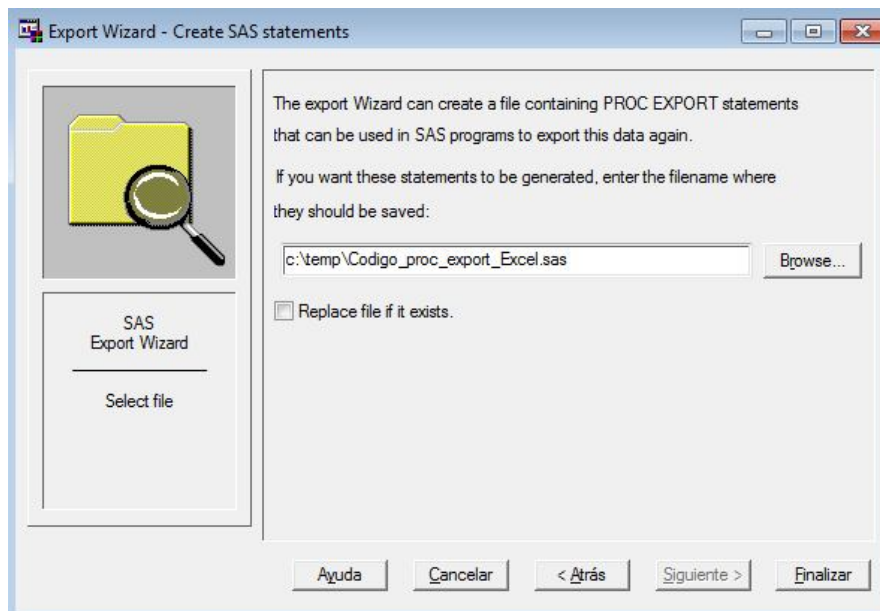
Especificamos el nombre de la hoja del libro Excel a exportar.

Escritura 8. Asistente para exportar a EXCEL. Especificar el nombre de la hoja.



Y por último SAS, en esta pantalla, da la opción de guardar el código del PROC EXPORT para poder utilizarlo en otras ocasiones sin tener que recurrir al asistente.

Escritura 9. Asistente para exportar a EXCEL. Opción de guardar código.



5.5 Escritura de ficheros ACCESS (mdb) usando PROC EXPORT

Con el procedimiento SAS visto en el apartado anterior, se pueden exportar también ficheros ACCESS. La sintaxis para escribir ficheros ACCESS usando PROC EXPORT es la siguiente:

```
FILENAME nombre 'directorio\nombre_fichero.mdb';
PROC EXPORT DATA= <libname>ficheroSAS
    OUTTABLE= nombre_tabla
    DBMS=ACCESS REPLACE;
DATABASE=nombre;
RUN;
```

O

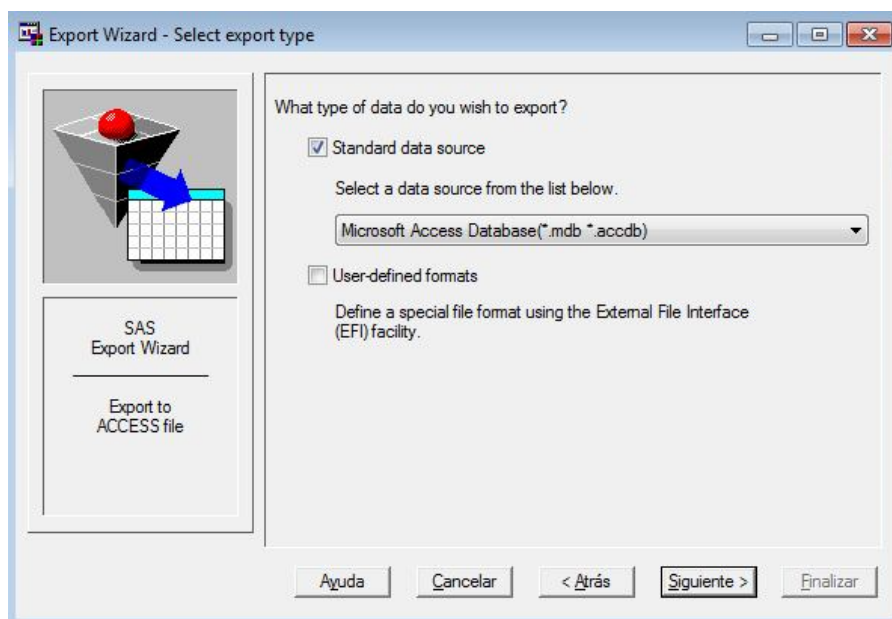
```
PROC EXPORT DATA = <libname>ficheroSAS
    OUTTABLE= nombre_tabla
    DBMS=ACCESS REPLACE;
DATABASE='directorio\nombre_fichero.mdb';
RUN;
```

Ejemplo:

```
PROC EXPORT DATA= WORK.CENSO2001
    OUTTABLE= "CENSOS"
    DBMS=ACCESS REPLACE;
DATABASE="C:\SAS\CURSOSAS\Censo2001.MDB";
RUN;
```

De nuevo existe la opción de exportar ficheros SAS a ACCESS sin tener que escribir el código, a través del asistente proporcionado por SAS. El procedimiento será el mismo que en el caso de exportar un fichero SAS a Excel teniendo en cuenta que, en el momento en que aparezca la pantalla de selección del tipo de fichero a exportar, deberemos especificar "Microsoft Access Database (*.mdb, *.accdb)".

Escritura 10. Asistente para exportar a ACCESS.



6 OPCIONES DE INFILE, SET Y FILE

6.1 Opciones INFILE: lectura de ficheros planos (.txt, .dat)

PAD: añade espacios al final de los registros más cortos consiguiendo que todos los registros tengan la misma longitud.

MISSOEVER: evita que la sentencia input salte al siguiente registro cuando SAS encuentra el final de registro antes de lo esperado. Es decir, en lugar de completar las variables vacías con la información del registro siguiente, hace que dichas variables tomen el valor missing. Sin embargo, para valores de variables con longitud inferior a la especificada, esta opción no funciona correctamente ya que ignora dichos valores y considera missings.

TRUNCOVER: evita que la sentencia input salte al siguiente registro cuando SAS encuentra el final de registro antes de lo esperado dando valor missing a las variables no informadas (igual que missover), pero además, si una variable no está vacía pero el fin del registro se encuentra antes de que ésta complete su longitud, “redefine internamente” el valor de dicha variable (completando la longitud esperada) para no perder información.

TRUNCOVER = PAD + MISSOEVER

STOPOVER detiene la lectura del fichero ASCII, si se encuentran valores missing al final de una línea.

DSD cambia el comportamiento que SAS tiene con los delimitadores:

- Establece como delimitador por defecto la coma
- Considera valor missing si encuentra dos delimitadores consecutivos
- Elimina las comillas de los valores entrecomillados

LRECL = Longitud del registro. Por defecto ,SAS lee ficheros planos de hasta 256 columnas. Esta opción se utilizará para leer ficheros mayores.

FIRSTOBS = n° primera observación a leer. Por defecto FIRSTOBS=1.

OBS = n° última observación a leer. Por defecto OBS=máximo.

OBS ≥ **FIRSTOBS**

$\left\{ \begin{array}{l} \text{Firstobs} = k_1 \\ \text{Obs} = k_2 \text{ (} k_2 \geq k_1 \text{)} \end{array} \right\} \Rightarrow \text{N}^\circ \text{ de observaciones a leer} = k_2 - k_1 + 1$

Caso particular:

Si $\text{Firstobs} = 1 \rightarrow \text{Obs} = \text{N}^\circ \text{ de observaciones a leer}$

Ejemplo:

```
FILENAME FIEN "F:\ESAL\ CUESTIONARIO";
DATA CUESTIONARIO;
INFILE FIEN FIRSTOBS=100 MISOVER LRECL=711;
INPUT
  @0001 PRNSS1  $11.
  @0012 MES      $2.
  @0014 ANNO     $4.
  @0018 RAM      $2.
  @0020 EST      $1.
  @0021 NORDEN   $8.
  [...]
  @0675 E15      12.
  @0687 E16      12.
  @0699 E17      12.;
RUN;
```

6.2 Opciones SET: lectura de ficheros SAS

END= Variable lógica (0 y 1) identifica el final del fichero. Tomará el valor 1 cuando se lea la última línea del fichero SAS.

FIRSTOBS = n° primera observación a leer. Por defecto **FIRSTOBS=1**.

OBS = n° última observación a leer. Por defecto **OBS=máximo**.

OBS ≥ **FIRSTOBS**

$\left\{ \begin{array}{l} \text{Firstobs} = k_1 \\ \text{Obs} = k_2 \ (k_2 \geq k_1) \end{array} \right\} \Rightarrow \text{Nº de observaciones a leer} = k_2 - k_1 + 1$

Caso particular:

Si $\text{Firstobs} = 1 \rightarrow \text{Obs} = \text{Nº de observaciones a leer}$

Ejemplo 1:

```
DATA salidaSAS;
SET entradaSAS (OBS=10);
RUN;
```

Se leen las 10 primeras observaciones del fichero de datos SAS (entradaSAS) y se guardan en el fichero SAS de salida (salidaSAS).

Ejemplo 2:

```
DATA salidaSAS;
SET entradaSAS END=fin;
RUN;
```

Se genera una variable lógica (fin) que valdrá 1 cuando se lea el final de fichero de datos de entrada (entradaSAS).

6.3 Opciones FILE: Escritura de ficheros planos (.txt, .dat...)

LRECL = Longitud del registro. Por defecto, SAS escribe ficheros planos de hasta 256 columnas. Esta opción se utilizará para escribir ficheros mayores.

MOD escribe a continuación de la última línea del fichero en caso de existir.

OLD reemplaza el contenido del fichero, es decir, crea uno nuevo. No es necesario especificarla pues es su comportamiento por defecto.

```
FILENAME fichero 'C:\SAS\CURSOSAS\Censos2011.txt';
```

```
DATA _NULL_;  
  SET CursoSAS.CENSO2011;  
  FILE fichero LRECL=67;  
  PUT CCAA $30.  
      TOTAL 12.  
      VARON 12.  
      MUJER 12.;  
RUN;
```

7 EL PASO DATA

El paso DATA es el método principal para la generación de ficheros SAS.

Su sintaxis más frecuente es:

```
DATA ficheroSAS_salida (opciones escritura);  
  SET ficheroSAS_entrada (opciones lectura);  
  Sentencias;  
RUN;
```

El paso DATA puede utilizarse entre otros para estos propósitos:

- o Seleccionar variables
- o Renombrar variables
- o Calcular nuevas variables.
- o Reemplazar valores de una variable ya existente.
- o Filtrar datos del fichero de entrada.
- o Realizar procesos iterativos.
- o Concatenar ficheros SAS.
- o Unir ficheros SAS.

7.1 Seleccionar y renombrar variables (KEEP, DROP, RENAME)

Podemos modificar en un paso DATA las variables de un fichero SAS, bien eliminando algunas, bien quedándonos con otras y/o bien renombrando otras. Estas tres acciones se pueden realizar en la lectura, en la escritura o durante la ejecución del paso, pero siempre se deberá tener presente que no deberemos renombrar o borrar una variable que se utilice más adelante.

Para eliminar variables utilizaremos **DROP**, para escoger variables utilizaremos **KEEP** y para renombrarlas **RENAME**.

Se pueden utilizar como opciones de la lectura

```
DATA ficheroSAS_salida;  
  SET ficheroSAS_entrada (DROP=variable RENAME=nomViejo=nomNuevo);7  
RUN;
```

o como opciones de la escritura

```
DATA ficheroSAS_salida (DROP=variable RENAME=nomViejo=nomNuevo);  
  SET ficheroSAS_entrada;  
RUN;
```

o como sentencias del paso DATA:

```
DATA ficheroSAS_salida;  
  SET ficheroSAS_entrada;  
  DROP variable;  
  RENAME nomViejo=nomNuevo;8  
RUN;
```

⁷ Un KEEP en la lectura sería **SET** ficheroSAS_entrada (KEEP=variable1 variable2);

⁸ Un KEEP como sentencia en el paso DATA: **KEEP** variable1 variable2;

Ejemplo:

Supongamos que partimos del siguiente fichero CENSO2011, que tiene las variables CCAA, PROVINCIA, TOTAL, VARON, MUJER.

	CCAA	Provincia	TOTAL	Varon	Mujer
1	Andalucía	Almería	688736	350963	337772
2	Andalucía	Cádiz	1244732	620133	624599
3	Andalucía	Córdoba	802575	394487	408088
4	Andalucía	Granada	922100	457840	464260
5	Andalucía	Huelva	519895	257808	262087

Y queremos quedarnos con las variables CCAA, PROVINCIA, Varon y Mujer o lo que es lo mismo, eliminar la variable TOTAL y renombrar la variable Varon por Hombre.

Para llevar a cabo este punto podemos utilizar las sentencias KEEP, DROP y RENAME de tres maneras:

1.- Como opciones de la lectura:

```
DATA salidaSAS1;  
SET WORK.CENSO2011 (DROP=TOTAL RENAME=Varon=Hombre);  
RUN;
```

Fichero salidaSAS1:

	CCAA	Provincia	Hombre	Mujer
1	Andalucía	Almería	350963	337772
2	Andalucía	Cádiz	620133	624599
3	Andalucía	Córdoba	394487	408088
4	Andalucía	Granada	457840	464260

2.- Como opciones de la escritura:

```
DATA salidaSAS2 (DROP=Total RENAME=Varon=Hombre);  
SET censo2011;  
RUN;
```

3.- Como sentencias en el paso DATA:

```
DATA salidaSAS3;  
SET censo2011;  
DROP TOTAL;  
RENAME Varon=Hombre;  
RUN;
```

7.2 Cambiar atributos de una variable (ATTRIB)

Esta sentencia se utiliza dentro de un paso DATA para cambiar los atributos (características) de una variable.⁹

Sintaxis:

ATTRIB variable **FORMAT**=formato. **LABEL**=‘etiqueta’ **LENGTH**=<\$>longitud;¹⁰

⁹ Modifica las características definidas en la Parte del Descriptor del fichero de datos SAS.

