UNIVERSIDAD DE LOS ANDES.
FACULTAD DE CIENCIAS ECONÓMICAS Y SOCIALES.
INSTITUO DE ESTADÍSTICA APLICADA Y COMPUTACIÓN
MAESTRÍA EN ESTADÍSTICA

Computación Estadística

Prof. Elizabeth Torres Rivas

Mérida, Septiembre 2014

CONTENIDO

INTRODUCCION AL SAS

1. INICIO DE UN TRABAJO EN SAS

Qué es el SAS?

El Sistema SAS nació en la década del 60 como un paquete estadístico, sin embargo, los avances científicos y tecnológicos lo han transformado en lenguaje de propósito general que maneja datos y una librería de procedimientos que trabajan en conjunto como un sistema. Tanto el mundo científico como el de los negocios se basan en datos para el proceso de toma de decisiones.

El SAS toma en consideración esto, pues, es un sistema que permite realizar 4 grandes tareas: acceso, manejo, análisis y presentación de los datos, que han sido transformados como una información útil para el proceso de toma de "buenas" decisiones. Es decir, es un sistema modular integrado de aplicaciones de software, que permite:

- Ingresar, recuperar y administrar datos
- Realizar reportes escritos y gráficos
- Realizar análisis estadísticos y matemáticos
- Realizar predicciones y apoyar la toma de decisiones en investigación o en negocios
- Investigación de operaciones y planificación de proyectos
- Desarrollo de aplicaciones

Esas aplicaciones están diseñadas para ayudar al usuario a realizar fácilmente dichas tareas. El núcleo del SAS es el software BASE, el cual consiste de:

- Un lenguaje de programación que le permite manejar sus datos
- Procedimientos, que son las herramientas del software para realizar análisis y reportes.
- Facilidad para programar macros, que es un lenguaje de programación.
- Un ambiente interactivo de ventanas, llamadas Display Manager System (DMS).

Desde el punto de vista estadístico, se puede definir el SAS como un sistema integrado para manejo de datos y análisis estadísticos, mediante la combinación de versátiles procedimientos estadísticos con la capacidad de hacer reportes escritos (Ver Figura 1). Es

decir, el manejo de datos o procesador de datos se denomina "DATA STEP", el cual produce datos en el ambiente SAS, llamados DATA SET y los análisis estadísticos son realizados mediante procedimientos, denominados PROC STEP.

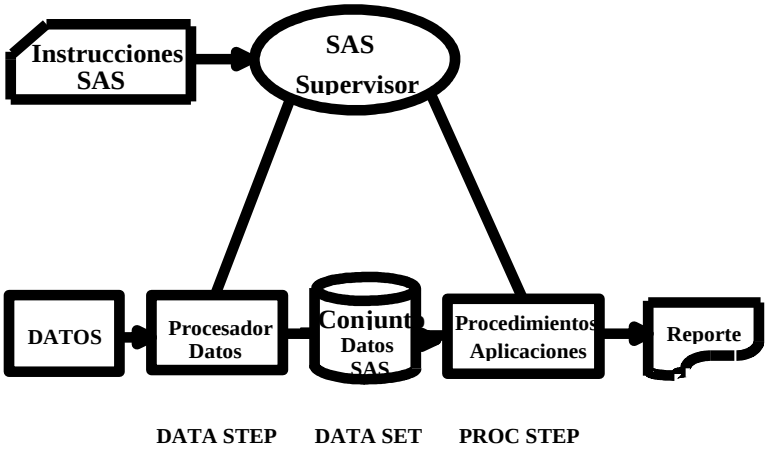

Figura 1

1.1 Computador

La forma de ejecución del SAS depende de la máquina (Host System) en que este instalado, si está conectado en red o no y del sistema operativo. Sin embargo, se puede decir, que las versiones de SAS son similares, en algunas cambia ligeramente el ambiente de ventanas.

1.2 Cómo Ejecutar el SAS en Windows

Para ejecutar el SAS haga clic sobre el icono del SAS y de forma inmediata aparecerán 2 ventanas, primero la referida a la portada de esta guía y segundo, la figura 2.

2 CÓMO ES EL AMBIENTE DEL SISTEMA SAS BAJO WINDOWS?

El ambiente del SAS denominado DISPLAY MANAGER SYSTEM (**DMS**) se define como un ambiente interactivo de ventanas en las que se puede escribir, ejecutar, probar, depurar y observar los resultados de programas.

2.1 Ventanas Primarias

La presentación de este ambiente es estándar, pero éste puede ser personalizado por el usuario. Se presenta como un conjunto de 3 ventanas primarias: **Explorer, Program Editor** y **Log,** las cuales aparecen al ejecutar el SAS, tal como se muestra en la Figura 2. Existe otra ventana primaria denominada **Output**, pero no aparece activa hasta que se ejecuta un programa SAS. También aparece la barra de menú denominada **ToolBox**, la cual se observa en la parte superior de la figura 2 (también se puede ver en la figura 3) y de acuerdo a la ventana en la que se trabaje están activas todas las opciones de la barra.

Aunque se pueden ver las 3 ventanas, solo una está activa. Cuando se inicia el SAS la ventana activa, en este caso es la del **Program Editor,** pues el cursor aparece en negrita y titilando.

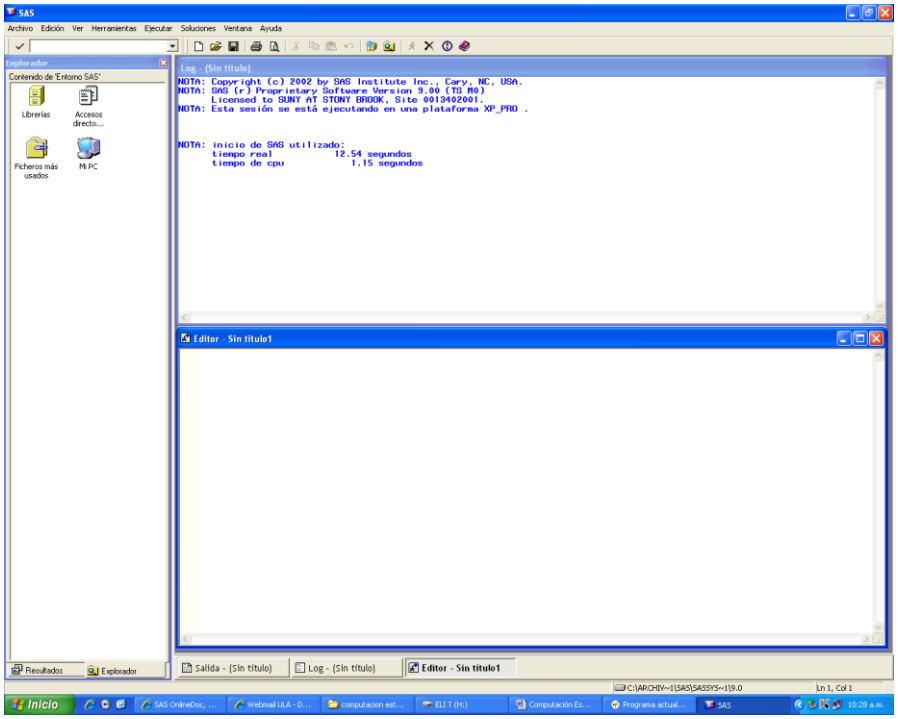

Figura 2

1. **Program Editor**: es la ventana de edición de programas SAS, está activa cuando se ingresa por primera vez en el sistema. Esta ventana viene acompañada de una barra de menú denominada **ToolBox**.
2. **Log**: ventana que muestra la ejecución sintáctica de un programa SAS; es útil para estudiar el tipo de errores encontrados en un programa que ha sido ejecutado, lo que permite su posterior corrección y depuración.
3. **Output**: en esta ventana aparecen los resultados de la ejecución de los procedimientos de un programa; está ubicada debajo de las dos primeras ventanas la primera vez que se ingresa al sistema, y
4. **Toolbox**: Esta ventana es una barra de menú, la cual aparece en la parte superior de las ventanas SAS (Ver Figura 3), ésta permite realizar operaciones sobre el archivo fuente de un programa SAS.

Figura 3

La figura 3 muestra las operaciones, de izquierda a derecha, que se pueden realizar en la ventana **ToolBox**:

Barra de comandos o línea de comandos: es una mini ventana ubicada debajo de las palabras "Archivo, Edición, Ver y Herramientas", la cual permite escribir comandos globales, como los siguientes:

- **Open**: abre una tabla (viewtable) para crear un conjunto de datos SAS
- **Save**: despliega una ventana de dialogo que permite guardar el archivo de datos SAS.
- **Print**: imprime el archivo. (No funciona bajo conexión remota).
- **Undo**: deshace última operación realizada.
- **Dir**: muestra los nombres de referencia de las librerías activas.
- **Assist**: invoca el asistente del SAS para realizar una tarea determinada.
- **Explorer**: despliega el entorno SAS y las librerías activas, lo cual permite realizar búsqueda y operaciones con archivos.
- **Submit**: ejecuta un programa fuente SAS.
- **Help**: invoca el Sistema de ayuda en línea del SAS

Cada vez que se realiza una de las anteriores operaciones, la ventana **ToolBox** se desactiva, es decir, el cursor se coloca en la ventana del editor. La cual puede ser invocada por el usuario, si así lo desea, para realizar operaciones de acuerdo a la ventana primaria que este activa. Solo sobre la ventana **Program Editor** se pueden realizar todas las operaciones de la barra **ToolBox**, en las otras se permiten algunas, como guardar un archivo e invocar la ayuda. A continuación se definen los comandos.

2.2 Comandos Globales

Los comandos globales se refieren a las operaciones que se pueden realizar sobre las ventanas primarias y se clasifican según el tipo de ventanas. Son operaciones que se pueden realizar para: el manejo de archivos, manejo de las ventanas, recorrido de la pantalla (**scrolling**), edición de texto, búsqueda en edición, invocación a ventanas o comandos, sobre el editor de texto, sobre el sistema operativo y máquina (**Host**), de la posición y tamaño de las ventanas, sobre las salidas y el color de las pantallas, entre otras.
Desde el punto de vista sintáctico los comandos globales son palabras claves que se utilizan para realizar una tarea determinada. Se permite escribir los primeros 4 caracteres del comando. Los comandos se pueden escribir en la línea de comandos o en la barra de comandos del **ToolBox**. Dicha línea o ventana puede ser invocada al seleccionar en la barra de **menú,** que se encuentra en la parte superior de cualquiera de las ventanas primarias (**PGM Editor, Log, Output**) la opción **Globals** y luego **Command** o **Command Line.** Es importante señalar que el SAS no diferencia entre mayúsculas y minúsculas.

Entre otros, comandos globales:

- **PGM**: activa la ventana de edición de programas SAS.
- **LOG**: activa la ventana de **Log**, es la que muestra la ejecución sintáctica del SAS, es decir, muestra los errores de sintáxis ocurridos durante la ejecución de un programa.
- **OUTPUT**: activa la ventana **Output** es la que muestra la salida de resultados.
- **COMMAND**: Si se escribe sobre la barra de comandos (**Command**) activa la línea de comandos (**Command Line**) en la ventana que está activa. Y si se escribe sobre la línea de comandos (**command line o mandato**) cierra dicha línea de mandatos. Es decir que es un interruptor que activa o desactiva la línea de comandos o mandatos.
- **X[' comando UNIX o DOS']**: permite salir temporalmente del sistema SAS al sistema operativo (**DOS o UNIX**) para ejecutar comandos del mismo. Es de hacer notar que este comando no funciona en **UNIX**, a menos que esté acompañado de los argumentos, los cuales son los comandos del sistema operativo **UNIX**. Los siguientes ejemplos muestran el uso de este comando global:
- **x 'ls -l'**: muestra el listado de directorios y archivos de su subdirectorio de trabajo.
- **x 'mkdir direc'**: crea en su cuenta un subdirectorio de nombre **direc** (o el nombre que Ud. elija).
- **Catalog**: muestra el directorio de catálogos librerías activas SAS, permite manejarlos y sus entradas o para crear nuevos catálogos.
- **Dir:** muestra información acerca de los conjuntos de datos SAS y catálogos SAS almacenados en la librería de datos SAS que está activa.
- **FileName:** muestra las referencias de archivos activos (**filerefs**) y los nombres de los archivos a los cuales se refieren.
- **FootNotes:** proporciona un menú para ingresar hasta 10 líneas para notas de pie páginas, para que aparezcan en la salida de un procedimiento o de un paso de datos.

- **Titles:** proporciona un menú para ingresar hasta 10 líneas para que aparezcan sobre un procedimiento o resultado de un paso de datos.
- **Help:** muestra ayuda en línea acerca del Sistema SAS.
- **Keys:** es una facilidad que permite mostrar, editar y almacenar las funciones a cumplir por las teclas función.
- **Libname:** muestra todas las librerias activas (**librefs**) y los nombres de los caminos (path) o directorios a los cuales ellas refieren.
- **Menú:** muestra el menú del SAS/Assist, que es una aplicación que usa una serie de paneles a llenar y lo guían a través de los procedimientos del sistema.
- **Notepad:** permite crear y almacenar notas de documentación o información.
- **Options:** muestra el entorno de opciones, es decir**,** despliega un submenú de opciones de edición.
- **Bye:** finaliza la sesión de trabajo en el SAS. También puede utilizar ENDSAS para abandonar el sistema.
- **End:** remueve una ventana especial de la pantalla, también permite cerrar y guardar la información de un proceso dado. Si está activa la ventana del editor PGM se ejecuta el programa SAS.
- **Zoom:** [ON/OFF] minimizao maximiza la ventana activa sobre la pantalla entera, trabaja como un switch ON/OFF. Esta misma operación se realiza con los iconos que se encuentran en la esquina superior derecha de cada ventana.
- **Numbers** [ON/OFF]: activa o desactiva el área de números de la ventana del editor PGM.
- **Assist:** activa el asistente para la ejecución de procedimientos relativos al manejo de datos, análisis de datos y gráficos.
- **Include** o **Inc 'Archivo':** copia el contenido de un archivo en la ventana del editor.
- **Graph**: invoca o activa la ventana de graficos **Graph**.

Los comandos globales siguientes pueden ser escritos en el **Toolbox,** en la barra de comandos, los cuales actúan como interruptores **ON/OFF** en los siguientes casos:

- **Num**: activa (**ON**) o desactiva (**OFF**) el área de números en la ventana del editor de programas **PGM**.
- **End:** activa (**ON**) o desactiva (**OFF**) una ventana.

En la esquina superior derecha de cada una de estas ventanas primarias (**Editor**, **Log**, **Output**) aparecen tres iconos y se refiere a: 1. Minimizar la ventana, 2. Desplegar (zoom) o maximizar la ventana y 3. Cerrar la ventana o Salir (X) del sistema SAS, esta última se activa solamente en la ventana del Manager del Sistema.

Cada una de las ventanas primarias tiene en la parte superior una barra de menú, y al hacer clic con el ratón se despliegan submenús referidos a las operaciones que se pueden realizar dentro de cada ventana. Los menús se desactivan pulsando la tecla **ESC** o realizando otra

operación de la barra de menú. Es importante resaltar, que algunas operaciones no están permitidas sobre las ventanas **Log** y **Output**.
En dicha barra de menú se encuentra una serie de operaciones que pueden ser aplicadas a las ventanas primarias o sobre la ventana que se encuentre activa. Las operaciones que se pueden realizar están referidas a: operaciones con archivos, opciones sobre la configuración y presentación del ambiente de trabajo del SAS, sobre la presentación de las ventanas, ya sean en forma de cascada o mosaico y sobre la ayuda en línea del sistema.

Como por ejemplo, en la ventana del editor (**Program Editor**) este menú es el que se observa en la Figura 4.

Dependiendo si el sistema SAS se presenta en inglés o en español, al seleccionar la opción **File (Archivo)** se despliega un menú que contiene todas las operaciones relacionadas con el manejo de archivos como: abrir un archivo de programa fuente SAS, guardar en un archivo la ventana activa, imprimir un archivo o la ventana activa, configurar la impresora a utilizar, ejecutar comandos del sistema operativo (Windows o **UNIX**) y salir del SAS al finalizar la sesión de trabajo.

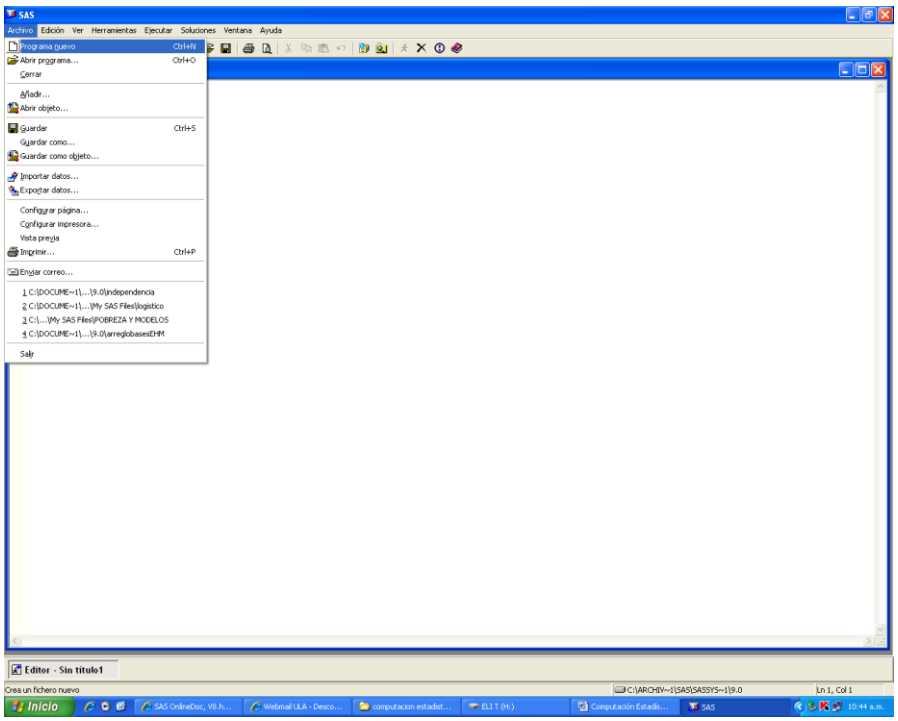

Figura 4

El menú de la opción **File (Archivo)** que se muestra en la figura 4 es el siguiente:

- **New Program (Nuevo Programa):** abre un nuevo archivo de un programa fuente SAS.
- **Open Program (Abrir Programa)**: abre un archivo de un programa fuente SAS.
- **Close (Cerrar)**: cierra el archivo de un programa fuente SAS.

- **Open Object (Abrir Objeto)**: abre un objeto.
- **Save (Guardar)**: guarda un archivo de un programa fuente SAS.
- **Save as (Guardar como)**: guarda el archivo activo del programa SAS con otro nombre
- **Save as Object (Guardar como objeto)**: guarda como un objeto.
- **Import data (Importar Datos)**: importa datos.
- **Export data (Exportar Datos)**: exporta datos.
- **Page Setup (Configurar página)**: configura página.
- **Print Setup (Configurar Impresora**): configura impresora.
- **Print Preview (Vista Previa)**: muestra página a imprimir.
- **Print (Imprimir)**: imprime el archivo activo en la pantalla.
- **Send Mail (Enviar correo):** envía correo, y
- **Exit (Salir)**: para salir del Sistema SAS.

La opción **Edit (Edición)** de la Figura 4 se refiere a las operaciones de edición de programas fuentes SAS, similares a las que presenta **ToolBox**, como:

- **Undo (Deshacer)**: deshace la última operación de edición.
- **Redo (Rehacer)**: rehace la última operación de edición.
- **Cut (Cortar)**: corta un área marcada de texto.
- **Copy (Copiar)**: copia un área marcada de texto.
- **Paste (Pegar)**: pega un área marcada de texto.
- **Clear (Borrar)**: borra un área marcada de texto.
- **Clear all (Borrar todo)**: borra la pantalla activa.
- **Select all (Seleccionar todo):** selecciona todo el archivo que está en pantalla.
- **Collapse all (contraer todo)**: a las instrucciones SAS las precede un simbolo + o -, si se elige esta opción, las instrucciones serán precedidas del simbolo +.
- **Expand all (expandir todo)**: despliega las instrucciones, es lo contrario de la opción anterior.
- **Find (Buscar)**: busca una determinada palabra o cadena de caracteres.
- **Replace (Reemplazar)**: busca una cadena de caracteres o palabra y la reemplaza por otra.

La opción **View** (**Ver**) permite navegar entre ventanas (ver Figura 4) y tiene el siguiente submenú:

- **Enhanced Editor (Editor Avanzado)**: activa el editor inteligente.
- **Program Editor**: activa la ventana de edición de programas.
- **Log**: activa la ventana del LOG que muestra la ejecución sintáctica de un programa SAS.
- **Output**: activa la ventana de resultados de procedimientos.
- **Graph**: activa la ventana de gráficos.
- **Results (Resultados)**: muestra el directorio de resultados para los procedimientos ejecutados.
- **Explorer (Explorador)**: activa la ventana que muestra los directorios de activos del ambiente SAS.

- **Contents only (Solo contenido)**: activa la ventana que muestra solo los directorios de activos del ambiente SAS.
- **My favorite folders (Mis carpetas)**: activa la ventana que muestra el directorio de carpetas creadas por el usuario.

La opción **Tools (Herramientas) (**de la Figura 4) tiene el siguiente submenú:

- **Query**: activa ventana para realizar operaciones de bases de datos.
- **Table Editor (Editor de tablas)**: activa la ventana del editor de datos.
- **Graphics Editor (Editor de gráficos)**: activa la ventana del editor de gráficos.
- **Report Editor (Editor de informes)**: activa la ventana del editor de reportes.
- **Image Editor (Editor de imagenes)**: activa la ventana del editor de imágenes.
- **Text Editor (Editor de textos)**: activa la ventana del editor de programas.
- **Keyboard macros (Macros del teclado)**: despliega otro submenú referido a comandos globales del sistema, para editar o modificar títulos, notas de pie de página, definición de símbolos, patrones, ejes, leyendas y colores para los gráficos. Además, permite activar la ventana **Command** o la línea de comandos (**Command Line**) para escribir directamente comandos globales, entre otros.
- **Add Abbreviation** (**Añadir abreviaturas**): permite añadir abreviaturas.

