Chapter 8: Tic Control

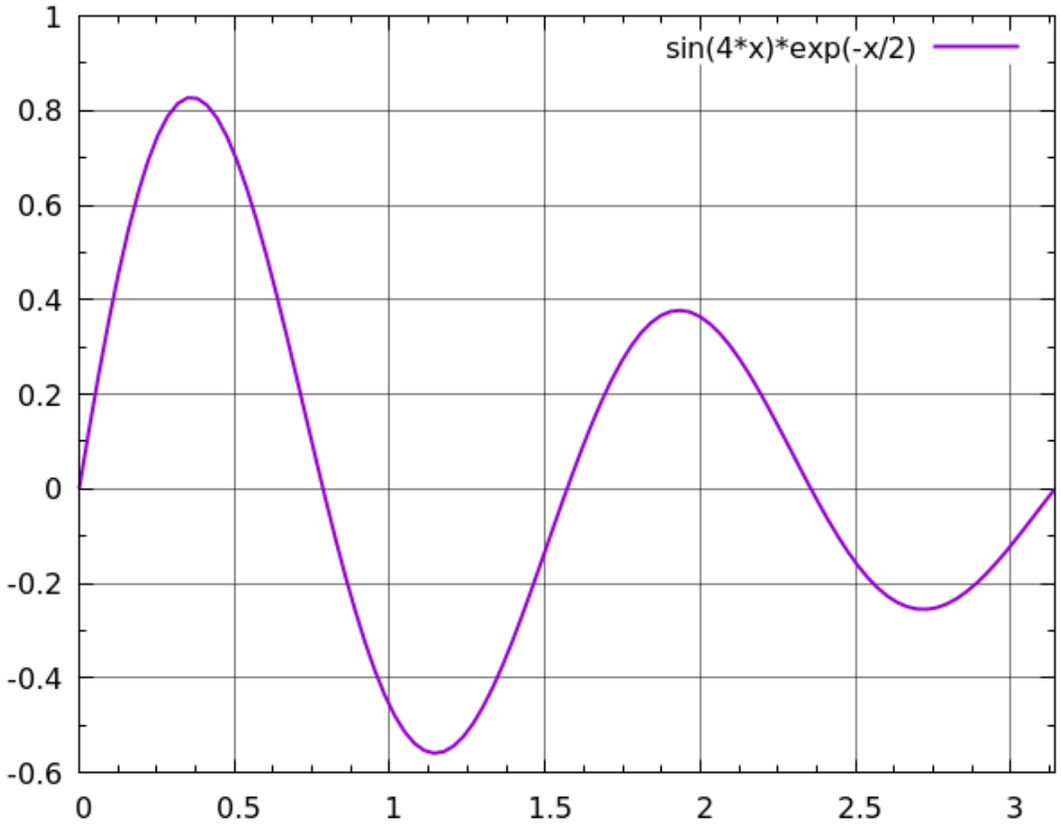
This chapter is about tics in gnuplot. Tics are important: they are the things that connect the curves, colors, and surfaces in your plots to actual numbers. The subject of "tics" includes not merely those little line segments dividing up the axes of your plots, but the labels associated with them. In this chapter you will learn how to get complete control of the tics and their labels: a somewhat tricky subject, it turns out. You'll learn about some features of gnuplot that we've been saving for this chapter, because they're really part of tic creation: gnuplot's handling of dates and times, and of latitude and longitude.

Minor Tics

Here is a simple plot with a subtle feature that we haven't seen before in any of our examples. There are smaller (shorter) tic marks between the major, labeled tic marks. These are called "minor tics". Notice that the grid is aligned with the major tic marks. As you might have guessed, the `set mxtics` command sets the minor tics on the x-axis and `set mytics` creates minor tics on the y-axis. The number following the keyword (`mxtics` or `mytics`) establishes the number of spaces between tics (not the actual number of minor tics). The figure should make it clear how the commands work. The purpose of minor tics is to make it easier to extract quantitative information from the graph by facilitating interpolation between the numerically labeled values associated with the major tic marks.

```
set grid lt -1
set mxtics 4
set mytics 2
set xr [0 : pi]
plot sin(4*x)*exp(-x/2) lw 2
```

Open script

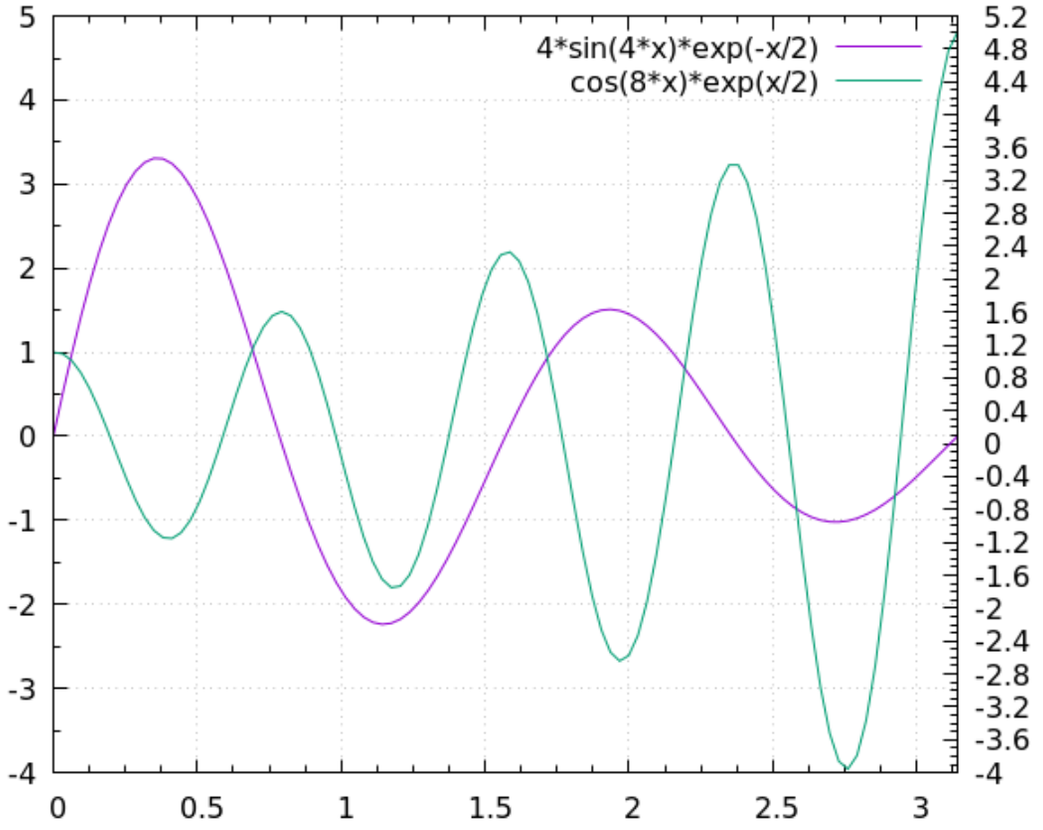


...On a Second Axis

Way back in the first chapter we learned how to put an independent scale and associated tic marks on a second axis. You can also set up minor tics on second axes, as in the following example (there is also a command `set mx2tics`, for a second x-axis):

```
set grid
set mxtics 4
set mytics 2
set xr [0 : pi]
set ytics nomirror
set y2tics 0.4
set my2tics 4
plot 4*sin(4*x)*exp(-x/2), cos(8*x)*exp(x/2)
```

Open script

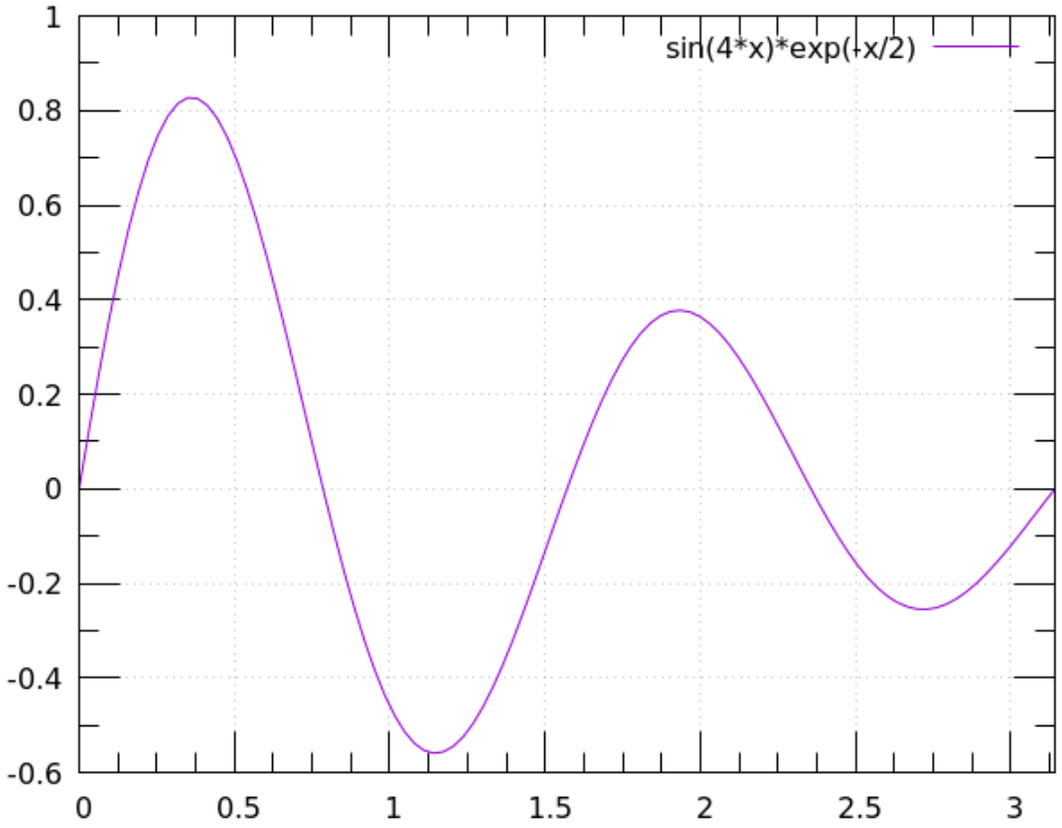


Adjusting the Tic Size

The default length gnuplot uses for tics is a bit small, and makes the minor tics nearly disappear on small plots. Naturally, the tic size can be adjusted to any size you like. The following figure employs longer tic marks than in the previous examples; the number "3" in the command will set the length of the major tics to be three times the default length, which depends on the terminal in use:

```
set grid
set tics scale 3
set mxtics 4
set mytics 2
set xr [0 : pi]
plot sin(4*x)*exp(-x/2)
```