¹⁰ FORMAT da formato a la variable. LABEL fija su etiqueta y LENGTH define su longitud, se escribirá \$ delante, si es alfanumérica.

Ejemplo: Supongamos que en un paso DATA queremos crear una variable que sea el nombre de la Comunidad Autónoma según el valor de la variable prov (nombre de la provincia). Podríamos programarlo de la siguiente forma:

```
DATA salidaSAS;
  SET entradaSAS;
  IF prov IN ('04','11','14','18','21','23','29','41') THEN ccaa='Andalucía';
  IF prov IN ('22','44','50') THEN ccaa='Aragón';
  IF prov='33' THEN ccaa='Asturias (Principado de)';
  IF prov='07' THEN ccaa='Balears (Illes)';
  IF prov IN ('35','38') THEN ccaa='Canarias';
  IF prov='39' THEN ccaa='Cantabria';
  IF prov IN ('02','13','16','19','45') THEN ccaa='Castilla y León';
  IF prov IN ('05','09','24','34','37','40','42','47','49') THEN ccaa='Castilla-La Mancha';
  IF prov IN ('08','17','25','43') THEN ccaa='Cataluña';
  IF prov IN ('03','12','46') THEN ccaa='Comunidad Valenciana';
  IF prov IN ('06','10') THEN ccaa='Extremadura';
  IF prov IN ('15','27','32','36') THEN ccaa='Galicia';
  IF prov='28' THEN ccaa='Madrid (Comunidad de)';
  IF prov='30' THEN ccaa='Murcia (Región de)';
  IF prov='31' THEN ccaa='Navarra (Comunidad Foral de)';
  IF prov IN ('01','20','48') THEN ccaa='País Vasco';
  IF prov='26' THEN ccaa='Rioja (La)';
  IF prov='51' THEN ccaa='Ceuta';
  IF prov='52' THEN ccaa='Melilla';
```

RUN;

Al no decirle a SAS, en este grupo de sentencias, ninguna pauta para crear la nueva variable CCAA, SAS se fija en el primer valor que se le asigna en el código. En nuestro caso este valor es *Andalucía*, lo que lleva a SAS a crear una variable de nombre CCAA, de tipo alfanumérica y de longitud 9 (el número de letras que componen la palabra Andalucía).

¿Cuál es el problema? Que tras la ejecución existirán registros sin el nombre de la Comunidad Autónoma completo debido a que el valor que le corresponde tiene una longitud mayor de 9. Es el caso de *Castilla y León*, *Comunidad Valenciana* etc.

DATA 1. Fichero SAS de salida. Work.salidaSAS

	prov	ccaa
1	01	País Vasc
2	02	Castilla
3	03	Comunidad
4	04	Andalucía
5	05	Castilla-
6	06	Extremadu
7	07	Balears (
8	08	Cataluña
9	09	Castilla-

Para solucionar este problema se puede utilizar la sentencia ATTRIB de la manera siguiente.

```

DATA salidaSAS;
SET entradaSAS;
ATTRIB ccaa LENGTH=$30;
IF prov IN ('04','11','14','18','21','23','29','41') THEN ccaa='Andalucía';
IF prov IN ('22','44','50') THEN ccaa='Aragón';
IF prov='33' THEN ccaa='Asturias (Principado de)';
IF prov='07' THEN ccaa='Balears (Illes)';
IF prov IN ('35','38') THEN ccaa='Canarias';
IF prov='39' THEN ccaa='Cantabria';
IF prov IN ('02','13','16','19','45') THEN ccaa='Castilla y León';
IF prov IN ('05','09','24','34','37','40','42',
            '47','49') THEN ccaa='Castilla-La Mancha';
IF prov IN ('08','17','25','43') THEN ccaa='Cataluña';
IF prov IN ('03','12','46') THEN ccaa='Comunidad Valenciana';
IF prov IN ('06','10') THEN ccaa='Extremadura';
IF prov IN ('15','27','32','36') THEN ccaa='Galicia';
IF prov='28' THEN ccaa='Madrid (Comunidad de)';
IF prov='30' THEN ccaa='Murcia (Región de)';
IF prov='31' THEN ccaa='Navarra (Comunidad Foral de)';
IF prov IN ('01','20','48') THEN ccaa='País Vasco';
IF prov='26' THEN ccaa='Rioja (La)';
IF prov='51' THEN ccaa='Ceuta';
IF prov='52' THEN ccaa='Melilla';

```

RUN;

DATA 2. Fichero SAS de salida. Work.salidaSAS

	prov	ccaa
1	01	País Vasco
2	02	Castilla y León
3	03	Comunidad Valenciana
4	04	Andalucía
5	05	Castilla-La Mancha
6	06	Extremadura
7	07	Balears (Illes)
8	08	Cataluña
9	09	Castilla-La Mancha

Nota: Cuando sólo se cambia un atributo de la variable es frecuente escribir únicamente la característica a cambiar. En nuestro caso podríamos haber escrito simplemente **LENGTH** ccaa \$30; Si, por el contrario, quisiéramos cambiar su formato utilizaríamos: **FORMAT** ccaa \$30.; y si tan solo estuviésemos interesados en darle una etiqueta: **LABEL** ccaa='Comunidad Autónoma';

7.3 Retener el valor de una variable (RETAIN)

Mediante la sentencia RETAIN podremos retener el valor de una variable de una observación o registro al siguiente.

Forma general de la sentencia RETAIN:

RETAIN variable valor-inicial...;

Ejemplo:

Supongamos que queremos conocer la suma de la variable Z a lo largo de todo el fichero. A la suma acumulada la llamaremos SUMA.

```
DATA salidaSAS;  
  SET entradaSAS;  
  RETAIN SUMA 0;  
  SUMA=SUMA+Z;  
RUN;
```

VIEWTABLE: Work.Salidasas		
	Z	SUMA
1	1	1
2	2	3
3	3	6
4	4	10
5	5	15
6	6	21
7	7	28
8	8	36
9	9	45
10	10	55

Un uso adicional, al ya visto, de la sentencia RETAIN es conseguir un orden específico de las variables en el conjunto SAS de salida de un paso DATA.

DATA 3. Uso adicional sentencia RETAIN. Conseguimos cambiar el orden de aparición de las variables.

VIEWTABLE: SasHELP.Class					
	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5

VIEWTABLE: Work.Clase					
	SEX	AGE	NAME	Height	Weight
1	M	14	Alfred	69	112.5
2	F	13	Alice	56.5	84
3	F	13	Barbara	65.3	98
4	F	14	Carol	62.8	102.5

```
DATA CLASE;  
  RETAIN SEX AGE NAME;  
  SET SASHELP.CLASS;  
RUN;
```

7.4 Sentencias condicionales

Una sentencia condicional es una instrucción o grupo de instrucciones que se pueden ejecutar o no en función del valor de una condición.

El tipo más conocido de sentencia condicional es:

```
SI condicionA ENTONCES accionA;  
EN CASO CONTRARIO SI condicionB ENTONCES accionB;  
.....  
EN CASO CONTRARIO accionZ;
```

Con las sentencias correspondientes:

```
IF condicionA THEN accionA;  
ELSE IF condicionB THEN accionB;  
.....  
ELSE accionZ;
```

O

```
IF condicionA THEN DO; accionA1; ... accionAn; END;  
ELSE IF condicionB THEN DO; accionB1; ... accionBn; END;  
.....  
ELSE DO; accionZ1; ... accionZn; END;
```

Algunas de las acciones que se pueden llevar a cabo con las sentencias condicionales son:

- Crear una variable que tome valores según el valor de una condición.
- Filtrar datos de un fichero de entrada quedándonos con aquellos que cumplan una determinada condición.
- Generar varios ficheros de salida a la vez teniendo en cuenta una condición lógica.

Veamos cada una de estas acciones con un ejemplo.

Crear una variable según el valor de una condición

Ejemplo: Calcular los ingresos según destino del vuelo partiendo del fichero SAS, CursoSAS.MADBCN,

	Flight	Date	Dest	FirstClass	Economy
1	439	11/12/2000	MAD	20	137
2	921	11/12/2000	BCN	20	131
3	114	12/12/2000	MAD	15	170
4	982	12/12/2000	BCN	5	85
5	439	13/12/2000	MAD	14	196
6	982	13/12/2000	BCN	15	116
7	431	14/12/2000	MAD	17	166
8	982	14/12/2000	BCN	7	88
9	114	15/12/2000	MAD	.	187
10	982	15/12/2000	BCN	14	31

y teniendo en cuenta la tabla siguiente:

DESTINO	CLASE	PRECIO
MAD	First	250
	Economy	100
BCN	First	260
	Economy	90

Sentencias SAS:

```
DATA ingresos;
  SET cursoSAS.MADBCN;
  IF dest='MAD' THEN
    Ingresos=sum (250*FirstClass, 100*Economy);
  ELSE IF dest='BCN' THEN
    Ingresos=sum (260*FirstClass,90*Economy);
  FORMAT Ingresos commax10.;
RUN;
```

Fichero SAS de salida **WORK.ingresos**:

	Flight	Date	Dest	FirstClass	Economy	Ingresos
1	439	11/12/2000	MAD	20	137	18.700
2	921	11/12/2000	BCN	20	131	16.990
3	114	12/12/2000	MAD	15	170	20.750
4	982	12/12/2000	BCN	5	85	8.950
5	439	13/12/2000	MAD	14	196	23.100
6	982	13/12/2000	BCN	15	116	14.340
7	431	14/12/2000	MAD	17	166	20.850
8	982	14/12/2000	BCN	7	88	9.740
9	114	15/12/2000	MAD	.	187	18.700
10	982	15/12/2000	BCN	14	31	6.430

Nota: El valor ausente en FirstClass del vuelo 114 no afecta al cálculo de la función SUM().

Si hiciésemos varias acciones para una misma condición el código sería:

```
DATA ingresos;
  SET CursoSAS.MADBCN;
  LENGTH Ciudad $9.;
  IF dest='MAD' THEN DO;
    Ingresos=sum (250*FirstClass, 100*Economy);
    Ciudad= 'Madrid';
  END;
  ELSE IF dest='BCN' THEN DO;
    Ingresos=sum (260*FirstClass,90*Economy);
    Ciudad= 'Barcelona';
  END;
  FORMAT Ingresos commax10.;
RUN;
```

Flight	Date	Dest	FirstClass	Economy	Ciudad	Ingresos
439	11/12/2000	MAD	20	137	Madrid	18.700
921	11/12/2000	BCN	20	131	Barcelona	16.990
114	12/12/2000	MAD	15	170	Madrid	20.750
982	12/12/2000	BCN	5	85	Barcelona	8.950
439	13/12/2000	MAD	14	196	Madrid	23.100
982	13/12/2000	BCN	15	116	Barcelona	14.340
431	14/12/2000	MAD	17	166	Madrid	20.850
982	14/12/2000	BCN	7	88	Barcelona	9.740
114	15/12/2000	MAD	.	187	Madrid	18.700
982	15/12/2000	BCN	14	31	Barcelona	6.430

Filtrar datos de un fichero SAS de entrada

Siguiendo con los usos de la sentencia IF, ahora vamos a ver cómo **filtrar datos** de un fichero SAS de entrada en función de si se cumple o no una condición.

Ejemplo: Partiendo del fichero anterior, cursosas.MADBCN, supongamos que queremos construir un fichero sólo con los vuelos con destino Barcelona (Dest='BCN'). Para llevar a cabo este ejercicio, existen dos opciones equivalentes de programación; una utilizando la opción IF-DELETE y otra usando simplemente IF.

```
DATA SoloBCN;
SET cursosas.MADBCN;
IF Dest='MAD' THEN DELETE;
RUN;
```

```
DATA SoloBCN;
SET cursosas.MADBCN;
IF Dest ne 'MAD';
RUN;
```

Fichero SAS de salida: **WORK.SoloBCN**

	Flight	Date	Dest	FirstClass	Economy
1	921	11/12/2000	BCN	20	131
2	982	12/12/2000	BCN	5	85
3	982	13/12/2000	BCN	15	116
4	982	14/12/2000	BCN	7	88
5	982	15/12/2000	BCN	14	31

Generar varios ficheros de salida a la vez teniendo en cuenta una condición lógica.

Ejemplo: Supongamos que ahora queremos generar, a partir del fichero de entrada cursoSAS.MADBCN, dos ficheros de salida: uno con los vuelos de destino Madrid y otro con los vuelos de destino Barcelona.

```
DATA SoloMAD SoloBCN;
SET cursosas.MADBCN;
IF Dest='MAD' THEN OUTPUT SoloMAD;
IF Dest='BCN' THEN OUTPUT SoloBCN;
RUN;
```

Fichero SAS entrada

VIEWTABLE: Cursosas.Madbcn					
	Flight	Date	Dest	FirstClass	Economy
1	439	11/12/2000	MA	20	137
2	921	11/12/2000	BCN	20	131
3	114	12/12/2000	MA	15	170
4	982	12/12/2000	BCN	5	85
5	439	13/12/2000	MA	14	196
6	982	13/12/2000	BCN	15	116
7	431	14/12/2000	MA	17	166
8	982	14/12/2000	BCN	7	88
9	114	15/12/2000	MA	.	187
10	982	15/12/2000	BCN	14	31

Fichero SAS salida

VIEWTABLE: Work.Solobcn					
	Flight	Date	Dest	FirstClass	Economy
1	921	11/12/2000	BCN	20	131
2	982	12/12/2000	BCN	5	85
3	982	13/12/2000	BCN	15	116
4	982	14/12/2000	BCN	7	88
5	982	15/12/2000	BCN	14	31

Fichero SAS salida

VIEWTABLE: Work.Solomad					
	Flight	Date	Dest	FirstClass	Economy
1	439	11/12/2000	MAD	20	137
2	114	12/12/2000	MAD	15	170
3	439	13/12/2000	MAD	14	196
4	431	14/12/2000	MAD	17	166
5	114	15/12/2000	MAD	.	187

7.5 Filtrar datos con WHERE

En este apartado veremos otra manera de filtrar datos dentro de un paso DATA.