La opción **Run (Ejecutar) (**de la Figura 4) tiene el siguiente submenú:

- **Submit (Procesar)**: ejecuta un programa SAS.
- **Recall last submit (Recuperar el último proceso)**: trae a pantalla el último programa ejecutado, ya que el DMS crea automáticamente una pila de programas ejecutados.
- **Submit top line (Procesar línea superior)**: ejecuta un programa desde la línea del tope.
- **Submit n lines (Procesar N líneas):** ejecuta n líneas de un programa.
- **Remote submitn (Proceso remoto)**: ejecuta un programa en forma remota. Las otras opciones tienen que ver con ejecución remota.

La opción **Solutions (Soluciones)** (ver Figura 4) tiene el siguiente submenú:

- **Analysis (Análisis)**: muestra el submenú de procedimientos de análisis de datos.
- **Development and Programming (Desarrollo y Programación)**: muestra el submenú sobre procedimientos para realizar aplicaciones y programación..
- **Reporting (Generación de Informes)**: abre como ventana activa el editor de reportes.
- **Accesories (Accesorios)**: presenta algunas utilidades como: muestra de gráficos que es un test para probar los patrones gráficos; editor de registros, explorador de metadatos, entre otros.
- **Assist**: muestra el asistente del SAS en ambiente de ventanas para facilitar al usuario la realización de operaciones sobre archivos, análisis de datos, realizar gráficos, etc.
- **Desktop (Escritorio)**: muestra el escritorio del SAS.
- **EIS/OLAP Application Builder (Desarrollo de aplicaciones EIS/OLAP)**: activa el módulo EIS referido al manejo y desarrollo de aplicaciones.

La opción **Window (Ventana)** muestra las operaciones que se pueden realizar sobre las ventanas, como ventana nueva, minimizar todas las ventanas, Cascade, Mosaico Vertical, Mosaico Horizontal, Ajustar tamaño de la ventana, Desacoplar ventanas y muestras el listado de ventanas.

La opción **Help (Ayuda)** tiene el siguiente submenú:

- **Using this window (Cómo usar la ventana)**: activa la ayuda en línea del SAS sobre la ventana en la cual Ud. solicitó ayuda.
- **Sas Help and documentation (Ayuda y Documentación SAS)**: muestra la ayuda extendida del sistema.
- **Books and training**: muestra un submenú que activa la documentación en línea (SAS onlineDoc) o el tutorial (SAS onlineTutor).
- **Getting started with SAS Software (Introducción al software SAS)**: activa el tutorial del SAS para que de una manera rápida el usuario se familiarice con el software SAS, eligiendo determinados tópicos como: uso del espacio de trabajo, acceso y manejo de sus datos, presentación y análisis de datos.
- **SAS on the WEB (SAS en la Web)**: conecta con la página WEB del Instituto SAS.
- **About SAS 9 (Acerca de SAS 9)**: muestra información acerca de la versión del SAS.

Las ventanas primarias **Log** y **Output** también poseen en la parte superior de la ventana esta barra de menú, sólo que algunas de las opciones no se encuentran activas, sino para el editor de programas. El menú **ToolBox** también se activa para estas ventanas y aquí también se observa que sólo algunas operaciones se pueden realizar (Ver Figuras 5 y 6).

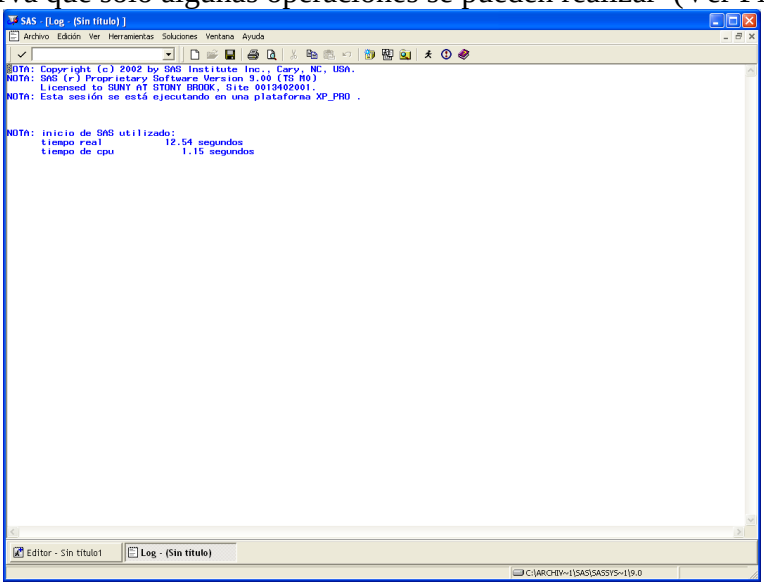

Figura 5

La selección de un item del menú, en cualquiera de las ventanas primarias, se realiza colocando el cursor o ratón y luego se presiona **ENTER** o se hace clic con el primer botón

(izquierda) del ratón. Los menús se desactivan pulsando la tecla **ESC** o realizando otra operación de la barra de menú.

Si Ud. selecciona, en la ventana **Program Editor**, **Open** dentro de la opción **File**, se abrirá una ventana que muestra el directorio activo para el usuario (Ver Figura 7.), luego debe elegir entre leer un archivo o leer un objeto.

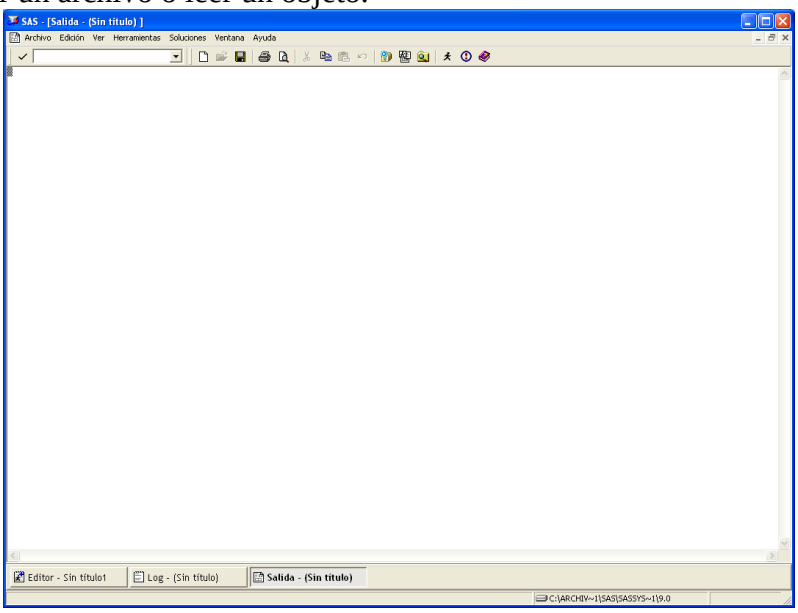

Figura 6

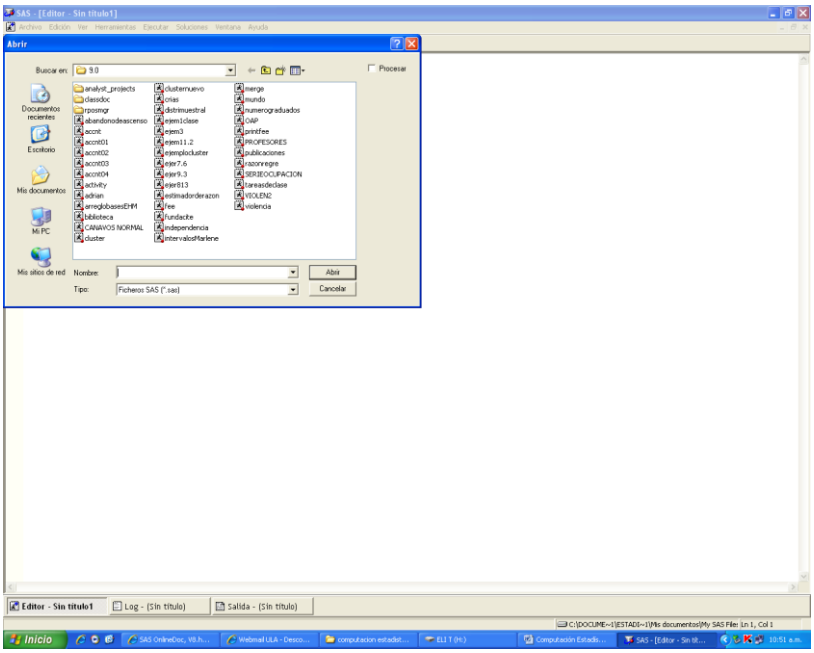

Figura 7

2.3 Algunos Comandos del Editor de Texto

Estos comandos que pueden ser ejecutados desde la línea de comandos o al seleccionar en el menú de la ventana del editor la opción **Edit**. Dichos comandos están referidos a:

- **Edición General**

- **Clear:** remueve el contenido de todas las líneas del texto.
- **N, num, Numbers**[ON/OFF]: cambia a ON u OFF los números de las líneas de datos en la ventana **PGM** o de las líneas de notas cuando se ejecuta el comando **Notepad** en cualquiera de las ventanas del sistema.

- **Manejo de Archivos**

- **File ['nombre']:** escribe el contenido total del editor de texto en un archivo externo. Si este comando se ejecuta en la ventana del **Output** o del **Log** permite almacenar el contenido de la misma en un archivo texto, que luego puede ser transportado al Windows ó **DOS** utilizando **FTP**.

- **Recorrido de la Pantalla** (scrolling)

- **Backward, back** [PAGE/HALF/MAX/n]: mueve el contenido del editor de texto hacia atrás (hasta el comienzo), la cantidad la coloca el comando VSCROLL. También puede especificar la cantidad a retroceder con los siguientes subcomandos:
- **Forward:** mueve el contenido del editor de texto hacia delante (hasta el fin), la cantidad la coloca el comando **VSCROLL**. También puede especificar la cantidad a avanzar con los siguientes subcomandos:

- **Cortar y Pegar**

- **Cut** <SHIFT F3> remueve el texto marcado en la ventana activa y lo almacena (en el buffer).
- **Mark** <SHIFT F1> marca el bloque que desea manipular.
- **Paste** <SHIFT F5> inserta el texto marcado (almacenado en el buffer) en la posición indicada por el cursor.

2.4 Comandos de Línea

Estos comandos son válidos únicamente para la ventana **PGM Editor**. Son muy útiles porque permiten insertar, copiar, borrar y mover líneas. Se especifican en el área de números de la ventana **PGM** con la opción **Edit, Options** y luego **Numbers**. También se puede activar con el comando global **Numbers**.

- **Línea Sencilla**

- **A,B A** quiere decir después y B antes. Se utilizan con los comandos de línea copiar C, mover M y con los comandos de bloque copiar CC y mover MM.
- **C[n]** copia (n) una o más líneas consecutivas a otro lugar en el archivo, indicado por el comando de la línea destino (A ó B).
- **CU** convierte todos los caracteres de la línea especificada a mayúscula.
- **D[n]** borra (n) una o más líneas.
- **I[n] IA[n] IB[N]** inserta una o más líneas después o antes de la línea en la cual se ejecuta el comando.
- **M[n]** mueve una o más líneas a otro sitio en el archivo.

- **Bloque de Líneas**

Estos comandos se utilizan para marcar tanto el inicio como el fin de un bloque marcado, que puede ser copiado, movido o borrado. Después de marcado el bloque, se debe indicar el destino o posición del cursor; es decir, se debe escribir la letra B (antes de) o A (después de).

- **CC** Indica el inicio o el fin del bloque marcado para ser copiado.
- **MM** Indica el inicio o el fin del bloque marcado para ser movido a otro lugar.
- **DD** Indica el inicio o el fin del bloque marcado para ser borrado. Una vez indicado el fin de línea a borrar, inmediatamente desaparece las líneas marcadas. Se recomienda asegurarse cuales líneas borrar, ya que el SAS no le advierte, sino simplemente borra.

- **Comandos Globales del Program Editor**

Estos comandos pueden ser ejecutados en el **ToolBox**, estando activa la ventana del **Program Editor**, al presionar el tercer botón del ratón y desplegando el menú, ya sea en las opciones: **Run** (Submit o Recall) y **File** (Save o Save as). También pueden ser ejecutados activando la línea de comandos:

- **Submit** ejecuta un programa SAS.
- **Recall** trae las instrucciones a la ventana del editor de programa.
- **File** "nombre" almacena un programa con el archivo nombre.
- **File** "prn:" envia el programa SAS a la impresora.
- **Include** "pgmz. sas" copia el archivo pgmz. sas en la ventana PGM. Generalmente se utiliza en forma abreviada **INC**. También, se puede incluir un archivo seleccionando del menú la opción **File** y luego **Open**.
- **Clear** limpia borra el contenido de la ventana.

2.5 Modos de Ejecución de un Programa SAS

El SAS le ofrece al usuario 3 opciones para ejecutar un programa:

a. **Modo Interactivo**: para iniciar una sesión en modo línea interactivo invoque el SAS con la opción NODMS (NO DISPLAY MANAGER SYSTEM) así: `sas -nodms` <ENTER>

Luego aparecerá en pantalla el siguiente mensaje (prompt) **1?**

Allí puede comenzar a escribir instrucciones SAS, como por ejemplo:

```
1?  Data prueba;
2?  Input variedad $ precio;
3?  Cards;
criolla    300
valencia 400
californi 780
mandarin 880
;
proc print;
endsas;
```

b. **Modo Interactivo con el Display Manager System (DMS)**: es el modo activo por defecto en el ambiente X-Window cada vez que invoca el sistema así: **sas&** <ENTER>

c. **Con el asistente SAS/ASSIST**: es un modo interactivo de ventanas que proporciona el manejador del sistema (DMS) para que el usuario elija del menú las opciones o tareas que desea realizar, ya sea para crear, leer conjuntos de datos, procedimientos gráficos o estadísticos. Este módulo genera automáticamente las instrucciones ejecutadas; las mismas se pueden ver en la pantalla del editor (**PGM**) al seleccionar del menú **Run** la opción **Recall Text**.

d. **Modo no Interactivo**: para activar este modo escriba el comando `sas` seguido del nombre del archivo que contiene el programa a ser ejecutado; por ejemplo escriba: **sas prueba**

Automáticamente el sistema crea dos archivos en su directorio actual en Hydra para almacenar los resultados (**Output**) y la ejecución sintáctica del SAS (**Log**) en: **prueba.lst** y **prueba.log**, respectivamente. Si el archivo prueba tiene la extensión (**.sas)** no es necesario indicarla, ya que el comando **sas** utiliza por defecto dicha extensión.

i. **Modo Batch**: para ejecutar un trabajo escriba el comando **sas** seguido del nombre del archivo fuente del programa SAS (que ha sido

creado previamente con un editor de texto ASCII, con la extensión **.sas**), las opciones apropiadas del sistema y el símbolo amperson (**&**) al final del comando, como por ejemplo: **sas prueba -print prueba.out -log prueba.log &**

3 INTRODUCCIÓN AL LENGUAJE SAS

Como todo lenguaje de programación, el SAS tiene su propio vocabulario y sintaxis. Es importante hacer notar que el lenguaje SAS no establece diferencia entre mayúsculas y minúsculas. Sin embargo, si el usuario desea ejecutar comandos del sistema operativo **UNIX** desde el SAS debe tener en cuenta que si existe diferencia entre ellas.

3.1 Características del Lenguaje

Instrucciones SAS

Una instrucción SAS es una cadena de palabras claves, nombres, caracteres especiales y operadores, y finaliza con un punto y coma.

Ejemplos:

```
. put x $15.;
. data one;
. format valor  10.3 ;
```

Palabras Claves

Muchas de las instrucciones SAS empiezan con palabras claves como **Put**, **Data, Proc**, **Infile, Do.** Cabe resaltar que una palabra clave no es una palabra reservada, como es el caso del PASCAL.

Nombres SAS

El SAS utiliza diferentes tipos de nombres para: variables, conjunto de datos, formatos, procedimientos, opciones, arreglos, librerías o directorios. Los nombres en SAS pueden tener hasta 8 caracteres alfabéticos de largo; el primer carácter debe ser una letra o subrayado (_). No pueden aparecer espacios en blanco ni caracteres especiales como: $, @, #.
Los nombres para archivos o conjuntos de datos SAS pueden tener dos niveles, el primer nivel se refiere al nombre de referencia a la librería de almacenamiento del archivo y el segundo nivel se refiere al nombre propiamente dicho del archivo. Un ejemplo sería:

```
Libname milib '/home/faces/eliza/ejemplo';
Data   Milib.prueba; /*   Milib.prueba es un nombre con dos
niveles de un archivo de datos SAS */
```

3.2 Conjuntos de Datos, Constantes y Variables del SAS

Conjuntos de Datos (Data Set)
Un conjunto de datos es una colección de valores que toman las variables para varias observaciones. Se pueden crear uno o más conjuntos de datos dentro de un trabajo; los cuales tendrán su propio nombre SAS. Como ya se mencionó, un nombre SAS puede tener hasta 8 caracteres, el primero debe ser cualquier letra, los siguientes pueden ser letras, números o el signo especial de subrayado "_".

Algunos ejemplos de nombres SAS: DATA 1 D12_A POSGRADO A NUEVO_09 FORESTAL.

Variables
Las variables en el SAS son nombres suministrados por el usuario para representar las variables que serán usadas en los análisis; estos nombres de variables siguen las mismas reglas de los nombres SAS.

Las variables pueden ser numéricas o alfanuméricas, pero una variable numérica debe tener solamente datos numéricos. El máximo número de variables en un conjunto de datos es 1024.

Tipos de Datos

Los datos son clasificados como numéricos o alfanuméricos.

Datos Numéricos: Son números los cuales pueden estar precedidos por un signo más (+) o menos (-), las comas no pueden aparecer en un dato numérico; un entero puede ser escrito con o sin punto decimal, ya que no hay especificaciones en relación con enteros y reales, también se puede usar la notación E. Ejemplos de datos numéricos:
71 .00038 -4 8214.7221 +98.045 -4.00 8.546 E 02

Datos Alfanuméricos: Consiste en cadenas de caracteres, hasta 200 caracteres, estos pueden ser letras, números, blancos y caracteres especiales.

OTROS ASPECTOS BASICOS DEL SAS

Observaciones
Una observación es un grupo de valores que representan diferentes atributos que corresponden a una unidad para análisis como: una persona, un animal, una región geográfica, un mes en particular, etc.

Separadores
En SAS se utilizan uno o más espacios en blanco para separar variables, palabras claves, datos, etc.

Toda instrucción termina con punto y coma "**;**" ejemplo: VAR + VAR * 5.6769; PROC PRINT ; INPUT X1 X2 X3 ;

Comentarios
Se pueden incluir comentarios en cualquier lugar de un programa utilizando al inicio y final los siguientes símbolos:

```
/* ESTO ES UN COMENTARIO */
*- OTRO COMENTARIO;
** ESTE ES OTRO COMENTARIO;
```

ESTRUCTURA DE UN PROGRAMA SAS

La estructura general de un trabajo de análisis usando SAS es la siguiente:

- **Paso de Datos**: Creación de los conjuntos de datos, mediante el ingreso de los datos y definición de variables, con la instrucción DATA nombre SAS en combinación con alguna o algunas de las instrucciones de entrada de datos, ejemplo:

  DATA PRIMERA; INPUT NOMBRE $ 1-19 X1 20-25 X2 27 SEX 30;

  (Esta instrucción determina los nombres de las variables a usar y como encontrar los datos)
  Modificaciones o transformaciones de valores de variables, creación de nuevas variables utilizando las variables definidas en la instrucción de entrada de datos ejemplo:
  X2= X1**2 ;

- **Paso de Procedimientos**: Ejecución de los diversos análisis requeridos usando las instrucciones PROC (PROC es una palabra clave para llamar las diversas rutinas), las cuales se incluyen después de los datos.