Open script

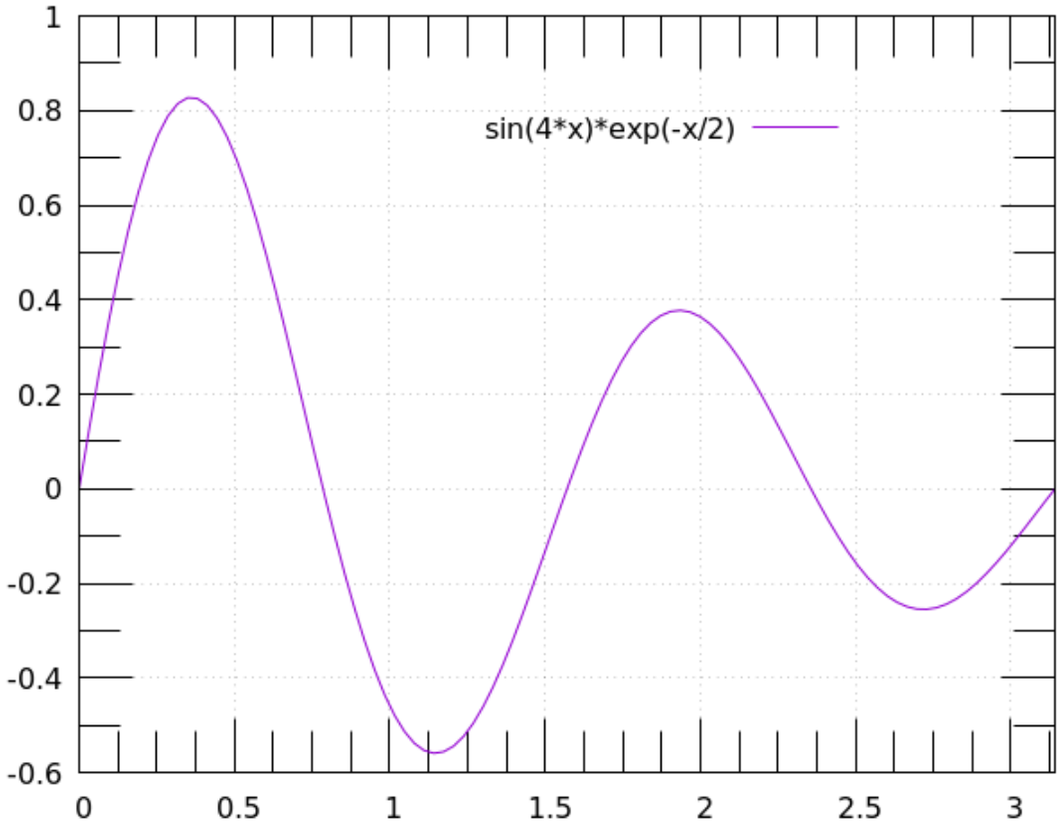


...Of Minor Tics

By default, gnuplot assigns a length to the minor tics that is one-half the length of the major tics, so in the previous example the minor tics have a scale of 1.5. If you want to set a non-default scale for the minor tics, the command becomes `set tics scale a, b`, where a is the scale for the major tics and b is the scale for the minor tics. You can even make the minor tics longer than the major tics if, for some reason, you want to.

```
set grid
set tics scale 4, 3
set key at 2.5, 0.8
set mxtics 4
set mytics 2
set xr [0 : pi]
plot sin(4*x)*exp(-x/2)
```

Open script

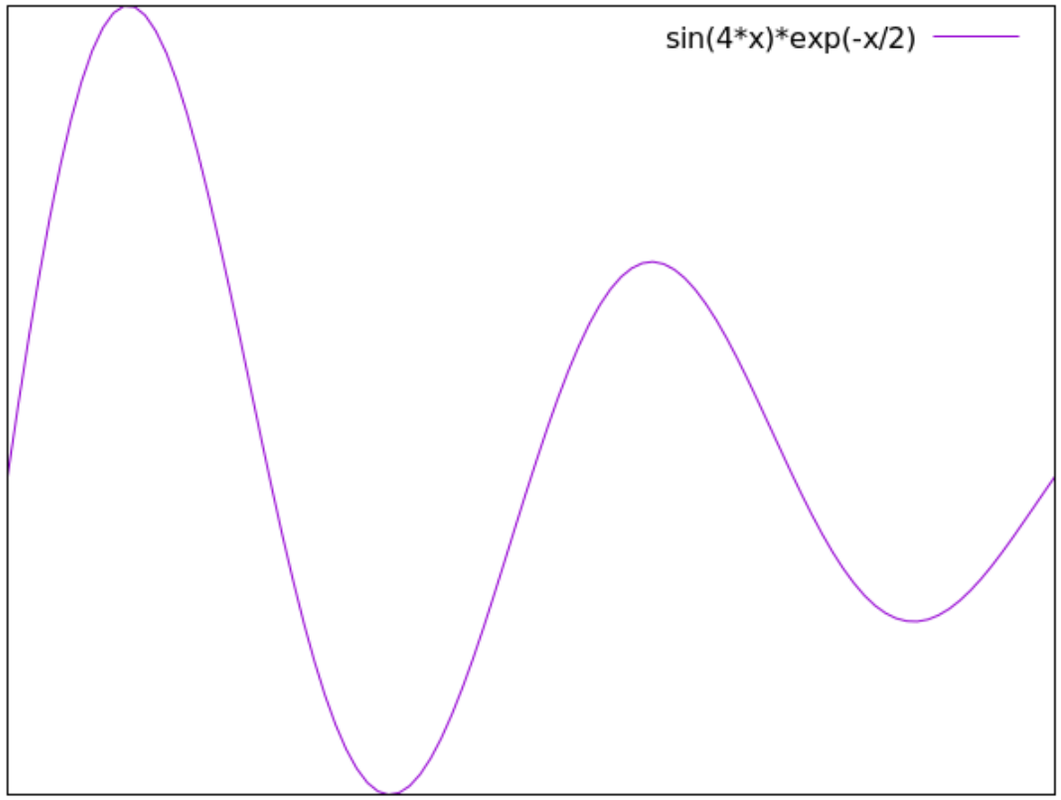


Removing All The Tics

You can remove the tics entirely from the plot. This gives a simplified appearance for situations where you need an illustration and the reader is not expected to glean quantitative information from the graph.

```
unset tics
set xr [0 : pi]
plot sin(4*x)*exp(-x/2)
```

Open script

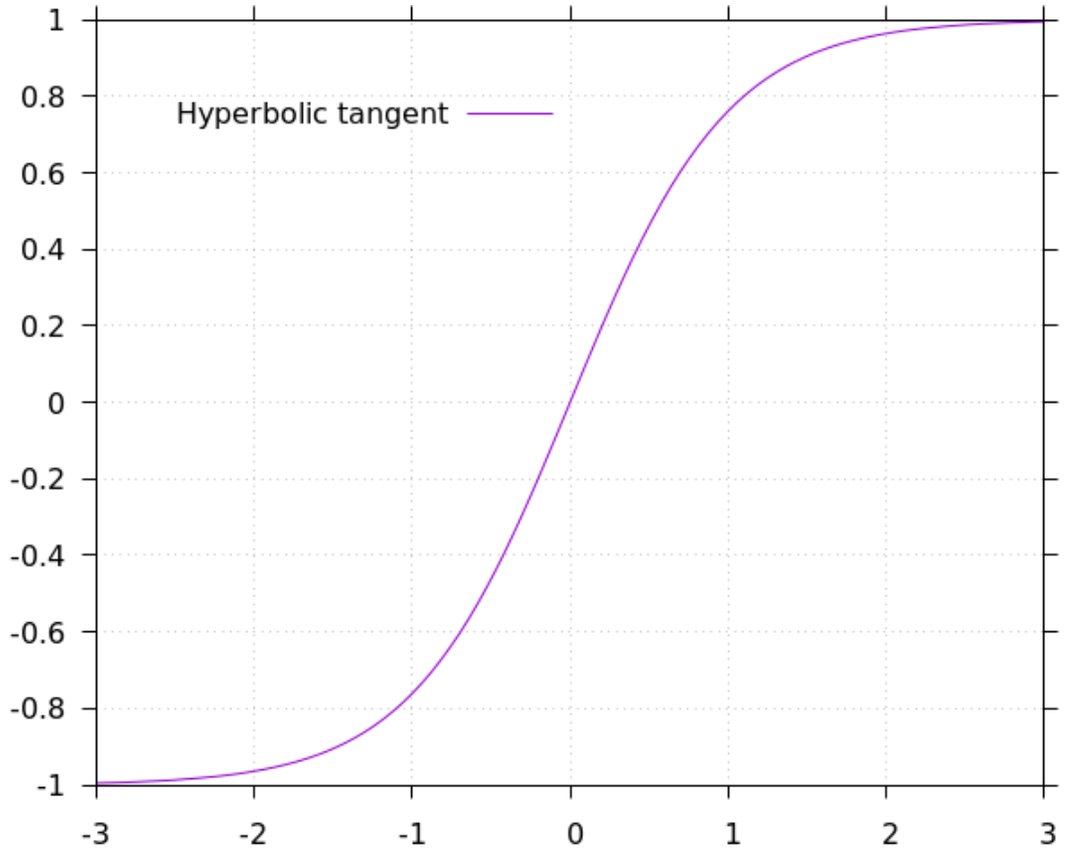


Making the Tics Stick Out

The following example illustrates another tic style available in gnuplot, where the tics stick out of the plot rather than intrude on the interior of the graph. This style is useful when the default tic marks might obscure your plot elements; for example, when part of the curve lies close to an axis. You can apply it to only one axis (`set xtics out`), too. You can also get the same effect by using negative numbers in the tic scaling command.

```
set tics out
set grid
set xr [-3 : 3]
set key at 0, 0.8
plot tanh(x) title "Hyperbolic tangent"
```

