```
// Creates a 2D wave consisting of 3 rows and 2 columns
Make wave2 = {{1,2,3},{4,5,6}}
```

See **Lists of Values** on page II-78 for further discussion of the use of curly braces.

The Duplicate operation can create an exact copy of a multidimensional wave or, using the /R flag, extract a subrange. Here is the syntax of the /R flag:

```
Duplicate/R=[startRow,endRow][startCol,endCol] and so on...
```

You can use the character * for any end field to specify the last element in the given dimension or you can just omit the end field. You can also specify just [] to include all of a given dimension. If the source wave has more dimensions than you specify in the /R flag, then all of those dimensions are copied. For example:

```
// Make a 3D wave to play with
Make/N=(5,4,3) wave3A = p + 10*q + 100*r

// Duplicate rows 1 through 2, columns 2 through the end, and all layers
Duplicate/R=[1,2][2,*] wave3A, wave3B     // 2 rows, 2 columns, 3 layers

// Create a 3D wave consisting of all rows of column 2, layer 0
Duplicate/R=[][2,2][0,0] wave3A, wave3C   // 5 rows, 1 column, 1 layer
```

Igor considers wave3C to be 3 dimensional wave and not 1 dimensional, even though it consists of just one column of data, because the number of columns and layers are greater than zero. This is a subtle distinction and can cause confusion. For example, you may think you have extracted a 1D wave from a 3D object but you will find that wave3C will not show up in dialogs where 1D waves are required.

You can turn the 3D wave wave3C into a 1D wave using the following command:

```
Redimension/N=(-1,0) wave3C
```

The -1 value does not to change the number of rows whereas the 0 value for the number of columns indicates that there are no dimensions past rows (in other words, no columns, layers or chunks).

# Programmer Notes

For historical reasons, you can treat the symbols x and p like global variables, meaning that you can store into them as well as retrieve their values by referencing them. But this serves no purpose and is not recommended.

Unlike x and p, y, z, t, q, r and s act like functions and you can't store into them.

Here are some functions and operations that are useful in programming with multidimensional waves:

```
DimOffset, DimDelta, DimSize
FindDimLabel, SetDimLabel, GetDimLabel
```

# Dimension Labels

A dimension label is a name associated with a dimension (rows, columns, layers or chunks) or with a specific dimension index (row number, column number, layer number or chunk number).

Dimension labels are primarily an aid to the Igor procedure programmer when dealing with waves in which certain elements have distinct purposes. Dimension labels can be set when loading from a file, and can be displayed, created or edited in a table (see **Showing Dimension Labels** on page II-235).

You can give names to individual dimension indices in multidimensional or 1D waves. For example, if you have a 3 column wave, you can give column 0 the name "red", column 1 the name "green" and column 2 the name "blue". You can use the names in wave assignments in place of literal numbers. To do so, you use the % symbol in front of the name. For example:

```
wave2D[][%red] = wave2D[p][%green]     //Set red column equal to green column
```