Entonces, la estructura de un programa SAS es:
- **Data MERIDA; Input** varnomi n-n varnom2 n-n; varnom3 =expresión con operaciones (+,-,*,/,LOG, funciones trigonométricas, lógicas, etc)
- **Cards**; (indica comienzo de los datos)
- **(.)** (una o más líneas de datos por caso u observación, o varias observaciones por líneas).
- **(;)** (el punto y coma indica fin de datos)

- **Proc Rutina** (nombre de la rutina o procedimiento que se quiere utilizar, OPCIONES Y PARAMETROS de la rutina)
- **Title**....xxxxxxxxxxxx.....(imprime el título deseado, máximo 132 caracteres) (otras instrucciones PROC)

Ejemplo 3.2

Para este modelo se tienen datos sobre varios estudiantes (nombre, edad, sexo, peso, estatura); se quiere: 1) imprimir los datos, 2) calcular las estadísticas básicas como: número de observaciones, media, desviación estándar, valor mínimo, valor máximo y error estándar de la media para las variables especificadas, 3) hallar la correlación entre peso y estatura, y 4) ordenar los datos de acuerdo al nombre e imprimirlos.

```
/* EJEM3.2: EJEMPLO DE UN PROGRAMA SAS */
* Esto es un comentario; *- También es comentario;
DATA CLASE ;
INPUT NOMBRE $ 1-30  EDAD 32-33  SEXO 35  PESO 37-42  ESTATURA
44-48 edocivil $ 50-51;
CARDS;
PEREZ JUAN                      22 1 93.50  1.86  S
RODRIGUEZ ARMANDO               24 1 69.5   1.67  C
GIL PABLO                       27 1 83.5   1.84  S
MACHADO CARMEN                  23 2 58     1.62  S
BERMUDEZ ALBERTO                30 1 62     1.70  V
GRANADOS CARMEN                 24 2 60     1.64  S
ALVAREZ ANTONIO                 25 1 63     1.68  S
ZAPATA FELIPE                   22 1 75     1.72  C
BRACHO OMAIRA                   22 2 56     1.62  S
LOPEZ GILDA                     23 2 57     1.65  C
CHACON TERESA                   21 2 49     1.56  C
LEON VIRGINIA                   22 2 55     1.62  C
PADRON HENRY                    28 1 78     1.69  V
BAPTISTA JUAN                   26 1 95     1.83  V
PE¤A LUIS                       23 1 72     1.58  S
ANDRADE CARMEN                  20 2 60     1.70  S
RAMIREZ SONIA                   24 2 60     1.64  S
run;
*Se crea las etiquetas o formato para variables;
PROC FORMAT;
*es un formato para una variable numérica;
 VALUE SEX 1='MASCULINO' 2='FEMENINO';
 *es un formato para una variable Alfnumérica;
 value $edo 'S'='SOLTERO'
            'C'='CASADO'
                 'V'='VIUDO';
PROC PRINT ;
 TITLE  'IMPRESION DE LOS DATOS DEL PROBLEMA';
 FORMAT SEXO SEX. edocivil $edo.;
run;
PROC MEANS ;
VAR  EDAD  PESO  ESTATURA  ;
TITLE 'IMPRESION DE LAS ESTADISTICAS BASICAS' ;
run;
PROC CORR  DATA=CLASE
  PEARSON
```

```
 ;
VAR PESO ;
WITH ESTATURA  ;
TITLE ' RESULTADOS DE LA CORRELACION ENTRE PESO Y ESTATURA';

PROC SORT  ;   BY NOMBRE ;
PROC PRINT ;
TITLE 'DATOS DEL PROBLEMA ORDENADOS DE ACUERDO AL NOMBRE';
RUN ;
```

Análisis de los Resultados del Ejemplo 1

El conjunto de datos a utilizar en este análisis tiene como nombre SAS CLASE especificado en la instrucción DATA (DATA CLASE ;).

La instrucción INPUT indica que los datos van a ingresar en el programa de la manera siguiente:

NOMBRE dato alfanumérico, especificado por el signo $, y este dato se encuentra entre las columnas 1-19
EDAD dato numérico entre columnas 20-21
SEXO dato numérico en la columna 25
PESO dato numérico entre las columnas 28-31
ESTATURA dato numérico entre las columnas 34-37

Este tipo de lectura se denomina **tipo columna**.

Cards: indica comienzo de la entrada de datos. Se ve que para indicar fin de datos se necesita el punto y coma
Proc Format: procedimiento que permite crear etiquetas o formatos para variables numéricas o alfanuméricas.
Proc Print: imprime los datos en la forma como entraron y le coloca un título.
Proc Means: calcula los estadísticos básicos, anteriormente nombrados; para las variables especificadas en la instrucción VAR; e imprime los resultados y le coloca un título.
Proc Corr: Hace el análisis de correlación entre peso y estatura, e imprime resultados y título. El coeficiente de correlación calculado es el de Pearson, entre las variables PESO con ESTATURA.
Proc Sort; by Nombre; Ordena los datos contenidos en el conjunto de datos **Data Clase** de acuerdo a la variable nombre.
Proc Print: Imprime o muestra en la ventana del OUTPUT los datos anteriormente ordenados y le coloca el título indicado por la instrucción TITLES.

3.2 Paso de Datos

Se refiere a la creación de conjuntos de datos SAS, que son necesarios para realizar procedimientos gráficos, estadísticos, reportes, etc.
En general, en todos los pasos de datos el lugar donde se localizan éstos debe ser especificado con la instrucción: **Data** <nombre>;

Luego la siguiente instrucción puede ser **Infile** <nombre>; que indica que los datos se encuentran almacenados en un archivo externo (**ASCII**), mientras la instrucción **Cards** indica que los datos se encuentran a continuación y viene precedida de la instrucción **Input**.

Esto indica que existen dos maneras de leer datos en SAS, una desde archivos externos y otra, desde el mismo programa a continuación de la instrucción **Cards**. La instrucción de lectura utilizada en ambas formas es **Input**.

Es importante hacer notar que así como, se puede leer datos de forma interactiva (programa) y desde archivos externos; también la ejecución de los programas SAS se puede realizar: en forma interactiva (modo línea) y desde archivos externos (modo batch). En este último modo de lectura el programa SAS se escribe con procesadores de texto (**ASCII**), y se puede ejecutar de dos maneras: una forma es interactivamente, utilizando para ello, el comando general INCLUDE 'NOMBRE' o INC 'nombre'; y la otra forma es desde el sistema operativo así: **sas** nombre <enter>. Como se dijo, esta ejecución del SAS no interactiva, produce por defecto dos archivos, uno con extensión **.log** que muestra los mensajes sobre la ejecución de los comandos; y el otro archivo con extensión **.lis** contiene los resultados de la ejecución.

También se puede realizar la importación de acuerdo diferentes tipos de formatos de datos, como EXCEL, ACCESS, DBase, SPSS, Stata, entre otros. Es una opción que se puede ejecutar a través de la barra de menú principal en la opción **FILE** (**Archivo**), el submenú **Import** (**Importar datos**).

3.2.1 Instrucción Input

Permite la lectura de líneas de datos identificados con la instrucción **Infile** o **Cards**. Es una instrucción ejecutable y tiene tres modos de lectura: tipo columna, con formato y tipo lista.

La forma general de sintaxis es:
Input [pointer control] variable [format modifier] [$] [startcol-endcol] [informat]

Tipos de Lectura con la Instrucción Input

Con esta instrucción se pueden realizar lecturas de datos de varios tipos, entre otras, tipo columna, formato libre y con formato.

3.2.2 Tipo Columna

La lectura de cada variable se realiza especificando el inicio y fin de la columna a ser localizada. El ejemplo 1 de la página 19 realiza este tipo de lectura.

3.2.3 Formato Libre

Esta forma es apropiada para leer datos que no están localizados en columnas fijas. Por defecto, lee datos tipo carácter o numéricos, usa uno o más blancos para delimitar los campos; no puede leer valores tipo carácter con blancos intercalados, crea variables con una longitud de 8 y lee datos faltantes indicados con (.) punto.

Ejemplo 3.2.1

```
/* EJEMPLO 3.2.1: LECTURA FORMATO LIBRE */
OPTIONS LS=72 PS=55;
*Paso de datos por programa;
DATA CLASE;
INPUT NOMBRE $ SEXO $ EDAD ESTATURA PESO LUGAR $;
CARDS;
JUAN M 12 59.0 99.5 MERIDA
JAIME M 12 57.3 83.0 CARACAS
ALFREDO M 14 69.0 112.5 VALENCIA
WILLIAM M 13 62.5 112.0 CARACAS
JEFFREY M 13 62.5 84.0 MARACAY
RONALD M 15 67.0 133.0 BARQUISIMETO
TOMAS  M 11 57.5 85.0 MERIDA
FELIPE M 16 72.0 150.0 BARCELONA
ROBERTO M 12 64.8 128.0 PTO.LACRUZ
HENRY M 14 63.5 102.5 MERIDA
JANET F 15 62.5 112.5 MATURIN
JOYCE F 11 51.3 50.5 MARACAY
JUDY  F 14 64.3 90.0 CARACAS
CAROL F 14 62.8 102.5 SAN CRISTOBAL
JANE  F 12 59.8 84.5 BARINAS
LUISA F 12 56.3 77.0 BARINAS
BARBARA F 13 65.3 98.0 SAN FERNANDO
MARIA F 15 66.5 112.0 SAN TOME
ALICIA F 13 56.5 84.0 BARQUISIMETO
;
PROC PRINT;
```

Si observamos la ejecución de este ejemplo en la ventana OUTPUT, notamos que la variable LUGAR no fué leída correctamente, Por qué? Observe la salida siguiente.

Obs	NOMBRE	SEXO	EDAD	ESTATURA	PESO	LUGAR

1	JUAN	M	12	59.0	99.5	MERIDA
2	JAIME	M	12	57.3	83.0	CARACAS
3	ALFREDO	M	14	69.0	112.5	VALENCIA
4	WILLIAM	M	13	62.5	112.0	CARACAS
5	JEFFREY	M	13	62.5	84.0	MARACAY
6	RONALD	M	15	67.0	133.0	BARQUISI
7	TOMAS	M	11	57.5	85.0	MERIDA
8	FELIPE	M	16	72.0	150.0	BARCELON
9	ROBERTO	M	12	64.8	128.0	PTO.LACR
10	HENRY	M	14	63.5	102.5	MERIDA
11	JANET	F	15	62.5	112.5	MATURIN
12	JOYCE	F	11	51.3	50.5	MARACAY
13	JUDY	F	14	64.3	90.0	CARACAS
14	CAROL	F	14	62.8	102.5	SAN
15	JANE	F	12	59.8	84.5	BARINAS
16	LUISA	F	12	56.3	77.0	BARINAS
17	BARBARA	F	13	65.3	98.0	SAN
18	MARIA	F	15	66.5	112.0	SAN
19	ALICIA	F	13	56.5	84.0	BARQUISI

Observe que la variable fue truncada y en los casos en los cuales aparece un blanco intercalado.

Ejemplo 3.2.2

```
/* EJEMPLO 3.2.2: LECTURA FORMATO LIBRE CON BLANCOS INTERCALADOS EN LA
VARIABLE LUGAR */
OPTIONS LS=72 PS=55;
DATA CLASE;
INPUT NOMBRE $ SEXO $ EDAD ESTATURA PESO LUGAR & $13.;
CARDS;
JUAN M 12 59.0 99.5 MERIDA
JAIME M 12 57.3 83.0 CARACAS
ALFREDO M 14 69.0 112.5 VALENCIA
WILLIAM M 13 62.5 112.0 CARACAS
JEFFREY M 13 62.5 84.0 MARACAY
RONALD M 15 67.0 133.0 BARQUISIMETO
TOMAS  M 11 57.5 85.0 MERIDA
FELIPE M 16 72.0 150.0 BARCELONA
ROBERTO M 12 64.8 128.0 PTO. LA CRUZ
HENRY M 14 63.5 102.5 MERIDA
JANET F 15 62.5 112.5 MATURIN
JOYCE F 11 51.3 50.5 MARACAY
JUDY  F 14 64.3 90.0 CARACAS
```

```
CAROL F 14 62.8 102.5 SAN CRISTOBAL
JANE  F 12 59.8 84.5 BARINAS
LUISA F 12 56.3 77.0 BARINAS
BARBARA F 13 65.3 98.0 SAN FERNANDO
MARIA F 15 66.5 112.0 SAN TOME
ALICIA F 13 56.5 84.0 BARQUISIMETO
;
PROC PRINT;
RUN;
```

Aquí se corrige el anterior problema, pues permite la lectura con formato libre, con blancos intercalados y se especifica el formato de lectura para la variable lugar. Los blancos intercalados se reconocen por el carácter &.

Los caracteres **& y (:)** son modificadores del formato de lectura. Estos son útiles en la lectura tipo lista.

El carácter **(:)** permite especificar un formato de lectura para datos no estándar y el carácter & permite especificar un formato de lectura para datos no estándar y lee valores carácter que contienen blancos intercalados.

Ejemplo 3.2.3
El ejemplo 3.2.3 muestra la lectura tipo lista y con datos faltantes en la variable LUGAR, en las líneas 2 y 5.

```
/* EJEMPLO 3.2.3: LECTURA FORMATO LIBRE CON BLANCOS INTERCALADOS EN
   LA VARIABLE LUGAR Y VALORES FALTANTES*/
OPTIONS LS=72 PS=55;
DATA CLASE;
 INPUT NOMBRE $ SEXO $ EDAD ESTATURA PESO LUGAR & $13.;
 CARDS;
JUAN M 12 59.0 99.5 MERIDA
JAIME M 12 57.3 83.0
ALFREDO M 14 69.0 112.5 VALENCIA
WILLIAM M 13 62.5 112.0 CARACAS
JEFFREY M 13 62.5 84.0
RONALD M 15 67.0 133.0 BARQUISIMETO
TOMAS  M 11 57.5 85.0 MERIDA
FELIPE M 16 72.0 150.0 BARCELONA
ROBERTO M 12 64.8 128.0 PTO. LA CRUZ
HENRY M 14 63.5 102.5 MERIDA
JANET F 15 62.5 112.5 MATURIN
JOYCE F 11 51.3 50.5 MARACAY
```

```
JUDY  F 14 64.3 90.0 CARACAS
CAROL F 14 62.8 102.5 SAN CRISTOBAL
JANE  F 12 59.8 84.5 BARINAS
LUISA F 12 56.3 77.0 BARINAS
BARBARA F 13 65.3 98.0 SAN FERNANDO
MARIA F 15 66.5 112.0 SAN TOME
ALICIA F 13 56.5 84.0 BARQUISIMETO
;
PROC PRINT;
RUN;
```

Al ejecutar el programa se produjeron errores en la lectura, si observamos la salida de la ventana OUTPUT, qué pasó al ejecutar el programa? Observe el cuadro de la página siguiente.

EJECUCION DEL PROGRAMA

Obs	NOMBRE	SEXO	EDAD	ESTATURA	PESO	LUGAR
1	JUAN	M	12	59.0	99.5	MERIDA
2	JAIME	M	12	57.3	83.0	ALFREDO M 14
3	WILLIAM	M	13	62.5	112.0	CARACAS
4	JEFFREY	M	13	62.5	84.0	RONALD M 15 6
5	TOMAS	M	11	57.5	85.0	MERIDA
6	FELIPE	M	16	72.0	150.0	BARCELONA
7	ROBERTO	M	12	64.8	128.0	PTO. LA CRUZ
8	HENRY	M	14	63.5	102.5	MERIDA
9	JANET	F	15	62.5	112.5	MATURIN
10	JOYCE	F	11	51.3	50.5	MARACAY
11	JUDY	F	14	64.3	90.0	CARACAS
12	CAROL	F	14	62.8	102.5	SAN CRISTOBAL
13	JANE	F	12	59.8	84.5	BARINAS
14	LUISA	F	12	56.3	77.0	BARINAS
15	BARBARA	F	13	65.3	98.0	SAN FERNANDO
16	MARIA	F	15	66.5	112.0	SAN TOME
17	ALICIA	F	13	56.5	84.0	BARQUISIMETO

El siguiente cuadro es una vista parcial de la ventana **log** de ejecución del programa.

```
1     /* EJEMPLO 3.2.3: LECTURA FORMATO LIBRE CON BLANCOS
INTERCALADOS EN
2  LA VARIABLE LUGAR Y VALORES FALTANTES*/
3     OPTIONS LS=72 PS=55;
4 DATA CLASE;
5 INPUT NOMBRE $ SEXO $ EDAD ESTATURA PESO LUGAR & $13.;
6 CARDS;
26   ;
NOTE: SAS went to a new line when INPUT statement reached past
the end of a line.
NOTE: The data set WORK.CLASE has 17 observations and 6
  variables.
NOTE: The DATA statement used 7.00 seconds.
      27  PROC PRINT;

      28  RUN;

NOTE: The PROCEDURE PRINT used 6.00 seconds.
```

Observen que si especificamos los datos faltantes con punto (.) se corrige el error de lectura. Si los datos de este ejemplo se encuentran almacenados en el archivo tipo texto 'DATOS', como se deben leer?

Ejemplo 3.2.3a

En este ejemplo se realiza la lectura desde un archivo externo tipo texto (ASCII), en formato libre con blancos intercalados en la variable lugar y valores faltantes.

```
/* EJEMPLO 3.2.3a: LECTURA FORMATO LIBRE CON BLANCOS INTERCALADOS EN UNA
LA VARIABLE LUGAR Y VALORES FALTANTES*/
OPTIONS LS=72 PS=55;
DATA CLASE;
 INFILE 'C:\Users\Usuario\Documents\computacion estadistica\ejemsas\DATOS.DAT'
MISSOVER;
 INPUT NOMBRE $ SEXO $ EDAD ESTATURA PESO LUGAR & $13.;
PROC PRINT;
RUN;
```

La instrucción **Infile** abre para lectura el archivo **Datos** que se encuentra ubicado en el directorio raíz especificado por el usuario, y la opción **Missover** indica que debe leer todas las variables en el registro actual, lo que obliga la lectura hasta el fin de línea. De esa manera, si existen datos faltantes les asigna (**.)** y no continúa la lectura en el siguiente registro, como hace en el anterior ejemplo.

Con formato de lectura (informat)
Este modo es apropiado para la lectura de datos en columnas fijas, para leer datos tipo carácter y numéricos estándar y no estándar. También permite leer valores tipo fecha (date Value) y los convierte a tipo fecha SAS.

La forma general es:
informatnamew.d
informatname es el nombre del formato de lectura (informat)
- w especifica el ancho del campo, es opcional
- . es el separador obligatorio
- d opcionalmente especifica los dígitos decimales

Este tipo de lectura también utiliza el control del apuntador (absoluto o relativo) a las posiciones de columnas.

Control del Apuntador de Columnas:

Absoluto

@n dirige el apuntador de columna a la posición n.
n puede ser un entero positivo o una variable (con valores positivos) que esta el PDV (vector de datos del programa).

```
INPUT SSN $11. @15 SALARIO 5. @42 DEPTO $3.;
```

Esta instrucción indica que la variable SSN es tipo carácter, con ancho 11, luego se indica el avance del apuntador a la columna 15, a partir de allí se lee la variable numérica SALARIO con ancho 5. Después se indica que avance a la columna 42 y lee la variable DEPTO con ancho 3.

Relativo

+*n* dirige el apuntador de columna para que se mueva **n** columnas relativas a la posición actual.
n puede ser un entero positivo o una variable (con enteros positivos o negativos) que está en el vector de datos del programa (PDV).

El siguiente ejemplo:

```
INPUT SSN $11. +3 SALARIO 5. +22 DEPTO $3.;
```

Indica que la variable SSN es tipo carácter, con ancho 11, luego se indica el avance del apuntador 3 columnas, a partir de allí se lee la variable numérica SALARIO con ancho 5. Después se indica que avance 22 columnas y allí lee la variable DEPTO con ancho 3 caracteres. Los apuntadores a columnas pueden ser utilizados en cualquier modo de lectura.

Realice el siguiente ejemplo y observe la ejecución del mismo:

```
/* EJEMPLO 3.2.3b: LECTURA FORMATO FIJO (FORMATTED INPUT)
   ESPECIFICA LA POSICION DE INICIO, EL ANCHO DE CADA CAMPO
   MUEVE EL APUNTADOR A LA POSICION DE INICIO DEL CAMPO Y
   LUEGO ESPECIFICA LA VARIABLE Y EL FORMATO DE LECTURA (INFORMAT)
*/
OPTIONS LS=72 PS=55;
/* LAS OPCIONES LS Y PS CONTROLAN EL ANCHO Y LARGO DE LA PAGINA,
   RESPECTIVAMENTE. */
DATA CLASE1;
 INPUT NOMBRE $8. @11 SEXO $1. +1 EDAD 2. +1 ESTATURA 4. +1 PESO
5.;
 CARDS;
JUAN      M 12 59.0  99.5
JAIME     M 12 57.3  83.0
ALFREDO   M 14 69.0 112.5
WILLIAM   M 13 62.5 112.0
JEFFREY   M 13 62.5  84.0
RONALD    M 15 67.0 133.0
TOMAS     M 11 57.5  85.0
FELIPE    M 16 72.0 150.0
ROBERTO   M 12 64.8 128.0
HENRY     M 14 63.5 102.5
JANET     F 15 62.5 112.5
JOYCE     F 11 51.3  50.5
JUDY      F 14 64.3  90.0
CAROL     F 14 62.8 102.5
JANE      F 12 59.8  84.5
LUISA     F 12 56.3  77.0
BARBARA   F 13 65.3  98.0
MARIA     F 15 66.5 112.0
ALICIA    F 13 56.5  84.0
;
PROC PRINT;
PROC MEANS;
 VARIABLES ESTATURA PESO;
PROC PLOT;
 PLOT PESO*ESTATURA=SEXO; RUN;
```

3.2.4 Lectura de múltiples registros por observación

El control del apuntador de línea está disponible con la instrucción INPUT para la lectura de múltiples líneas de datos (en programa o en archivos) que son combinados en una sola observación.

El control del apuntador de línea puede ser:

- **Relativo**

  - (- /) carga el siguiente registro en el vector (buffer) de lectura y coloca el apuntador de columna en la posición 1.
  - un sólo buffer de lectura es utilizado.
  - este apuntador puede ser utilizado en cualquier modo de lectura.

Realice el siguiente ejemplo y visualice la lectura. Explique cómo se realizó la lectura?

Ejemplo 3.2.4.1: Use el control apuntador / para leer las etiquetas de direcciones de correo, donde cada etiqueta se almacenó en cuatro registros consecutivos.

```
DATA INFO;
 INPUT NOMBRE $15. / DIRECC $20.
         / CIUDAD  $15. EDO $2. / ZIP $5.;
 CARDS;
C.J. DORN
2341 PENN AVE.
WASHINGTON     DC
20016
B.A. GREEN
1180 WOODVIEW DRIVE
NEW YORK       NY
10032
;
  PROC PRINT DATA=INFO NOOBS;
RUN;
```

- **Absoluto**

  - #n dirige el apuntador de línea a la n-ésima línea.
  - n puede ser un entero positivo o variable entera positiva.
  - este apuntador puede ser utilizado en cualquier tipo de lectura.