Open script

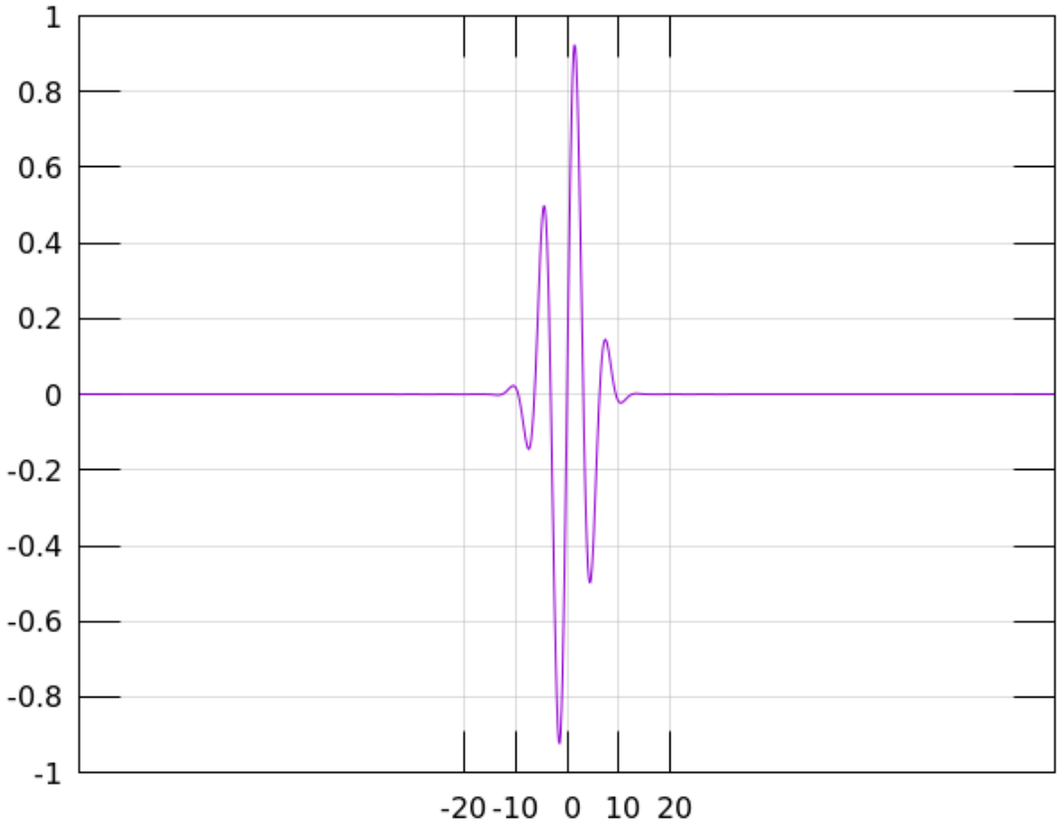


Setting Tic Values

You can set the locations of a plot's tics independently of the axes ranges, using the command `set xtics b, i, e`, where "b" is the starting tic value, "i" is the increment between tics, and "e" is the ending tic value (there are corresponding commands for the tics on the other axes, of course). In this way you can focus the plot on a particular area, where the reader may want to read off quantitative information. The grid lines, as before, will track the major tics. A simpler version of the commands, for example, `set xtics i`, just sets the increment, and allows gnuplot to set the beginning and ending values.

```
unset key
set samples 1000
set tics scale 3
set grid lt 1 lc "grey" lw .5
set xr [-30*pi : 30*pi]
set xtics -20, 10, 20
plot sin(x) * exp(-x**2/30)
```

Open script

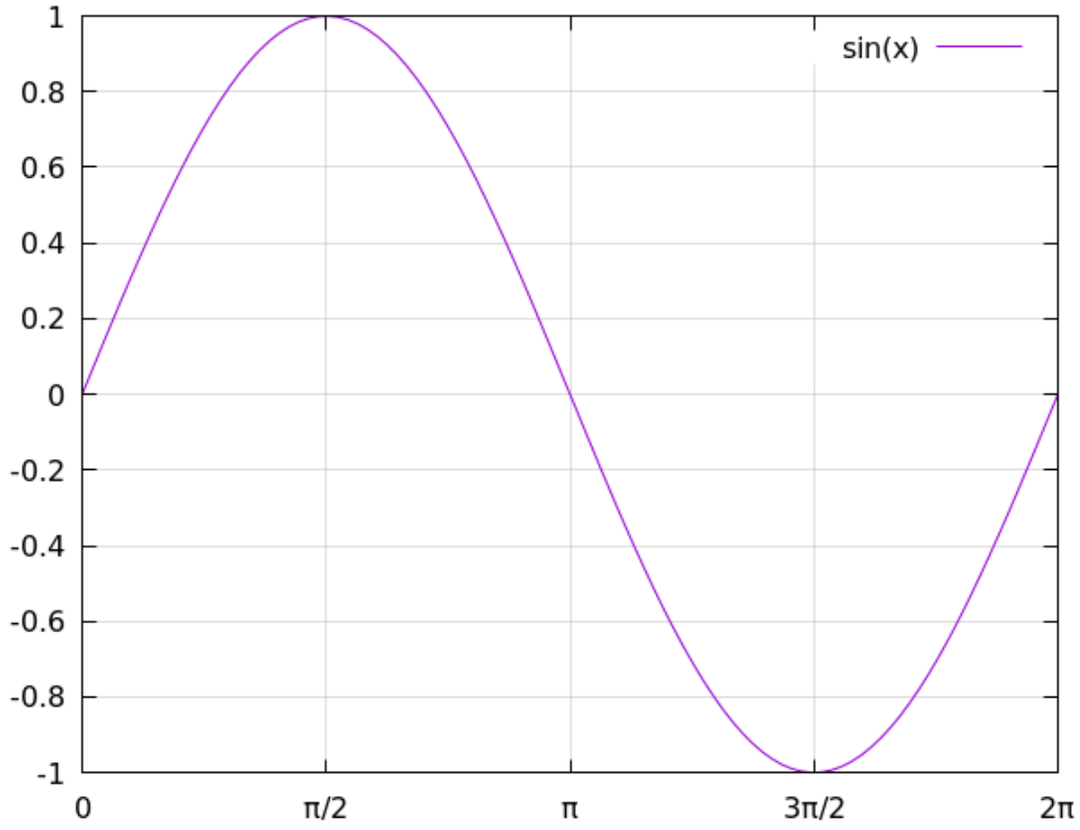


Setting Tics Manually

Sometimes gnuplot's automatic tic placement is not flexible enough and you need to take complete control of the position of every tic mark, or you need to place custom labels on the tics rather than rely on the automatically generated numerical labels. Gnuplot will cooperate. Notice the tic labels and positions along the x-axis in the following plot. The tics are aligned with the peaks and zero crossings of the sine wave, and are labeled using π rather than an approximate decimal. This is the natural way to label the axis when plotting this circular function. Gnuplot's automatically chosen tic positions and numerical labels would be placed at the positions 1, 2, 3, etc., and would have no particular relation to the function we are plotting.

```
set xr [0 : 2*pi]
set grid lt 1 lc "grey" lw .5
set xtics\
  ("π" pi, "π/2" pi/2, "2π" 2*pi, "3π/2" 3*pi/2, "0" 0)
plot sin(x)
```

Open script



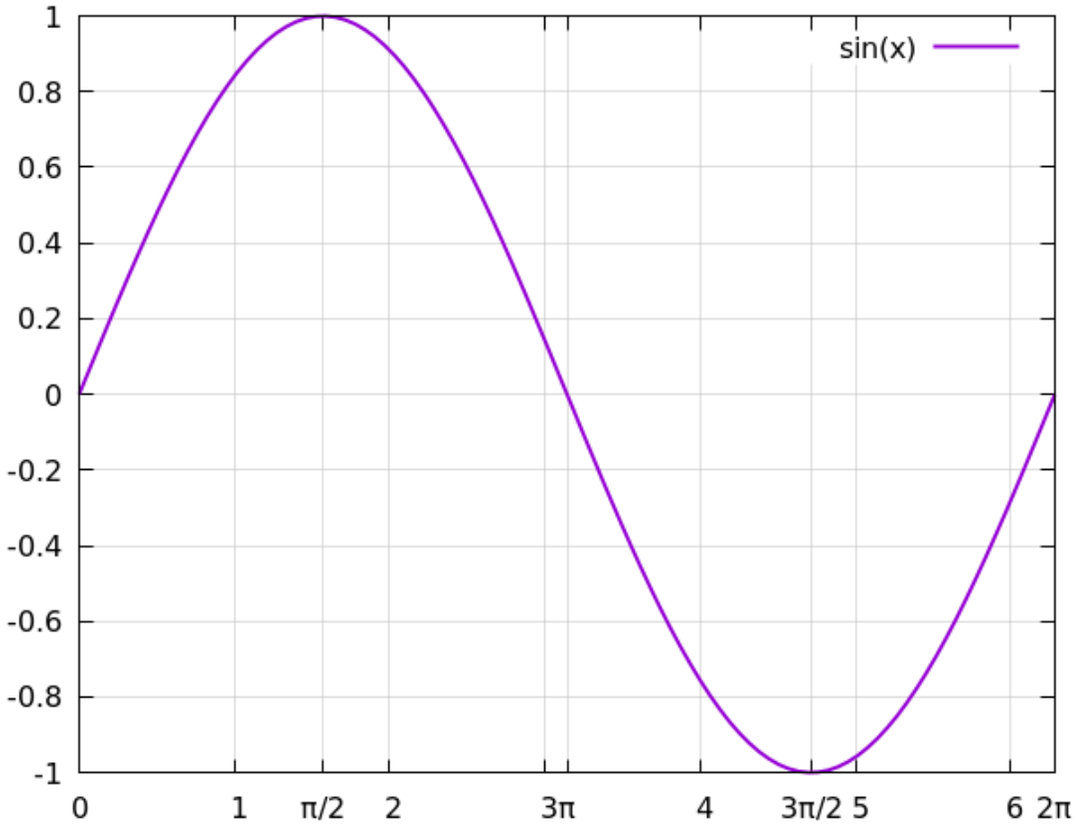
Combining Automated and Manual Tics

You can supplement gnuplot's automatic tics (either the fully automatic ones or the ones set using the `set xtics b, i, e` command) with any number of manual tics using the highlighted command in the script below. This is useful to call attention to a small number of salient coordinate positions. Notice that the grid lines are drawn both on the automatic tics and on the ones that you add.

