```
   KillDataFolder SubroutineResults          // We are done with results
End
```

Note that the NVAR statement must appear *after* the call to the procedure (Subroutine in this case) that creates the global variable. This is because NVAR has both a compile-time and a runtime behavior. At compile time, it creates a local variable that Igor can compile (theAvg in this case). At runtime, it actually looks up and creates a link to the global (variable gAvg stored in data folder SubroutineResults in this case).

Often a function needs to access a large number of global variables stored in a data folder. In such cases, you can write more compact code using the ability of NVAR, SVAR and WAVE to access multiple objects in one statement:

```
Function Routine2()
   Make aWave= {1,2,3,4}
   Subroutine(aWave)

   DFREF dfr = :SubroutineResults

   NVAR theAvg=dfr:gAvg, theMin=dfr:gMin     // Access two variables
   SVAR theName = dfr:gWName
   Print theAvg, theMin, theName

   KillDataFolder SubroutineResults          // We are done with results
End
```

Automatic Creation of WAVE, NVAR and SVAR References

The Igor compiler sometimes automatically creates WAVE, NVAR and SVAR references. For example:

```
Function Example1()
   Make/O wave0
   wave0 = p

   Variable/G gVar1
   gVar1= 1

   String/G gStr1
   gStr1= "hello"
End
```

In this example, we did not use WAVE, NVAR or SVAR references and yet we were still able to compile assignment statements referencing waves and global variables. This is a feature of *Make*, `Variable/G` and `String/G` that automatically create local references for simple object names.

Simple object names are names which are known at compile time for objects which will be created in the current data folder at runtime. `Make`, `Variable/G` and `String/G` do not create references if you use $ expression, a partial data folder path or a full data folder path to specify the object unless you include the /N flag, discussed next.

Explicit Creation of NVAR and SVAR References

If you create a global variable in a user-defined function using a path or a $ expression, you can explicitly create an NVAR or SVAR reference like this:

```
Function Example2()
   String path

   Variable/G root:gVar2A = 2
   NVAR gVar2A = root:gVar2A        // Create NVAR gVar2A

   path = "root:gVar2B"
   Variable/G $path = 2
```