Ejemplo 3.2.4.2: Use el control del apuntador para leer los datos de etiquetas de correo del ejemplo anterior.

```
*LECTURA DE MULTIPLES REGISTROS, CON CONTROL APUNTADOR DE LINEA
ABSOLUTO #;
DATA INFO;
 INPUT #1 NOMBRE $15. #2 DIRECC $20.
         #3 CIUDAD  $15. EDO $2. #4 ZIP $5.;
 CARDS;
C.J. DORN
2341 PENN AVE.
WASHINGTON     DC
20016
B.A. GREEN
1180 WOODVIEW DRIVE
NEW YORK       NY
10032
;
  PROC PRINT DATA=INFO NOOBS;
RUN;
```

Ejemplo 3.2.4.3: Las altas temperaturas de cada día en un lugar específico son almacenadas en un archivo 'temper'. Cada registro contiene las temperaturas para cuatro días consecutivos, así:

050186 74 050286 79 050386 81 050486 80
050586 82 050686 83 050786 83 050886 82
050986 83 051086 84 051186 83 051286 85

```
*Paso de datos desde un archivo externo y lectura de multiples
observaciones en un registro;
DATA TEMPS;
  INFILE 'temper.dat';
  INPUT FECHA : MMDDYY6. TEMP @@;
  FORMAT FECHA DATE7.;
  PROC PRINT;
RUN;
```

La instrucción INPUT indica que la variable FECHA (tipo fecha) utiliza un modificador de formato (:) y el formato (MMDDYY6.) mes, día y año, con ancho 6, mientras la variable TEMP es numérica y @@ especifica que se debe mantener la lectura en el registro hasta que no se encuentre fin de línea. Mientras que @ indica que se debe mantener la lectura del registro para la siguiente lectura. @@ típicamente es utilizado para la lectura de múltiples

observaciones desde una línea de datos; no debe ser usado con el control del apuntador @ en la lectura tipo columna.

La instrucción FORMAT especifica que la variable FECHA va ser almacenada en el archivo SAS de acuerdo al formato estándar DATE7. (2 caracteres para el día, 3 para el mes y dos para el año). El usuario puede definir el formato a utilizar con el procedimiento PROC FORMAT, el cual permite sustituir los valores de una variable por etiquetas descriptivas.
Su forma general es:

PROC FORMAT;
VALUE nombrefmt num1=’etiqueta1’
num2=’etiqueta2’;

A continuación se muestra un ejemplo donde se define un formato particular.

Ejemplo 3.2.4.4
Crear un conjunto de datos SAS llamado FINANZA, definir el formato para los valores de la variable DEPT.

```
DATA FINANZA;
  INPUT NOMBRE $10. @12 DEPT 3. @16 SALARIO 5.;
CARDS;
LAREZ, P.  101 39000
PEREZ, J.  101 21000
RUIZ, A.   105 26500
SUAREZ, L. 105 42600
;
PROC FORMAT;
 VALUE DEPTFMT 101='EDUCACION'
               105='PUBLICIDAD'
               OTHER='OTRO';
PROC PRINT LABEL DATA=FINANZA; FORMAT SALARIO DOLLAR8.
DEPT DEPTFMT.;  LABEL DEPT='DEPARTAMENTO';
TITLE 'SALARIO DEL PERSONAL POR DEPARTAMENTO';
FOOTNOTE 'INFORMACION CONFIDENCIAL';  RUN;
```

Hasta ahora hemos visto que, la lectura de múltiples registros por observación puede ser procesada con el control de apuntador de línea # (absoluto) o el / (relativo). También se puede realizar utilizando múltiples instrucciones INPUT. Un ejemplo sería:

Ejemplo 3.2.4.5

```
DATA CLASE1;
INFILE 'EJEM45.DAT';
 INPUT NOMBRE $8. @11 SEXO $1.;
 INPUT EDAD  4-5;
```

```
 INPUT ESTATURA : 5. PESO : 5.;   /*lectura tipo lista con
modificador  de formato */
PROC PRINT;
RUN;
```

3.2.5 Lectura de archivos jerárquicos

Muchos archivos son jerárquicos en su estructura porque tienen un registro encabezamiento y varios registros para detalles. Existen dos maneras para leer un archivo jerárquico:

- *Crear una observación por registro detalle y almacenar la información del registro encabezamiento como parte de cada observación.*

Ejemplo 3.2.5.1

Un archivo contiene el número de cuenta de crédito sobre un registro encabezado por H, seguido por uno o más registros de detalles, identificados por D=débitos y C=créditos que representan las transacciones de cada cuenta. Se pide crear un conjunto de datos SAS que contenga una observación por débito.

```
/* EJEMPLO 3.2.5.1*/
    DATA CUENTA (DROP=TIPO); /* EXCLUYE DEL ARCHIVO LA VARIABLE
    TIPO */
    RETAIN No_CTA; /*conserva el valor de la variable No.CTA */
    INFILE 'TRANSAC';
    INPUT TIPO $1. @;
    IF TIPO='H' THEN INPUT @3 No_CTA $4.;
    IF TIPO='D';  /* Aquí crea el subconjunto con D */
    INPUT @3 FECHA DATE7. TOTAL 7.;
    PROC PRINT NOOBS;
    FORMAT FECHA DATE7.;
    TITLE 'DEBITOS';
    RUN;
```

- *Crear una observación por registro encabezamiento y almacenar toda la información de los registros de detalle, que le corresponden como parte de esa observación.*

Ejemplo 3.2.5.2

Crear un archivo de datos SAS a partir del archivo ALIMENTO, en el cual las tres primeras líneas se refieren al registro encabezamiento (A), que contienen los nombres y direcciones de las familias.
Estos registros están seguidos por un número variable de registros de detalles; los cuales contienen información sobre las compras de alimentos y que son denotados con: L los

productos lácteos, F frutas y G granos o C carnes. Calcular el dinero total gastado por cada familia en las compras de carnes C, lácteos L y frutas F. Incluya sólo los nombres de las familias y totales.

```
/* EJEMPLO 3.2.5.2 */
DATA ALIMENT (KEEP=NOMBRE CARNE LACTEOS FRUTAS);  /* LA OPCION KEEP INDICA LAS VARIABLES QUE SE VAN A INCLUIR EN EL ARCHIVO, EN ESTE CASO: CARNES LACTEOS Y FRUTAS */
RETAIN NOMBRE;
INFILE 'ALIMENTO' END=EOF; /* EOF=1 CUANDO ES FIN DE ARCHIVO */
INPUT @1 TIPO $1. @;
IF TIPO='A' THEN
DO;
IF _N_ GT 1 THEN OUTPUT; /* _N_ ES UNA VARIABLE AUTOMATICA DEL SISTEMA QUE INDICA EL NUMERO DE REGISTROS LEIDOS */
CARNE=0;
LACTEOS=0;
FRUTAS=0;
INPUT @3 NOMBRE &  $19. / /;
END;
ELSE
DO;
INPUT @3 ITEM $ PRECIO;
IF TIPO='C' THEN CARNE+PRECIO; /* ES UNA INSTRUCCION DE TOTALIZACION */
ELSE IF TIPO='L' THEN LACTEOS+PRECIO;
ELSE IF TIPO='F' THEN FRUTAS+PRECIO;
END;
IF EOF=1 THEN OUTPUT;
PROC PRINT DATA=ALIMENT NOOBS;
RUN;
```

3.2.6 Procesamiento iterativo

Algunas aplicaciones requieren el procesamiento iterativo de lectura en el paso de datos. Los lazos DO loops proporcionan esta facilidad. Estos se pueden utilizar para realizar cálculos repetitivos, generar datos, eliminar código redundante, ejecutar instrucciones condicionales y leer datos.

El número de veces que un DO loop es ejecutado se puede definir especificando los valores de la variable índice (si es un lazo simple) o proporcionando una condición que indique cuando finalizar (procesamiento condicional).

En el procesamiento simple DO loop puede especificar los valores que toma la variable índice al inicio, como incremento y para finalizar. Estos valores pueden ser: numéricos discretos, caracteres o expresiones. Cuando se utilicen valores numéricos discretos se deben separar con comas entre sí. Los valores carácter se especifican entre apóstrofes (‘ ‘) y separados por comas. Algunos ejemplos donde se indica inicio, fin e incremento son:

DO I=2 TO 10 BY 2;
DO K=3.6 TO 4.8 BY .05;

Para valores discretos y separados por comas
DO N=1,5,15,30,60;

Para valores caracteres
DO MES=’ENE’,’FEB’,’MAR’;
Con expresiones
DO Z=K TO (N/10);

Ejemplo 3.2.6.1
Se invierten Bs. 20000 en una cuenta el 1o. de Enero 1985. Determine el valor de la cuenta 20 años después a una tasa de interés constante del 10% anual.

```
/* EJEMPLO 3.2.6.1 */
/*ILUSTRA EL USO DE ESTRUCTURAS REPETITIVAS */
DATA INVERS;
CAPITAL=20000;
DO WHILE (CAPITAL <= 500000);
  CAPITAL+(CAPITAL*0.1);
END; /* FIN DEL DO */
PROC PRINT DATA=INVERS;
 TITLE 'TOTAL CAPITALIZADO';
RUN;
```

Escriba y ejecute el anterior programa. Explique por qué el archivo contiene una sola observación y por qué la variable YEAR tiene el valor 2005 y no 2004.
Para generar una observación separada para cada año, se debe agregar la instrucción OUTPUT al final del lazo, la cual indica que se debe grabar la información en el archivo.

El Proc Print mostrará el capital acumulado para cada año.

```
DATA INVERS;
CAPITAL=20000;
DO YEAR= 1985 TO 2004;
CAPITAL+(CAPITAL*0.1) ;
OUTPUT;
```

```
END;
PROC PRINT DATA=INVERS;
TITLE ‘TOTAL CAPITALIZADO’;
RUN;
```

Se puede utilizar el procesamiento iterativo condicionalmente a través de la instrucción lógica IF-THEN dentro del lazo (loop) o utilizando las instrucciones DO WHILE o DO UNTIL.

Ejemplo 3.2.6.2
En el ejercicio anterior determine el número de años que tomará para que el capital invertido exceda de 500000, si el interés es del 10% anual.

```
/* EJEMPLO 3.2.6.2 ILUSTRA EL USO DE ESTRUCTURAS REPETITIVAS */
DATA INVERS;
RETAIN YEAR 0;
CAPITAL=20000;
DO WHILE (CAPITAL <= 500000);
CAPITAL+(CAPITAL*0.1);
YEAR+1;
END; /* FIN DEL DO */
PROC PRINT DATA=INVERS;
TITLE ‘TOTAL CAPITALIZADO’;
RUN;
```

Con la instrucción DO WHILE la repetición del bloque de instrucciones (lazo) continúa mientras una expresión es verdadera y ésta es evaluada al inicio del lazo (indica que si la expresión es falsa al inicio del lazo nunca se ejecuta). Su forma general es:

```
DO WHILE(expresión);
instrucciones
END;
```

Con la instrucción DO UNTIL la repetición del lazo continúa hasta que la condición es verdadera y ésta es evaluada al final del lazo. La evaluación al final del lazo significa que las instrucciones del mismo son ejecutadas al menos una vez. Su forma general es:

```
    DO UNTIL(expresión);

instrucciones
END;
```

Ejemplo 3.2.6.3
Utilice la instrucción DO UNTIL para resolver el ejercicio anterior.

```
/*EJEMPLO 6.3 */
```

```
DATA INVERS;
CAPITAL+20000;
DO UNTIL(CAPITAL GT 500000) ;
YEAR+1;
CAPITAL+(CAPITAL*0.1);
END;
PROC PRINT DATA=INVERS NOOBS;
TITLE ‘NUMERO DE AÑOS PARA QUE EL CAPITAL EXCEDA 500000’;
RUN;
```

También se pueden combinar expresiones condicionales con la instrucción simple DO loop. Su forma general es:

```
DO variable=inicio TO fin [BY incremento]
                [WHILE|UNTIL(expresión)];
instrucciones ;
END;
```

Ejemplo 3.2.6.4

Repita el ejercicio anterior pero determine el año y el mes en que la cuenta excederá de Bs. 500000, sí el interés es capitalizado mensualmente. Utilice la instrucción DO WHILE y DO loop condicional.

```
/*EJEMPLO 3.2.6.4 */
DATA INVERS;
DO WHILE(CAPITAL LE 500000) ;
YEAR+1;C
CAPITAL+20000;
DO MONTH=0 TO 11 WHILE(CAPITAL LE 500000);
CAPITAL+(CAPITAL*0.1/12);
END;
PROC PRINT DATA=INVERS NOOBS;
VAR YEAR MONTH CAPITAL;
TITLE ‘NUMERO DE AÑOS Y MES EN QUE EL CAPITAL EXCEDE 500000’;
RUN;
```

También se pueden utilizar lazos anidados DO loop para leer datos.

3.2.7 Uso de Arreglos

Un arreglo es una colección de variables SAS que son agrupadas bajo un solo nombre. Esas variables son llamadas elementos del arreglo y cada elemento es identificado por un subíndice que representa la posición del elemento en el arreglo.

Existen dos tipos de arreglos en SAS:

- arreglos con subíndice explícito
- arreglos con subíndice implícito

Los arreglos se pueden utilizar para: realizar cálculos repetitivos, crear varias variables de una forma similar, leer datos, hacer las mismas comparaciones para variables del mismo tipo.

Definición de un arreglo explícito

La instrucción **Array** es utilizada para definir el conjunto de variables a ser procesadas de igual forma. Su forma general es:

Array nombrear {n} [$] [longitud] elementos;
nombrear es el nombre SAS del arreglo

- n indica el número de elementos del arreglo
- $ indica que los elementos son de tipo carácter
- longitud define una longitud común para cada elemento
- elementos son las variables declaradas como elementos

Una instrucción **Array** debe contener todas las variables numéricas o todas tipos carácter. Debe ser utilizada para definir el arreglo, antes que ese arreglo pueda ser referenciado. Esta instrucción crea variables si estas aún no han sido definidas. No es una instrucción ejecutable.

Para procesar un arreglo explícito generalmente se utilizan los lazos **Do loop**. El valor del subíndice indica el elemento del arreglo a ser procesado. Su forma general es:

```
ARRAY nombrear {n} elementos;
DO subindi=1 TO n
instrucciones SAS utilizando el arreglo
END;
```

subindi define la variable subíndice que accederá cada elemento del arreglo.
nombrear{subindi} identifica un elemento específico del arreglo.

Ejemplo 3.2.7.1
El siguiente ejemplo se refiere al uso de arreglos para realizar cálculos repetitivos. Utilice arreglos para convertir los valores de cuatro variables sobre estatura de pulgadas a centímetros, y los valores de cuatro variables de peso de libras a kilogramos.

```
/* EJEMPLO 3.2.7.1 */
DATA METRICO(DROP=I);
```

```
INPUT TIPO $ AL1 P1 AL2 P2 AL3 P3 AL4 P4;
ARRAY ALT[4] AL1-AL4; /*Aquí se define el arreglo */
ARRAY PES[4] P1-P4;
DO I=1 TO 4;
ALT[I]=ALT[I]*2.54; /*CONVERSION DE PULGADAS A CENTIMETROS */
PES[I]=PES[I]*.454 ; /*CONVERSION LIBRAS A KILOGRAMOS */
END;
CARDS;
A 65 124 68 155 63 142 69 177
B 67 166 73 191 65 122 70 150
C 66 133 67 161 62 115 74 196
D 75 204 71 166 65 152 72 180
;
FORMAT AL1-AL4 P1-P4 5.1;
PROC PRINT NOOBS;
RUN;
```

Es importante hacer notar que los nombres de arreglos no se pueden utilizar en las instrucciones **Label, Format, Drop, Keep** o **Retain**.
El siguiente ejemplo muestra la creación de variables con arreglos y de dimensión 2. El número de elementos que debe contener un arreglo de dimensión 2 es igual al producto de la dimensión uno por la dos. En el ejemplo se va a generar una matriz aleatoria de ceros y unos, de orden 28 por 3 lo cual indica que contiene 84 elementos (variables).

Definición de un arreglo implícito

Para procesar un arreglo implícito generalmente se utilizan los lazos DO OVER. El valor del subíndice indica el elemento del arreglo a ser procesado. Su forma general es:

- **Array** nombrear[(índice)] [$] [longit] elementos;
- nombrear es un nombre válido SAS
    - índice es el nombre del índice del arreglo
    - $ indica que los elementos del arreglo son tipo carácter.
- longit define una longitud común para cualquier elemento del arreglo, al que no ha sido asignado previamente longitud.
- elementos son las variables declaradas como elementos del arreglo.

Si la variable índice del arreglo no está explícitamente definida, el supervisor del SAS crea automáticamente la variable _I_ y la coloca en el vector de datos del programa (PDV) pero no la escribe en el archivo de salida del SAS.

Es importante resaltar que, este vector de datos (PDV) es un área de trabajo en memoria, que es temporalmente almacenada para cada observación. Es construido para la información de todas las variables del programa. Las variables que contiene las clasifica en: definidas por el usuario y automáticas o definidas por el sistema. Dichas variables están contenidas en el PDV pero no son almacenadas en el conjunto de datos SAS.

Normalmente en cada paso de datos se crean automáticamente dos variables: _N_ y _ERROR_, la primera mantiene el número de veces que es ejecutado el paso de datos, es decir contiene el número de observaciones y la segunda es una variable dicotómica, es igual a cuando ocurre un error en la lectura de los datos, de otra manera es igual a cero.

Si existe un arreglo definido implícitamente automáticamente se crea otra variable, esto es _I_ que funciona como subíndice. La información PDV que incluye es la siguiente:

. el nombre, tipo y longitud de cada variable

. la posición de cada variable en el conjunto SAS

. atributos opcionales de cada variable (label, informat y format).

Anteriormente se indicó, que estos arreglos se pueden procesar utilizando el lazo DO OVER, para ejecutar un grupo de instrucciones para cada elemento del arreglo. La forma general de esta instrucción es:

DO OVER nombrearreglo;

instrucciones;

END;

El siguiente ejemplo ilustra el uso de estos arreglos.

Ejemplo 3.2.7.2
Utilice arreglos implícitos para convertir los valores de cuatro variables, de pulgadas a centímetros y los valores de cuatro variables de libras a Kilogramos.

```
/* EJEMPLO 3.2.7.2 ARREGLOS IMPLÍCITOS */
DATA MÉTRICO;
INPUT TIPO $1. AL1 P1 AL2 P2 AL3 P3 AL4 P4;
ARRAY ALT  AL1-AL4; /*Aquí se define el arreglo */
ARRAY PES  P1-P4;
DO OVER ALT;
ALT=ALT*2.54; /*CONVERSION DE PULGADAS A CENTÍMETROS */
PES=PES*.454; /*CONVERSION LIBRAS A KILOGRAMOS */
```

```
END;
CARDS;
A 65 124 68 155 63 142 69 177
B 67 166 73 191 65 122 70 150
C 66 133 67 161 62 115 74 196
D 75 204 71 166 65 152 72 180
;
FORMAT AL1-AL4 P1-P4 5.1;
PROC PRINT NOOBS;
RUN;
```

Ejecuten este mismo programa sustituyendo la instrucción DO OVER por la instrucción DO loop.
Los arreglos implícitos, al igual que los explícitos se pueden utilizar para crear variables.

Ejemplo 3.2.7.3

Los siguientes datos se refieren a las ventas trimestrales en millones de Bolívares para cada año. Determine el porcentaje del total para cada trimestre.

```
1984   15.1  12.2  16.2  22.1
1985   12.2   9.3  11.4  14.6
1987    17.3  18.1  15.4  24.8
1988    18.9  19.7  17.6  25.1
```

```
/* EJEMPLO 3.2.7.3 ARREGLOS IMPLÍCITOS Y CREANDO VARIABLES */
DATA CONVERT(DROP= PRIMERO SEGUNDO TERCERO CUARTO TOTAL);
INPUT YEAR PRIMERO SEGUNDO TERCERO CUARTO;
TOTAL=SUM(OF PRIMERO—CUARTO); /* UN SOLO GUIÓN PARA VAR P1-P4 */
ARRAY VENTAS PRIMERO—CUARTO; /*Aquí se define el arreglo */
ARRAY PORCENTA  P1-P4;
DO OVER VENTAS;
PORCENTA=100*VENTAS/TOTAL;
END;
CARDS;
1984    15.1 12.2 16.2 22.1
1985    12.2  9.3 11.4 14.6
1986    16.2 18.2 17.3 25.2
1987    17.3 18.1 15.4 24.8
1988    18.9 19.7 17.6 25.1
;
PROC PRINT NOOBS;
TITLE ‘USO DE ARREGLOS IMPLÍCITOS Y CREANDO VARIABLES’;
FORMAT P1-P4 3.0;
```

```
RUN;
```

Al igual que la introducción DO loop, la instrucción DO OVER se puede utilizar en forma anidada, pero se debe tener cuidado en la forma como se manejan los subíndices, el siguiente ejemplo ilustra esta situación.

Ejemplo 3.2.7.4
Los siguientes datos se refieren a la fecha, tres tasas de interés y cinco cantidades invertidas. Crear un conjunto SAS con una observación para cada posible combinación de interés, cantidad invertida para cada año. Calcule la ganancia obtenida para cada cantidad invertida. Imprima el conjunto de datos.

```
11/12/86  .065  .070  .075  1000  5000  10000  20000  25000
11/19/86  .061  .068  .072   500  7500  15000  30000  50000

/* EJEMPLO 3.2.7.4 ARREGLOS IMPLÍCITOS ANIDADOS */
DATA PRODUCTO(KEEP=FECHA TASA CANTIDAD RETURN);
INPUT FECHA MMDDYY8. T1 -T3 C1-C5;
FORMAT FECHA MMDDYY8.;
ARRAY TASAS T1-T3;
ARRAY INVERS C1-C5;
DO OVER TASAS;
DO OVER INVERS;
TASA=TASAS;          /*NO SE PUEDEN ALMACENAR LOS ARREGLOS DIRECTAMENTE */
CANTIDAD=INVERS;
RETURN=TASAS*INVERS;
OUTPUT;
END;
END;
CARDS;
11/12/86 .065 .070 .075 1000 5000 10000 20000 25000
11/19/86 .061 .068 .072  500 7500 15000 30000 50000
;
PROC PRINT NOOBS;
 FORMAT CANTIDAD RETURN DOLLAR10.2;
RUN;
```

Ejecute el programa y observe la VENTANA LOG. Por qué el arreglo del subíndice está fuera de rango? ¿Cómo deben variar los subíndices? ¿Qué pasaría si el lazo externo fuese el del INVERS y el interno del arreglo TASAS?