La expresión **WHERE** se utiliza para seleccionar dentro de un conjunto de datos SAS observaciones que cumplen una determinada condición.

Ejemplo: DATA alumnos;
SET SASHELP.CLASS;
WHERE age >14;
RUN;

Fichero SAS de salida Work.Alumnos

VIEWTABLE: Work.Alumnos						
	Name	Sex	Age	Height	Weight	
1	Janet	F	15	62.5	112.5	
2	Mary	F	15	66.5	112	
3	Philip	M	16	72	150	
4	Ronald	M	15	67	133	
5	William	M	15	66.5	111	

WHERE age >14;

La **ventaja** de utilizar WHERE para filtrar datos y no IF está en que, con la sentencia WHERE, SAS no lee el fichero completo sino que sólo lee aquellos registros que cumplen la condición. Esto supone un ahorro en tiempo de ejecución sobre todo cuando manipulamos ficheros de gran tamaño.

Los operadores que podrán utilizarse para formar la condición lógica del WHERE son:

DATA 4. Cuadro de operadores para cláusula WHERE.

TIPO DE OPERADOR	SÍMBOLO	DESCRIPCIÓN
Aritméticos		
	*	multiplicación
	/	división
	+	suma
	-	resta
	**	exponente
De comparación		
	= or EQ	igual
	^=, ^=, ^=, or NE	distinto
	> or GT	mayor que
	< or LT	menor que
	>= or GE	mayor o igual que
	<= or LE	menor o igual que
	IN	igual a un valor de una lista
Lógicos		
	& or AND	y - lógico
	or OR	o - lógico
	~, ^, ^, or NOT	o no - lógico
Otros		
	,	concatenación de variables tipo carácter
	BETWEEN-AND	dentro de un rango concreto de valores
	? or CONTAINS	contener cadena de caracteres
	IS NULL or IS MISSING	valores perdidos
	LIKE	patrones coincidentes
	=*	"suena como"
	SAME-AND	añade una cláusula a una sentencia WHERE ya existente sin escribirla de nuevo

NOTA: Salvo el operador de *concatenación*, el resto de operadores clasificados como "OTROS" son exclusivos de la expresión WHERE.

Ejemplos:

- **BETWEEN – AND:** selecciona las observaciones para las que los valores de las variables se encuentran dentro de un rango de valores.

`WHERE` variable BETWEEN valor1 AND valor2;

- **CONTAINS o ?:** Selecciona las observaciones que incluyen la cadena de caracteres especificada. Sólo admite variables de tipo carácter.

`WHERE` variable ? 'cadena';
`WHERE` variable CONTAINS 'cadena';

- **LIKE 'M_I%':** _ representa un carácter cualquiera y % indica un número de caracteres indefinido..

`WHERE` variable LIKE 'M_I%';

Por lo que nuestro ejemplo buscaría valores de la variable que comiencen por M seguido de un carácter cualquiera, a continuación una "I" y después cualquier cadena de caracteres. Por ejemplo, en caso de existir, seleccionaría valores como Melilla, Málaga etc..

Señalar que SAS diferencia entre mayúsculas y minúsculas cuando se trata de valores de variables.

- **IS NULL o IS MISSING:** Selecciona todas las observaciones para las que la variable es missing.

`WHERE` variable IS NULL;
`WHERE` variable IS MISSING;
`WHERE` variable IS NOT NULL;
`WHERE` variable IS NOT MISSING;

7.6 Procesos iterativos (bucle DO)

En este capítulo veremos cómo generar un proceso iterativo a través de un bucle DO acompañada de una variable índice.

Se pueden usar bucles DO para:

- Realizar cálculos repetitivos
- Generar conjuntos de datos SAS
- Eliminar código redundante

Sintaxis:

```
DO variable-índice = inicio TO final BY incremento;  
    Sentencias SAS  
END;
```

Las sentencias dentro de un bucle DO se ejecutan un número específico de veces (nº indicado por la variable índice).

Los valores de inicio, final e incremento:

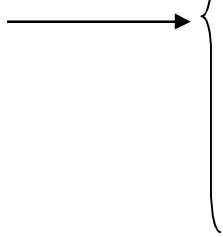
- Se deben fijar antes de la entrada al bucle
- No se pueden cambiar durante la ejecución del proceso iterativo
- Pueden ser números, variables o expresiones SAS.

El valor de la variable-índice sí se puede cambiar durante la ejecución del bucle.

Por defecto el incremento es 1 si se omite la sentencia BY.

Ejemplo 1: Si quisiéramos generar un fichero de salida con 10 números aleatorios, podríamos escribir un código SAS como este.

```
DATA aleatorio;  
aleatorio= ranuni(0); output;  
aleatorio= ranuni(0); output;  
aleatorio= ranuni(0); output;  
aleatorio= ranuni(0); output;  
aleatorio= ranuni(0); output;  
aleatorio= ranuni(0); output;  
aleatorio= ranuni(0); output;  
aleatorio= ranuni(0); output;  
aleatorio= ranuni(0); output;  
aleatorio= ranuni(0); output;  
RUN;
```



	aleatorio
1	0.774545831
2	0.0777920424
3	0.7365634515
4	0.4128596058
5	0.6779469613
6	0.5570747641
7	0.9532969692
8	0.9470777414
9	0.2029307635
10	0.0441963384

Sin embargo, podríamos reducir el código anterior utilizando un bucle DO de la manera siguiente:

```
DATA aleatorio;  
DO i =1 TO 10; /*variable índice*/  
    aleatorio= ranuni(0); output;  
END;  
DROP i;  
RUN;
```

Podemos también ejecutar el proceso iterativo sólo para ciertos valores de una lista:

```
DO variable-índice = valor1, valor2, valor3...;  
    Sentencias SAS  
END;
```

Los valores pueden ser numéricos o carácter.

Ejemplo 2: Otra opción podría haber sido ejecutar el proceso iterativo para una lista de meses del año.

```
DATA aleatorios;  
LENGTH mes $10;  
DO mes='Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre';  
    aleatorio= ranuni(0); output;  
END;  
RUN;
```

Fichero SAS de salida:

VIEWTABLE: Work.Aleatorios		
	mes	aleatorio
1	Enero	0.0594182969
2	Febrero	0.8009966671
3	Marzo	0.4712020287
4	Abril	0.8884501722
5	Mayo	0.7176847149
6	Junio	0.9688235749
7	Julio	0.3277143772
8	Agosto	0.3005963193
9	Septiembre	0.6589419137
10	Octubre	0.8310119257

7.7 Concatenar ficheros SAS

Otra tarea que se puede realizar con un paso DATA es concatenar conjuntos de datos SAS.

Para concatenar conjuntos de datos SAS basta con escribir uno detrás de otro mediante una sentencia SET.

Sintaxis:

```
DATA salidaSAS;  
  SET entradaSAS1 entradaSAS2.....;  
RUN;
```

Ejemplo: Partiendo de dos fichero SAS, work.A y work.B, se genera un tercer fichero work.AB resultado de la concatenación de los dos primeros.

Ficheros SAS de entrada:

VIEWTABLE: Work.A		
	Num	VARA
1	1	A1
2	2	A2
3	3	A3

VIEWTABLE: Work.B		
	Num	VARB
1	4	B1
2	5	B2
3	6	B3

Paso DATA:

```
DATA AB;  
  SET A B;  
RUN;
```

Fichero SAS de salida:

VIEWTABLE: Work.Ab			
	Num	VARA	VARB
1	1	A1	
2	2	A2	
3	3	A3	
4	4		B1
5	5		B2
6	6		B3

8 PROC SORT

Antes de explicar cómo unir conjuntos de datos SAS es necesario introducir teóricamente el procedimiento utilizado para ordenar conjuntos de datos SAS, PROC SORT.

Sintaxis

```
PROC SORT DATA=datos <opciones>;  
      BY (DESCENDING) <variables>;  
RUN;
```

Opciones

DATA= Conjunto de datos SAS que se desea ordenar. Si se omite será el último conjunto de datos SAS válido.

OUT= Conjunto de datos SAS de salida ordenado. Si se omite será el mismo que el indicado en la opción DATA.

NODUP: Eliminación de observaciones duplicadas.

NODUPKEYS: Elimina las observaciones con valores repetidos en las variables de ordenación.

Sentencias

BY variables; Variables de ordenación. Por defecto se realizará en orden ascendente. Si alguna variable se desea ordenar en descendente la sintaxis será **DESCENDING** variable.

Como ejemplo, supongamos que quisiéramos ordenar un conjunto de datos SAS por una variable V1 en orden ascendente y por V2 en descendente, en ese caso la sentencia sería: **BY V1 DESCENDING V2;**

9 UNIÓN DE CONJUNTOS SAS (MERGE)

Con la sentencia MERGE se pueden combinar observaciones de dos o más conjuntos de datos SAS.

Sintaxis general:

```
DATA salidaSAS;  
  MERGE entradaSAS1 entradaSAS2;  
  BY variablecomun;  
  <otras sentencias ...>;  
RUN;
```

Para unir los conjuntos de datos según la variable común indicada en la sentencia **BY** los conjuntos de datos deben estar ordenados por dicha variable, de modo que un paso previo a la unión de ficheros debe ser la ordenación de los mismos:

```
PROC SORT DATA=entradaSAS1; BY variablecomun; RUN;  
PROC SORT DATA=entradaSAS2; BY variablecomun; RUN;
```

El conjunto de datos SAS de salida contiene todas las variables que se encuentran en los conjuntos de datos mezclados.

Importante: Si al unir dos conjuntos de datos SAS existe alguna variable del mismo nombre en ambos que no es la de ordenación, los valores de dicha variable correspondientes al conjunto de datos escrito más a la izquierda serán reemplazados por los valores de la variable del conjunto más a la derecha.

Para conocer **cómo actúa el MERGE**, vamos a ver las 5 posibilidades de correspondencia que existen entre 2 ficheros:

- **1 a ninguno o viceversa:** ambos conjuntos de datos SAS tienen valores únicos en la variable indicada en la sentencia BY y estos valores son diferentes en cada fichero.

Ejemplo 1:

datos1	
NUM	VARA
1	A1
2	A2
5	A3

datos2	
NUM	VARB
3	B1
4	B2
6	B3

```
PROC SORT DATA=DATOS1; BY NUM; RUN;  
PROC SORT DATA=DATOS2; BY NUM; RUN;
```

```
DATA MERGE1;  
MERGE DATOS1 DATOS2;  
BY NUM;  
RUN;
```


MERGE 1. Fichero de salida MERGE1

merge datos1 datos2		
NUM	VARA	VARB
1	A1	
2	A2	
3		B1
4		B2
5	A3	
6		B3

- **1 a 1:** ambos conjuntos de datos SAS tienen valores únicos en la variable indicada en la sentencia BY y estos valores son iguales en los dos ficheros.

Ejemplo 2:

datos1		datos2	
NUM	VARA	NUM	VARB
1	A1	1	B1
2	A2	2	B2
3	A3	3	B3

```
PROC SORT DATA=DATOS1; BY NUM; RUN;  
PROC SORT DATA=DATOS2; BY NUM; RUN;
```

```
DATA MERGE2;  
MERGE DATOS1 DATOS2;  
BY NUM;  
RUN;
```

MERGE 2. Fichero de salida MERGE2

merge datos1 datos2		
NUM	VARA	VARB
1	A1	B1
2	A2	B2
3	A3	B3

- **1 a varios o viceversa:** uno de los dos conjuntos de datos tiene valores únicos en la variable indicada en la sentencia BY pero existen valores que se corresponden a estos en el otro conjunto de datos que sí están repetidos.

Ejemplo 3:

datos1		datos2	
NUM	VARA	NUM	VARB
1	A1	2	B1
2	A2	2	B2
3	A3	3	B3

```
PROC SORT DATA=DATOS1; BY NUM; RUN;  
PROC SORT DATA=DATOS2; BY NUM; RUN;
```

```
DATA MERGE3;  
MERGE DATOS1 DATOS2;  
BY NUM;  
RUN;
```

MERGE 3. Fichero SAS de salida MERGE3

merge datos1 datos2		
NUM	VARA	VARB
1	A1	
2	A2	B1
2	A2	B2
3	A3	B3

- o **varios a ninguno o viceversa:** ambos conjuntos de datos tienen valores duplicados en la variable indicada en la sentencia BY y éstos son diferentes en ambos conjuntos.

Ejemplo 4:

datos1	
NUM	VARA
2	A1
3	A2
3	A3

datos2	
NUM	VARB
1	B1
4	B2
4	B3

```
PROC SORT DATA=DATOS1; BY NUM; RUN;  
PROC SORT DATA=DATOS2; BY NUM; RUN;
```

```
DATA MERGE4;  
MERGE DATOS1 DATOS2;  
BY NUM;  
RUN;
```

MERGE 4. Fichero SAS de salida MERGE4

merge datos1 datos2		
NUM	VARA	VARB
1		B1
2	A1	
3	A2	
3	A3	
4		B2
4		B3

- o **varios a varios:** ambos conjuntos de datos tienen valores duplicados en la variable indicada en la sentencia BY y éstos coinciden en ambos conjuntos.

En este caso se obtiene una NOTA en la ventana LOG como la siguiente:

NOTA: La sentencia MERGE tiene más de un conjunto de datos con repeticiones de valores BY.

o

NOTE: MERGE statement has more than one data set with repeats of BY values.