In this example we've done something new: using the little-known ability of gnuplot to use Unicode symbols in variable names, we've make the script easier to read by setting "π" equal to pi, and using it in the following command setting the additional xtic values.

```
set xr [0 : 2*pi]
set grid lt 1 lc "grey" lw .5
π = pi
set xtics add\
   ("π" π, "π/2" π/2, "2π" 2*π, "3π/2" 3*π/2, "0" 0)
plot sin(x) lw 2
```

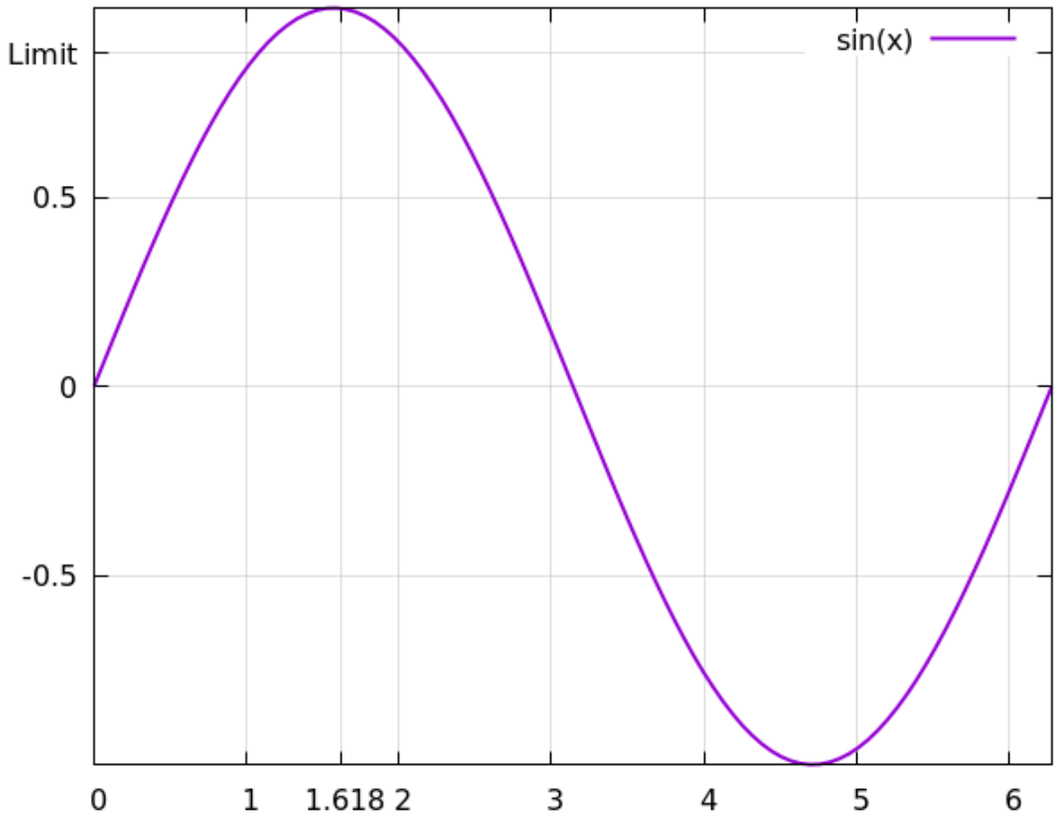
Open script



You can also set manual tics using `set xtics add` or just `set xtics` without using labels. In this case gnuplot will put the tics where you tell it to, and use its default numerical labels. You can mix and match these types of tics at will:

```
set xr [0 : 2*pi]
set grid lt 1 lc "grey" lw .5
set ytics (-.5, 0, .5, "Limit" .88)
set xtics add (1.618)
plot sin(x) lw 2
```

Open script

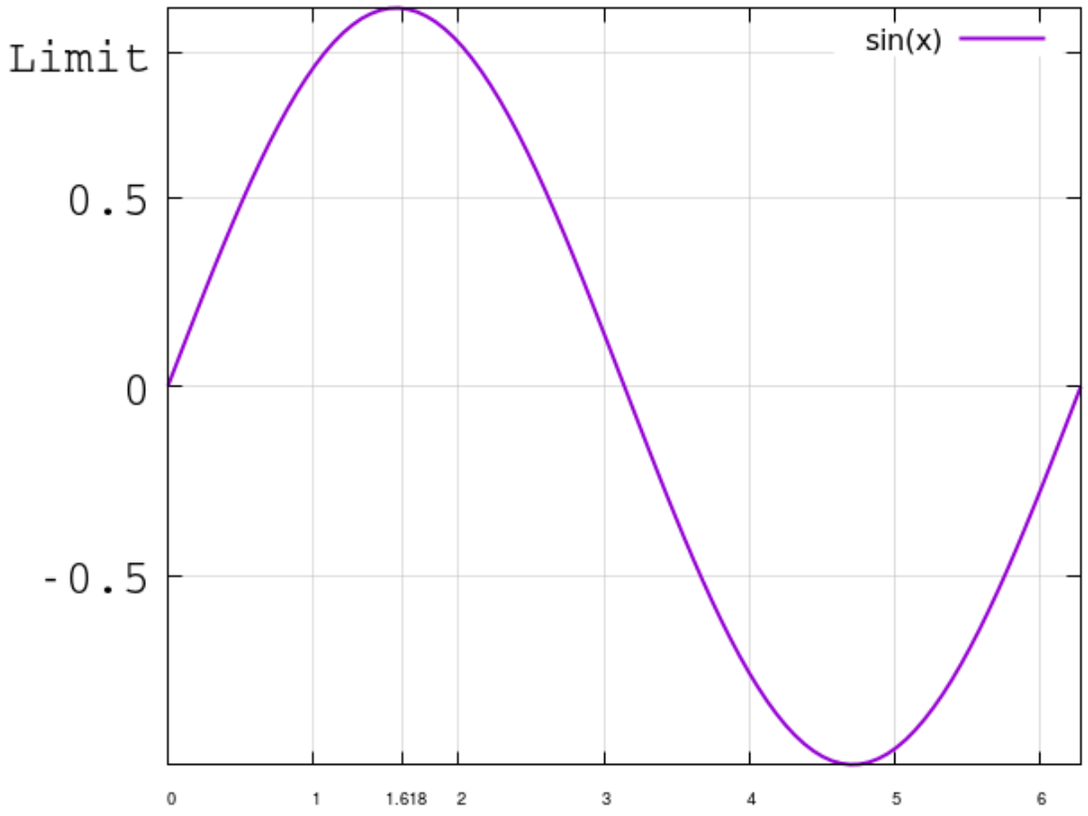


Formatting Tics

All of the options for string formatting that we learned about in Chapter 5 are available for formatting tics. You can use all the familiar font specifiers, too. We'll give an example of that first. When making custom tic labels using the manual command here, gnuplot does not take their width into account when setting plot margins. You will need to take measures to ensure that they fit. Here, we are obliged to increase the left margin.

```
set xr [0 : 2*pi]
set lmargin 10
set grid lt 1 lc "grey" lw .5
set ytics (-.5, 0, .5, "Limit" .88) font "Courier, 20"
set xtics add (1.618) font "Helvetica, 7"
plot sin(x) lw 2
```

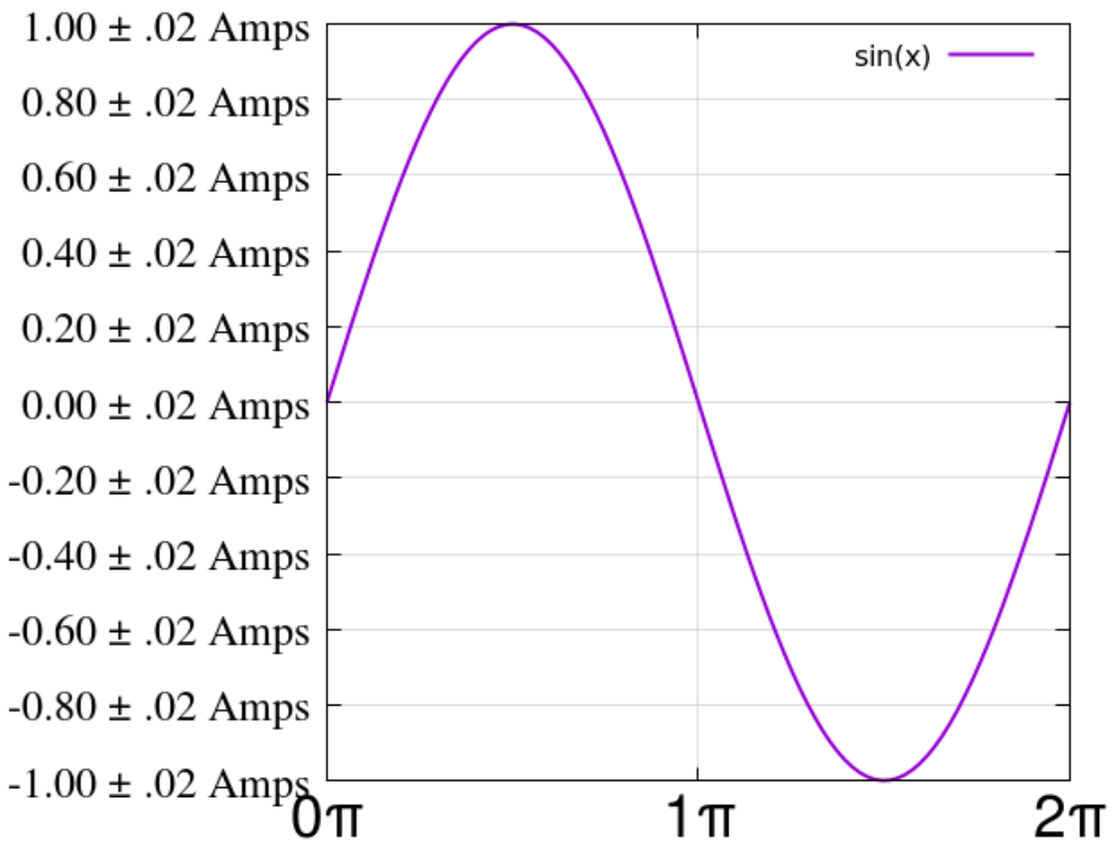
Open script



Here we'll use another of gnuplot's string formatting tricks to generate labels on the x-axis at multiples of π automatically. The version of the `set xtics` command in the third line sets the interval, and allows gnuplot to set the beginning and ending values. We'll also include some explanatory text, including an error estimate, within the tic labels on the y-axis. The commands for applying format specifiers to tic labels are `set format x`, `set format y`, etc. This example also shows how you can set the font of the tic labels separately from their other properties. Note that when using a format, as we do below, gnuplot has an idea of the width of the tic labels, and sets the margins appropriately itself.

```
set xr [0 : 2*pi]
set grid lt 1 lc "grey" lw .5
set xtics pi
set xtics font "Helvetica, 25"
set ytics font "Times, 18"
set format x "%.0Pπ"
set format y "%.2f ± .02 Amps"
plot sin(x) lw 2
```