Ejemplo 3.2.7.5
Ejecute nuevamente el anterior programa, pero coloque un subíndice diferente para el arreglo INVERS, digamos K.

```
/* EJEMPLO 3.2.7.5
ARREGLOS IMPLÍCITOS ANIDADOS */
DATA PRODUCTO(KEEP=FECHA TASA CANTIDAD RETURN);
INPUT FECHA MMDDYY8. T1 -T3 C1-C5;
FORMAT FECHA MMDDYY8.;
ARRAY TASAS T1-T3;
ARRAY INVERS(K) C1-C5;
DO OVER TASAS;
DO OVER INVERS;
TASA=TASAS;
CANTIDAD=INVERS;
RETURN=TASAS*INVERS;
OUTPUT;
END;
END;
CARDS;
11/12/86 .065 .070 .075 1000 5000 10000 20000 25000
11/19/86 .061 .068 .072  500 7500 15000 30000 50000
;
PROC PRINT NOOBS;
FORMAT CANTIDAD RETURN DOLLAR10.2;
RUN;
```

Entre los errores más comunes que pueden ocurrir utilizando arreglos, tanto explícitos como implícitos, están: definir el nombre de un arreglo idéntico al de una función SAS, asignar un valor a un subíndice mayor que el número de elementos del arreglo, o definir un arreglo en un paso de datos e intentar referirse a él en otro paso de datos.

3.2.8 Uso de funciones SAS

Existen más de 120 funciones en SAS, clasificadas en: matemáticas y estadísticas, para manipulación de cadenas de caracteres, para conversión de datos numéricos y caracteres, y para manipular datos de fechas.
Una función es una rutina que devuelve un valor determinado a partir de uno o más argumentos. La forma general de una función SAS es:

FUNCTIONNAME (argument, argument,...,argument)

Los argumentos de una función debe estar separados por comas entre sí. Muchas funciones aceptan como argumentos: constantes, variables, expresiones y funciones. Algunas funciones toman muchos argumentos en cualquier orden. Otras tienen un número específico de argumentos y en un orden fijo. En las

funciones que aceptan un número variable de argumentos, se puede utilizar la palabra OF seguida de la lista de argumentos así:

FUNCTIONNAME (OF n1-n50)

Otra forma puede ser: TOTAL=SUM(OF VENTAS—GASTOS);

Se suman los valores desde la variable VENTAS consecutivamente hasta la variable GASTOS.

A continuación se muestran ejemplos de algunas funciones SAS.

*3.2.8.1 Función **SUM:***

Esta función toma un número variable de argumentos y en cualquier orden, su forma es:

SUM (ARGUMENT, ARGUMENT,...**)**.

El Ejemplo 3.2.7.3 muestra el uso de esta función.

*3.2.8.2 Función **ROUND:***

La función ROUND redondea y tiene un número específico de argumentos y en un orden fijo, es decir:

ROUND (numero [, unidredo]**)**

el segundo argumento (uniredo) de la función es opcional, es la unidad de redondeo, si se omite se asume 1 por defecto. Matemáticamente es expresada así:

*ROUND(X,d)=INT(X/d+0.5)*d*

Ejemplo 3.2.8.2
Dada el área de parcelas cuadradas en acres, calcular la longitud de un lado usando: la precisión por defecto, la función ROUND para redondear al entero más cercano y redondeando con .5.

```
/*EJEMPLO 3.2.8.2*/
DATA LADOS;
INPUT AREA;
DEFECTO=SQRT (AREA*43560); /*SQRT ES LA RAIZ CUADRADA*/
ENTERO= ROUND (SQRT (AREA*43560));
MITAD=ROUND (SQRT (AREA*43560), 0.5);
CARDS;
1
50
100
;
```

PROC PRINT; FORMAT DEFECTO ENTERO MITAD 9.4;
RUN;

En el ejemplo anterior ROUND (SQRT (43560),0.5) se puede calcular:

ROUND(SQRT(43560),0.5)=int(208.7103/0.5+0.5)*.5=208.5

3.2.8.3 Manipulación de cadenas de caracteres

En muchas aplicaciones se puede extraer porciones de cadenas de caracteres, concatenar o modificar cadenas de caracteres. Las siguientes funciones son útiles para realizar esas tareas:

LENGTH determina la longitud de una cadena de caracteres

SUBSTR extrae una sub_cadena

SCAN busca por palabras

TRIM borra los blancos intercalados de una cadena

INDEX busca una cadena que funciona como clave

El sistema SAS también posee el operador (||) para concatenar variables y constantes de caracteres.

La función **LENGTH** devuelve la longitud de una expresión carácter especificada como argumento, su forma general es:

LENGTH (argument**)**

argument es cualquier constante, variable o expresión tipo carácter.

La función **SUBSTR** es utilizada para extraer de una cadena una subcadena de caracteres. Si la variable que recibe la subcadena no tiene predefinida la longitud, ésta tendrá la misma longitud que el argumento. Su forma general es:

SUBSTR (argument, inicio[,num]**)**

argument es una constante, variable o expresión carácter.

inicio es la posición inicial.

num es el número de caracteres a leer. Si es omitido la subcadena contiene el resto del argumento.

Ejemplo 3.2.8.3
Extraiga el último carácter de la variable ID.

```
/* EJEMPLO 3.2.8.3 */
DATA PRUEBA;
INPUT ID $5.;
TRY1=SUBSTR (ID,4,1);
TRY2=SUBSTR (ID,4);
TRY3=SUBSTR (ID,5);
TRY4=SUBSTR (ID, LENGTH (ID));
CARDS;
1021
10032
1042
10041
;
PROC PRINT;
RUN;
```

Ejemplo 3.2.8.4
Suponga en el anterior problema que la variable ID se refiere al código de identificación de los pacientes. Se sabe que el último dígito determina si el paciente es un Sr. o una Sra., es decir, si es 1 es una Sra. y si es 2 es un Sr.

```
/* EJEMPLO 3.2.8.4 */
DATA PRUEBA;
INPUT ID $5.;
/* DETERMINE EL TITULO DE CORTESIA */
IF SUBSTR (ID, LENGTH(ID))='1' THEN TITULO='Sra.';
ELSE TITULO='Sr.';
CARDS;
1021
10032
1042
10041
;
PROC PRINT;
RUN;
```

La función **SCAN** devuelve la n-ésima palabras de una cadena de caracteres. Es utilizada para extraer palabras de una cadena cuando el orden relativo de las palabras es conocido, pero no se sabe donde comienzan. La forma general es:

SCAN(argument,n[,delimitador]**)**
argument es una constante, variable o expresión carácter.
n especifica la n_ésima palabra a extraer.
delimitador define caracteres que delimitan o separan las palabras

Si ningún delimitador es especificado, sirven como separadores por defecto los siguientes caracteres:

blanco . < (+ | ! $ *) ; - / , %

Cualquier carácter o conjunto de caracteres puede servir como delimitador. Los delimitadores antes de la primera palabra no tienen efecto, es decir, no lo toma en cuenta. Si existen dos o más delimitadores contiguos son tratados como uno solo. Si existen menos palabras que n en la cadena, la función devuelve un valor faltante.

Ejemplo 3.2.8.5
Este ejemplo demuestra el uso de la función SCAN.

```
/*EJEMPLO 3.2.8.5 */
DATA BUSQUEDA;
LENGTH VAR1-VAR5 $6.; /* FIJA LA LONGITUD DE LA CADENA */
TEXT='VIERNES NOVIEMBRE 6, 1992)';
VAR1=SCAN(TEXT,1);
VAR2=SCAN(TEXT,4);
VAR3=SCAN(TEXT,5);
VAR4=SCAN(TEXT,2,',');
VAR5=SCAN(TEXT,2,',)');
PROC PRINT NOOBS;
VAR VAR1-VAR5;
PROC CONTENTS POSITION DATA=BUSQUEDA;
RUN;
```

En minicomputadores y PC´s la longitud de la variable que recibe el resultado de la función SCAN es 200 por defecto, a menos que previamente sea pre-definida. Observen el resultado del procedimiento PROC CONTENTS.

A continuación se muestra otro ejemplo sobre extracción de palabras de una cadena de caracteres.

Ejemplo 3.2.8.6
Este ejemplo determina el titulo y divide el nombre en nombre y apellido.

```
/* EJEMPLO 3.2.8.6 */
DATA PRUEBA;
INPUT ID : $5. NOMBRE :$35.;
/* DETERMINA EL TITULO DE CORTESIA */
```

```
IF SUBSTR(ID,LENGTH(ID))='1' THEN TITULO='Sra.';
ELSE IF SUBSTR(ID,LENGTH(ID))='2' THEN TITULO='Sr.';
/* DEFINE LA LONGITUD DE LAS SUBCADENAS DE CARACTERES */
LENGTH PRIMERNO $15. APELLIDO $20.;
/* DIVIDE EL NOMBRE */
PRIMERNO=SCAN(NOMBRE,2,',');
APELLIDO=SCAN(NOMBRE,1,',');
CARDS;
1021 Allison,Mary G.
10032 Best,John R.
1042 Gains,Elmer
10041 Giles,Sandra P.
;
PROC PRINT;
RUN;
```

Para concatenar cadenas de caracteres se utiliza el operador ||. La forma general es:
string1||string2

string1 y string2 pueden ser constantes de caracteres, variables o expresiones de caracteres.

La función **TRIM** extrae los blancos de una cadena de caracteres. Usualmente es utilizada antes de concatenar cadenas de caracteres con la finalidad de remover blancos extraños.

Ejemplo 3.2.8.7
Aquí se demuestra el uso del operador **||** y la función **TRIM**. Se forma el nombre de los pacientes como primer nombre y apellido y se le agrega el titulo Sr. o Sra.

```
/* EJEMPLO 3.2.8.7 */
DATA PRUEBA(KEEP= ID PACIENTE);
INPUT ID : $5. NOMBRE :$35.;
/* DETERMINA EL TITULO DE CORTESIA */
IF SUBSTR(ID,LENGTH(ID))='1' THEN TITULO='Sra.';
ELSE IF SUBSTR(ID,LENGTH(ID))='2' THEN TITULO='Sr.';
/* DEFINE LA LONGITUD DE LAS SUBCADENAS DE CARACTERES */
LENGTH PRIMERNO $15. APELLIDO $20.;
/* DIVIDE EL NOMBRE */
PRIMERNO=SCAN(NOMBRE,2,',');
APELLIDO=SCAN(NOMBRE,1,',');
/*FORMA EL NOMBRE DEL PACIENTE */
PACIENTE=TITULO||' '||TRIM(PRIMERNO)||' '||APELLIDO;
CARDS;
1021 Allison,Mary G.
```

```
10032 Best,John R.
1042 Gains,Elmer
10041 Giles,Sandra P.
;
PROC PRINT;
RUN;
```

La función **INDEX** busca en una variable o constante tipo carácter la posición de una cadena de caracteres específica y devuelve la posición. La forma general es:

INDEX(argument1,argument2**)**

argument1 es cualquier constante, variable o expresión de caracteres, donde se realiza la búsqueda.

argument2 especifica cualquier string o valor de una variable carácter a ser localizada en el argument1.

Si argument2 es localizado, la función INDEX devuelve la posición inicial de la primera ocurrencia de argument2. Si no es encontrado la función devuelve un cero (0).

Ejemplo 3.2.8.8

```
/* EJEMPLO 3.2.8.8 */
DATA INDEX;
TEXT='DELIMIT IT WITH BLANKS WHEN IN DOUBT';
/*BUSQUEDA DE UNA CONSTANTE */
POS1=INDEX(TEXT,'IT');
POS2=INDEX(TEXT,' IT ');
POS3=INDEX(TEXT,'it');
/*BUSQUEDA DE UN VALOR DE UNA VARIABLE */
LENGTH STRING $5.;
STRING='IT';
POS4=INDEX(TEXT,STRING);
POS5=INDEX(TEXT,TRIM(STRING));
POS6=INDEX(TEXT,' '||TRIM(STRING)||' ');
PROC PRINT;
VAR POS1-POS6;
RUN;
```

3.2.8.4 Uso de funciones para la conversión de datos

En general el resultado que devuelve una función SAS es carácter si sus argumentos son carácter, o numérico si sus argumentos son numéricos. En muchas aplicaciones se puede desear convertir un dato de un tipo en otro tipo. Algunas veces se desea transformar los dígitos de una variable carácter en un valor numérico. Otras veces, se puede desear convertir los valores numéricos en un valor carácter. Esta conversión se puede realizar de dos maneras:

- implícitamente el SAS obliga a hacerlo, debido a que permite que una expresión matemática contenga algún elemento tipo carácter, es decir que convierte automáticamente los dígitos carácter en numéricos.

- explícitamente con las funciones **INPUT** y **PUT**

A continuación se muestra un ejemplo sobre la conversión automática de carácter a numérico:

Ejemplo 3.2.8.4.1
El siguiente programa calcula el bono sobre el 10% del ingreso bruto de cada empleado.

```
/*EJEMPLO 8.41 */
DATA BONOS;
INPUT SSN  $11. INGRESO : 6$.;
BONO=.10*INGRESO;
CARDS;
201-92-2498 32000
482-87-7945 12000
330-40-7100 39000
203-37-4782 25000
;
PROC PRINT;
RUN;
```

Cuando ejecute este programa observe la ventana LOG. Qué muestra? El SAS convierte automáticamente un valor carácter en uno numérico, cuando el valor carácter es utilizado en un contexto numérico, como en uno de los siguientes casos:

. para asignar un valor a una variable numérica

. en la comparación lógica con un valor numérico

. en operaciones aritméticas

. en funciones que requieren argumentos numéricos

La función **INPUT** convierte los dígitos carácter en valores numéricos. Su forma es:

INPUT(argument,informat**)**

argument es una constante, variable o expresión tipo carácter.

informat es un formato de lectura SAS.

Si la función **INPUT** es utilizada para crear una variable que no ha sido definida previamente, entonces el tipo y la longitud de la variable es definida por el formato de lectura (informat).

Ejemplo 3.2.8.4.2
En el ejemplo anterior utilice la función INPUT para convertir el valor carácter en numérico.

```
/*EJEMPLO 3.2.8.4.2 */
DATA BONOS;
INPUT SSN :$11. INGRESO : 6$.;
BONO=.10*INPUT(INGRESO,6.);
CARDS;
201-92-2498 32000
482-87-7945 12000
330-40-7100 39000
203-37-4782 25000
;
PROC PRINT;
RUN;
```

El SAS permite también la conversión automáticamente de valores numéricos a carácter.

Ejemplo 3.2.8.4.3
Utilice la conversión automática de numérico a carácter.

```
/*EJEMPLO 3.2.8.4.3 */
DATA TELEFONO;
INPUT CODIGO NUMERO $;
TLF=CODIGO||'/'||NUMERO;
CARDS;
303 393-0956
919 770-8292
301 449-5239
;
PROC PRINT;
RUN;
```

Qué pasará cuando una variable numérica, como CODIGO es utilizada en una expresión carácter?

El SAS convierte automáticamente un valor numérico en uno carácter, cuando el valor numérico es utilizado en un contexto carácter, como:

. en la asignación a una variable carácter

. en operaciones de concatenación

. en funciones que requieran argumentos tipo carácter

La función **PUT** convierte un valor numérico en uno carácter. Su forma general es:
PUT(argument,format**)**

argument es una constante, variable o expresión

format es un formato SAS predefinido o definido por el usuario.

Esta función devuelve siempre una cadena de caracteres.

Ejemplo 3.2.8.4.4
En el ejercicio anterior utilizar la función PUT para convertir el valor numérico en carácter.

```
/*EJEMPLO 3.2.8.4.4 */
DATA TELEFONO;
INPUT CODIGO NUMERO $;
TLF=PUT(CODIGO,3.)||'/'||NUMERO;
CARDS;
303 393-0956
919 770-8292
301 449-5239
;
PROC PRINT;
RUN;
```

3.2.8.5 Funciones estadísticas

Existen múltiples funciones estadísticas clasificadas en: funciones de números aleatorios, funciones de estadísticos muestrales, funciones de probabilidad y funciones de cuantiles.

Entre las funciones de números aleatorios están:
NORMAL(semilla) genera una variable pseudo aleatoria normal(0,1).

RANBIN(semilla,n,p) genera una observación de una distribución binomial, con parámetros n y p.

RANUNI(semilla) genera un número aleatorio uniforme.

RANNOR(semilla) genera un número aleatorio normal.

RANPOI(semilla,lambda) genera una observación de una distribución Poisson con parámetro Lambda.

RANEXP(semilla) genera una exponencial.

RANGAM(semilla,alfa) genera una distribución Gamma con alfa mayor que cero.

UNIFORM(semilla) genera una variable pseudo aleatoria distribuida en el intervalo (0,1).

Se pueden generar números aleatorios para varias distribuciones usando las funciones de números aleatorios. Estas funciones utilizan un argumento para seleccionar el valor inicial de la semilla y dependiendo de éste valor la inicialización puede ser:

a. Menor o igual a cero (<=0)

Si asigna cero (0), en la primera ejecución de la función dicho argumento inicializa la semilla con un valor igual al reloj del computador. Para las siguientes ejecuciones la función devuelve una observación generada con la semilla actual.

b. Mayor que cero (>0)

Durante la primera ejecución de la función el argumento es utilizado para inicializar la semilla y devolver una observación. En las siguientes ejecuciones, es devuelta una observación generada con la semilla actual.

Ejemplo 3.2.8.5.1

```
/*Este ejemplo genera números aleatorios de acuerdo a la distribución uniforme*/
DATA A;
RETAIN SEMILLA1 SEMILLA2 1613218064;
DO I=1 TO 5;
X1=RANUNI(SEMILLA1);
X2=RANUNI(SEMILLA2);
OUTPUT;
END;
PROC PRINT;
TITLE 'GENERA NUMEROS ALEATORIOS UTILIZANDO UNA FUNCION DE UNA DISTRIBUCION UNIFORME';
RUN;
```

Aunque la semilla actual cambie cada vez que es ejecutada la función, el valor del argumento permanece inalterado. No se pueden controlar los valores de las semillas después de la inicialización. Si se desea controlarla se debe utilizar la instrucción CALL que permite llamar a subrutinas sobre números aleatorios.

Los números aleatorios generados a través de subrutinas se invocan con la instrucción CALL; la forma general de sintáxis es:

CALL subrutina(semilla,variable);
donde:

subrutina es el nombre de cualquier función SAS (excepto NORMAL o UNIFORM).

semilla es el nombre de una variable que mantiene los valores de la semilla, es la semilla del siguiente número aleatorio. Esta debe ser inicializada antes de realizarse la primera llamada a la rutina, puede ser inicializada con la instrucción RETAIN.

variable se refiere al nombre de una variable que contiene el número aleatorio generado.

Ejemplo 3.2.8.5.2

```
DATA;
RETAIN SEMILLA3 SEMILLA4 1613218064;
DO I=1 TO 5;
CALL RANUNI(SEMILLA3,X3);
CALL RANUNI(SEMILLA4,X4);
OUTPUT;
END;
PROC PRINT;
TITLE 'GENERA NUMEROS ALEATORIOS UTILIZANDO LA LLAMADA A UNA RUTINA EN VEZ DE UNA FUNCION';
RUN;
```

La utilización de las rutinas, en vez de las funciones, para generar números aleatorios, le permite inicializar la semilla a más de un número aleatorio, es decir que la semilla se puede iniciar cada vez. Mientras que en las funciones mas de un conjunto de números aleatorios puede ser creado, pero todos ellos provienen de una sola fuente (semilla) que es la dada en la inicialización (Ver Ejemplos 3.2.8.5.1 y 3.2.8.5.2).

Observe que aunque los valores iniciales de semilla1 y semilla2 son iguales los números generados y depositados en x1 y x2 son diferentes. Esto se debe a que el programa inicializa la semilla una sola vez, es decir ignora el valor asignado para la semilla2.

Ahora observe el resultado del Ejemplo 8.5.2, allí los valores de X3 y X4 son idénticos para cada observación, porque la instrucción CALL que permite varias semillas, fue inicializada con el mismo valor. Si le asigna a semilla3 y semilla4 diferentes valores se pueden obtener números aleatorios que provengan de fuentes independientes. Note que con las rutinas es posible ver el valor actual de la semilla, mientras que esto no es posible con las funciones.

Entre las funciones de estadísticos muestrales están:

CSS calcula la suma de cuadrados corregida

CV calcula el coeficiente de variación

KURTOSIS calcula la curtosis

MAX devuelve el máximo valor

MIN devuelve el mínimo valor

MEAN calcula el promedio aritmético

RANGE calcula el recorrido o rango

SKEWENESS calcula el coeficiente de asimetría

VAR calcula la varianza

STD calcula la desviación estándar

Cada una de estas funciones requiere varios argumentos, al menos uno, que deben ser valores numéricos. Su forma general es:

nombrefunc(num1,num2,...,numn)

Ejemplo 3.2.8.5.3

A continuación se muestra un ejemplo en el que se utilizan estas funciones:

```
/*EJEMPLO 3.2.8.5.3 */
DATA MEDIAS;
MEDIAED=MEAN(12,14,10,19,17);
VARIANZA=CSS(12,14,10,19,17)/4; STD=SQRT(VARIANZA);
MINIMO=MIN(12,14,10,19,17);
MAXIMO=MAX(12,14,10,19,17);
PROC PRINT;
RUN;
```

Entre las funciones para el cálculo de probabilidades, se encuentran:

POISSON(lambda,n)	devuelve la probabilidad de que una observación sea menor o igual a n.
PROBBNML(p,n,m)	devuelve la probabilidad que sea menor o igual a m, con parámetros p y n.