Ejemplo 5:

datos1				datos2	
NUM	VARA			NUM	VARB
2	A1			1	B1
3	A2			3	B2
3	A3			3	B3

```
PROC SORT DATA=DATOS1; BY NUM; RUN;
PROC SORT DATA=DATOS2; BY NUM; RUN;
```

```
DATA MERGE5;
MERGE DATOS1 DATOS2;
BY NUM;
RUN;
```

MERGE 5. Fichero SAS de salida MERGE5

merge datos1 datos2		
NUM	VARA	VARB
1		B1
2	A1	
3	A2	B2
3	A3	B3

Observación: En este ejemplo, si cambiamos el orden de las observaciones en uno de los ficheros de entrada, por ejemplo en **datos1**, el resultado del **MERGE** es diferente al que se acaba de obtener:

datos1				datos2	
NUM	VARA			NUM	VARB
2	A1			1	B1
3	A3			3	B2
3	A2			3	B3

```
PROC SORT DATA=DATOS1; BY NUM; RUN;
PROC SORT DATA=DATOS2; BY NUM; RUN;
```

```
DATA MERGE5_BIS;
MERGE DATOS1 DATOS2;
BY NUM;
RUN;
```

MERGE 6. Fichero de salida MERGE5_BIS

merge datos1 datos2		
NUM	VARA	VARB
1		B1
2	A1	
3	A3	B2
3	A2	B3

Identificación de los conjuntos de datos que forman las observaciones

Cuando se unen distintos conjuntos de datos SAS en un solo paso DATA, se puede usar la opción **IN=** para saber qué conjunto de datos de entrada contribuye a cada observación del fichero de salida.

La sintaxis es la siguiente:

```
PROC SORT DATA=Datos1; BY variablecomun; RUN;
PROC SORT DATA=Datos2; BY variablecomun; RUN;

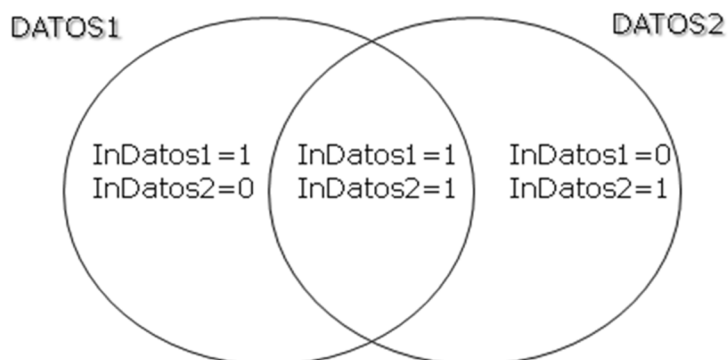
DATA salidaSAS;
MERGE Datos1 (IN=inDatos1)
      Datos2 (IN=inDatos2);
BY variablecomun;
RUN;
```

- **InDatos1** y **InDatos2** son nombres de variables SAS válidos.
- Las variables creadas con la opción **IN=** son variables numéricas temporales, sólo estarán disponibles durante la ejecución y no se van a escribir en **salidaSAS**.
 - **InDatos1** toma el valor 1 ó 0 dependiendo de si **Datos1** ha formado o no la observación actual.
 - **InDatos2** toma el valor 1 ó 0 dependiendo de si **Datos2** ha formado o no la observación actual.
- Por lo tanto, el programador aporta el nombre de la variable en la opción **IN=** y el sistema SAS aporta su valor.

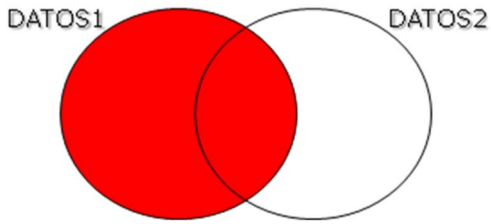
Detallamos a continuación los distintos casos que pueden presentarse a la hora de unir horizontalmente **datos1** y **datos2** dependiendo de los valores que toman las variables temporales detalladas en la opción **IN=** y de los operadores lógicos AND y OR aplicándolo al ejemplo 5 (varios a varios):

datos1	
NUM	VARA
2	A1
3	A2
3	A3

datos2	
NUM	VARB
1	B1
3	B2
3	B3



MERGE 7. Unión ficheros SAS. Caso 1.

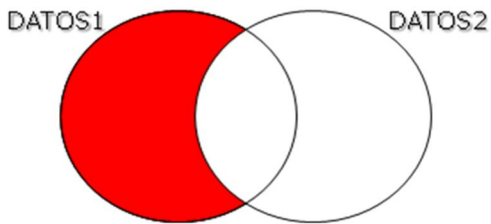


```
PROC SORT DATA=DATOS1; BY NUM; RUN;
PROC SORT DATA=DATOS2; BY NUM; RUN;

DATA MERGEDATOS;
MERGE DATOS1 (IN=InDATOS1)
      DATOS2 (IN=InDATOS2);
BY NUM;
IF InDATOS1=1;
RUN;
```

if indatos1=1		
merge datos1 datos2		
NUM	VARA	VARB
2	A1	
3	A2	B2
3	A3	B3

MERGE 8. Unión ficheros SAS. Caso 2.

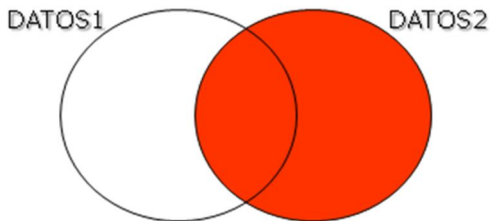


```
PROC SORT DATA=DATOS1; BY NUM; RUN;
PROC SORT DATA=DATOS2; BY NUM; RUN;

DATA MERGEDATOS;
MERGE DATOS1 (IN=InDATOS1)
      DATOS2 (IN=InDATOS2);
BY NUM;
IF InDATOS1=1 and InDATOS2=0;
RUN;
```

if indatos1=1 and indatos2=0		
merge datos1 datos2		
NUM	VARA	VARB
2	A1	

MERGE 9. Unión ficheros SAS. Caso 3.

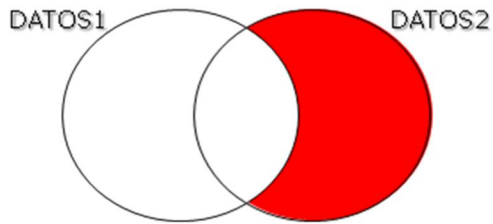


```
PROC SORT DATA=DATOS1; BY NUM; RUN;
PROC SORT DATA=DATOS2; BY NUM; RUN;

DATA MERGEDATOS;
MERGE DATOS1 (IN=InDATOS1)
      DATOS2 (IN=InDATOS2);
BY NUM;
IF InDATOS2=1;
RUN;
```

if indatos2=1		
merge datos1 datos2		
NUM	VARA	VARB
1		B1
3	A2	B2
3	A3	B3

MERGE 10. Unión ficheros SAS. Caso 4.

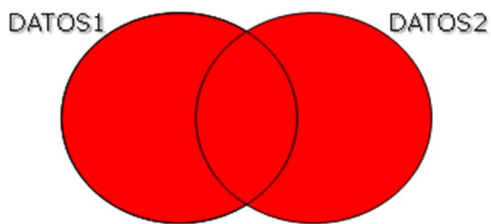


```
PROC SORT DATA=DATOS1; BY NUM; RUN;
PROC SORT DATA=DATOS2; BY NUM; RUN;
```

```
DATA MERGEDATOS;
MERGE DATOS1 (IN=InDATOS1)
      DATOS2 (IN=InDATOS2);
BY NUM;
IF InDATOS1=0 and InDATOS2=1;
RUN;
```

if indatos1=0 and indatos2=1		
merge datos1 datos2		
NUM	VARA	VARB
1		B1

MERGE 11. Unión ficheros SAS. Caso 5.

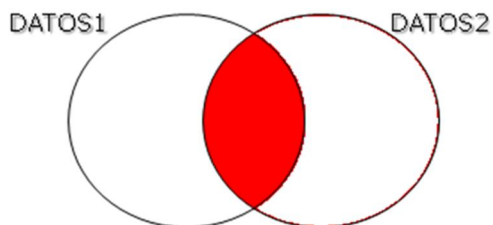


```
PROC SORT DATA=DATOS1; BY NUM; RUN;
PROC SORT DATA=DATOS2; BY NUM; RUN;
```

```
DATA MERGEDATOS;
MERGE DATOS1 (IN=InDATOS1)
      DATOS2 (IN=InDATOS2);
BY NUM;
IF InDATOS1=1 or InDATOS2=1;
RUN;
```

if indatos1=1 or indatos2=1		
merge datos1 datos2		
NUM	VARA	VARB
1		B1
2	A1	
3	A2	B2
3	A3	B3

MERGE 12. Unión ficheros SAS. Caso 6.

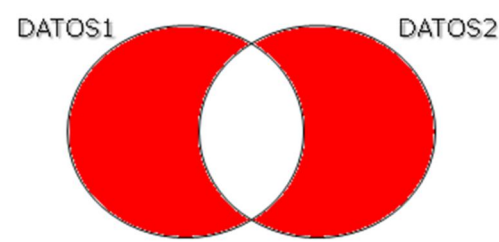


```
PROC SORT DATA=DATOS1; BY NUM; RUN;
PROC SORT DATA=DATOS2; BY NUM; RUN;
```

```
DATA MERGEDATOS;
MERGE DATOS1 (IN=InDATOS1)
      DATOS2 (IN=InDATOS2);
BY NUM;
IF InDATOS1=1 and InDATOS2=1;
RUN;
```

if indatos1=1 and indatos2=1		
merge datos1 datos2		
NUM	VARA	VARB
3	A2	B2
3	A3	B3

MERGE 13. Unión ficheros SAS. Caso 7.



```
PROC SORT DATA=DATOS1;BY NUM;RUN;
PROC SORT DATA=DATOS2;BY NUM;RUN;

DATA MERGEDATOS;
MERGE DATOS1 (IN=InDATOS1)
      DATOS2 (IN=InDATOS2);
BY NUM;
IF InDATOS1+InDATOS2=1;
RUN;
```

IF ((INDATOS1=1 AND INDATOS2=0) OR
(INDATOS1=0 AND INDATOS2=1));

if indatos1+indatos2=1		
merge datos1 datos2		
NUM	VARA	VARB
1		B1
2	A1	

9.1 Sentencia BY. Variables FIRST Y LAST.

En el apartado anterior hemos utilizado la sentencia BY, para unir conjuntos de datos SAS mediante MERGE. Bien pues en este apartado veremos otro uso de esta sentencia frecuentemente utilizado.

Importante: Siempre que utilicemos la sentencia BY, ya sea en un SET o con un MERGE, el o los conjuntos de datos de entrada SAS han estar ordenados por la variable o variables indicadas en el BY. Para ello se utiliza el procedimiento PROC SORT.

Ejemplo: Supongamos que partimos del fichero SAS work.Alumnos siguiente:

	Clase	Nombre
1	1ªA	Tomás
2	1ªA	Luis
3	1ªA	Jorge
4	1ªA	María
5	1ªB	Alejandra
6	1ªB	Sara
7	1ªB	Juan
8	1ªB	Alejandra
9	2ªA	Patricia
10	2ªB	Inés
11	2ªB	Carlos

y escribimos código:

```
PROC SORT DATA=Alumnos; BY clase; RUN;  
DATA Alumnos;  
SET Alumnos;  
BY clase;  
RUN;
```

Al ejecutarse la sentencia `BY clase;` automáticamente se crean unas variables temporales denominadas **FIRST.clase** y **LAST.clase** que pueden ser utilizadas para procesar condicionalmente los datos clasificados o agrupados por *clase*.

¿Qué valores toman estas dos variables?

- La variable **FIRST.clase** toma el valor 1 para el primer registro dentro de su grupo de clasificación, en nuestro caso la clase, y 0 en el resto de los casos.
- La variable **LAST.clase** toma el valor 1 para el último registro dentro del grupo de clasificación, en nuestro caso clase, y 0 en el resto de los casos.

Para visualizar su *funcionamiento*, modificaremos el paso DATA anterior guardando los valores de las variables temporales FIRST.clase y LAST.clase en dos variables nuevas PrimerReg y UltimoReg.

```
PROC SORT DATA=Alumnos; BY clase; RUN;
```

```
DATA Alumnos;  
SET Alumnos;  
BY clase;  
PrimerReg=First.clase; *guardamos el valor de las variables temporales FIRST y LAST;  
UltimoReg=Last.clase;  
RUN;
```


El fichero SAS de salida que obtendríamos sería work.Alumnos y tendría el siguiente aspecto:

	Clase	Nombre	PrimerReg	UltimoReg
1	1ªA	Tomás	1	0
2	1ªA	Luis	0	0
3	1ªA	Jorge	0	0
4	1ªA	María	0	1
5	1ªB	Alejandra	1	0
6	1ªB	Sara	0	0
7	1ªB	Juan	0	0
8	1ªB	Alejandra	0	1
9	2ªA	Patricia	1	1
10	2ªB	Inés	1	0
11	2ªB	Carlos	0	1

Como se puede observar la variable PrimerReg, que guarda los valores de **First.Clase**, toma el valor 1 para el primer registro de la clase y 0 para el resto y la variable UltimoReg, que guarda los valores de **Last.clase**, toma el valor 1 para el último registro de la clase.

Ahora veamos una utilidad que se puede obtener con la sentencia BY y las variables temporales FIRST y LAST.

Ejemplo: Dado el conjunto de datos SAS, CursoSAS.AlumnosSG.

	subdireccion	nombre
1	Gabinete Presidencia	Margarita
2	S.G.Censos y Padrón	Concha
3	S.G.Informática	Maria Luisa
4	S.G.Mercado Laboral	Luis
5	S.G.Mercado Laboral	Alberto
6	S.G.Mercado Laboral	Esther
7	S.G.Mercado Laboral	Beatriz
8	S.G.Metodología	Jose Manuel
9	S.G.Precios	Teresa
10	S.G.Precios	Alberta
11	S.G.Precios	Alicia
12	S.G.Recogida de Datos	Eva
13	S.G.Servicios	Mercedes
14	S.G.Servicios	Ana María
15	S.G.Servicios	Marta
16	S.G.Sociales	Carmen

Supongamos que queremos generar una variable nueva (que llamaremos contador) que cuente el número de personas asistentes al curso de SAS por Subdirección (variable subdirección).

