Taller de Programación

07 - Miscelánea

Mgs. Lic. Marcos Prunello

2018

Miscelánea

En la última clase vamos a hablar brevemente de algunos conceptos que nos permitirían desarrollar y ejecutar programas más generales y más complejos:

- La línea de comandos de Windows
- Ejecutar SAS desde la línea de comandos
- Operaciones vectorizadas en SAS-IML

La línea de comandos de Windows

- Windows 10 es un sistema operativo que nos permite interactuar con la computadora muy ágilmente a través de su interfaz gráfica.
- Sin embargo, es posible usar Windows de otra forma, a través de su **línea de comandos** o **símbolo del sistema** (también conocida como *consola de Windows*).
- Mediante este sistema, podemos comunicarnos con la computadora a través de comandos especiales para hacer cualquier tipo de actividad sin utilizar la interfaz gráfica.
- Esto se vuelve muy útil para realizar tareas administrativas, tareas que afectan varios archivos u objetos, lanzar a correr proyectos grandes, etc.
- Durante mucho tiempo, esta era la única forma disponible de usar la computadora (sistema operativo MS-DOS).
- La línea de comandos actual de Windows es compatible con la de MS-DOS, pero con extensiones.

Cómo abrir la consola de Windows

Escribir *cmd* en inicio y dar enter, o buscar *Símbolo del Sistema*.

Comandos útiles de la consola de Windows

- *cls*: limpia la ventana de CMD
- *exit*: cierrar la consola
- *cd*: muestra el directorio (carpeta) en la que estamos parados. También se usa para cambiar a otra carpeta (*cd = current directory/change directory*). Para cambiar de carpeta: *cd rutadecarpeta*. Para ir a una carpeta anterior: *cd ...*
- *ls* o *dir*: lista todos los archivos y carpetas que tenemos en el directorio actual
- *help*: muestra todos los comandos disponibles y una breve descripción. Para obtener ayuda para un comando específico, escribir el comando seguido de una barra diagonal y un signo de interrogación. Por ejemplo: cd /?.

Comandos útiles de la consola de Windows

- *copy origen destino*: copia un archivo de origen en una nueva ubicación de cualquier ruta dentro del disco duro.
- *move origen destino*: similar al anterior, pero elimina al archivo de la ubicación original (cortar).
- *del archivoocarpeta*: elimina un archivo o todo el contenido de la carpeta indicada, pero no elimina la carpeta en sí.
- *rd carpeta*: elimina toda la carpeta (*rd* = *remove directory*).
- *rename origen destino*: renombrar un archivo
- *md nombredecarpeta* o *mkdir nombredecarpeta*: crea una nueva carpeta (*make directory*)

Ejecutar SAS desde la línea de comandos

- Desde la línea de comandos podemos hacer que se ejecute nuestro programa de SAS.
- Por ejemplo, en el escritorio de mi PC tengo una carpeta llamada *demoSAS*, la dirección física de la carpeta es: *C:\Users\marco\Desktop\demoSas*.
- Ahí guardé el archivo *prueba1.sas* con el siguiente código:

```
proc iml;
    a = 1:10;
    print a;
quit;
```

Ejecutar SAS desde la línea de comandos

- En la consola, voy hasta esa carpeta con *cd*:

```
cd C:\Users\marco\Desktop\demoSas
```

- Y corro el programa haciendo:

```
sas prueba1.sas
```

Ejecutar SAS desde la línea de comandos

- Como resultado se crearon dos archivos:
  - **prueba1.lst**: contiene el resultado o "output" de nuestro programa, lo que veríamos en el Viewer de SAS.
  - **prueba1.log**: contiene la descripción del procedimiento ejecutado, incluye, si hubo, explicaciones sobre los errores encontrados, es lo que saldría en la ventana *log* de SAS.
- Estos son archivos de texto comunes, que se pueden abrir, por ejemplo, con el Block de Notas (botón derecho, abrir con...)

Variables de entorno

- En general, en lugar de correr sólo `sas prueba1.sas`, se debe especificar la ruta entera donde SAS ha sido instalado en la computadora, para que la misma sepa con qué programa debe ejecutar dicho comando.
- Por ejemplo, en mi computadora debo hacer:

```
C:\Program Files\SASHome\SASFoundation\9.4\sas prueba1.sas
```

- Pero para evitar tener que escribir tanto cada vez que quiero correr algo en SAS, le puedo avisar a la computadora que cada vez que escribo `sas` hago referencia al programa que está guardado allí.