PROBCHI(x,df,nc)	calcula la probabilidad de una variable aleatoria Chi-Cuadrado con df grados de libertad, que sea < x. El tercer argumento es opcional, si no es especificado o es cero calcula la Chi-Cuadrado Central. De lo contrario, calcula el valor de no centralidad.
PROBF(x,ndf,ddf,nc)	calcula la probabilidad de una variable con distribución F, con ndf y ddf grados de libertad en el numerador y denominador. El argumento nc es opcional, significa parámetro de no centralidad. Para encontrar el nivel de significación realice p=1-probf(x,ndf,ddf).
PROBHYPR(nn,k,n,x,or)	calcula la probabilidad <=x de una observación de una distribución Hipergeométrica, para una muestra total nn con un total k éxitos y n fracasos.
PROBIT(argument)	esta función es la inversa de la función de distribución Normal, es decir calcula el valor crítico Z. El valor de **argument** debe ser 0 ó 1.
PROBNORM(x)	calcula la probabilidad de una v.a Normal(0,1).
PROBT(x,df,,nc)	calcula la probabilidad de una variable aleatoria T-Student con df grados de libertad, cuya area sea menor que x. El tercer argumento es opcional, es el valor de no centralidad.

A continuación se muestra un ejemplo sobre el cálculo de estas probabilidades.

Ejemplo 3.2.8.5.4
Cálculo de probabilidades.

```
/*EJEMPLO 8.5.4 */
DATA;
file print; *Abre la ventana del OUTPUT como un archivo;
title 'CALCULO DE PROBABILIDADES';
p=poisson(1,2); put p;
p=probbnml(.5,10,4);  put p;
p=1-probchi(31.264,11); put p;
p=1-probf(3.32,2,30);  put p;
x=probhypr(10,5,3,2); put x;
x=probit(0.025);  put x;
x=probnorm(1.96); put x;
x=probt(.9,5); put x;
RUN;
```

El contenido de la ejecución de un programa y en tiempo de ejecución, se puede enviar a la a un archivo. Para enviar a un archivo ASCII, escriba:

file 'C:\Users\Usuario\Documents\My SAS Files\archivo';

Esta última instrucción indica que los resultados de la ejecución de un programa SAS, son escritos en un archivo, situado en su directorio actual de **My SAS Files** y denominado **archivo**. Este es un documento tipo texto o ASCII, el cual se puede leer con el bloc de notas de Windows.

3.2.9 Union de Archivos (MERGE)

La unión de varios archivos en uno solo se puede realizar de varias maneras (Ver Figura 8):

- leyendo dos o más archivos con la instrucción INFILE en un solo paso de datos.
- utilizando la instrucción SET para leer archivos de datos SAS.
- utilizando la instrucción MERGE

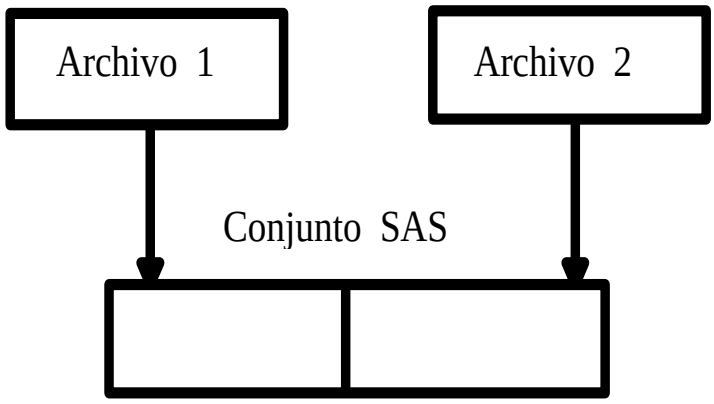

Figura 8.

*3.2.9.1 Unión con la instrucción **INFILE***

En la primera forma se supone que los archivos a unir contienen el mismo número de observaciones y al menos una variable en común.

Ejemplo 3.2.9.1
Crear un conjunto de datos SAS a través de la unión de dos archivos externos que contienen las calificaciones de un grupo de estudiantes, la cual está divida en dos partes, una se refiere a las notas de Matemáticas y la otra a las notas de Inglés. Se pide además calcular el promedio para cada estudiante e imprimir el conjunto de datos resultante.

```
/* EJEMPLO 3.2.9.1 UNION DE ARCHIVOS (INFILE) */
DATA NOTAS (DROP=TOTAL);
*Nota: Cambiar la ruta de almacenamiento del archivo de datos;
INFILE 'MATEMAT';
INPUT ESTUDIAN $14. @16 MAT1 2. @19 MAT2 2.;
INFILE 'FISICA;
INPUT @16 FISI1 3. @19 FISI2 2.;
PROM=MEAN (MAT1,MAT2,FISI1,FISI2);
PROC PRINT NOOBS;
TITLE 'PROMEDIO DE NOTAS DE LOS CURSOS MATEMATICAS Y FISICA;
RUN;
```

Se pueden unir archivos que contengan diferente número de registros, pero la unión contendrá el mismo número de registros que el archivo más corto, ya que el paso de datos finaliza cuando es detectada la marca de fin de archivo. El siguiente ejemplo muestra esta situación.

Ejemplo 3.2.9.2
Crear un conjunto de datos SAS uniendo dos archivos externos, uno contiene las ventas mensuales para el año 1986 y el otro contiene las ventas de los primeros cinco meses de 1987.

```
/* EJEMPLO 3.2.9.2 UNION DE DOS ARCHIVOS */
DATA VENTAS;
*Nota: Cambiar la ruta de almacenamiento del archivo de datos;
INFILE 'Y1986';
INPUT MES 2. @5 VENTA1 5.;
INFILE 'Y1987';
INPUT @5 VENTA2;
PROC PRINT;
RUN;
```

Sin embargo, es posible utilizar la opción END de la instrucción INFILE para evitar que el paso de datos finalice cuando detecte la marca de fin de archivo en un archivo. Ejecute el siguiente ejemplo, el cual es una modificación del anterior y observe la ventana LOG. Por qué el SAS LOG indica que el conjunto SAS contiene 13 observaciones? Porque la condición se cumple antes de la asignación.

Ejemplo 3.2.9.3

Repita el ejemplo anterior y utilice la opción END de la instrucción INFILE para evitar la terminación prematura del paso de datos.

```
/* EJEMPLO 3.2.9.3: Repita el ejemplo anterior y utilice la opci¢n
   END de la instrucci¢n INFILE para evitar la terminaci¢n
    prematura del paso de datos */
DATA VENTAS;
  IF E1=0 THEN
     DO;
       INFILE 'C:\Users\Usuario\Documents\Computacion
Estadisitica14\ejemsas14\Y1986' END=E1;
       INPUT MES 2. @5 VENTA1 5.;
     END;
  IF E2=0 THEN
     DO;
       INFILE 'C:\Users\Usuario\Documents\Computacion
Estadisitica14\ejemsas14\Y1987' END=E2;
       INPUT @5 VENTA2 5.;
     END;
PROC PRINT;
RUN;
```

A continuación se presenta otra modificación de este ejemplo donde se indica que el paso de datos finaliza cuando se ha procesado la última observación.

Ejemplo 3.2.9.4

Repita el ejemplo anterior y utilice la opción END de la instrucción INFILE para evitar la terminación prematura del paso de datos. Detenga el paso de datos cuando la última observación sea procesada.

```
/* EJEMPLO 3.2.9.4: Repita el ejemplo anterior y utilice la opción
   END de la instrucción INFILE para evitar la terminación
    prematura del paso de datos. Detenga el paso de datos cuando la
   última observación sea procesada */
DATA VENTAS;
  IF E1=1 AND E2=1 THEN STOP;
  IF E1=0 THEN
     DO;
       INFILE 'C:\Users\Usuario\Documents\Computacion
Estadisitica14\ejemsas14\Y1986' END=E1;
       INPUT MES 2. @3 VENTA1 7.;
     END;
  IF E2=0 THEN
     DO;
       INFILE 'C:\Users\Usuario\Documents\Computacion
Estadisitica14\ejemsas14\Y1987' END=E2;
       INPUT @3 VENTA2 7.;
     END;
  TOTAL=VENTA1+VENTA2;
RUN;
PROC PLOT DATA=VENTAS NOLEGEND;
 PLOT VENTA1*MES='1' VENTA2*MES='2' TOTAL*MES='T' / OVERLAY
```

```
   VAXIS=15000 TO 50000 BY 3000
   HAXIS=1 TO 12 BY 1;
 LABEL MES='MES DEL AÑO'
       VENTA1='TOTAL VENTAS';
 FORMAT VENTA1 VENTA2 DOLLAR7.;
 TITLE 'COMPARACION DE LAS VENTAS EN 1986 Y 1987';
PROC PRINT;
RUN;
```

_

3.2.9.2 Unión de archivos con la instrucción SET

La instrucción SET es una instrucción para lectura de archivos SAS. En tiempo de compilación esta instrucción lee el descriptor de datos (PDV) de cada conjuto SAS y coloca todas las variables en el vector de datos (PDV) del conjunto SAS que se está creando. En tiempo de ejecución lee cada una de las observaciones del conjunto SAS indicado. La forma de sintaxis es:

SET [[SASdataset[(doptions)] ...] [setoptios]];

SASdataset indica el nombre o las nombres de los archivos SAS existentes a ser leidos.
doptions especifica la opciones del conjunto SAS a ser leido.

setoptions especifica las opciones de la instrucción SET.

La instrucción SET sin nombre lee el conjunto SAS creado recientemente.

Ejemplo 3.2.9.2.1
Escriba el siguiente ejemplo, en el que se crean varios archivos SAS en un solo paso de datos y que luego serán leidos con instrucción SET en un ejemplo posterior.

```
*SE CREA EL NOMBRE DEL DIRECTORIO O CARPETA DE ALMACENAMIENTO
DE LOS DATOS;
LIBNAME C C:\Users\Usuario\Documents\Computacion Estadisitica14\ejemsas14';
DATA C.A;
INPUT NUM VAR_A $;
CARDS;
1 A1
2 A2
5 A3
;
PROC PRINT;
DATA C.C;
INPUT NUM VAR $;
```

```
CARDS;
1 C1
2 C2
2 C3
3 C4
;
PROC PRINT;
DATA C.D;
INPUT NUM VAR $;
CARDS;
2 D1
3 D2
3 D3
PROC PRINT;
RUN;
```

En este ejemplo la instrucción LIBNAME se utiliza para declarar la unidad y/o directorio de almacenamiento de los conjuntos de datos (DATA SET SAS). Los archivos A, C y D son creados por el sistema con la extensión .SSD y guardados en el directorio o librería de referencia C o en el directorio de su cuenta en Hydra '/home/faces/usuario'.

Como se recordará los nombres SAS deben contener un máximo de 8 caracteres de longitud, pueden comenzar con cualquier carácter alfabético o de subrayado y los siguientes caracteres pueden ser alfanuméricos. Específicamente, los nombres de archivos tienen dos niveles en el nombre, en el ejemplo DATA C.A significa: C es la unidad de almacenamiento o directorio y A es el nombre del archivo, quiere decir que el archivo llamado A se encuentra almacenado en el directorio de su cuenta '/home/faces/usuario'. Por defecto, el SAS crea y almacena temporalmente los datos en la librería de referencia WORK; así que, si observan la ventana LOG después de la ejecución de un programa, les mostrará que el archivo creado es WORK.nombre. Recuerde también que en una sesión SAS se pueden realizar varios pasos de datos.

Ejemplo 3.2.9.2.2

Ejecute el siguiente ejemplo en el que se utiliza la instrucción SET para leer archivos SAS previamente creados.

```
/*EJEMPLO 3.2.9.2.2 */
*SE CREA EL NOMBRE DEL DIRECTORIO O CARPETA DE ALMACENAMIENTO DE LOS DATOS;
LIBNAME C 'C:\Users\Usuario\Documents\Computacion Estadisitica14\ejemsas14';
DATA B;
SET C.A;
VAR_B=TRIM(VAR_A)||'B';
```

```
PROC PRINT;
RUN;
```

Se pueden leer varios archivos SAS estableciendo un orden de lectura por una variable, utilizando la opción BY.

Ejemplo 3.2.9.2.3

El siguiente ejemplo muestra la unión vertical de dos archivos, es decir, que la unión se realiza desde el punto de vista de las observaciones. Estos archivos contienen las mismas variables.

```
LIBNAME C 'C:\Users\Usuario\Documents\Computacion Estadisitica14\ejemsas14';
DATA COMBINA;
SET C.C C.D;
BY NUM;
PROC PRINT;
RUN;
```

Cuando se ejecuta la instrucción SET C.C C.D el procesador establece precedencia en el orden de lectura, primero lee el archivo C y luego el D.

Cuando los archivos contienen las mismas variables, la unión se realiza verticalmente, es decir se agregan las observaciones del archivo D después las del archivo C.

Cuando es utilizada la instrucción BY con la instrucción SET automáticamente se crean dos variables: FIRST.var y LAST.var, las cuales son temporales y se utilizan para procesar condicionalmente la ordenación o agrupación de los datos. Por ejemplo, suponga que el conjunto SAS Personas es ordenado por (BY) sexo. Las instrucciones:

```
DATA TEMP;
SET PERSONAS;
BY SEXO;
RUN;
```

crean las variables automáticas en el PDV:

FIRST.SEX esta variable es igual a 1 para la primera observación del grupo BY, de otra manera es igual a 0.

LAST.SEX es igual a 1 para la última observación del grupo BY, en caso contrario es 0.

En este caso el vector de datos del programa contiene las siguientes variables automáticas:

_N_	_ERROR_	FIRST.SEX	LAST.SEX
NUM8	NUM8	NUM8	NUM8
DROP	DROP	DROP	DROP

Se puede observar que estas variables son (todas) de tipo numérico, de longitud 8 y DROP quiere decir que son excluidas del archivo.

Ejemplo 3.2.9.2.4
El archivo SAS llamado C.VENTAS contiene las ventas del año 1984 para varias regiones del país. Este archivo está almacenado en el directorio c:\sas. Se pide:

1. Ordene dicho archivo por REGION

2. Calcule las ventas totales para cada región.

```
/*EJEMPLO 3.2.9.2.4 */
LIBNAME C 'C:\Users\Usuario\Documents\Computacion Estadisitica14\ejemsas14';
PROC SORT DATA=C.VENTAS OUT=C.VENTAS;
BY REGION;
DATA TOTALES(KEEP=REGION TOTAL);
SET C.VENTAS;
BY REGION;  /*by descending region; */
/* SE INICIALIZA EL TOTAL PARA CADA REGION */
IF FIRST.REGION THEN TOTAL=0;
TOTAL+VENTA;
/* SI ES LA ULTIMA OBSERVACION DE LA REGION SE ALMACENA */
IF LAST.REGION;
PROC PRINT DATA=TOTALES NOOBS;
TITLE ' TOTAL DE VENTAS PARA 1984';
RUN;
```

Para calcular las ventas por región se hace necesario ordenar primero el archivo de acuerdo a la variable REGION. Observe que el procedimiento SORT tiene dos opciones: DATA se refiere al conjunto de datos a ordenar y OUT se refiere al nombre del archivo donde se almacena.

Ejemplo 3.2.9.2.5
En el archivo C.VENTAS del ejemplo anterior, utilizar la lectura con múltiples variables en la instrucción BY.

```
/*EJEMPLO 3.2.9.2.5 */
LIBNAME C 'C:\Users\Usuario\Documents\Computacion Estadisitica14\';
```

```
DATA VENTAS;
SET C.VENTAS;
BY MES REGION NOTSORTED;
FMES=FIRST.MES;
LMES=LAST.MES;
FREGION=FIRST.REGION;
LREGION=LAST.REGION;
PROC PRINT DATA=VENTAS;
VAR VENTA MES FMES LMES REGION FREGION LREGION;
TITLE ‘ VENTAS PARA 1984’;
RUN;
```

En este ejemplo se demuestra el uso de las variables FIRST y LAST cuando dos variables son llamadas por la instrucción BY.

La opción NOTSORTED indica que el archivo no está ordenado por la variable indicada por BY.

Se pueden utilizar múltiples instrucciones SET para unir (MERGE) conjuntos SAS que tienen una variable en común y un número desigual de observaciones (registros). Este caso la variable común se solapa y contiene solo el número de observaciones comunes. Cada instrucción SET tiene un apuntador (pointer) independiente para los conjuntos SAS listados.

Ejemplo 3.2.9.2.5.1
Este ejemplo ilustra el uso de múltiples instrucciones **SET**.

```
/*EJEMPLO 3.2.9.2.5.1 ILUSTRA EL USO DE MULTIPLES INST.  SET */
LIBNAME C ‘C:\Users\Usuario\Documents\Computacion Estadisitica14\’;
DATA COMBINA;
SET C.A;
SET C.C;
PROC PRINT;
RUN;
```

El archivo resultante contendrá todas las variables leídas de todos los conjuntos de datos leídos con la instrucción **SET**. El número de observaciones depende de cuándo detecta el fin de archivo.

La instrucción **SET** normalmente realiza la lectura de las observaciones en forma secuencial. Sin embargo, se puede realizar la lectura en forma indexada (modo de lectura de acceso directo). La forma general para esta lectura es:

SET archivoSAS **POINT=**apuntador;

donde apuntador puede ser:

. una variable de tipo numérica que contiene el número de la observación a ser leída.
. debe ser un valor dado antes de la ejecución de la instrucción SET.

Es importante hacer notar que, en el modo de lectura de acceso directo el fin de archivo no es detectado. Por lo tanto, se debe programar de una manera lógica para asegurar que el paso de datos finalice, para ello se utiliza la instrucción STOP.

La opción **POINT** de la instrucción **SET** a menudo se utiliza con la opción NOBS. Su forma de sintaxis es:

SET archivoSAS **POINT=**apuntador **NOBS=**variable;

donde variable:

. es una variable numérica especial a la que se le asigna un valor durante la compilación.

. contiene el número total de observaciones del archivo SAS que es llamado en la instrucción SET.

Si múltiples archivos aparecen en la instrucción SET, en la variable de la opción NOBS se coloca el número total de observaciones de todos los archivos SAS leídos.

Ejemplo 3.2.9.2.6
El siguiente ejemplo muestra el uso de las opciones POINT y NOBS de la instrucción SET. Aquí se genera una muestra aleatoria con reposición de tamaño fijo (5) a partir de un conjunto SAS llamado C.CLASE. Este conjunto de datos fue creado en el ejemplo 1 y contiene información sobre las variables NOMBRE, EDAD SEXO, PESO Y ESTATURA. En esta muestra aleatoria las observaciones se pueden duplicar.

```
LIBNAME C ‘C:\Users\Usuario\Documents\Computacion Estadisitica14\’;
DATA MUESTRA;
TAM=5;  /* TAMAÑO DE LA MUESTRA */
DO I=1 TO TAM;
X=CEIL(RANUNI(0)*TOTOBS);
SET C.CLASE POINT=X NOBS=TOTOBS;
OUTPUT;
END;
STOP; /* EN ACCESO DIRECTO  EL FIN DE ARCHIVO NO ES DETECTADO */
PROC PRINT;
RUN;
```

La función **RANUNI** devuelve un valor entre 0 y 1 generado de una Distribución Uniforme y la función **CEIL** redondea al entero inmediato superior. Si desea generar la misma muestra aleatoria otra vez utilice una semilla diferente de cero.

Ejemplo 3.2.9.2.7
Crear una muestra aleatoria sin reposición de tamaño fijo (5) a partir del conjunto C.CLASE.

```
/*EJEMPLO 3.2.9.2.7 */
LIBNAME C 'C:\SAS\';
DATA NUMEROA(KEEP=I);
TAM=5;  /* TAMAÑO DE LA MUESTRA */
DO WHILE(TAM>0);
I+1;
IF UNIFORM(0)<TAM/TOTOBS THEN
DO;
        OUTPUT;
        TAM=TAM-1;
END;
TOTOBS=TOTOBS-1;
END;
STOP; /* EN ACCESO DIRECTO  EL FIN DE ARCHIVO NO ES DETECTADO */
/* EN TIEMPO DE COMPILACION SE ASIGNA EL NUMERO DE OBSERVACIONES
A LA VARIABLE TOTOBS */
SET C.CLASE POINT=_N_ NOBS=TOTOBS;
DATA MUESTRA;
SET NUMEROA;
SET C.CLASE POINT=I;
PROC PRINT;
RUN;
```

El conjunto (data set) NUMEROA contiene los 5 números aleatorios enteros entre 1 y el número total de observaciones seleccionados del conjunto C.CLASE. Este conjunto determina cuales observaciones deben ser leídas del archivo C.CLASE.

Esta técnica representa una manera muy eficiente para seleccionar una muestra aleatoria de observaciones, ya que solo n operaciones de lectura son realizadas de un gran conjunto de datos, donde n es el tamaño de la muestra. Auí se realiza muestreo aleatorio simple, es decir, sin reposición.

Cuando se utiliza para unir archivos SAS con la instrucción SET pueden ocurrir problemas como los siguientes:

. las variables que contienen información similar tienen diferentes atributos.

. los nombres de las variables son diferentes.

. las variables tienen diferentes longitudes, formatos (formats, informats), o etiquetas.

. las variables son de diferente tipo.