Para llevar a cabo esta tarea utilizaremos:

1. Una sentencia BY subdireccion; para poder hacer uso de las variables temporales FIRST.subdireccion y LAST.subdireccion teniendo en cuenta sus valores.
2. Una sentencia RETAIN.
3. Una sentencia condicional del tipo IF-THEN-ELSE.

Las líneas de código SAS serían las siguientes:

*Ordenamos alumnosSG por subdireccion;

PROC SORT DATA= CursoSAS.alumnosSG; **BY** subdireccion; **RUN**;

*Paso DATA;

DATA AlumnosSG;

SET CursoSAS.AlumnosSG;

BY subdireccion;

RETAIN contador;

IF FIRST.subdireccion=**1** **THEN** contador=**1**;

ELSE contador=contador+**1**;

RUN;

Retenemos el valor de *contador* de un registro a otro.

Iniciamos *contador* a 1 al encontrarnos con el primer registro de la subdirección.

Si las ejecutásemos y guardásemos las variables temporales FIRST.subdireccion y LAST.subdireccion para ver sus valores y poder entender, de una manera más clara, el código anterior, obtendríamos el siguiente conjunto de datos SAS:

Fichero SAS de salida **work.AlumnosSG** con las variables contador, FIRST.subdireccion y LAST.subdireccion

	contador	FIRST.subdireccion	LAST.subdireccion	subdireccion	nombre
1	1	1	1	Gabinete Presidencia	Margarita
2	1	1	1	S.G.Censos y Padrón	Concha
3	1	1	1	S.G.Informática	Maria Luisa
4	1	1	0	S.G.Mercado Laboral	Luis
5	2	0	0	S.G.Mercado Laboral	Alberto
6	3	0	0	S.G.Mercado Laboral	Esther
7	4	0	1	S.G.Mercado Laboral	Beatriz
8	1	1	1	S.G.Metodología	Jose Manuel
9	1	1	0	S.G.Precios	Teresa
10	2	0	0	S.G.Precios	Alberta
11	3	0	1	S.G.Precios	Alicia
12	1	1	1	S.G.Recogida de Datos	Eva
13	1	1	0	S.G.Servicios	Mercedes
14	2	0	0	S.G.Servicios	Ana María
15	3	0	1	S.G.Servicios	Marta
16	1	1	1	S.G.Sociales	Carmen

Por último comentaremos que se puede especificar más de una variable en la sentencia BY y que por lo tanto para cada una de las variables especificadas en el BY se crearán dos variables temporales FIRST y LAST.

Sintaxis general:

DATA SalidaSAS;

SET EntradaSAS;

BY variable1 variable2;

IF FIRST.variable1 **THEN** ...;

ELSE IF FIRST.variable2 **THEN** ...;

RUN;

A modo de resumen recordaremos que:

- ❑ La sentencia BY se usa para el control de las operaciones realizadas por las sentencias SET y MERGE.
- ❑ Sólo una sentencia BY puede aplicarse a cada SET y MERGE.
- ❑ El o los conjuntos de datos SAS indicados en SET o MERGE deben estar ordenados por las mismas variables que se especifican en la sentencia BY, por lo que se precisa un PROC SORT previo.
- ❑ Si se utiliza la sentencia BY con un SET para leer dos o más conjuntos de datos SAS, se intercalarán los valores de todos ellos para formar cada uno de los grupos, es decir no se realiza una concatenación pura.

10 OPCIONES GLOBALES. SENTENCIA OPTIONS.

Mediante la sentencia **OPTIONS** podremos modificar las opciones globales que por defecto se encuentran en la configuración de SAS. Veamos algunas de ellas:

NOCENTER: para evitar que la salida a la ventana OUTPUT se centre en la ventana. Por defecto la opción es CENTER.

NODATE: para evitar la impresión de la fecha en la ventana OUTPUT. Por defecto la opción es DATE.

NONUMBER: para evitar la numeración de páginas en la impresión de la ventana OUTPUT. Se debe tener presente que la ventana OUTPUT acumula las salidas, y aunque la borremos, a lo largo de nuestra sesión SAS las páginas que deseemos imprimir no serán de la 1 en adelante. Por defecto es NUMBER.

ERRORS = n° número máximo de errores que se mostrarán en el LOG. Por defecto **ERRORS** = 20. Si se detectan más errores que el número máximo fijado, el proceso continúa, pero SAS no muestra mensajes de ERROR para los errores adicionales.

FIRSTOBS = n° primera observación a leer en un paso. Por defecto **FIRSTOBS** = 1

LINESIZE = n° número de posiciones por línea de la ventana OUTPUT. Con esta opción conseguiremos colocar toda la información deseada en líneas completas de la ventana OUTPUT.

PAGESIZE = n° número de líneas por página de la ventana OUTPUT. Con esta opción evitaremos, según la configuración de nuestra impresora, saltos de página indeseables en el momento de la impresión.

Ejemplo de sintaxis:

```
OPTIONS NOCENTER NODATE LINESIZE=100;
```

Las opciones globales se mantendrán activas durante la sesión SAS hasta que se modifiquen por una nueva sentencia **OPTIONS**.

10.1 Títulos y notas a pie de página

En muchas ocasiones deseamos poner títulos o notas a pie de página a nuestras salidas en la ventana OUTPUT.

Esto es posible por dos medios: por el menú HERRAMIENTAS – OPCIONES – TITULOS (o PIES DE PAGINA), o mediante sentencias TITLE y FOOTNOTE en nuestro programa.

10.1.1 Sentencias TITLE

Cada línea del título estará asociada con una sentencia TITLE:

```
TITLE1 'primera línea';  
TITLE2 'segunda línea';  
Etc.
```

Se pueden anular en cualquier momento de la siguiente forma:

```
TITLE1;  
TITLE2;
```

10.1.2 Sentencias FOOTNOTE

Se utilizan de la misma manera que TITLE:

```
FOOTNOTE1 'primera línea de pie de página';  
Etc.
```

Aunque es posible utilizar las sentencias TITLE y FOOTNOTE en cualquier parte del programa, si las utilizamos en diversos momentos para varias salidas se recomienda programarlas asociadas al PROC del que deseamos obtener la salida. Es posible, en caso contrario, obtener salidas con títulos no deseados.

11 PROC PRINT

Procedimiento utilizado para obtener informes en nuestra ventana OUTPUT.

Sintaxis

```
PROC PRINT opciones;  
    VAR variables;  
    ID variable;  
    BY variables;  
        PAGEBY variable;  
        SUMBY variable;  
    SUM variables;  
    FORMAT variable formato;  
RUN;
```

Opciones

DATA= conjunto de datos SAS del que se obtiene el informe. Si se omite tomará el último válido.

DOUBLE obtiene la salida a doble espacio.

NOOBS elimina el número de observación de la ventana OUTPUT.

LABEL muestra las etiquetas de las variables del conjunto de datos SAS del que se obtiene el informe.

Sentencias

VAR variables; Variables a obtener en el informe. Si se omite se mostrarán todas.

ID variable; Si deseamos visualizar una variable a la izquierda del informe.
(especifica una o más variables que proc print utiliza para identificar las observaciones del informe en lugar del número de observación)

BY variables; Para la obtención del informe por grupos. Se precisa un PROC SORT previo por el mismo criterio de ordenación que el indicado en esta sentencia BY.

PAGEBY variable; Se realiza un salto de página por cada cambio de valor de la variable indicada en esta sentencia. Dicha variable debe incluirse en la sentencia BY.

SUMBY variable; Se obtienen sumas parciales para cada cambio de valor de la variable indicada. Dicha variable debe incluirse en la sentencia BY.

SUM variables; Se obtienen sumas totales de las variables **numéricas** indicadas en la sentencia.

FORMAT Para obtener en el informe un formato para alguna variable, por ejemplo para fechas.

12 FORMATOS

Un formato es una característica o atributo de la variable que se utiliza para visualizar sus valores de una manera determinada.

Podemos entender un formato como una “máscara” que aplicamos a los datos para que tengan el aspecto que queremos. Los formatos permiten al usuario visualizar de distintas formas la información almacenada sin necesidad de crear variables auxiliares.¹¹

FORMATOS.1: Visualización de un formato definido por el usuario

Fichero de datos sin formato

	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83

Definición Formato:

M=Hombre

F= Mujer

Fichero de datos con formato

	Name	Sex	Age	Height	Weight
1	Alfred	Hombre	14	69	112.5
2	Alice	Mujer	13	56.5	84
3	Barbara	Mujer	13	65.3	98
4	Carol	Mujer	14	62.8	102.5
5	Henry	Hombre	14	63.5	102.5
6	James	Hombre	12	57.3	83

FORMATOS.2: Visualización formato commax12. con separación de miles

	cod_prov	PROVINCIA	TOTAL	Hombres	Mujeres
1	04	Almería	536.731	272023	264708
2	11	Cádiz	1.116.491	552463	564028
3	14	Córdoba	761.657	372464	389193
4	18	Notar las diferencias de visualización entre la población Total y población Hombres	821.660	401638	420022
5	21		462.579	229013	233566
6	23		643.820	317343	326477
7	29	Málaga	1.287.017	630902	656115

12.1 Formatos predefinidos por SAS

SAS proporciona al usuario un conjunto de formatos predeterminados que permiten acomodar la información a una gran variedad de formas de presentación. Se muestran algunos ejemplos en la siguiente tabla:

Formato	Definición	Rango de amplitud	Amplitud por defecto
Tipo Carácter			
\$UPCASEw.	Convierte datos de tipo carácter a mayúsculas	1–32767	Longitud de la variable u 8
\$w.	Escribe datos en formato carácter estándar (por defecto es el que se muestra)	1–32767	Longitud de la variable o 1

¹¹ El concepto de formato fue introducido al hablar de fechas SAS en el tema de Datos, operadores y funciones y ha sido un concepto importante en la lectura y escritura de ficheros planos.

Tipo Fecha y hora			
DATEw.	Escribe valores de fecha Sas (tipo date) en formato ddmonyy o ddmonyyyy	5-9	7
DATETIMEw.d	Escribe valores de "fechahora" SAS (tipo datetime) en formato ddmmyy:hh:mm:ss.ss	7-40	16
DTDATEw.	Escribe valores de "fechahora" SAS (tipo datetime) en formato ddmonyy o ddmonyyyy	5-9	7
EURDFDDw.	Escribe valores de fecha Sas (tipo date) en formato dd.mm.yy o dd.mm.yyyy	2-10	8
JULIANw.	Escribe valores de fecha Sas (tipo date) en formato calendario juliano yyddd o yyyyddd	5-7	5
MMDDYYw.	Escribe valores de fecha Sas (tipo date) en formato mm/dd/yy o mm/dd/yyyy	2-10	8
TIMEw.d	Escribe valores de hora Sas (tipo time) en formato hh:mm:ss.ss	2-20	8
WEEKDATEw.	Escribe valores de fecha Sas (tipo date) en formato día de la semana, nombre del mes-dd, nombre del mes-yy o nombre del mes-yyyy	3-37	29
WORDDATEw.	Escribe valores de fecha Sas (tipo date) en formato nombre del mes-dd o nombre del mes-yyyy	3-32	18

- Los valores "fecha" SAS representan los días transcurridos desde el 1 de enero de 1960.
- Los valores "hora" SAS representan los segundos transcurridos desde la medianoche.
- Los valores "fechahora" SAS representan los segundos transcurridos desde la medianoche del 1 de enero de 1960.

Formato	Definición	Rango de amplitud	Amplitud por defecto
Tipo Numérico			
BESTw.	SAS elige el mejor formato para datos numéricos (por defecto es el que se muestra)	1-32	12
COMMAw.d	Escribe los separadores de miles con comas y los decimales con puntos	2-32	6
COMMAXw.d	Escribe los separadores de miles con puntos y los decimales con comas	2-32	6
Ew.	Escribe los números con notación científica	7-32	12
EUROXw.d	Escribe los separadores de miles con puntos y antepone el símbolo €	2-32	6
PERCENTw.d	Escribe los datos numéricos como porcentajes	4-32	6
w.d	Escribe datos en formato numérico estándar	1-32	Ninguno

Ejemplos:

Formato	Valor	Resultado	Valor	Resultado
Tipo Carácter				
\$UPCASE10.	Lassen	LASSEN	St. Helens	ST. HELENS
\$10.	Lassen	Lassen	St. Helens	St. Helens
Tipo Fecha y hora				
DATE9.	366	01JAN1961	396	31JAN1961
DATETIME16.	37800	01JAN60:10:30	2629800	31JAN60:10:30
DTDATE9.	37800	01JAN1960	2629800	31JAN1960

EURDFFD10.	366	01.01.1961	396	31.01.1961
JULIAN7.	366	1961001	396	1961031
MMDDYY10.	366	01/01/1961	396	01/31/1961
TIME8.	37800	10:30:00	37815	10:30:15
WEEKDATE15.	366	Sun, Jan 1, 61	396	Tue, Jan 31, 61
WORDDATE12.	366	Jan 1, 1961	396	Jan 31, 1961
Tipo Numérico				
BEST10.	1000001	1000001	-12.34	-12.34
BEST6.	1000001	1E6	100001	100001
COMMA12.2	1000001	1,000,001.00	-12.34	-12.34
COMMAX12.2	1000001	1.000.001,00	-12,34	-12,34
E10.	1000001	1.00E+06	-12.34	-1.234E+01
EUROX13.2	1000001	€ 1.000.001,00	-12.34	\$-12.34
PERCENT9.2	0.05	5.00%	-1.20	(120.00%)
10.2	1000001	1000001.00	-12.34	-12.34

Para más información sobre los formatos predeterminados en SAS:

<http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000309859.htm>

12.2 Formatos definidos por el usuario: PROC FORMAT

Además de los formatos proporcionados por SAS, un usuario puede definir sus propios formatos con ayuda del procedimiento PROC FORMAT.