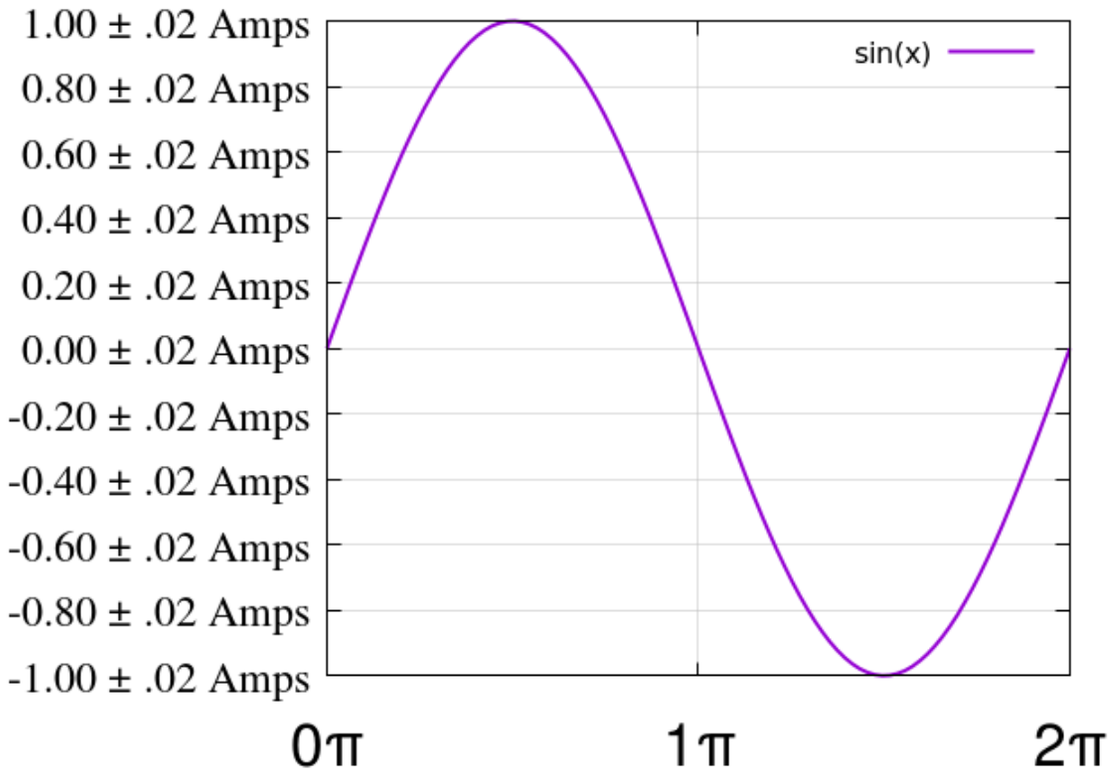
Open script



You might have noticed that, because of the large font sizes we chose for the tic labels in the previous example, the first x-axis tic collided somewhat with the lowest y-axis tic. We can set an offset to shift the tic label positions. It works just the way offsets for labels work.

```
set xr [0 : 2*pi]
set grid lt 1 lc "grey" lw .5
set xtics pi
set xtics offset 0, -1
set bmargin 5
set xtics font "Helvetica, 25"
set ytics font "Times, 18"
set format x "%.0Pπ"
set format y "%.2f ± .02 Amps"
plot sin(x) lw 2
```

Open script

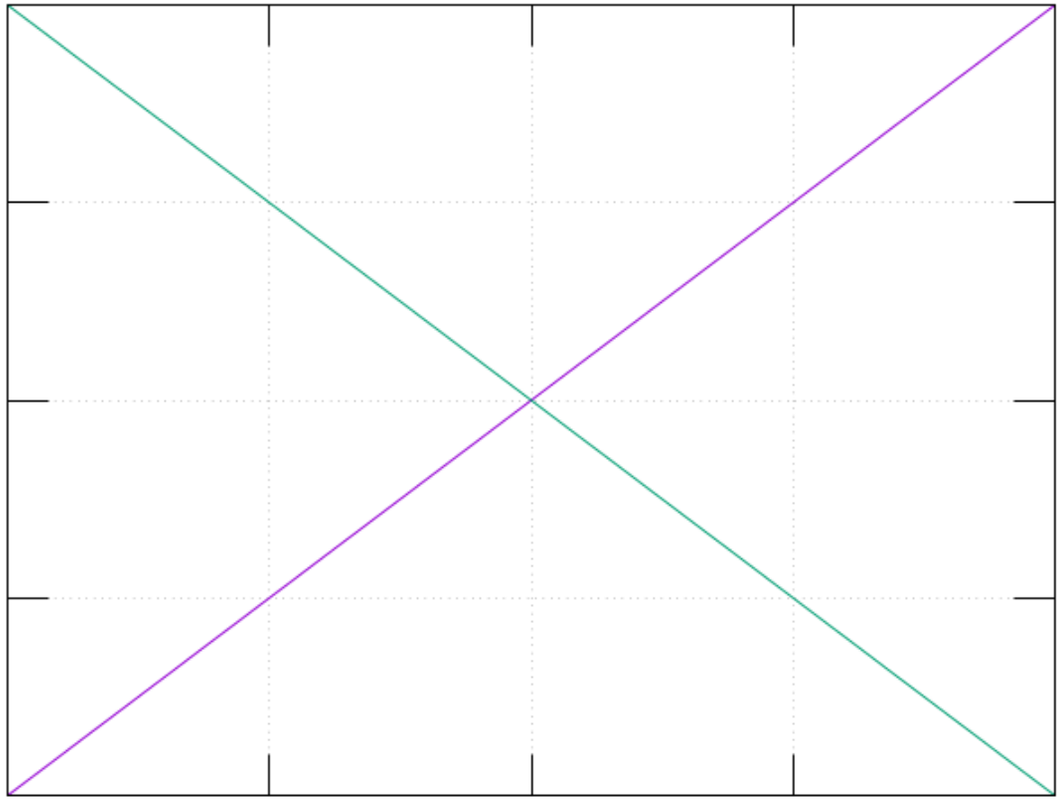


Tics With No Labels

You might want tic marks with no labels at all — for example, if you are adding labels outside of gnuplot. As we show in this example, you can accomplish this by simply setting an empty format. If, for some reason, you want labels with no tic marks, try `set tics scale 0`.

```
set format y ""
set format x ""
unset key
set tics scale 3
set grid
plot x, -x
```

Open script



Dates and Times

Gnuplot has extensive support for plotting using dates and times. There are two components to this: telling gnuplot about the data/time format that your data is stored in, and telling it how to represent data/time data in the plot.

If you are plotting date/time data, you are almost certainly dealing with files. Here is a small file for experimenting with date/time plotting. It consists of a few lines of data, each line containing a date, time, and a number to be plotted. The dates are in the form day/month/2-digit-year, and the times are in the form hour:minute. You copy the lines directly, use the "Open file" button, or make your own; in any case, you will need to save a file called "timedat.dat" in the directory in which you are running gnuplot to use the scripts in this section as written, and it should have this format, although, of course, the values can be different.

The Example File

```
1/1/17 19:00 72.01
12/3/17 06:15 12.03
3/6/17 13:13 9.23
7/12/17 17:14 72.09
23/2/18 09:15 66.06
29/7/18 14:13 55.55
9/9/18 22:19 63.12
```

Open file

Defining the Input Format

Before gnuplot can interpret your data as dates and/or times, you must tell it to expect date/time data with the command `set xdata time`. Then, you can tell it the format in which you have stored the data. The command for that is `set timefmt f`, where "f" is the date/time format you are using. These formats use special codes: `%d` to stand for the numerical day, `%H` to stand for the numerical hour using a 24-hour clock, and so on. For a summary of all the formats that gnuplot understands, type `help timefmt` at the interactive prompt. For our example data, the format should be `%d/%m/%y`

`%H:%M`. This tells gnuplot about the single space between the date and time, the colon between the hours and minutes, etc. The format must be complete and exact. Only then will the program know when the date/time input ends and the following data columns begin.