Cualquiera de estos problemas se puede solucionar. Si desea unir dos archivos que contienen información similar pero tienen nombres de variables diferentes, puede utilizar la opción **RENAME** con la instrucción SET. Un ejemplo sería:

```
DATA MES;
SET Y1986 Y1987(RENAME=(VENTAS=VENTA));
RUN;
```

En el caso que se quieran unir archivos que contienen variables con diferente longitud, formato o etiqueta, puede pasar que, por defecto los atributos de una variable del primer conjunto leído se asignen para el archivo resultante. Si se quiere definir explícitamente los atributos o formatos para las variables del archivo resultante de la unión, entonces se debe utilizar la instrucción FORMAT. Un ejemplo sería:

```
DATA COMBINA;
FORMAT MES $3. QTR 1. VENTA DOLLAR12.;
LENGTH QTR 4;
SET Y1985 Y1986;
RUN;
```

Un ejemplo para unir archivos con variables de diferente tipo, sería:

```
DATA COMBINA;

SET Y84(IN=IN84 RENAME=(QTR=TEM))

Y85;
IF IN84=1 THEN QTR=TEMP;
RUN;
```

Supongamos en este ejemplo que, las variables TEMP y QTR son de diferente tipo, es decir la primera es tipo CHAR y la otra numérica. Así que en la instrucción IF IN84=1 THEN QTR=TEMP; se utiliza la conversión automática.

3.2.9.3 Unión de archivos con la instrucción MERGE

La instrucción **MERGE** lee y combina observaciones de dos o más archivos SAS. En tiempo de compilación, esta instrucción lee la porción descriptora del archivo y todas las variables encontradas son colocadas en el vector de datos (PDV).

En tiempo de ejecución, si no es utilizada la opción BY la instrucción MERGE lee todos los archivos nombrados de izquierda a derecha y combina las observaciones una a una. Mientras que si es utilizada la opción BY, las observaciones son unidas de acuerdo a los valores de las variables en el BY

Ejemplo 3.2.9.3.1
En este programa se crean dos archivos SAS que luego son unidos con la instrucción MERGE.

```
data a;
input num var_a $;
cards;
1 a1
2 a2
3 a3
;
proc print data=a noobs;
data b;
input num var_b $;
cards;
1 b1
2 b2
3 b3
;
proc print data=b noobs;
data combina;
merge a b;
proc print data=combina noobs;
run;
```

Observe que los archivos contienen una variable en común y tienen el mismo número de observaciones. También puede ocurrir que aún cuando tengan una variable en común, tengan un número desigual de observaciones; entonces, el valor de esa variable para el segundo archivo sobre-escribe el valor del primer archivo.
La unión se puede realizar utilizando una variable común con la opción BY de la instrucción MERGE. Las variables nombradas en el BY deben ser del mismo tipo y deben tener los mismos nombres en cada uno de los archivos SAS a unir. Cada variable BY genera las variables automáticas FIRST. y LAST.

Ejemplo 3.2.9.3.2
Unir los archivos c.a y c.b de acuerdo a la variable común num.

```
libname c '/home/faces/usuario';
data combina;
merge c.a c.b;
by num;
proc print data=combina noobs;
run;
```

En la unión de archivos por una variable en común, puede ocurrir que existan en ambos archivos algunos valores en común y algunos no. En este caso, las variables leídas de los conjuntos de datos son inicializadas con faltante (.) (missing) cuando el valor de la variable BY cambia en ambos conjuntos. De otra manera, los valores se mantienen. En algunos casos se utiliza la instrucción MERGE con la finalidad de actualizar la información de un archivo, en función de otro, tal como se realiza a continuación:

Ejemplo 3.2.9.3.3

Se crean dos archivos que contienen información similar y luego se unen con la finalidad de actualizar la información. El archivo PAGO contiene información sobre los salarios de los empleados; mientras que el archivo denominado OTRO contiene los nuevos salarios para los empleados ascendidos. Actualice el archivo PAGO con los nuevos salarios.

```
/*EJEMPLO 3.2.9.3.3  ACTUALIZACION CON MERGE */
DATA PAGO;
INPUT SSN : $9. DEPT : $5. SALARIO 5.;
CARDS;
111223333     ADMIN         25600
222334444     VENTA         24000
333445555     VENTA         31200
444556666     CONT.         27400
555667777     CONT.18000
666778888     ADMIN         15000
777889999     VENTA         21000
888990000     VENTA        26500
;
PROC PRINT;
DATA OTRO;
INPUT SSN :$9. SALARIO 5. DEPT : $5.;
CARDS;
222334444 26400  .
555667777 21000  .
666778888 18000  .
666778888     . CONT.
888990000 29000 .
;
```

```
PROC PRINT;
DATA NUEVOPA;
MERGE PAGO OTRO;
BY SSN;
PROC PRINT DATA=NUEVOPA;
RUN;
```

En vista que la actualización con la instrucción MERGE no es del todo perfecta, el SAS tiene una instrucción específica UPDATE.

3.2.9.4 Instrucción UPDATE

Esta instrucción actualiza un archivo maestro con los datos de otro archivo. La sintaxis general de esta instrucción es:

UPDATE archivomaestro otroarchivo;
BY variable(s);

Con esta instrucción se pueden cambiar valores de variables y añadir observaciones al archivo maestro.
Las restricciones en el uso de esta instrucción son:

- Solo dos nombres de archivos pueden aparecer en la instrucción UPDATE, es decir sólo se puede actualizar un archivo a partir de otro.
- El primer archivo listado es el maestro.
- Ambos archivos deben estar ordenados por la o las variables BY.
- El archivo maestro no debe contener más de una observación para un mismo valor de la variable BY.

Ejemplo 3.2.9.4.1

Utilice los archivos creados en el ejemplo anterior. Actualice el archivo PAGO con la información sobre los nuevos salarios contenida en el archivo OTRO.

```
/*EJEMPLO 3.2.9.4.1  ACTUALIZACION CON UPDATE */
PROC PRINT DATA=PAGO;
PROC PRINT DATA=OTRO;
DATA NUEVO;
UPDATE PAGO OTRO;
BY SSN;
PROC PRINT DATA=NUEVO;
RUN;
```

Observe que, los valores faltantes en el segundo archivo (de transacción), llamado en este caso OTRO no tienen efecto sobre los valores del archivo maestro.

La instrucción **UPDATE** tiene otras opciones como **IN**. Su forma general es:

UPDATE maestro **IN=**variable1 otro **IN=**variable2;

donde:

. **maestro** es el archivo maestro a actualizar.
. **otro** es el archivo que contiene la actualización a realizar.
. **IN=**variable crea una con el nombre indicado en variable.

Una variable diferente en IN=variable debe estar asociada a cada archivo SAS.

La operación UPDATE indica cual de los conjuntos contribuye con datos a la observación actual, es decir que, asigna a la variable IN valores 0 ó 1. Si variable1 es igual a 1 indica que la información de la observación actual proviene del archivo maestro, de otra forma este valor es cero. Variable1 y variable2 serán iguales a 1 si ambos conjuntos contribuyen con información a la actualización.

Ejemplo 3.2.9.4.2

El archivo ORDENES contiene las ordenes recibidas por un fabricante y el archivo INVENTAR contiene el inventario actualizado de las cantidades disponibles de los productos. Estos archivos contienen la siguiente información:

ARCHIVO INVENTARIO

PRODUCT	CUENTA
A1412	2538
A2172	2291
B8126	938
C1279	5281
D1002	172
D1039	309

ARCHIVO DE ÓRDENES RECIBIDAS

FECHA	VENDEDOR	PRODUCT	CANTIDAD
10MAR87	105	D1002	100
12MAR87	101	D1002	100
12MAR87	104	B8126	10
13MAR87	105	A2172	15
13MAR87	105	C1279	35
16MAR7	101	A1412	50
17MAR87	105	D1039	12
17MAR87	101	B8126	5

Se pide:

1. Crear los archivos SAS respectivos.
2. Actualizar el archivo maestro INVENTAR, restando las cantidades ordenadas si no exceden la cantidad disponible del producto.

```
/*EJEMPLO 3.2.9.4.2 USO DE ACTUALIZACION */
DATA INVENTAR;
INPUT PRODUCT CUENTA;
CARDS;
A1412 2538
A2172 2291
B8126  938
C1279 5281
D1002  172
D1039  309
;
DATA ORDENES;
INPUT FECHA $ VENDEDOR $ PRODUCT CANTIDAD;
CARDS;
10MAR87     105    D1002        100
12MAR87     101    D1002        100
12MAR87     104    B8126         10
13MAR87     105    A2172         15
13MAR87     105    C1279         35
16MAR87     101    A1412         50
17MAR87     105    D1039         12
17MAR87     101    B8126          5
;
PROC SORT DATA=ORDENES OUT=ORDENES;
BY PRODUCT FECHA;
DATA NUEVAC(KEEP=PRODUCT CUENTA)
```

```
DESPACHO(KEEP=PRODUCT CUENTA) DEVOLUC;
UPDATE INVENTAR ORDENES(IN=INORDER);
BY PRODUCT;
IF INORDER THEN
IF CUENTA-CANTIDAD<0 THEN OUTPUT DEVOLUC;
ELSE
DO;
OUTPUT DESPACHO;
CUENTA=CUENTA-CANTIDAD;
END;
IF LAST.PRODUCT THEN OUTPUT NUEVAC;
PROC PRINT DATA=DESPACHO;
TITLE 'ORDENES ENVIADAS';
PROC PRINT DATA=DEVOLUC;
TITLE 'ORDENES DEVUELTAS';
PROC PRINT DATA=NUEVAC;
TITLE 'NUEVO INVENTARIO';
RUN;
```

3.2.10 Ingresando Datos por Pantalla

El Sistema SAS permite el ingreso de datos por pantalla. Las instrucciones WINDOW y DISPLAY le permitirán crear sus propias ventanas de entrada de datos, ya sea para ingresar los datos a un conjunto de datos SAS o para exhibir sus datos en la pantalla del computador. También se puede ingresar datos por pantalla utilizando el SAS/ASSIST.

La instrucción WINDOW se utiliza para dar especificaciones de diseño para la creación de una ventana. El siguiente ejemplo muestra un paso de datos y exhibe una ventana con campos para la entrada de un registro de llamadas telefónicas diarias y su duración:

```
/*EJEMPLO 3.2.10.1  ILUSTRA DEFINICION DE VENTANAS */

DATA TELEFONO;
WINDOW DIARIO COLOR=GREEN ROWS=14 COLUMNS=52
#1 @15 'Diario Telefónico'
#3 @1  'Tlf. ' @15 'Nombre'  @45 'Tiempo'
#5 tlfnum $char12. @15 nombre $15. @45 longilla time8.;
display diario;
run;
proc print;
format longilla time8.;
label longilla 'TIEMPO';
run;
```

La instrucción WINDOW nombra a la ventana DIARIO; la opción COLOR= dará el color del fondo de la pantalla; ROWS= será el número de filas; COLUMNS= será el ancho de la ventana en columnas. El símbolo #1 @15 le indicará al SISTEMA SAS que en la fila 1 columna 15 coloque la frase "Diario Telefónico". La tercera fila #3 tiene tres encabezamientos de columnas: Tlf., Nombre y Tiempo. La fila 5 contiene los nombres de tres variables SAS: tlfnum, nombre y longilla cuyos valores se ingresan para producir el registro de llamadas. Esta instrucción tiene similitud con la instrucción PUT.

Cuando la sentencia DISPLAY es ejecutada, presentará la ventana DIARIO. En este momento todas las variables presentarán valores faltantes, por lo tanto, los campos de tlfnum y nombre se encuentran en blanco y el campo de longilla está marcado con un punto, ya que se refiere a un valor numérico faltante del Sistema SAS.

Ingrese los valores del número de teléfono, el nombre de la persona llamada y la duración de la llamada en horas y minutos. Cuando haya llenado cada campo presione ENTER. Al introducir una observación (todos los campos) de inmediato el Sistema SAS archivará esos datos en el conjunto TELEFONO y a la vez, limpiará los campos de manera que Ud. pueda ingresar un nuevo registro de datos (u observación). Para finalizar el ingreso de datos escriba en la línea de comandos (COMMAND ==>) la palabra END o presione la tecla F8, al ejecutarse este comando se eliminará de la pantalla la ventana DIARIO (SAS, Guía Introductoria al SAS, 1989).

3.2.11 Instrucciones Utilizadas en Cualquier Lugar

El SAS tiene un conjunto de instrucciones o comandos globales que pueden ser escritas en un programa sin importar el orden, como por ejemplo:

```
Title  ' xxxxxxx ';
Footnote 'xxxxxxx';
Options ps=55 ls=72  nodate;
```

3.3 PASO DE PROCEDIMIENTOS

La instrucción PROC (o PROCEDURE) es utilizada para invocar un procedimiento SAS. Estos son programas que: leen datos, calculan estadísticas descriptivas, realizan análisis de regresión y correlación, análisis multivariantes, análisis de datos categóricos, imprimen resultados y crean conjuntos de datos, entre otros.

Una vez que Ud. ha creado un conjunto de datos SAS, está listo para utilizar procedimientos que le permitan procesar y analizar el conjunto de datos.

La forma general de esta instrucción o paso de procedimientos es:

PROC program opciones**;**

Hasta el momento hemos utilizado en casi todos los ejemplos presentados en esta guía, el PROC PRINT; este procedimiento lee su conjunto de datos SAS y se los exhibe en pantalla.

Es importante hacer notar que el Sistema SAS permite ejecutar pasos de datos y luego de procedimientos, en forma encadenada. También, permite realizar pasos de procedimientos sobre archivos de datos permanentes o ya creados. Una vez ejecutada la instrucción LIBNAME podrá invocar directamente el archivo de datos en un procedimiento particular; el cual Ud. especificará en la Opción DATA del procedimiento.

A continuación se muestran ejemplos de procedimientos:

```
PROC MEANS;

PROC MEANS SUM MAXDEC=2 DATA=CLASE;
```

En general todos los procedimientos tienen opciones. La opción SUM= indica que se va a calcular el TOTAL para cada variable, MAXDEC= se refiere al número máximo de decimales con que se imprimirán los resultados, DATA= especifica el archivo SAS a ser procesado. Si es omitido, el procedimiento utilizará el archivo SAS creado más recientemente. La opción OUT= especifica el archivo donde se almacenarán los resultados de la ejecución.

Instrucciones frecuentemente utilizadas en los procedimientos:

. Instrucción VARIABLES

Especifica las variables a ser utilizadas en el procedimiento. La forma general es:

```
VARIABLES listavariables;  o
```

VAR listavariables;

. Instrucción BY

Permite el procesamiento de los datos por grupos o subgrupos. La sintáxis general es: BY lista;

. Instrucción TITLE

Define el título a ser escrito en la línea de encabezamiento de la(s) página(s) de salida. Hasta 10 títulos pueden ser especificados. La forma de sintáxis es:

TITLEn 'Titulo'

donde n es el número del título.

. Instrucción FOOTNOTE

Define el texto a ser escrito como notas de pie de páginas, en las salidas. Pueden ser especificados hasta 10 notas de pie de página. Su forma general es:

FOOTNOTEn 'Nota de pie de página';

donde n es el número de la nota de pie de página. Se pueden escribir hasta 10 líneas de pie de página.

A continuación se explican algunos procedimientos disponibles en el Sistema SAS.

. PROC FORMAT

Se utiliza para definir los formatos de lectura (informat) o escritura (formats) para variables carácter o numéricas. Es decir, este procedimiento crea dos tipos de formatos:

. valor (value formats) que convierte o escribe los valores para las salidas en diferente forma, ya sea: numérica a carácter, o carácter a otro carácter diferente. La instrucción VALUE genera las valores (value format) para el formato.

PROC FORMAT;
VALUE SEXO 1='MASCULINO' 2='FEMENINO';

en este ejemplo, la instrucción VALUE define un formato llamado SEXO que convierte una variable numérica a carácter. Los valores de esta variable son

almacenados como números (1 ó 2) pero son formateados como carácter para efectos de escritura de resultados. Si la variable es carácter el nombre del formato debe estar precedido por el carácter $, por ejemplo:

```
PROC FORMAT;
VALUE $French 'OUI'='SI' 'NON'='NO';
```

Una vez creado el formato, para utilizarlo se debe invocar la instrucción FORMAT junto a algún procedimiento. Por ejemplo:

```
PROC PRINT;
FORMAT FRENC FRENCH.;
```

La forma general de la instrucción FORMAT es:

```
FORMAT VAR FMT. CHAR $FMT.;
```

donde:

VAR es una variable (o una lista de variable numéricas)
FMT. es el nombre del formato
CHAR es una variable (lista de variable de carácteres)
$FMT. es el nombre del formato asociado a una variable carácter

Si desea almacenar permanentemente los formatos utilice las siguientes instrucciones:

```
LIBNAME LIBRARY '/home/local/usuario';
PROC FORMAT LIBRARY=LIBRARY;
VALUE ...
```

. PROC MEANS

Este procedimiento lee el conjunto de datos (especificado en la opción DATA= y si no es especificado lee el archivo más recientemente creado),y luego produce estadísticas descriptivas univariantes para variables numéricas. La instrucción BY causará que el procedimiento calcule las estadísticas descriptivas separadamente por grupos de observaciones, definidos por la(s) variable(s) en BY. Opcionalmente, imprime los resultados o los almacena en un archivo SAS.

La forma general es:

PROC MEANS opciones;

Las Opciones pueden ser:

DATA= archivo indica el nombre del archivo a ser analizado.
NOPRINT no imprime los resultados.
MAXDEC=n indica el número de decimales a utilizar en la impresión de los resultados.

Puede especificar las estadísticas que requiera:

N	RANGE	CV	MAX	
NMISS	SUM	CSS	MEAN	
VAR	STDERR	STD	USS	
T	PRT	MIN		

Se pueden especificar las variables a analizar y los grupos, es decir:
BY lista variables;
VAR lista;
OUTPUT OUT=archivo opcion=lista...;

. PROC FREQ

Calcula las frecuencias y produce tablas de frecuencias simples o tablas cruzadas. Realiza las tablas de distribución de valores de variables, combina la frecuencia para dos o mas variables, asigna peso a las frecuencias, calcula las medidas de asociación y pruebas estadísticas para tablas de contigencia de dos variables. Su forma general es:

PROC FREQ DATA=archivo;
Se debe utilizar la instrucción **TABLES** para especificar el tipo de tabla de frecuencia que se desea producir, así:

TABLES variables / opciones;

una sola via	TABLES sexo;
dos vias	TABLES sexo*ingreso;
tres vias	TABLES grupo*sexo*ingreso;
varias tablas	TABLES sexo*(edad ingreso)

esta última instrucción realiza dos tablas cruzadas para: sexo*edad y sexo*ingreso.

Opciones de las tablas (tables):

EXPECTED	NOFREQ	NOCUM	ALL
DEVIATION	NOPERCENT	MISSING	LIST
CELLCHI2	NOROW	NOPRINT	SPARSE

CHISQ NOCOL OUT=archivo

Ejemplo 3.3.1

El siguiente ejemplo muestra el uso de algunos procedimientos como: FREQ, FORMAT, CORR y SORT.

```
/*EJEMPLO 3.3.1. */
PROC FORMAT;
VALUE SEX 1='MASCULINO' 2='FEMENINO';
DATA CLASE ;
INPUT NOMBRE $ 1-30  EDAD 32-33  SEXO 35  PESO 37-42  ESTATURA
44-48;
CARDS;
PEREZ JUAN                          22 1  93.50  1.86
RODRIGUEZ ARMANDO              24 1  69.5   1.67
GIL PABLO                           27 1  83.5   1.84
MACHADO CARMEN                 23 2  58     1.62
BERMUDEZ ALBERTO             30 1  62     1.70
GRANADOS CARMEN                24 2  60     1.64
ALVAREZ ANTONIO                25 1  63     1.68
ZAPATA FELIPE                    22 1  75     1.72
BRACHO OMAIRA                    22 2  56     1.62
LOPEZ GILDA                      23 2  57     1.65
CHACON TERESA                    21 2  49     1.56
LEON VIRGINIA                    22 2  55     1.62
PADRON HENRY                     28 1  78     1.69
BAPTISTA JUAN                    26 1  95     1.83
PE$A LUIS                        23 1  72     1.58
ANDRADE CARMEN                 20 2  60     1.70
RAMIREZ SONIA                    24 2  60     1.64
;
PROC PRINT ;
TITLE  'IMPRESION DE LOS DATOS DEL PROBLEMA';
FORMAT SEXO SEX.;
PROC MEANS ;
VAR  EDAD  PESO  ESTATURA  ;
TITLE 'IMPRESION DE LAS ESTADISTICAS BASICAS' ;
PROC PLOT;
PLOT PESO*ESTATURA/
 ;
TITLE 'DIAGRAMA DE DISPERSION';
GOPTIONS DEVICE=WIN; /* WIN SE REFIERE A LA PANTALLA.   PARA IMPRIMIR EN
EPSON ESCRIBIR FX185 */
PROC GPLOT;
PLOT PESO*ESTATURA/
 ;
PROC CORR  DATA=CLASE
PEARSON
 ;
VAR PESO ;
WITH ESTATURA  ;
TITLE 'RESULTADOS DE LA CORRELACION ENTRE PESO Y ESTATURA';
PROC SORT  ;   BY NOMBRE ;
PROC PRINT ;
TITLE 'DATOS DEL PROBLEMA ORDENADOS DE ACUERDO AL NOMBRE';
RUN ;
```

. PROC PLOT

Produce un gráfico de una variable vs. otra. Puede ser utilizado para graficar variables tipo carácter, especifica el símbolo a graficar, especifica el largo y ancho del gráfico, se pueden superponer dos o más gráficos (plot). Forma general:

PROC PLOT opciones**;**

Opciones pueden ser:

DATA=archivo indica el archivo a utilizar.

Instrucciones usadas con el PROC PLOT

BY lista;
PLOT vertical * horizontal .../ opciones; vertical y horizontal se refieren a las variables a graficar en cada eje.