Básicamente, los formatos creados por los usuarios permiten establecer un conjunto de etiquetas que sustituyen valores concretos o rangos de valores en los datos.

Consideraciones para su definición:

- Los formatos son del tipo de las variables a las que se aplican, así los formatos numéricos se aplican a variables numéricas y los formatos alfanuméricos a variables alfanuméricas.
- Todo formato tiene un nombre, el cual:
 - sólo puede contener letras, números o “subrayados” (_)
 - no puede superar los 32 caracteres en longitud
 - no puede empezar ni terminar con un número
- El nombre de los formatos alfanuméricos debe empezar con el símbolo \$, el cual se contabiliza dentro de los 32 caracteres permitidos en la longitud.

Creación de un formato de tipo alfanumérico

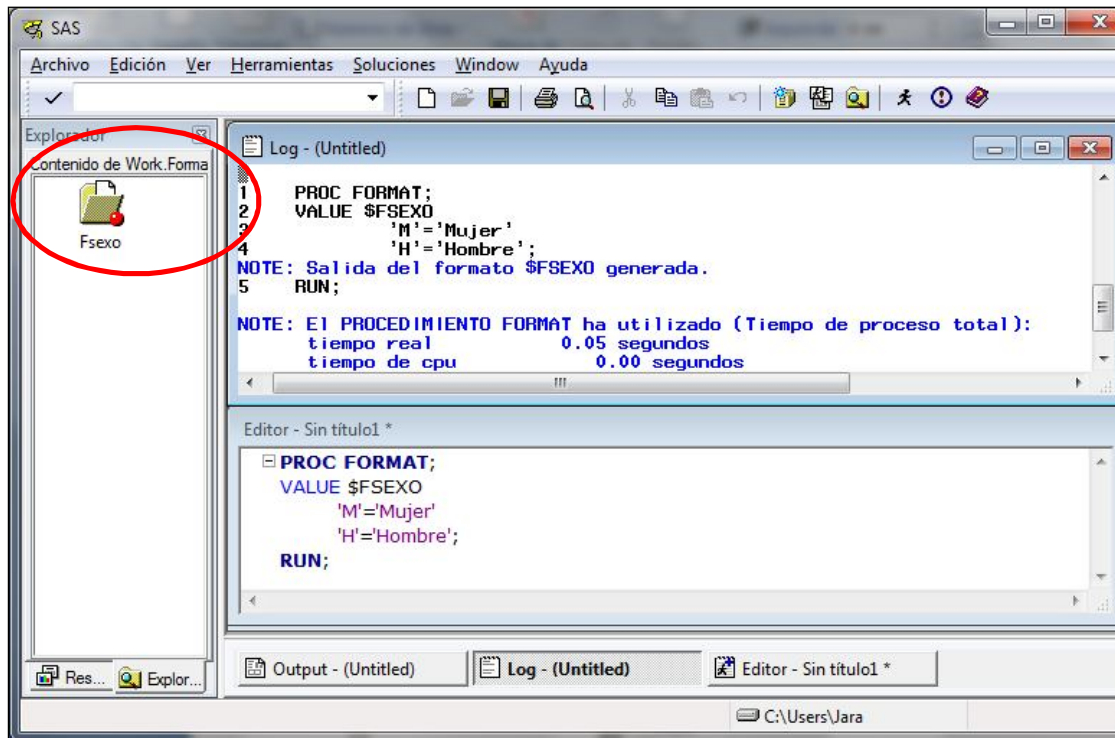
Sintaxis:

```
PROC FORMAT;
VALUE $nombrefmt
'cadena1'='etiqueta1'
'cadena2'='etiqueta2'
...
'cadenak'='etiquetak'
;
RUN;
```

Ejemplo:

```
PROC FORMAT;  
VALUE $FSEXO  
  'M'='Mujer'  
  'H'='Hombre';  
RUN;
```

FORMATOS.3: Creación de un formato de tipo alfanumérico.



Creación de un formato de tipo numérico

Sintaxis:

```
PROC FORMAT;  
VALUE nombrefmt  
  valor1='etiqueta1'  
  valor2='etiqueta2'  
  ...  
  valork='etiquetak';  
RUN;
```

Ejemplo:

```
PROC FORMAT;  
VALUE fsexo  
  1='Hombre'  
  6='Mujer';  
RUN;
```

Podremos etiquetar los valores missing en la definición del formato de la manera siguiente:

Formato numérico: . = 'etiqueta'
Formato alfanumérico: " = 'etiqueta'

Ejemplos:

```
PROC FORMAT;  
VALUE $fsexo  
  'M'='Mujer'  
  'H'='Hombre'  
  ''='Error'  
;  
RUN;
```

```
PROC FORMAT;  
VALUE fsexo  
  1='Hombre'  
  6='Mujer'  
  .='Error'  
;  
RUN;
```

Especificación de rangos

Además de valores individuales, PROC FORMAT admite la especificación de rangos.

- Los rangos pueden especificarse con cadenas de caracteres (formatos alfanuméricos) o valores (formatos numéricos) separados por comas:

- `'cadena1', 'cadena2', ... , 'cadenai'='etiquetaa'`
- `valor1, valor2, ... , valorj'='etiquetab'`

- Los rangos pueden representar intervalos:

<code>valor_{inf} – valor_{sup}</code>	→ se incluyen los 2 extremos: <code>[valor_{inf} , valor_{sup}]</code>
<code>valor_{inf} < – valor_{sup}</code>	→ se incluyen extremo superior: <code>(valor_{inf} , valor_{sup}]</code>
<code>valor_{inf} – < valor_{sup}</code>	→ se incluyen extremo inferior: <code>[valor_{inf} , valor_{sup})</code>
<code>valor_{inf} < – < valor_{sup}</code>	→ no se incluyen los extremos: <code>(valor_{inf} , valor_{sup})</code>

- Existen palabras clave para simplificar la expresión de ciertos intervalos:

HIGH

LOW

- Para formatos numéricos, NO incluye los valores missing
- Para formatos alfanuméricos, Sí incluye los valores missing

OTHER

- Tanto para formatos numéricos como alfanuméricos, Sí incluye los valores missing, a menos que se especifique otra etiqueta para dichos valores.

Ejemplo

```
PROC FORMAT;
```

```
VALUE $fcaa  
  '04','11','14','18','21','23','29','41'='Andalucía'  
  '22','44','50'='Aragón'  
  '33'='Asturias, Principado de'  
  '07'='Balears, Illes'  
  '35','38'='Canarias'  
  '39'='Cantabria'  
  '05','09','24','34','37','40','42','47','49'='Castilla y León'  
  '02','13','16','19','45'='Castilla - La Mancha'  
  '08','17','25','43'='Cataluña'  
  '03','12','46'='Comunitat Valenciana'
```

```

'06','10'='Extremadura'
'15','27','32','36'='Galicia'
'28'='Madrid, Comunidad de'
'30'='Murcia, Región de'
'31'='Navarra, Comunidad Foral de'
'01','20','48'='País Vasco'
'26'='Rioja, La'
'51'='Ceuta'
'52'='Melilla'
OTHER='ERROR';
VALUE ftramo
LOW-<1='Menos de 1 Ha'
1-<5='De 1 Ha a menos de 5 Ha'
5-<25='De 5 Ha a menos de 25 Ha'
25-<50='De 25 Ha a menos de 50 Ha'
50-HIGH='50 Ha o más';

```

RUN;

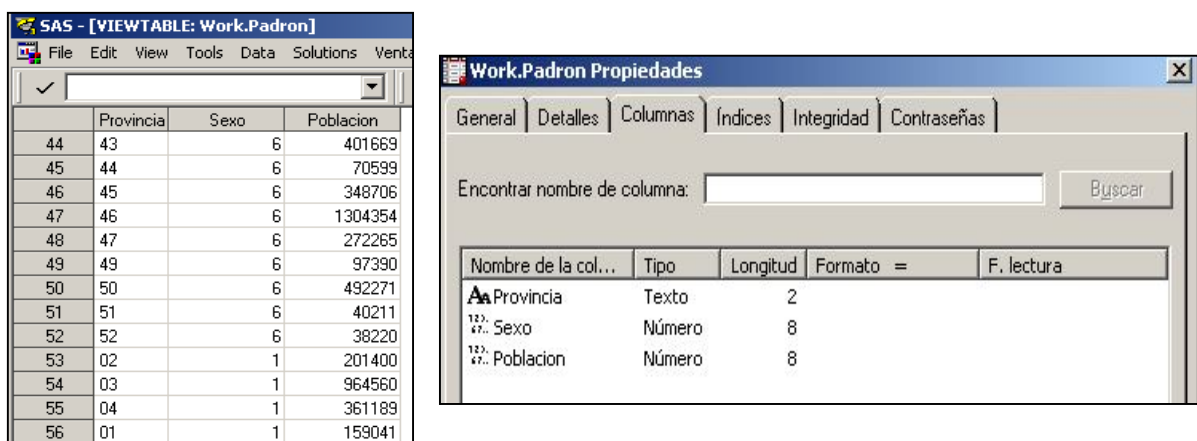
Asignación y cambio del formato de una variable en un conjunto de datos SAS

Dado un conjunto de datos SAS, el modo de cambiar (o asociar, si inicialmente no tiene) el formato a una variable de dicho fichero es a través de la sentencia **FORMAT**.

- Se asignan formatos con:
FORMAT *variable1* *formato1*. *variable2* *variable3* *formato2*. ... ;
- Se desasignan formatos con:
FORMAT *variable1* *variable2* ...; **FORMAT** _all_ ;

Ejemplo:

Consideramos el conjunto de datos Work.Padron, el cual recoge las cifras de hombres y mujeres empadronados por provincia a fecha 1 de enero de 2011. Las variables de Work.Padron no tienen asociado ningún formato:



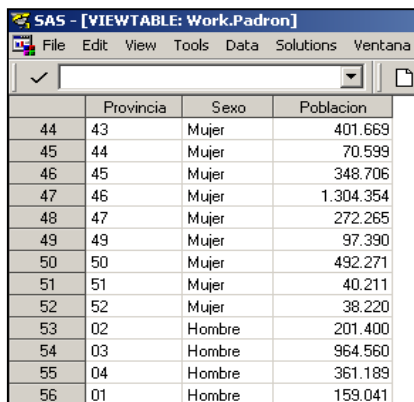
	Provincia	Sexo	Poblacion
44	43	6	401669
45	44	6	70599
46	45	6	348706
47	46	6	1304354
48	47	6	272265
49	49	6	97390
50	50	6	492271
51	51	6	40211
52	52	6	38220
53	02	1	201400
54	03	1	964560
55	04	1	361189
56	01	1	159041

Nombre de la col...	Tipo	Longitud	Formato =	F. lectura
Provincia	Texto	2		
Sexo	Número	8		
Poblacion	Número	8		

Creemos un formato adecuado para asignárselo a la variable sexo y a la variable población le asignamos el formato COMMAX, formato predeterminado de SAS que separa los miles por puntos.

Código SAS:

```
/*Definición del formato para sexo*/  
PROC FORMAT;  
VALUE fsexo  
    1='Hombre'  
    6='Mujer';  
RUN;  
/*Asignación de formatos*/  
DATA PADRON;  
SET PADRON;  
FORMAT sexo fsexo. Poblacion commax10.;  
RUN;
```



	Provincia	Sexo	Poblacion
44	43	Mujer	401.669
45	44	Mujer	70.599
46	45	Mujer	348.706
47	46	Mujer	1.304.354
48	47	Mujer	272.265
49	49	Mujer	97.390
50	50	Mujer	492.271
51	51	Mujer	40.211
52	52	Mujer	38.220
53	02	Hombre	201.400
54	03	Hombre	964.560
55	04	Hombre	361.189
56	01	Hombre	159.041



Nombre de la col...	Tipo	Longitud	Formato	=	F. lectura
Provincia	Texto	2			
Sexo	Número	8	SEXO.		
Poblacion	Número	8	COMMAX10.		

Devolvemos Work.Padron a su situación inicial desasignando los formatos:

```
DATA padron;  
SET padron;  
FORMAT sexo poblacion;  
RUN;
```

Aplicación de formatos en los PROC

Aquellos PROC de SAS destinados a obtener informes de resultados admiten la sentencia **FORMAT** para la visualización de datos formateados. Esto es muy útil en aquellas situaciones donde el usuario no está interesado en realizar modificación alguna en el fichero de partida pero sí desea visualizar los datos de sus informes con un aspecto determinado.

En este caso, la sintaxis de la sentencia **FORMAT** coincide con la vista para los conjuntos de datos SAS, y para que tenga efecto basta incluirla dentro del código del PROC:

```
FORMAT variable1 formato1. variable2 variable3 formato2. ... ;
```

Ejemplo:

Consideramos el conjunto de datos Work.Padron donde sus variables no tienen ningún formato asociado. Queremos obtener por pantalla el número de hombres y mujeres empadronados en Madrid (provincia=28) a 1 de enero de 2011. La información debe visualizarse de modo que, para sexo, en lugar de los valores 1 y 6 aparezcan las etiquetas *Hombre* y *Mujer* respectivamente, y las cifras de población asociadas separen los miles por puntos.

```

/*Definición de formato*/
PROC FORMAT;
VALUE FSEXO
    1='Hombre'
    6='Mujer';
RUN;

/*Aplicación formato en PROC PRINT*/
PROC PRINT DATA=padron noobs;
WHERE provincia='28';
FORMAT sexo fsexo. poblacion commax10.;
RUN;

```

Ventana OUTPUT:

Provincia	Sexo	Poblacion
28	Mujer	3.356.836
28	Hombre	3.132.844

Formatos temporales frente a formatos permanentes

Los formatos predeterminados de SAS son formatos permanentes pues están definidos internamente en el propio software y, por tanto, siempre están disponibles para ser utilizados.

