To return multiple waves, you can return a "wave reference wave". See **Wave Reference Waves** on page IV-77 for details.

Writing Functions that Process Waves

The user function is a powerful, general-purpose analysis tool. You can do practically any kind of analysis. However, complex analyses require programming skill and patience.

It is useful to think about an analysis function in terms of its input parameters, its return value and any side effects it may have. By return value, we mean the value that the function directly returns. For example, a function might return a mean or an area or some other characteristic of the input. By side effects, we mean changes that the function makes to any objects. For example, a function might change the values in a wave or create a new wave.

This table shows some of the common types of analysis functions. Examples follow.

Input Parameters	Return Value	Side Effects	Example Function
A source wave	A number	None	WaveSum
A source wave	Not used	The source wave is modified	RemoveOutliers
A source wave	Wave reference	A new destination wave is created	LogRatio
A source wave and a destination wave	Not used	The destination wave is modified	
An array of wave references	A number	None	WavesMax
An array of wave references	Wave reference	A new wave is created	WavesAverage

The following example functions are intended to show you the general form for some common analysis function types. We have tried to make the examples useful while keeping them simple.

WaveSum Example

Input:	Source wave
Return value:	Number
Side effects:	None

```
// WaveSum(w)
// Returns the sum of the entire wave, just like Igor’s sum function.
Function WaveSum(w)
   Wave w

   Variable i, n=numpnts(w), total=0
   for(i=0;i<n;i+=1)
      total += w[i]
   endfor

   return total
End
```

To use this, you would execute something like

```
Print "The sum of wave0 is:", WaveSum(wave0)
```

RemoveOutliers Example

Input:	Source wave
Return value:	Number
Side effects:	Source wave is modified

Often a user function used for number-crunching needs to loop through each point in an input wave. The following example illustrates this.

```
// RemoveOutliers(theWave, minVal, maxVal)
// Removes all points in the wave below minVal or above maxVal.
// Returns the number of points removed.
Function RemoveOutliers(theWave, minVal, maxVal)
   Wave theWave
   Variable minVal, maxVal

   Variable i, numPoints, numOutliers
   Variable val
   numOutliers = 0
   numPoints = numpnts(theWave)         // number of times to loop

   for (i = 0; i < numPoints; i += 1)
      val = theWave[i]
      if ((val < minVal) || (val > maxVal))  // is this an outlier?
         numOutliers += 1
      else                                   // if not an outlier
         theWave[i - numOutliers] = val      // copy to input wave
      endif
   endfor

   // Truncate the wave
   DeletePoints numPoints-numOutliers, numOutliers, theWave
   return numOutliers
End
```

To test this function, try the following commands.

```
Make/O/N=10 wave0= gnoise(1); Edit wave0
Print RemoveOutliers(wave0, -1, 1), "points removed"
```

RemoveOutliers uses the for loop to iterate through each point in the input wave. It uses the built-in numpnts function to find the number of iterations required and a local variable as the loop index. This is a very common practice.

The line "`if ((val < minVal) || (val > maxVal))`" decides whether a particular point is an outlier. `||` is the logical OR operator. It operates on the logical expressions "`(val < minVal)`" and "`(val > maxVal)`". This is discussed in detail under **Bitwise and Logical Operators** on page IV-41.

To use the WaveMetrics-supplied RemoveOutliers function, include the Remove Points.ipf procedure file:

```
#include <Remove Points>
```

See **The Include Statement** on page IV-166 for instructions on including a procedure file.

LogRatio Example

Input:	Source waves
Return value:	Wave reference
Side effects:	Output wave created

```
// LogRatio(source1, source2, outputWaveName)
// Creates a new wave that is the log of the ratio of input waves.
// Returns a reference to the output wave.
Function/WAVE LogRatio(source1, source2, outputWaveName)
   Wave source1, source2
   String outputWaveName

   Duplicate/O source1, $outputWaveName
   WAVE wOut = $outputWaveName
   wOut = log(source1/source2)
   return wOut
End
```

To call this from a function, you would execute something like:

```
Wave wRatio = LogRatio(wave0, wave1, "ratio")
Display wRatio
```

The LogRatio function illustrates how to treat input and output waves. Input waves must exist before the function is called and therefore are passed as wave reference parameters. The output wave may or may not already exist when the function is called. If it does not yet exist, it is not possible to create a wave reference for it. Therefore we pass to the function the desired name for the output wave using a string parameter. The function creates or overwrites a wave with that name in the current data folder and returns a reference to the output wave.

WavesMax Example

Input:	Array of wave references
Return value:	Number
Side effects:	None

```
// WavesMax(waves)
// Returns the maximum value in waves in the waves array.
Function WavesMax(waves)
   Wave/WAVE waves

   Variable theMax = -INF

   Variable numWaves = numpnts(waves)
   Variable i
   for(i=0; i<numWaves; i+=1)
      Wave w = waves[i]
      Variable tmp = WaveMax(w)
      theMax = max(tmp, theMax)
   endfor

   return theMax
End
```

This function illustrates how you might call WavesMax from another function:

```
Function DemoWavesMax()
   Make/FREE/N=10 w0 = p
   Make/FREE/N=10 w1 = p + 1

   Make/FREE/WAVE waves = {w0, w1}

   Variable theMax = WavesMax(waves)

   Printf "The maximum value is %g\r", theMax
End
```

WavesAverage Example

Input:	An array of wave references
Return value:	Wave reference
Side effects:	Creates output wave

```
// WavesAverage(waves, outputWaveName)
// Returns a reference to a new wave, each point of which contains the
// average of the corresponding points of a number of source waves.
// waves is assumed to contain at least one wave reference and
// all waves referenced by waves are expected to have the same
// number of points.
Function/WAVE WavesAverage(waves, outputWaveName)
   Wave/WAVE waves         // A wave containing wave references
   String outputWaveName
```