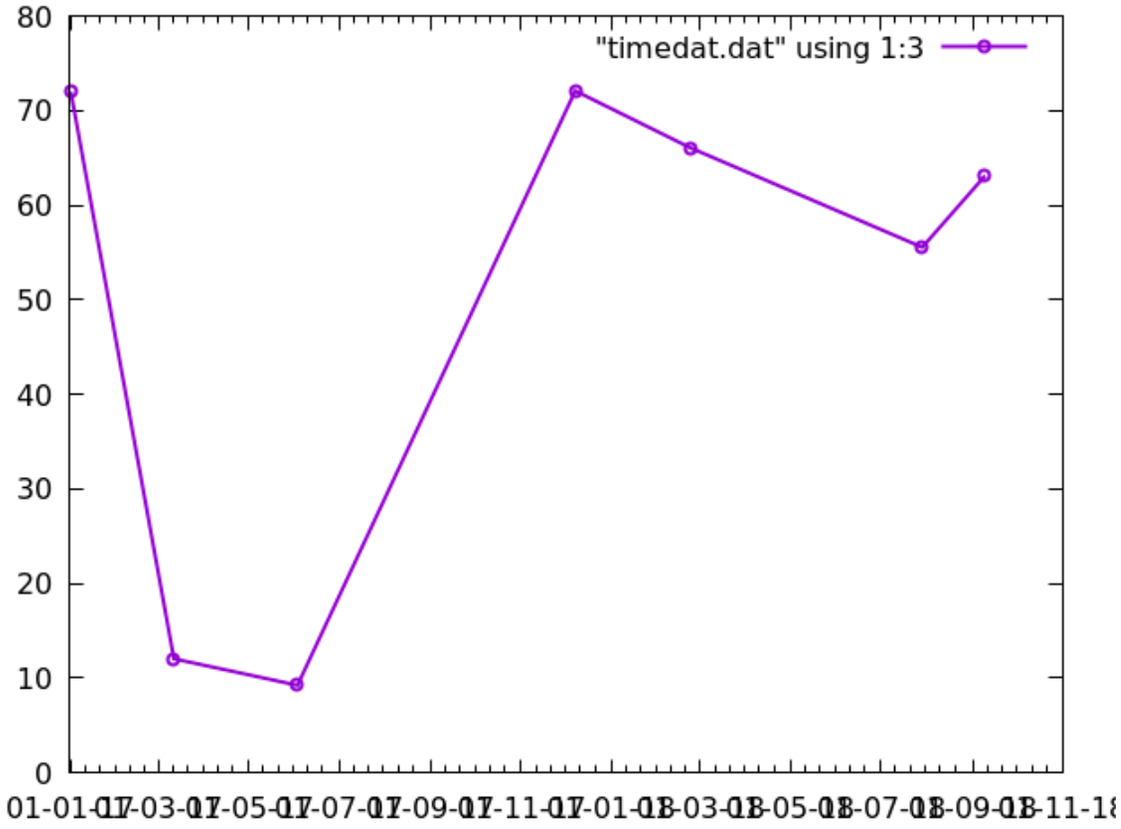
Defining the Output Format

The output format that gnuplot will use for printing dates and times is completely independent of the input format that you tell it to expect in your data. You can print any portion of the data (the time only, the year only, etc.) in any format. The format specifier uses the same codes available for the input specifier. If you've given the command `set xdata time`, gnuplot will expect the x-axis (in this case) to be formatted as a date/time. Use the same command that we used to format normal tic labels, `set format x`, but using the date/time formatting symbols (again, try `help timefmt` to see the complete set). Let's plot our little file with different output formats to see how this works.

This script will plot the dates only, using the format “day-month-year”. One quirk that you must remember: you are required to include a `using` command when plotting date/time data, and you must take into account the columns used in the date/time portion. Since we’re using two columns for the x-data, the first containing the date and the second the time, the actual data starts in column three.

```
set xdata time
set timefmt "%d/%m/%y %H:%M"
set format x "%d-%m-%y"
plot "timedat.dat" using 1:3 with linespoints lw 2 pt 6
```

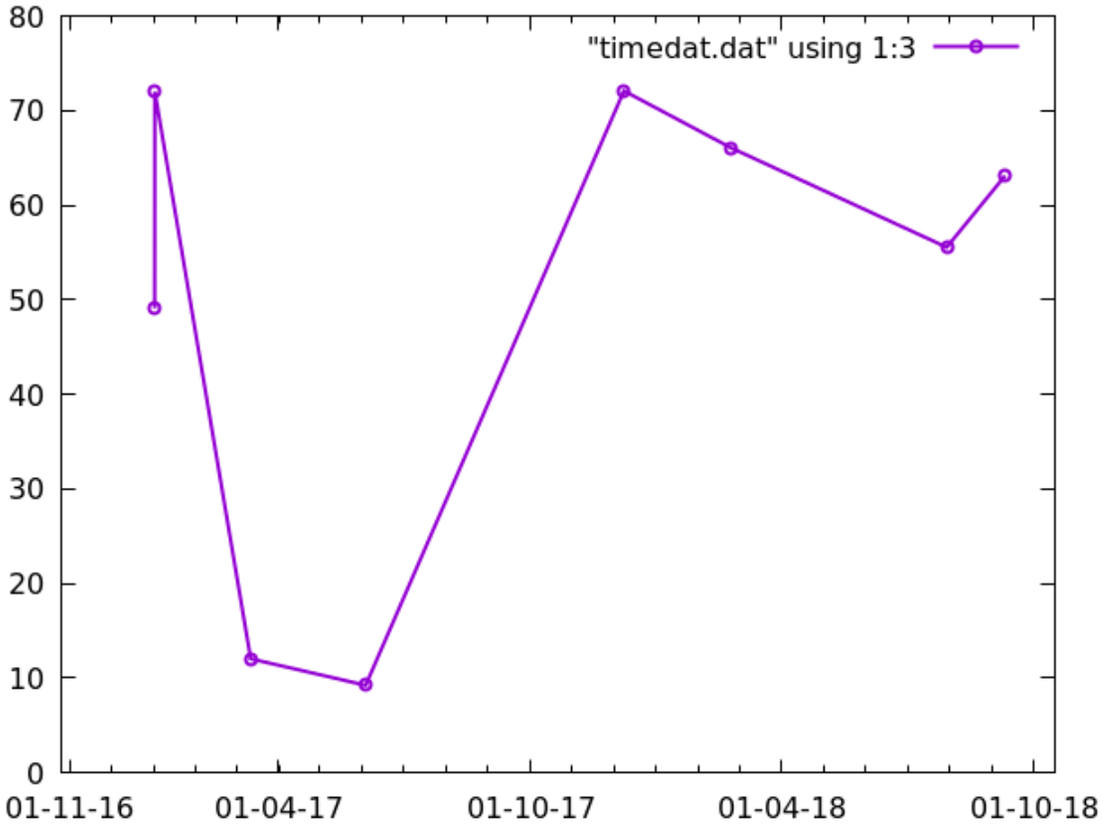
Open script



You might have noticed a problem with the previous graph. There are too many tic labels, and they overlap. We could make the font smaller, but we might prefer to simply plot fewer tics. Previously, we accomplished such things by adjusting the tic interval — but how so we do that when the tics represent temporal data? You can do it the same way, using a number of seconds as the interval:

```
set xdata time
set timefmt "%d/%m/%y %H:%M"
set format x "%d-%m-%y"
hour = 60*60
day = 24*hour
set xtics 180*day
plot "timedat.dat" using 1:3 with linespoints lw 2 pt 6
```

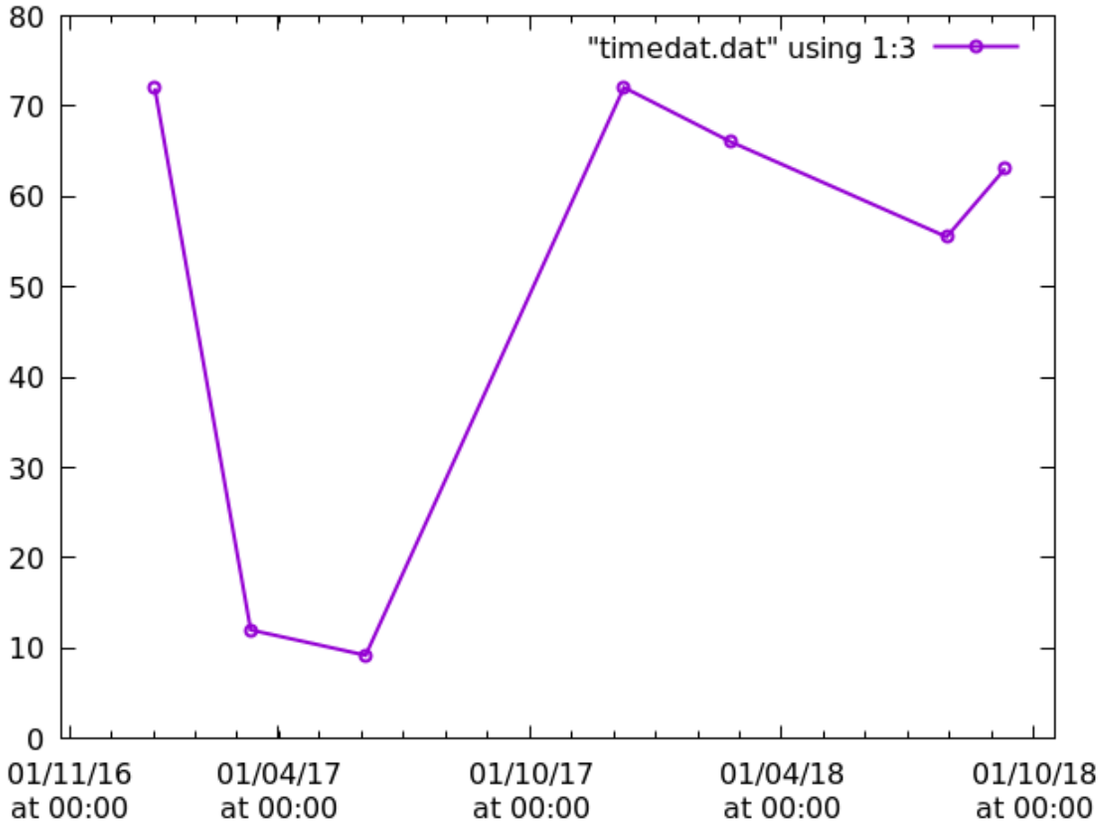
Open script



As you add more information, temporal tic labels have a tendency to take up a lot of space, requiring a strategy to get it all to fit. Below we use a format with the time set below each date. The escape code “\n” in the format specification produces a new line in the label, which puts the time of day below the date and saves on horizontal space.

```
set xdata time
set timefmt "%d/%m/%y %H:%M"
set format x "%d/%m/%y\nat %H:%M"
hour = 60*60
day = 24*hour
set xtics 180*day
plot "timedat.dat" using 1:3 with linespoints lw 2 pt 6
```