Por su parte, los formatos definidos por el usuario pueden ser temporales o permanentes, según se opte en su definición.

Hasta ahora en la sintaxis del PROC FORMAT no hemos hecho referencia a ninguna librería en la que guardar la definición del formato. Cuando esto ocurre, SAS, por defecto, lo almacena en el catálogo Formats de la librería WORK. Lo que quiere decir que ese formato es temporal y sólo estará disponible en la sesión activa de SAS y no en ninguna otra.

Para conseguir que nuestras definiciones de formatos sean permanentes deberemos especificar la librería dónde guardarlo. ¿Cómo? De la siguiente manera:

```
LIBNAME CURSOSAS 'c:\sas\cursosas';
```

```

PROC FORMAT LIBRARY=CURSOSAS;
VALUE fsexo
    1='Hombre'
    6='Mujer';
RUN;

```

Con esto conseguimos guardar en el catálogo Formats de la librería CursoSAS el formato fsexo.

Si cerrásemos esa sesión de SAS y quisiéramos trabajar en un día posterior con ese formato sin tener que definirlo de nuevo, bastaría con utilizar la opción `fmtsearch` para indicarle en qué librería buscar el catálogo que guarda nuestros formatos.

```
options fmtsearch=(cursosas);
```

De manera general, en la opción `fmtsearch` se podrá especificar un listado de librerías de búsqueda de catálogos de formatos. El orden de búsqueda es de izquierda a derecha

```
options fmtsearch=(lib1 lib2 ...);
```

Si no se activa la opción `fmtsearch`, por defecto, SAS busca los formatos en el catálogo `Formats` de la librería `WORK`. Si no los encuentra, se produce un error y SAS deja de procesar las sentencias enviadas a ejecutar.

Para evitar que SAS dé error cuando no encuentra un formato, se puede añadir `nofmterr` a la opción `fmtsearch`.

```
options fmtsearch=(lib1 lib2 ...) nofmterr ;
```

12.3 Opción `notsorted`

Cuando se crea un formato, por defecto `PROC FORMAT` lo almacena teniendo en cuenta el orden creciente de los valores que se etiquetan (orden alfabético de las cadenas en el caso de formatos alfanuméricos). Dicha ordenación es la que posteriormente se emplea en la presentación de informes.

Sin embargo, cuando los informes se generan con `PROC MEANS` o `PROC TABULATE` (por ejemplo), el usuario puede mantener el orden especificado en la definición del formato siempre y cuando añada la opción (`NOTSORTED`) junto al nombre del mismo en el momento de definirlo:

```
value nombformato (notsorted)
```

Importante: cuando se emplea la opción `NOTSORTED`, para mantener el orden definido en la sentencia `VALUE`, es necesario utilizar las opciones `PRELOADFMT` y `ORDER=DATA` en el `PROC`.

Ejemplo:

Opción A: Sin opción `NOTSORTED`

```
proc format;
value $f1sexo
'M'='Hombre'
'F'='Mujer' ;
run;
```

```
proc means data=sashelp.class;
format sex $f1sexo. ;
class sex;
run;
```

Procedimiento MEANS						
Sex	Número de observaciones	Variable	Número de observaciones	Media	Desviación estándar	Mínimo
Mujer	9	Age	9	13.2222222	1.3944334	11.0000000
		Height	9	60.5888889	5.0183275	51.3000000
		Weight	9	90.1111111	19.3839137	50.5000000
Hombre	10	Age	10	13.4000000	1.6465452	11.0000000
		Height	10	63.9100000	4.9379370	57.3000000
		Weight	10	108.9500000	22.7271864	83.0000000

Opción B: Con opción **NOTSORTED** y PROC con **preloadfmt order=data**

```
proc format;
value $f2sexo (notsorted)
    'M'='Hombre'
    'F'='Mujer' ;
run;
```

```
proc means data=sashelp.class;
format sex $f2sexo. ;
class sex / preloadfmt order=data;
run;
```

		Procedim	
Sex	Número de observaciones	Variable	Número observaci
Hombre	10	Age Height Weight	
Mujer	9	Age Height Weight	

13 PROC FREQ

Procedimiento utilizado para la obtención de tablas de frecuencias y de tabulación cruzada entre variables indicadas en la sentencia TABLES.

Sintaxis

```
PROC FREQ DATA=entradaSAS <opciones>;  
    WEIGHT factor;  
    FORMAT variable1 fmtvar1. variable2 fmtvar2.;  
    TABLES variable1 variable2... variable1*variable2... /OUT=salidaSAS;  
RUN;
```

- La sentencia PROC FREQ. Presenta las siguientes **opciones**:

DATA = conjunto de datos SAS para el que queremos obtener el estudio de frecuencias. Si se omite lo realizará del último conjunto de datos SAS válido.

NOPRINT elimina cualquier salida en la ventana OUTPUT. Esta opción se suele utilizar cuando se quiere guardar la salida en un fichero de datos SAS.

ORDER=DATA | FORMATTED | FREQ | INTERNAL especifica el orden en el que se muestran los valores de la variable en el informe de salida.

DATA ordena los valores de acuerdo a su orden en el fichero SAS de entrada.

FORMATTED ordena los valores según los valores con formato (en orden ascendente)

FREQ ordena los valores por su frecuencia (en orden descendente).

INTERNAL ordena los valores según los valores sin formato.

Por defecto ORDER=INTERNAL. La opción ORDER= no se aplica a valores missing, los cuales serán mostrados en primer lugar siempre.

FORMCHAR= especifica la cadena de caracteres que definirá la línea de división de las tablas de frecuencias. Dicha cadena deberá ser de longitud tres, de tal manera que el primer carácter (1) es utilizado para los separadores verticales, el segundo para los separadores horizontales (2) y el tercero para las intersecciones tanto verticales como horizontales (7). Si no se especifica nada, PROC FREQ utiliza por defecto FORMCHAR(1,2,7)='| -+'.

NLEVELS muestra el número de modalidades o valores diferentes para todas las variables de las tablas.

COMPRESS comienza a mostrar la siguiente tabla de frecuencias, indicada en el TABLES, en la misma página que la que le precede. Esta opción no es válida con la opción PAGE.

PAGE muestra una tabla por página.

- **Sentencia WEIGHT.** Identifica la variable numérica que proporciona los factores de elevación (pesos).
- **Sentencia FORMAT.** Identifica los formatos a aplicar a cada una de las variables que intervienen en la sentencia TABLES.

- **Sentencia TABLES.**

Para obtener **tablas de frecuencias** para alguna de las variables del fichero de datos SAS se indican sus nombres, separados por espacios en blanco, en la declaración TABLES. *Ejemplo: TABLES var1 var2 var3;*

Para **tablas de tabulación cruzada**, se indica primero la variable que se desea en el lado izquierdo de la tabla, luego un asterisco * y luego la variable que se desea mostrar en la parte superior. En el caso de querer cruzar tres variables, estas se unirán con asteriscos. *Ejemplo: TABLES var1*var2*var3;*

Dentro de la sentencia TABLES existe con la opción OUT= podemos especificar el fichero SAS donde guardar los resultados del PROC FREQ. El fichero incluirá las variables COUNT (frecuencia absoluta) PERCENT (frecuencia relativa).

Se pueden pedir tantas tablas como se necesiten en una declaración TABLES, teniendo en cuenta que en el fichero de salida especificado en el OUT= sólo se guardarán los datos referentes a la última tabla especificada.

Ejemplos:

1.- Tabla de frecuencias

Partiendo del fichero de datos SAS, SASUSER.PILOTS, que contiene información sobre pilotos de una compañía aérea, imaginemos que queremos conocer el número de pilotos por puesto de trabajo (jobcode)

Fichero de entrada: Sasuser.Pilots

	ID	LastName	FirstName	City	State	Gender	JobCode	Salary	Birth	Hired	HomePhone
1	1333	BLAIR	JUSTIN	STAMFORD	CT	M	PT2	88606	02APR49	13FEB69	203/781-1777
2	1739	BOYCE	JONATHAN	NEW YORK	NY	M	PT1	66517	28DEC52	30JAN79	212/587-1247
3	1428	BRADY	CHRISTINE	STAMFORD	CT	F	PT1	68767	07APR58	19NOV79	203/781-1212
4	1404	CARTER	DONALD	NEW YORK	NY	M	PT2	91376	27FEB41	04JAN68	718/384-2946
5	1118	DENNIS	ROGER	NEW YORK	NY	M	PT3	111379	19JAN32	21DEC68	718/383-1122
6	1905	GRAHAM	ALVIN	NEW YORK	NY	M	PT1	65111	19APR60	01JUN80	212/586-8815
7	1407	GRANT	DANIEL	MT. VERNON	NY	M	PT1	68096	26MAR57	21MAR78	914/468-1616
8	1410	HARRIS	CHARLES	STAMFORD	CT	M	PT2	84685	06MAY55	10NOV74	203/781-0937
9	1439	HARRISON	FELICIA	BRIDGEPORT	CT	F	PT1	70736	09MAR52	13SEP78	203/675-4987
10	1545	HUNTER	CLYDE	STAMFORD	CT	M	PT1	66130	15AUG47	01JUN78	203/781-1119
11	1777	LUFKIN	ROY	NEW YORK	NY	M	PT3	109630	26SEP39	24JUN69	718/383-4413
12	1106	MARSHBURN	JASPER	STAMFORD	CT	M	PT2	89632	09NOV45	19AUG72	203/781-1457
13	1333	NEWKIRK	SANDRA	PRINCETON	NJ	F	PT2	84536	08SEP54	15APR76	201/812-3331
14	1478	NEWTON	JAMES	NEW YORK	NY	M	PT2	84203	12AUG47	27OCT78	212/587-5549
15	1556	PENNINGTON	MICHAEL	NEW YORK	NY	M	PT1	71349	25JUN52	14DEC79	718/383-5681
16	1890	STEPHENSON	ROBERT	NEW YORK	NY	M	PT2	85896	23JUL39	28NOV67	718/384-9874
17	1107	THOMPSON	WAYNE	NEW YORK	NY	M	PT2	89977	12JUN42	13FEB67	718/384-3785
18	1830	TRIPP	KATHY	BRIDGEPORT	CT	F	PT2	84471	30MAY45	01FEB71	203/675-2479
19	1928	UPCHURCH	LARRY	WHITE PLAINS	NY	M	PT2	89858	19SEP42	16JUL78	914/455-5009
20	1076	VENTER	RANDALL	NEW YORK	NY	M	PT1	66558	17OCT60	06OCT79	718/383-2321

Para obtener el resultado pedido no tendremos más que ejecutar este PROC FREQ:

```
PROC FREQ DATA=Sasuser.Pilots ;
    TABLES JobCode;
RUN;
```

Y, dado que no se ha especificado la opción NOPRINT, en la ventana del OUTPUT se mostrará la información que queríamos conocer.

Ventana OUTPUT:

The FREQ Procedure				
Job Code	Frequency	Percent	Cumulative Frequency	Cumulative Percent
PT1	8	40.00	8	40.00
PT2	10	50.00	18	90.00
PT3	2	10.00	20	100.00

2.- Tablas cruzadas de frecuencias

Partiendo del fichero de datos SAS anterior, supongamos que queremos obtener una tabla cruzada por puesto y salario pero clasificando el salario según si es menor de 70.000\$, está entre 70.000 y 100.000\$ o si es más de 100.000\$.

El PROC FREQ que nos devolvería esa petición, tras definir un formato apropiado, sería:

```
PROC FORMAT;
VALUE Tramos
  low-<70000='Menos de 70.000'
  70000-100000='70.000- 100.000'
  100000<-high='Más de 100.000';
RUN;

Title 'Distribución salarial según puesto de trabajo';
PROC FREQ DATA=Sasuser.Pilots ;
  FORMAT Salary Tramos.;
  TABLES JobCode*Salary;
RUN;
```

Y en la ventana OUTPUT obtendríamos la tabla cruzada con la información.

Ventana del OUTPUT:

Distribución salarial según puesto de trabajo				
The FREQ Procedure				
Tabla de JobCode por Salary				
JobCode	Salary			Total
Frecuencia	Menos de 70.000	70.000- 100.000	Más de 100.000	
PT1	6	2	0	8
PT2	0	10	0	10
PT3	0	0	2	2
Total	6	12	2	20

14 PROC SUMMARY / PROC MEANS

- **PROC SUMMARY y PROC MEANS** son procedimientos que se utilizan para el estudio de frecuencias y el análisis descriptivo de las variables de un conjunto de datos SAS.
- Por defecto estos procedimientos ofrecen los informes de los datos en la ventana OUTPUT pero también se pueden guardar estas salidas en conjuntos de datos SAS con distintas estructuras.
- En este capítulo se estudiará la sintaxis de PROC SUMMARY. La sintaxis de PROC MEANS es análoga a ésta, de modo que las particularidades que se van a estudiar de PROC SUMMARY se deben entender también como características propias de PROC MEANS.

Sintaxis

```
PROC SUMMARY DATA= <opciones>;  
    VAR variables;  
    CLASS variables;  
    BY variable;  
    OUTPUT OUT = salidaSAS <estadísticos>;  
RUN;
```

- La sentencia PROC SUMMARY. Presenta las siguientes **opciones**:

DATA = conjunto de datos SAS para el que se obtendrá el estudio de frecuencias. Si se omite el estudio se realizará del último conjunto de datos SAS válido.

MISSING trata en el estudio de frecuencias los valores missing. Recomendable utilizarlo siempre.

PRINT obtiene la salida en la ventana OUTPUT.

