Using

The most common datafile modifier is **using**.

Syntax:

```
plot 'file' using {<entry> {:<entry> {:<entry> ...}}} {'format'}
```

If a format is specified, each datafile record is read using the C library's 'scanf' function, with the specified format string. Otherwise the record is read and broken into columns at spaces or tabs. A format cannot be specified this way for time-format data (instead use **set xdata time**).

The resulting array of data is then sorted into columns according to the entries. Each 5#5entry6#6 may be a simple column number, which selects the datum, an expression enclosed in parentheses, or empty. The expression can use $1 to access the first item read, $2 for the second item, and so on. It can also use **column(x)** and **valid(x)** where x is an arbitrary expression resulting in an integer. **column(x)** returns the x'th datum; **valid(x)** tests that the datum in the x'th column is a valid number. A column number of 0 generates a number increasing (from zero) with each point, and is reset upon encountering two blank records. A column number of -1 gives the dataline number, which starts at 0, increments at single blank records, and is reset at double blank records. A column number of -2 gives the index number, which is incremented only when two blank records are found. An empty 5#5entry6#6 will default to its order in the list of entries. For example, **using ::4** is interpreted as **using 1:2:4**.

N.B. -- the **call** command also uses $'s as a special character. See **call (p.)** for details about how to include a column number in a **call (p.)** argument list.

If the **using** list has but a single entry, that 5#5entry6#6 will be used for y and the data point number is used for x; for example, "**plot 'file' using 1**" is identical to "**plot 'file' using 0:1**". If the **using** list has two entries, these will be used for x and y. Additional entries are usually errors in x and/or y. See **set style (p.)** for details about plotting styles that make use of error information, and **fit (p.)** for use of error information in curve fitting.

'scanf' accepts several numerical specifications but **gnuplot** requires all inputs to be double-precision floating-point variables, so "%lf" is essentially the only permissible specifier. A format string given by the user must contain at least one such input specifier, and no more than seven of them. 'scanf' expects to see white space -- a blank, tab ("3#3t"), newline ("3#3n"), or formfeed ("3#3f") -- between numbers; anything else in the input stream must be explicitly skipped.

Note that the use of "3#3t", "3#3n", or "3#3f" requires use of double-quotes rather than single-quotes.

Examples:

This creates a plot of the sum of the 2nd and 3rd data against the first: The format string specifies comma- rather than space-separated columns. The same result could be achieved by specifying **set datafile separator ","**.

```
plot 'file' using 1:($2+$3) '%lf,%lf,%lf'
```

In this example the data are read from the file "MyData" using a more complicated format:

```
plot 'MyData' using "%*lf%lf%*20[^\n]%lf"
```

The meaning of this format is:

```
%*lf        ignore a number
%lf         read a double-precision number (x by default)
%*20[^\n]   ignore 20 non-newline characters
%lf         read a double-precision number (y by default)
```

One trick is to use the ternary **?:** operator to filter data:

```
plot 'file' using 1:($3>10 ? $2 : 1/0)
```

which plots the datum in column two against that in column one provided the datum in column three exceeds ten. **1/0** is undefined; **gnuplot** quietly ignores undefined points, so unsuitable points are suppressed.

In fact, you can use a constant expression for the column number, provided it doesn't start with an opening parenthesis; constructs like **using 0+(complicated expression)** can be used. The crucial point is that the expression is evaluated once if it doesn't start with a left parenthesis, or once for each data point read if it does.

If timeseries data are being used, the time can span multiple columns. The starting column should be specified. Note that the spaces within the time must be included when calculating starting columns for other data. E.g., if the first element on a line is a time with an embedded space, the y value should be specified as column three.

It should be noted that **plot 'file'**, **plot 'file' using 1:2**, and **plot 'file' using ($1):($2)** can be subtly different: 1) if **file** has some lines with one column and some with two, the first will invent x values when they are missing, the second will quietly ignore the lines with one column, and the third will store an undefined value for lines with one point (so that in a plot with lines, no line joins points across the bad point); 2) if a line contains text at the first column, the first will abort the plot on an error, but the second and third should quietly skip the garbage.

In fact, it is often possible to plot a file with lots of lines of garbage at the top simply by specifying

```
plot 'file' using 1:2
```

However, if you want to leave text in your data files, it is safer to put the comment character (#) in the first column of the text lines.

http://www.gnuplot.info/demo/using.htmlFeeble using demos.

If gnuplot is built with configuration option -enable-datastrings, then additional modifiers to **using** can specify handling of text fields in the datafile. See **datastrings (p.)**, **using xticlabels (p.)**, **using title (p.)**.

Subsections

Ethan Merritt 2007-03-03