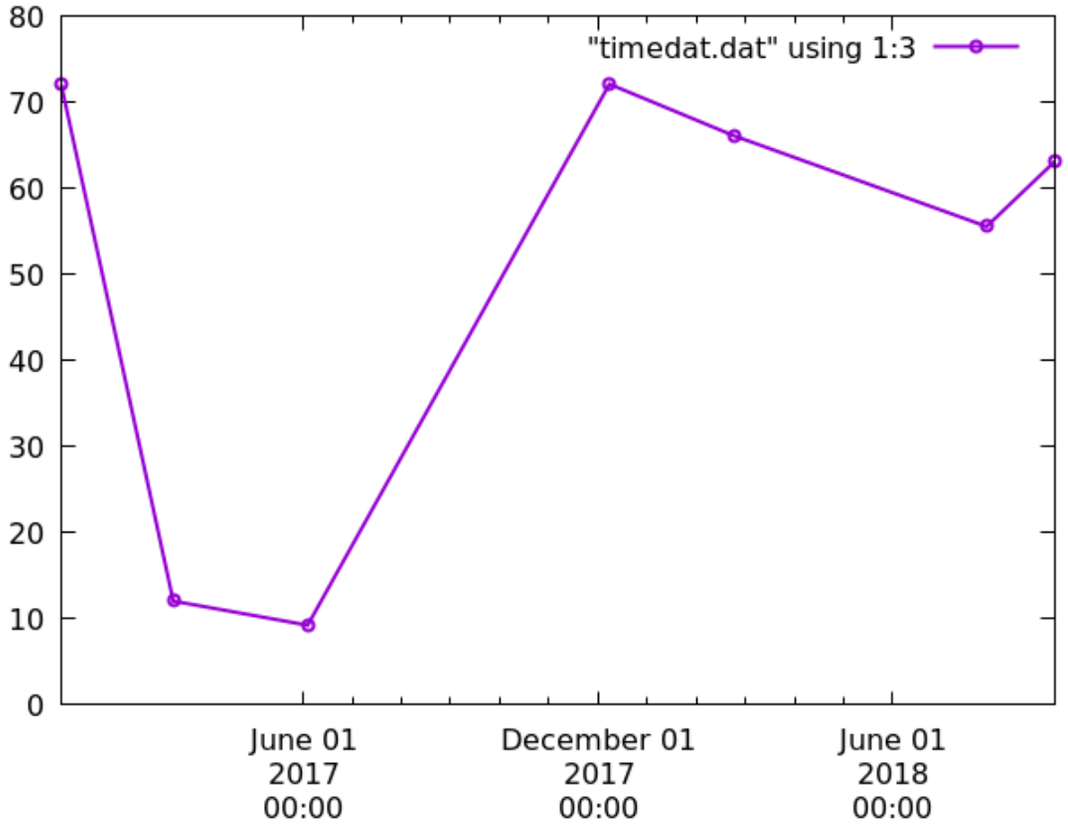
Open script



Another example of date/time formatting might be instructive. Here we also show how to set beginning and ending values for tic ranges. Once the `timefmt` is set, you can use strings in that input format to define these values:

```
set xdata time
set timefmt "%d/%m/%y %H:%M"
set format x "%B %d\n%Y\n%H:%M"
hour = 60*60
day = 24*hour
set xtics "1/6/17", 180*day, "1/6/18"
plot "timedat.dat" using 1:3 with linespoints lw 2 pt 6
```

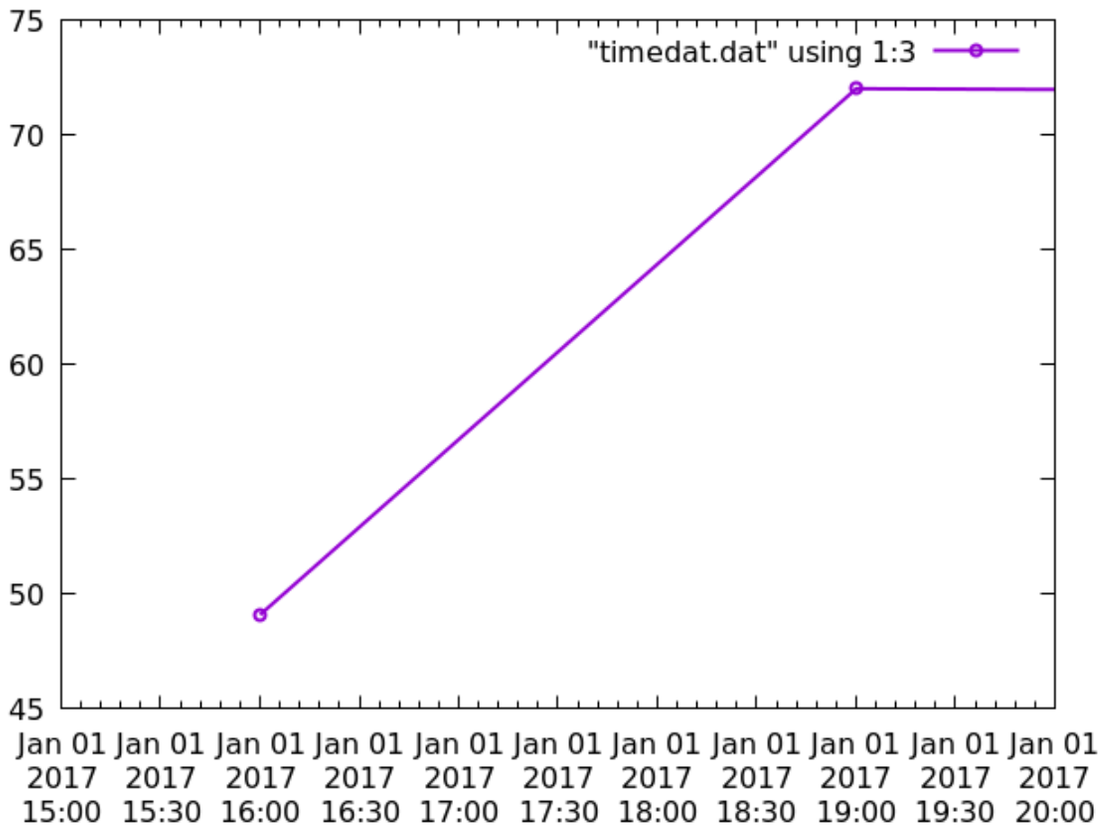
Open script



As you can see, gnuplot plots each date at time 00:00 (midnight), so, for long ranges of days such as these, we might as well leave the times out of the output format. If we restrict the date range, however, the time of day becomes more relevant:

```
set xdata time
set timefmt "%d/%m/%y %H:%M"
set format x "%b %d\n%Y\n%H:%M"
hour = 60*60
set xr ["1/1/17 15:00" : "1/1/17 20:00"]
set xtics 0.5 * hour
plot "timedat.dat" using 1:3 with linespoints lw 2 pt 6
```

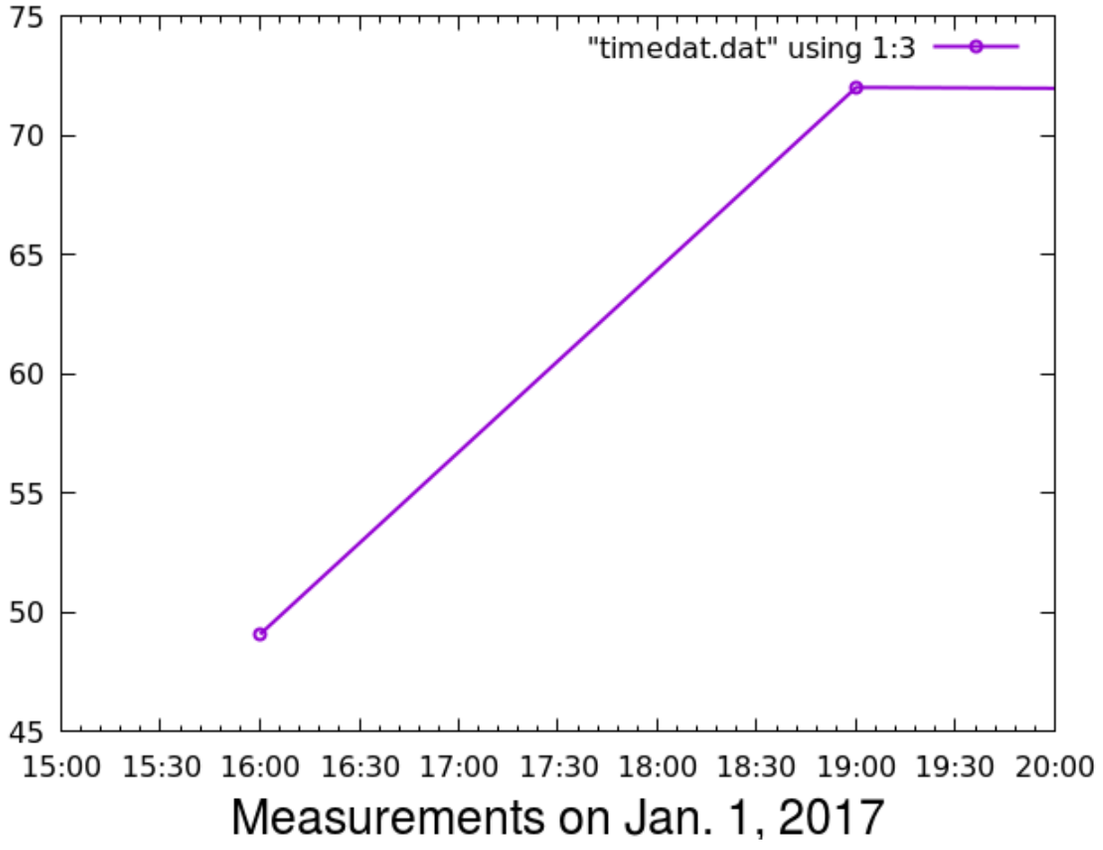
Open script



The previous plot would be more attractive if we extracted the redundant date information into an axis label:

```
set xdata time
set timefmt "%d/%m/%y %H:%M"
set xlab "Measurements on Jan. 1, 2017" font "Helvetica, 20
set format x "%H:%M"
hour = 60*60
set xr ["1/1/17 15:00" : "1/1/17 20:00"]
set xtics 0.5 * hour
plot "timedat.dat" using 1:3 with linespoints lw 2 pt 6
```