Variables de entorno

- Esto se logra, editando las **variables de entorno**, que son cadenas de texto que contienen información acerca del entorno para determinar, por ejemplo, dónde buscar algunos archivos.
- En este link se puede ver cómo se editan las variables de entorno. Si uno quiere agregar SAS, tiene que seguir esos pasos reemplazando el ejemplo de `C:\Ruby22` por la dirección donde esté instalado SAS, que en mi computadora es `C:\Program Files\SASHome\SASFoundation\9.4`.
- Haciendo esto, me alcanza con ejecutar el comando `sas prueba1.sas` para que corra SAS.

Enviar información desde la consola hacia SAS

- De esta manera podríamos emular en cierto modo el comportamiento de *LEER* que incluimos en nuestros algoritmos cuando queremos dejar algo a elección del usuario.
- Para esto vamos a hacer uso de *variables macro* y la función *scan* (comentarios en clase).
- Vamos a ver cómo se usa con el siguiente ejemplo.

Enviar información desde la consola hacia SAS

- Tenemos el archivo *prueba2.sas* con el siguiente código:

```
%let limite = %scan(&sysparm, 1);
%let valor = %scan(&sysparm, 2);
%let dia = %scan(&sysparm, 3);

proc iml;
    vector = 1:&limite;
    print vector;
    if &valor > 10 then print &valor " es mayor que 10";
    else print &valor " es menor que 10";
    print "El doble de " &valor " es " (&valor * 2);
    print "Hoy es " &dia;
    if &dia = "sabado" then print "que bueno, es sabado";
    else print "que mal, no es sabado";
quit;
```

Enviar información desde la consola hacia SAS

- Si en la línea de comandos corremos

```
sas prueba2.sas -sysparm '5 7 "lunes"'
```

la variable de SAS *limite* recibirá el valor 5, *valor* recibirá el valor 7 y *dia* recibirá el valor de tipo caracter "lunes".

- Podemos operar con ellas dentro de IML, haciendo referencia a las mismas con un & delante de sus nombres.

Enviar información desde la consola hacia SAS

- Podemos usar esto para enviar al programa el nombre de un archivo que queremos leer. Por ejemplo, esto es mi archivo *prueba3.sas*.

```
%let limite = %scan(&sysparm, 1);
%let archivo = %scan(&sysparm, 2);

proc import out = Work.datos
            datafile = &archivo
            dbms = EXCEL REPLACE;
run;

proc iml;
    vector = 1:&limite;
    print vector;
    use Work.datos;
    read all into matriz;
    close Work.datos;
    print (nrow(matriz)) (ncol(matriz));
quit;
```

Enviar información desde la consola hacia SAS

- Lo voy a correr desde la consola y voy a hacer que se importe el conjunto de datos del archivo *Diametro* que está en la misma carpeta que este script.

```
sas prueba3.sas -sysparm '5 "Diametros"'
```

- Notar que puse el nombre del archivo sin su extensión ".xls".

Otras opciones para enviar información desde la consola hacia SAS

- La función *sysget* (me gusta más pero no pude hacer que las variables tipo caracter sean interpretadas bien dentro de IML, si alguien quiere explorarlo, bienvenido!). Con el comando:

```
sas prueba4.sas -set limite 5 -set valor 7
```

se puede correr el siguiente programa guardado en el archivo *prueba4.sas*:

```
%let limite = %sysget(limite);
%let valor = %sysget(valor);

proc iml;
    vector = 1:&limite;
    print vector;
    if &valor > 10 then print &valor " es mayor que 10";
    else print &valor " es menor que 10";
    print "El doble de " &valor " es " (&valor * 2);
quit;
```

Operaciones vectorizadas en SAS-IML

- Las reglas que aprendimos para escribir el pseudocódigo nos permite traducir nuestros algoritmos a cualquier lenguaje de computación de manera muy general y sencilla.
- Sin embargo, cada lenguaje de programación tiene diseñados un conjunto de comandos, funciones, estructuras de datos, etc., que facilitan algunas tareas, pero son específicos de ese lenguaje y no siempre generalizables.
- Una vez que hemos incorporado los conceptos básicos de la programación, podemos dedicarnos a aprender las profundidades de un lenguaje en particular.

Operaciones vectorizadas en SAS-IML

- Por ejemplo, IML significa **Interactive Matrix Language** porque está pensando para hacer operaciones vectorizadas, es decir, para operar al mismo tiempo todos los elementos de los vectores y matrices (no con cada celda una por una como aprendimos a realizar).
- Ver en el archivo *operaciones_vectorizadas.sas* ejemplos de cómo se pueden "acortar" muchas de las tareas que aprendimos a programar desde cero usando aspectos específicos de IML.
- El archivo *pr7_ej8_con_otras_funciones.sas* muestra cómo podemos hacer uso de ellas para resolver de otra forma el ejercicio 8 de la práctica 7.