Opciones del PLOT:

. para superponer gráficos
OVERLAY

. para dibujar líneas del gráfico
HREF=lista de valores
VREF=lista de valores
HREFCHAR='C'
HREFCHAR='C'

. para definir la escala de los ejes
VAXIS=lista de valores
HAXIS=lista de valores
. largo y ancho del gráfico
HPOS=n
VPOS=n
HSPACE=n
VSPACE=n

. PROC GPLOT

Produce un gráfico de alta resolución de una variable vs. otra. Puede ser utilizado para graficar variables tipo carácter, especifica el símbolo a graficar, especifica el largo y ancho del gráfico, se pueden superponer dos o más gráficos (plot). Forma general:

PROC GPLOT opciones;

Las salidas (output) de los procedimientos gráficos pueden tomar una o mas de las siguientes formas:

. se muestra el gráfico en un monitor, terminal o estación de trabajo.

. el gráfico es enviado a un periférico (impresora o plotter).

. el gráfico es almacenado en un catalogo. El contenido de este catalogo puede ser visto con PROC GREPLAY.

La instrucción GOPTIONS debe ser ejecutada antes del procedimiento GPLOT.

Ejemplo 3.3.2

El siguiente ejemplo ilustra el uso del procedimiento PROC GPLOT y la forma de imprimir un gráfico. Asimismo, en este ejemplo se realiza un análisis de correspondencias referido a los tipos de propietarios de automóviles y los atributos de los mismos. Este es un ejemplo clásico de correspondencias que se presenta en los manuales del SAS con el nombre CRSPEX1.SAS.

```
/*************************************************************/
 /*        S A S   S A M P L E   L I B R A R Y                */
 /*                                                           */
 /*    NAME: CRSPEX1                                          */
 /*   TITLE: CAR OWNERS AND CAR ORIGIN                        */
 /* PRODUCT: STAT                                             */
 /*  SYSTEM: ALL                                              */
 /*    KEYS: MARKET PSYCHOME CAT GRAPHICS                     */
 /*   PROCS: CORRESP GPLOT                                    */
 /*    DATA:                                                  */
 /*                                                           */
 /*    REF: P-179, PROC CORRESP, EXAMPLE 1.                   */
 /*    MISC: YOU MUST LICENSE SAS/GRAPH SOFTWARE TO RUN PROC  */
 /*        GPLOT.  YOU MUST ALSO SPECIFY A GOPTIONS STATEMENT. */
 /*                                                           */
 /*************************************************************/
title 'Car Owners and Car Origin';
data cars;
  missing a;
  *---Read Numeric Variables---;
  input (norigin nsize ntype nhome nincome nmarital nkids nsex)
       (1.) @@;

  *---Check for End of Line---;
```

```
  if n(of norigin -- nsex) eq 0 then do;
    input;
    return;
    end;

  *---Create Character Variables---;
  if      norigin  = 1 then origin  = 'American';
  else if norigin  = 2 then origin  = 'Japanese';
  else if norigin  = 3 then origin  = 'European';
  if      nkids    > 0 then do;
    if   nmarital = 1 then marital = 'Single w Kids ';
    else  marital = 'Married w Kids';
    end;
  else do;
    if   nmarital = 1 then marital = 'Single        ';
    else  marital = 'Married       ';
    end;
  keep marital origin;
  output;
  return;
  cards;
13111221212111012111220113121101121122112211212113112212321122221221220 1
12112202312122123221110112212202212111012211210213111221112111011231110 1
21111211321122312112220222112211131112313121110232112222322122022122110 1
12212202212122021121220122112202112211013211220221311211133122612222110 1
1212110231AA220232112212113112112121220212212202112111022222110212121221
21121101221122221221110131311211312122012111221212111221221122222111211
22111101111220122212201131211013121220113112222131112012131110221112211
12111221221112112111220132112231131122111311221221321101312122022122110 1
13321101121222023331110221311102321112212131222221221211111122212111211
13311201121211221211221222222022131222222121101111122022211220113112212
21111201223222012122110221321101113122012121220121112211233122023331220 2
22212201211122021211220122112211221222022222122113111220121111101122121 2
11222201113111222121220232221102122211012122110133321101223211013221210 1
22322201311122011221110121121102211211021221110222112202111122011211121 1
11112202212111011331112232211112222121022221110121212202121122123211220 2
13311101131122112132220121312212111122122112202133122021212111212.2212
121122.22121210233112212222121011311122121211102211122112121110121212101
31121202223122111211221121121131222122121311221222112202222211013121220 2
21312221131122121211222211312222122122021311122112121122121122122112210 2
13112221121122022122210122311201211122121211110222312211131122212111110 2
21211101211122021331222223111221213122121121.2101312122012111122112112202
11121202312111011111222121211101221122022132110122121112212122011211112
```

```
212211022111110122221101121112112122110122122322211222122112212121122 02
213122112211110212121201113211012221110232111102212211012112220121212202
221112011211220121221101211211022211221112121101111112212121221111221201
211122122122111212112221111122312132110113121101121122222111220222121102
221211012122110221221102312111012122201211211011211222211112222122211102
212122021222120113112202121122212121110113111101123112212111220113111101
221112211321210131212211121211011222110122112222123122023121223112212202
311211012131110131221102112211021131220213122201222111022121221221312202
131.225232211101222122211311124122112202211211121312220221222201221221 01
212111011311220221312202221122123221210121222202223122121211221221111112
211111121211221221212201113122122131220222112222211122011311110112312211
211222013221220121211211312122122221220122112201111222011211110122311112
312111021231220122121101211112112.221102221122121211221222111101211121 01
121211013211222121112222321112112112110121321101113111012221220121312201
213211012212220221211101321122121111220221121101122211021122110213112212
212122011211122131221101121211022212220212121101
;

*---Perform Simple Correspondence Analysis---;
proc corresp all data=cars outc=coor;
  tables marital, origin;
  run;
*---Create Annotate Data Set---;
data coor;
  set coor;
  y     = dim1;   /* y-axis variable                    */
  x     = dim2;   /* x-axis variable                    */
  xsys  = '2';    /* x-axis label area is axis min to axis max */
  ysys  = '2';    /* y-axis label area is axis min to axis max */
  text  = _name_; /* text string to position in the plot       */
  size  = 2;      /* relative size of the text string          */
  label y = 'Dimension 1'
        x = 'Dimension 2';
  keep x y text xsys ysys size;
  run;

*---Plot the Simple Correspondence Analysis Results---;

*---Note, SAS/GRAPH Software and a GOPTIONS Statement are Required---;
proc gplot data=coor;
  symbol1  v=none;
  *---Note: Lengths are Device Specific---;
  axis1    length=2.5 IN order=-0.25 to 0.25 by 0.25;
```

```
plot y*x=1 / annotate=coor frame haxis=axis1 vaxis=axis1
        href=0 vref=0;
run;
```

. PROC IML (Lenguaje Matricial)

Este procedimiento permite realizar operaciones matriciales sobre conjuntos de datos. Es un Módulo Independiente del SAS Base que implementa el Lenguaje Matricial IML (Interactive Matrix Languaje) del Sistema SAS. Es un lenguaje muy amigable debido a que le permite pensar y expresarse en términos matriciales.

Para ejecutar el PROC IML escriba en el editor de programas (PGM):

PROC IML;

y luego presione la tecla Función F8 para activar el módulo; de inmediato el IML responderá con el mensaje:

IML ready

Una vez que ha invocado el IML no necesita escribir nuevamente PROC IML, debido a que es un modulo independiente e interactivo. Es decir, cuando Ud. invoca el módulo la primera vez ya se encuentra en el ambiente del IML, por lo tanto, para ejecutar otras instrucciones del IML debe escribirlas y luego presionar la tecla F8.

Para salir del IML escriba: **QUIT;** y presione la tecla F8.

Como todo lenguaje de programación, el IML tiene un vocabulario, expresiones, operadores operaciones e instrucciones. Entre las operaciones que puede realizar están: suma, resta, multiplicación, inversa común e inversa generalizada, entre otras.

A continuación se presenta un ejemplo que ilustra algunas operaciones e instrucciones de este lenguaje matricial.

Ejemplo 3.3.3

```
/*********************************************/
/*  Se ilustra la estimación del acuerdo tipo Kappa en- */
/*  tre varios observadores, asi  como los componentes */
/*  individuales para cada una de las L categorias  de  */
/*  respuesta en un diseño balanceado.  Se comparan    *
/*  original y duplicado para el diente patrón IPS17_16*/
/*                                                     */
/*  De:  Landis, J. R. and Koch, G. G.(1977). A one-   */
/*           way components of Variance Model for      */
```

```
/*          categorical Data. Biometrics 33,671-679      */
/*********************************************/
data prueba2;
input count1  count2  count3;
cards;
            2   0   0   2
            0   0   2   2
            0   0   2   2
            2   0   0   2
            1   1   0   2
            0   0   2   2
            0   0   2   2
            0   2   0   2
            1   0   1   2
            2   0   0   2
            2   0   0   2
            2   0   0   2
            2   0   0   2
            2   0   0   2
            2   0   0   2
            2   0   0   2
            2   0   0   2
            0   0   2   2
            0   0   2   2
            2   0   0   2
            2   0   0   2
            2   0   0   2
            2   0   0   2
;
run;
proc iml;       /* Se invoca al módulo IML */
use prueba2; /* Instrucción de Lectura USE para un archivo creado */
read all into y;  /*Indica que el contenido de la lectura se convierte en la matriz Y */
/*reset print; */
yt=y[+,]; /*Aqui se totaliza por columnas*/
di=y[,+]; /*Aqui se totaliza por filas*/
d=2;     ymed=y/d;
Nn=sum(di);  yt=yt/Nn;  dif=ymed;  suma1=(dif[1,] - yt)`*(dif[1,] - yt);
suma2=(dif[2,]- yt)`*(dif[2,]- yt);   suma3=(dif[3,]- yt)`*(dif[3,]- yt);
suma4=(dif[4,]- yt)`*(dif[4,]- yt);   suma5=(dif[5,]- yt)`*(dif[5,]- yt);
suma6=(dif[6,]- yt)`*(dif[6,]- yt);   suma7=(dif[7,]- yt)`*(dif[7,]- yt);
suma8=(dif[8,]- yt)`*(dif[8,]- yt);   suma9=(dif[9,]- yt)`*(dif[9,]- yt);
suma10=(dif[10,]- yt)`*(dif[10,]- yt);   suma11=(dif[11,]- yt)`*(dif[11,]- yt);
suma12=(dif[12,]- yt)`*(dif[12,]- yt);   suma13=(dif[13,]- yt)`*(dif[13,]- yt);
```

```
suma14=(dif[14,]- yt)`*(dif[14,]- yt);   suma15=(dif[15,]- yt)`*(dif[15,]- yt);
suma16=(dif[16,]- yt)`*(dif[16,]- yt);   suma17=(dif[17,]- yt)`*(dif[17,]- yt);
suma18=(dif[18,]- yt)`*(dif[18,]- yt);   suma19=(dif[19,]- yt)`*(dif[19,]- yt);
suma20=(dif[20,]- yt)`*(dif[20,]- yt);  suma21=(dif[21,]- yt)`*(dif[21,]- yt);
suma22=(dif[22,]- yt)`*(dif[22,]- yt);
ss=suma1+suma2+suma3+suma4+suma5+suma6+suma7+suma8+suma9+suma10+
suma11+suma12+suma13+suma14+suma15+suma16+suma17+suma18+suma19+
suma20+suma21+suma22;
n=nrow(y);  ss=d*ss;   MSS=SS/(n-1);   YTm=YT;  dif=y;
SSt=Nn*(Diag(YTM)-YTM`*YTM);
file print; /*Abre el archivo OUTPUT para escribir sobre el */
MST=SSt/(Nn-1);
put /;  /* Instrucción de escritura */
put @15 'CUADRADO MEDIO TOTAL'; PUT /;  /* @15 indica posición olumna 15 */
DO I=1 TO 3 BY 1;
DO J=1 TO 3 BY 1; CM=MST[i,j];
PUT @(11*J) CM @;
END;
PUT /;
END; PUT/;
print " CUADRADO MEDIO TOTAL" ,MSt;
SSe=SSt-SS;
/*Cuadrado Medio del Error*/
MSe=SSe/(Nn-n);   print "CUADRADO MEDIO DEL ERROR", MSe ;
put @15 'CUADRADO MEDIO DEL ERROR'; PUT /;
DO I=1 TO 3 BY 1;
DO J=1 TO 3 BY 1; CM=MSe[i,j];
PUT @(11*J) CM @;
END;
PUT /;
END; PUT/;
/* Sigma Cuadrado  de S */
SQs=diag((1/d)*(MSS- MSe));
Sqe=diag(MSe);  /*Determina la matriz Diagonal*/
Vp=(1/d)*(MSS+(d-1)*MSe);
SigmaQ=SQs+Sqe; /* Sigma Cuadrado estimado */
Sigma=sqrt(sigmaQ);
DS=diag(sigma);
/* Matriz de Correlación */
/* En la diagonal principal de esta matriz se encuentran las medidas de acuerdo tipo kappa
para cada categoría */
```

```
P=(1/d)*inv(DS)*(MSS - MSe)*inv(DS); /* inv(DS) es la inversa de la matriz DS  y P es la
matriz de Correlación */
put @15 'MATRIZ DE CORRELACION'; PUT /;
DO I=1 TO 3 BY 1;
DO J=1 TO 3 BY 1; CM=P[i,j];
PUT @(11*J) CM @;
END;   PUT /;  END; PUT/;
R=sum(SQS)/sum(SigmaQ);  PUT @15 'COEFICIENTE DE CORRELACION'; PUT /;
PUT @20 R;
print "MATRIZ DE CORRELACION", p;
print "COEFICIENTE DE CORRELACION",R;
quit;  /*Para salir del módulo IML */
```

4 USO DEL ASISTENTE SAS/ASSIST

El **SAS/ASSIST** es un software de tareas dirigidas, es un menú de iconos que simplifica el uso del SAS. Puede ser utilizado para el manejo de datos, escribir reportes, producir presentaciones gráficas, analizar datos, planificar proyectos, producir cartas de control de calidad o desarrollar aplicaciones.

Para utilizar el **ASSIST** seleccione de la barra de menú, **Soluciones** y luego **ASSIST (**Ver Figuras 9 y 10). Otra forma de invocar el **ASSIST** es a través de la barra **ToolBox** seleccione de derecha a izquierda el penúltimo icono y aparecerá una pantalla como la de la Figura 9.

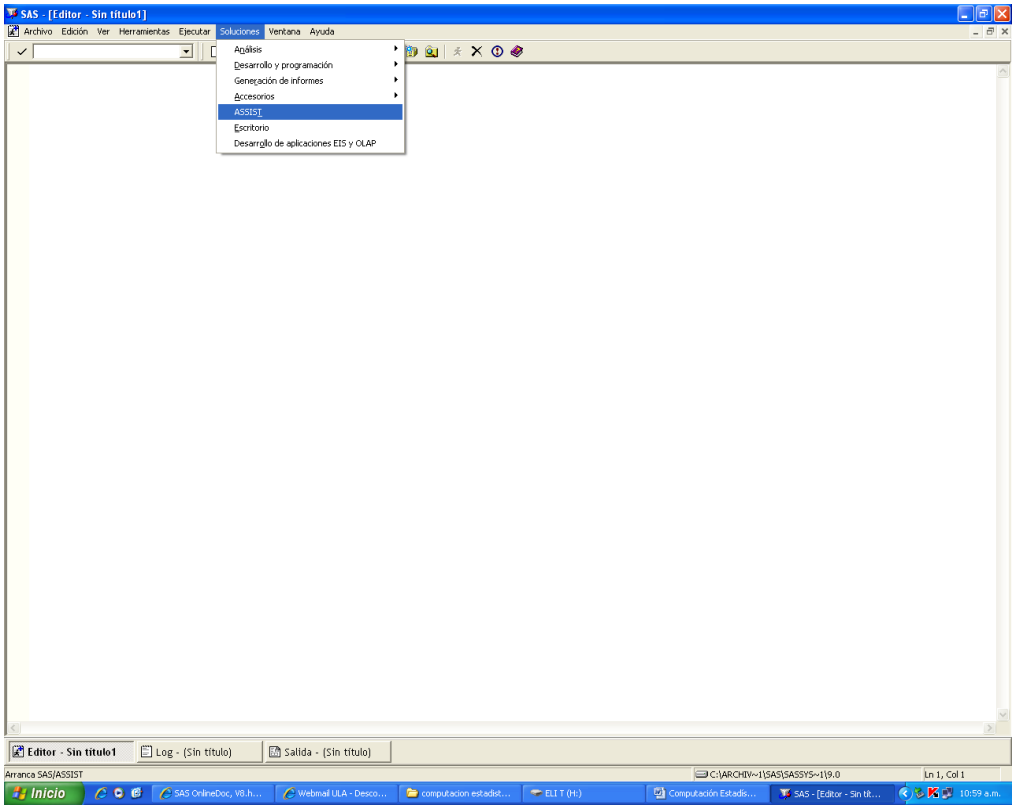

Figura 9.

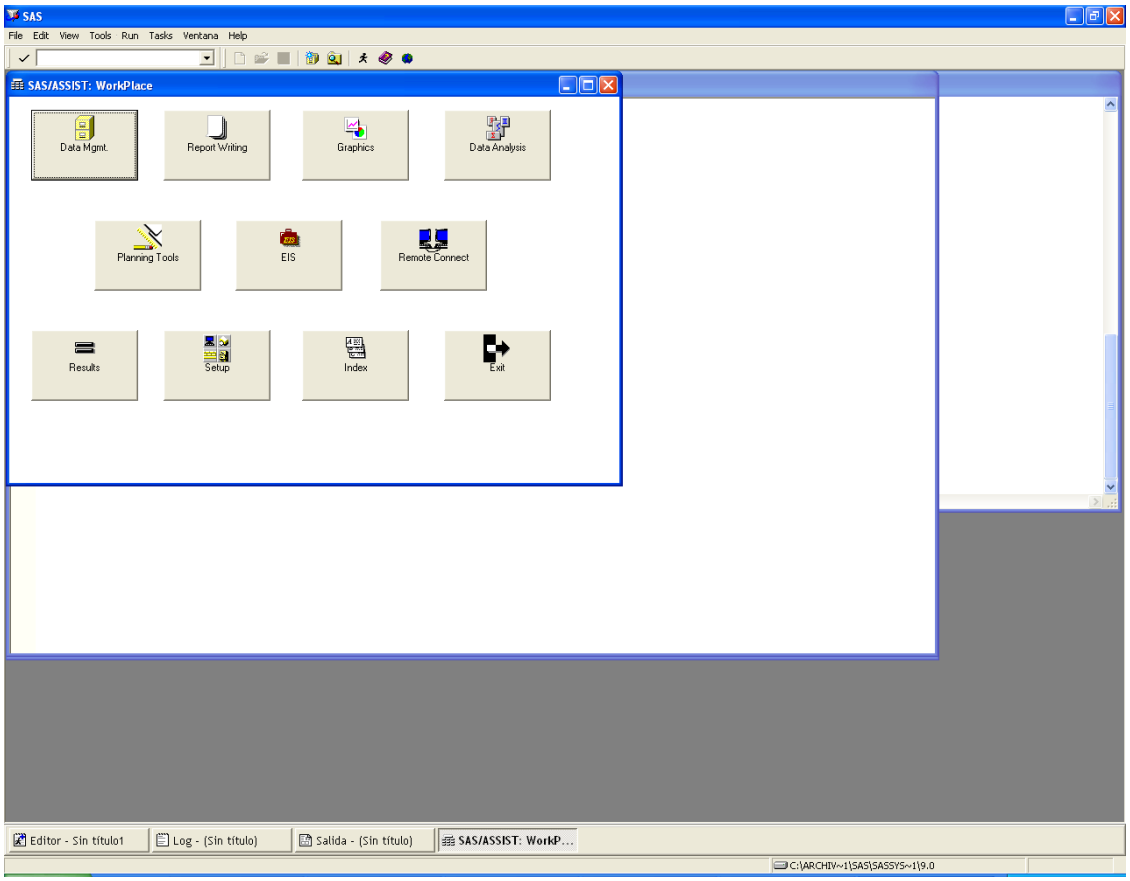
SAS
File Edit View Tools Run Tasks Ventana Help
SAS/ASSIST: WorkPlace
Data Mgmt.
Report Writing
Graphics
Data Analysis
Planning Tools
EIS
Remote Connect
Results
Setup
Index
Exit
Editor - Sin título1
Log - (Sin título)
Salida - (Sin título)
SAS/ASSIST: WorkP...
C:\ARCHIV~1\SAS\SASSYS~1\9.0

Figura 10.

BIBLIOGRAFIA

Aster, R. and Seidman, R. Professional SAS Programming Secrets. McGraw-Hill. 1991.

SAS Institute Inc., Guía Introductoria al SAS, Versión 6, CARY, NC, USA, 1989.

_____ , SAS Language Guide, Release 6.03, CARY, NC, USA, 1988a, 2nd printing 1989.

_____ , SAS Procedures Guide, Release 6.03 EDITION, CARY, NC, USA, 1988b, 4th printing 1990.

_____ , IML User's Guide, Release 6.03 EDITION, CARY, NC, USA, 1988c

_____ , SAS/STAT User's Guide, Release 6.03, NC, USA, 1988d, 3rd. printing 1991.

_____ , SAS/GRAPH User's Guide, Release 6.03, NC, USA, 1988e.

_____ , SAS Technical Report: P-179, CARY, NC, USA, 1988f.

_____ , SAS/ASSIST Software. Your Interface to the SAS System Version 6, First Edition. SAS Institute Inc, Cary, NC, USA, 1990.

_____ , SAS Language. Reference, Version 6, First Edition, SAS Institute Inc, Cary, NC, USA, 1990.

_____ , Getting Started with the SAS System Using SAS/ASSIST Software. Version 6, First Edition. SAS Institute Inc, Cary, NC, USA, 1991.