Por defecto, PROC MEANS envía la salida a la ventana de OUTPUT de modo que esta opción no aporta nada. En su lugar debería ser sustituida por la opción que anula la acción por defecto, es decir **NOPRINT**.

NWAY obtiene en la salida sólo el cruce entre variables y no las frecuencias marginales.

- **Sentencia VAR.** Identifica las variables de análisis.
- **Sentencia CLASS.** Identifica las variables que definirán subgrupos para el análisis.
- **Sentencia BY.** Calcula estadísticos de forma separada para cada grupo identificado por los valores de la variable aquí indicada. Por tanto será preciso un PROC SORT previo por la misma variable como criterio de ordenación.
- **Sentencia OUTPUT OUT = .**

salidaSAS conjunto de datos SAS de salida.

Estadísticos MIN (mínimo), MAX (máximo), SUM (suma), MEAN (media), VAR (varianza), STD (desviación típica), de cada uno de los grupos del estudio de frecuencias.

La diferencia entre PROC SUMMARY y PROC MEANS está en la salida. Por defecto, PROC SUMMARY no envía la salida a la ventana output por lo que esta sentencia **OUTPUT OUT= salidaSAS** es obligatoria en la sintaxis del procedimiento para poder ver los resultados. Por su parte, PROC MEANS por defecto envía la salida a la ventana de output de modo que sólo se requiere esta sentencia **OUTPUT OUT= salidaSAS** cuando se precisa guardar los resultados en un conjunto de datos SAS.

El conjunto de datos SAS de salida indicado en la sentencia OUTPUT tiene las variables indicadas en la sentencia CLASS y dos nuevas denominadas **_FREQ _** y **_TYPE _**. Si no se especifica ningún estadístico en las opciones se tendrá una tercera variable nueva llamada **_STAT _** que recogerá los nombres de los estadísticos que el procedimiento muestra por defecto. El resto de variables del fichero de lectura desaparecen.

_TYPE _ por defecto es una variable numérica. Identifica la combinación de variables clasificatorias que se está teniendo en cuenta en cada caso para calcular los estadísticos.

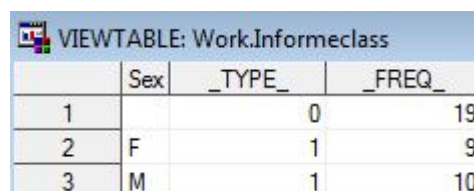
_FREQ _ indica la frecuencia de cada posible cruce de las variables indicadas en la sentencia CLASS.

Ejemplo1:

Supongamos que queremos conocer el número de observaciones por sexo del conjunto de datos SAS **sashelp.class**. Para ello ejecutaremos el siguiente código:

```
PROC SUMMARY data=sashelp.class;  
CLASS Sex;  
OUTPUT OUT=InformeClass;  
RUN;
```

El conjunto de datos SAS de salida, **InformeClass**, será el siguiente:



	Sex	_TYPE_	_FREQ_
1		0	19
2	F	1	9
3	M	1	10

El valor **0 de _TYPE _** representa la no existencia de cruce, es decir el total de observaciones del conjunto de datos SAS de lectura.

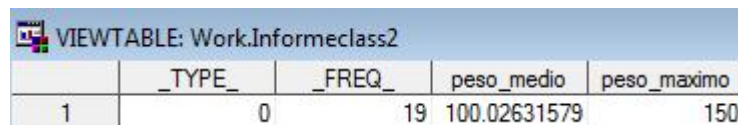
El valor **1 de _TYPE _** representa los valores tomados por la variable de clasificación SEX.

Ejemplo2:

Partiendo del mismo conjunto de datos SAS anterior llamado. Consideramos ahora como variable de análisis el peso para obtener el peso medio y el peso máximo del total de observaciones.

```
PROC SUMMARY data=sashelp.class;  
VAR weight;  
OUTPUT OUT=InformeClass2 MEAN=peso_medio MAX=peso_maximo;  
RUN;
```

El conjunto de datos SAS de salida, **InformeClass2**, será el siguiente:



	TYPE	_FREQ_	peso_medio	peso_maximo
1	0	19	100.02631579	150

El valor **0 de _TYPE_** representa la no existencia de cruce, por tanto el peso medio y el peso máximo obtenido tiene en cuenta el total de observaciones del conjunto de datos **sashelp.class**.

Ejemplo3:

Partiendo, de nuevo, del conjunto de datos SAS **sashelp.class**. Consideramos ahora como variables de clasificación el sexo y la edad, a la que habremos aplicado el formato GrupoEdad, y como variable de análisis el peso y obtendremos el peso medio y el peso máximo en cada grupo dado por las dos variables de clasificación mencionadas:

```
PROC FORMAT;  
VALUE GrupoEdad  
LOW-12='< 13 años'  
13-14='Entre 13 y 14 años'  
15-HIGH='> 14 años';  
RUN;  
  
PROC SUMMARY DATA=sashelp.class;  
CLASS sex age;  
FORMAT age GrupoEdad.;  
VAR weight;  
OUTPUT OUT=InformeClass3 MEAN=Peso_medio MAX=Peso_maximo;  
RUN;
```

El conjunto de datos SAS de salida, **InformeClass3**, será el siguiente:

VIEWTABLE: Work.InformeClass3						
	Sex	Age	_TYPE_	_FREQ_	Peso_medio	Peso_maximo
1			0	19	100.02631579	150
2		< 13 años	1	7	86.785714286	128
3		Entre 13 y 14 años	1	7	96.214285714	112.5
4		>14 años	1	5	123.9	150
5	F		2	9	90.111111111	112.5
6	M		2	10	108.95	150
7	F	< 13 años	3	3	70.666666667	84.5
8	F	Entre 13 y 14 años	3	4	93.625	102.5
9	F	>14 años	3	2	112.25	112.5
10	M	< 13 años	3	4	98.875	128
11	M	Entre 13 y 14 años	3	3	99.666666667	112.5
12	M	>14 años	3	3	131.666666667	150

El valor **0 de _TYPE_** representa la no existencia de cruce, es decir el total de observaciones del conjunto de datos SAS de lectura.

El valor **1 de _TYPE_** representa los valores tomados por la variable de clasificación Age después de aplicarle el formato GrupoEdad.

El valor **2 de _TYPE_** representa los valores tomados por la variable de clasificación Sex.

El valor **3 de _TYPE_** representa el cruce de las 2 variables de clasificación anteriores: Sex y Age.

En total el **InformeClass3** tiene 12 observaciones: 1 el total (TYPE=0) + 3 para los niveles de **Age** (TYPE=1) + 2 para los niveles de **Sex** (TYPE=2) + 2x3 por el cruce los niveles de **Sex y Age**. Tenemos 6 variables: las variables clasificatorias, los tipos, la frecuencia de las observaciones y las variables peso_medio y peso_máximo asociadas a los estadísticos media y máxima especificados en la sentencia OUTPUT.

En este mismo ejemplo si utilizamos la opción **NWAY** sólo se obtendrán en la salida los registros correspondientes al **_TYPE_** máximo (no se obtendrán las frecuencias marginales):

```
PROC SUMMARY DATA=sashelp.class NWAY;
CLASS sex age;
FORMAT age GrupoEdad.;
VAR weight;
OUTPUT OUT=InformeClass3 MEAN=Peso_medio MAX=Peso_maximo;
RUN;
```

VIEWTABLE: Work.InformeClass3						
	Sex	Age	_TYPE_	_FREQ_	Peso_medio	Peso_maximo
1	F	< 13 años	3	3	70.666666667	84.5
2	F	Entre 13 y 14 años	3	4	93.625	102.5
3	F	>14 años	3	2	112.25	112.5
4	M	< 13 años	3	4	98.875	128
5	M	Entre 13 y 14 años	3	3	99.666666667	112.5
6	M	>14 años	3	3	131.666666667	150

15 Anexo

15.1 Código para generar los datos de los ejemplos

Código para la generación de los ficheros de datos utilizados en el manual.

```
/* Fichero WORK.Censo2011 */
data Censo2011;
infile datalines dlm='/' truncover dsd;
input ccaa :$50. total varon mujer;
datalines;
Andalucía/8343655/4138125/4205530
Aragón/1331190/665005/666180
Asturias, Principado de/1069275/513990/555285
Balears, Illes/1096905/551065/545840
Canarias/2078280/1037445/1040835
Cantabria/589175/289130/300050
Castilla y León/2515755/1250500/1265255
Castilla - La Mancha/2092395/1057675/1034720
Cataluña/7472935/3701250/3771685
Comunitat Valenciana/4990345/2481205/2509135
Extremadura/1097695/546830/550865
Galicia/2759890/1336460/1423430
Madrid, Comunidad de/6387250/3084035/3303215
Murcia, Región de/1458250/736125/722125
Navarra, Comunidad Foral de/635175/317570/317600
País Vasco/2173265/1062375/1110890
Rioja, La/319460/159855/159605
Ceuta/83185/42515/40670
Melilla/80655/41545/39110
;
run;
```

```
/* Fichero WORK.FECHAS */
```

```
data work.fechas;
infile datalines dlm=' ';
input ID $ Fecha $10.;
datalines;
01,24/11/2008
02,25/11/2008
03,26/11/2008
04,27/11/2008
05,28/11/2008
06,29/11/2008
07,30/11/2008
08,01/12/2008
09,02/12/2008
10,03/12/2008
11,04/12/2008
12,05/12/2008
;
run;
```



```

/* Fichero SASHELP.CLASS */
DATA CLASS;
INFILE datalines dlm=' ';
INPUT Name $ Sex $ Age Height Weight;
datalines;
Alfred,M,14,69,112.5
Alice,F,13,56.5,84
Barbara,F,13,65.3,98
Carol,F,14,62.8,102.5
Henry,M,14,63.5,102.5
James,M,12,57.3,83
Jane,F,12,59.8,84.5
Janet,F,15,62.5,112.5
Jeffrey,M,13,62.5,84
John,M,12,59,99.5
Joyce,F,11,51.3,50.5
Judy,F,14,64.3,90
Louise,F,12,56.3,77
Mary,F,15,66.5,112
Philip,M,16,72,150
Robert,M,12,64.8,128
Ronald,M,15,67,133
Thomas,M,11,57.5,85
William,M,15,66.5,112
;
RUN;

```

```

/* Fichero MADBCN */
DATA MADBCN;
INFILE datalines dlm=' ' dsd ;
INPUT Flight $ Date $ Dest $ FirstClass Economy ;
datalines;
439,11/12/2000,MAD,20,137
921,11/12/2000,BCN,20,131
114,12/12/2000,MAD,15,170
982,12/12/2000,BCN,5,85
439,13/12/2000,MAD,14,196
982,13/12/2000,BCN,15,116
431,14/12/2000,MAD,17,166
114,15/12/2000,MAD,,187
982,14/12/2000,BCN,7,88
982,15/12/2000,BCN,14,31
;run;

```

```

/* Fichero ALUMNOS */

data alumnos;
infile datalines dlm=' ';
input Clase $ Nombre $10.;
datalines;
1ºA,Tomás
1ºA,Luis
1ºA,Jorge
1ºA,María

```

```

1°B,Alejandra
1°B,Sara
1°B,Juan
1°B,Alejandra
2°A,Patricia
2°B,Inés
2°B,Carlos
;
run;

/* Fichero ALUMNOSSG */
data AlumnosSG;
infile datalines dlm=' ';
input Subdireccion :$21. Nombre :$11.;
datalines;
S.G.Censos y Padrón,Concha
S.G.Informática,Maria Luisa
S.G.Mercado Laboral,Luis
S.G.Mercado Laboral,Alberto
S.G.Mercado Laboral,Esther
S.G.Mercado Laboral,Beatriz
S.G.Metodología,Jose Manuel
S.G.Precios,Teresa
S.G.Precios,Alberta
S.G.Precios,Alicia
S.G.Recogida de Datos,Eva
S.G.Servicios,Mercedes
S.G.Servicios,Ana María
S.G.Servicios,Marta
S.G.Sociales,Carmen
;
run;

/*Fichero PADRON*/
data Padron;
infile datalines dlm=' ';
input Provincia $ Sexo Poblacion;
datalines;
02,1,201400
02,6,200918
03,1,964560
03,6,969567
04,1,361189
04,6,341630
01,1,159041
01,6,160186
33,1,518571
33,6,562916
05,1,87085
05,6,85619
06,1,344291
06,6,349630
07,1,557577
07,6,555537
08,1,2715628
08,6,2813471
48,1,560178

```

48,6,595594
09,1,189652
09,6,186005
10,1,206573
10,6,208873
11,1,615865
11,6,627654
39,1,289872
39,6,303249
12,1,302855
12,6,301489
13,1,264078
13,6,266097
14,1,395858
14,6,409999
15,1,551476
15,6,595648
16,1,111052
16,6,108086
20,1,347653
20,6,361954
17,1,381448
17,6,375362
18,1,457084
18,6,467466
19,1,131532
19,6,124929
21,1,259362
21,6,262606
22,1,116224
22,6,112137
23,1,333384
23,6,337216
24,1,243316
24,6,254483
25,1,225388
25,6,216920
27,1,170854
27,6,180676
28,1,3132844
28,6,3356836
29,1,801126
29,6,824701
30,1,741581
30,6,728488
31,1,320656
31,6,321395
32,1,160570
32,6,172687
34,1,85118
34,6,86550
35,1,551241
35,6,545739
36,1,466691
36,6,496820
26,1,161582
26,6,161373

```
37,1,172584
37,6,180402
38,1,510350
38,6,519439
40,1,82967
40,6,81202
41,1,945766
41,6,983196
42,1,48347
42,6,46876
43,1,409732
43,6,401669
44,1,74008
44,6,70599
45,1,358536
45,6,348706
46,1,1274365
46,6,1304354
47,1,262609
47,6,272265
49,1,95993
49,6,97390
50,1,481054
50,6,492271
51,1,42165
51,6,40211
52,1,40256
52,6,38220
;
run;
```