Open script



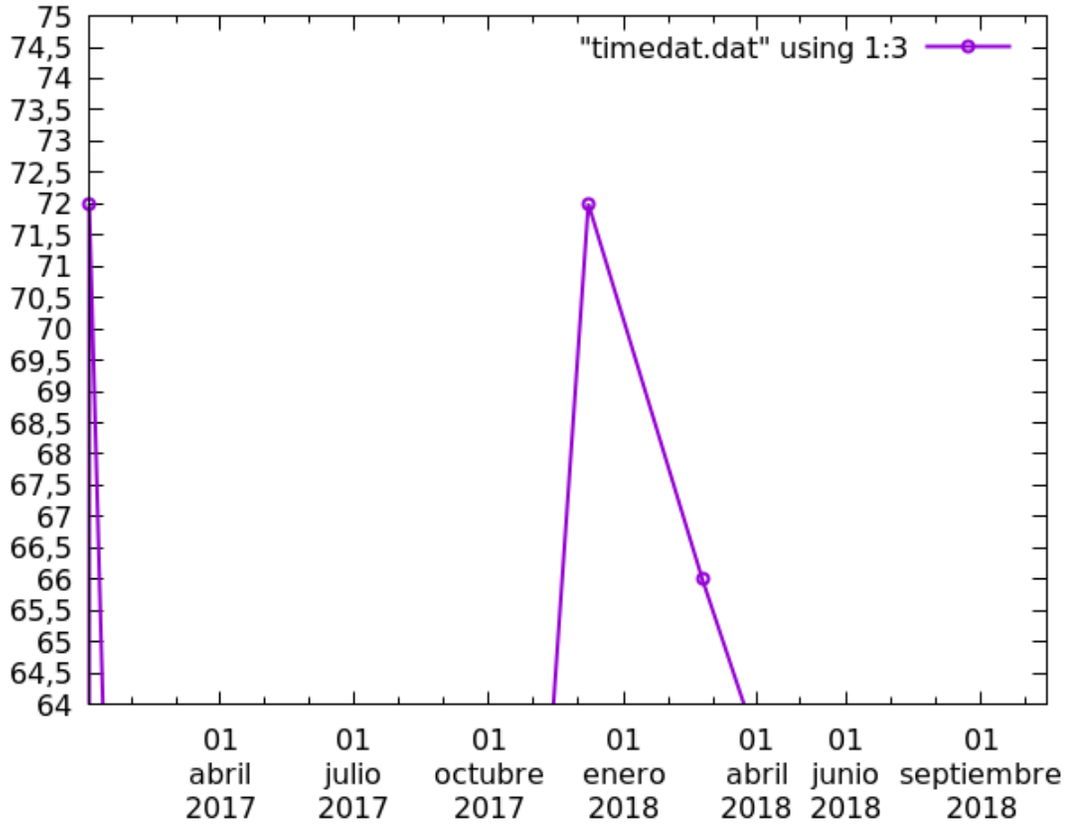
Internationalization of Dates

Gnuplot knows the names of the months in several languages. The languages available, and how to designate them, depend on your system. This example should work with any Linux and some other unix-derived systems. It prints the names of the months in Spanish. We also set another flag that uses commas rather than dots as decimal points, as is the standard in many European countries. This affects the formatting of the numbers on the y-axis.

If you get an error about a missing locale, you can check (on Linux, etc.) for available locales with the shell command `locale -a`. If you want to install the Spanish locale, try, as root, `locale-gen es_ES` followed by `update-locale`.

```
set xdata time
set locale "es_ES"
set decimalsign locale "es_ES"
set timefmt "%d/%m/%y %H:%M"
set format x "%d\n%B\n%Y"
hour = 60*60
day = 24*hour
set xtics "01/01/17", 90 * day
set yr [64 : 75]
set ytics 0.5
plot "timedat.dat" using 1:3 with linespoints lw 2 pt 6
```

Open script

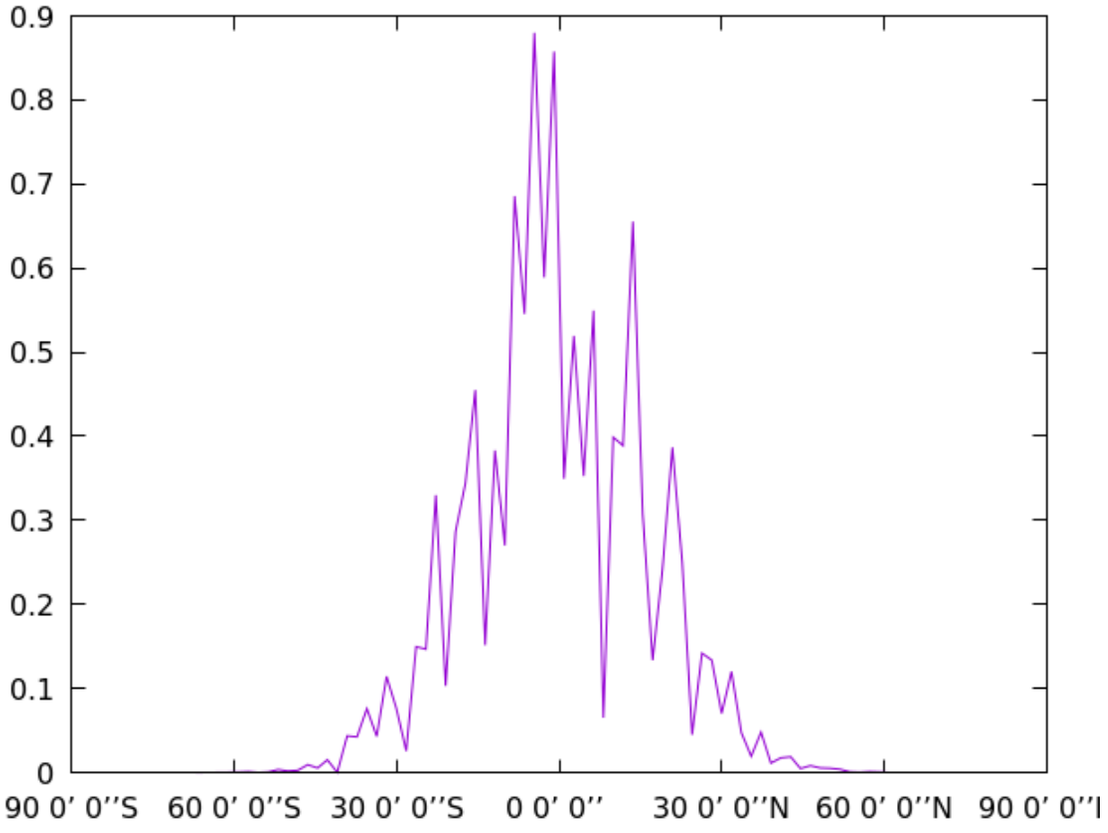


Geographic Coordinates

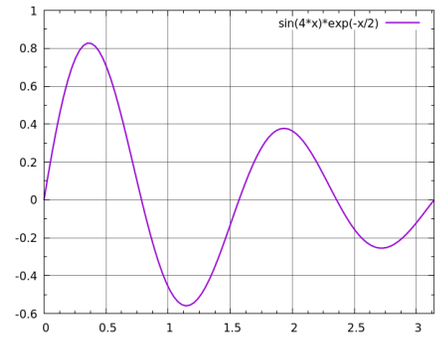
If you `set xtics geographic`, gnuplot can convert coordinate values into degrees, minutes, and seconds, including translating into E and W longitudes or N and S latitudes. Formatting uses a few special codes (type `help geo` for a list): `D` for integer degrees, `M` for minutes, `S` for seconds, and `N` to get "N" and "S" rather than plus or minus, or `E` to get E and W. This is useful when drawing maps or plotting measurements taken at various places on the surface of a planet. Here's how it works:

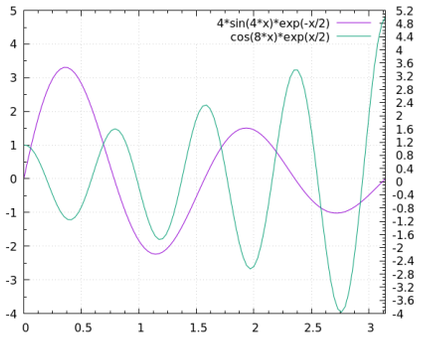
```
set xtics geographic
set xtics format "%D %M' %S''%N"
set xr [-90:90]
set xtics 30
plot rand(0)*exp(-x**2/500) notitle
```

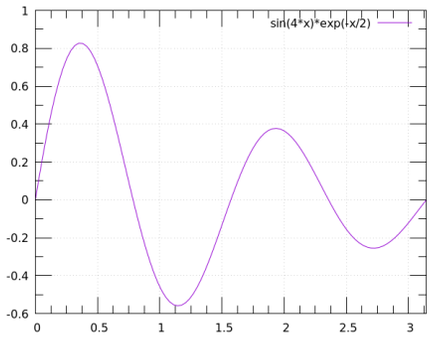
Open script

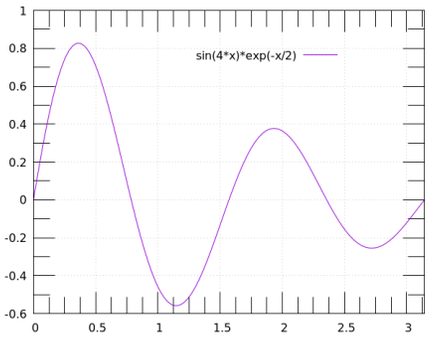


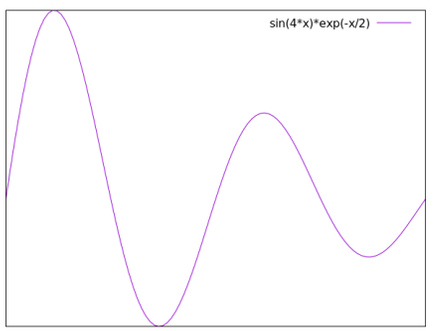
Index of Plots

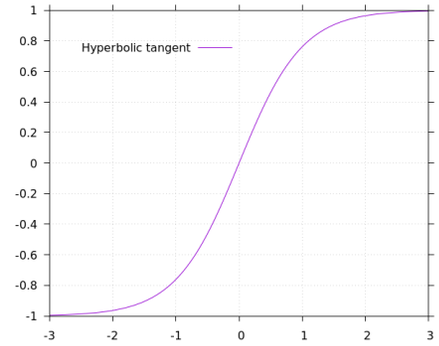


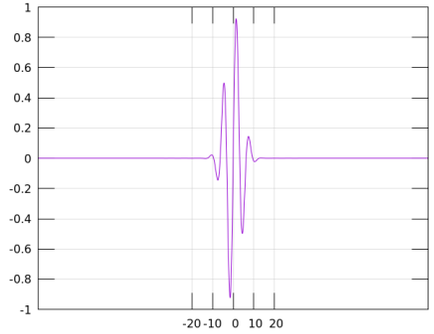


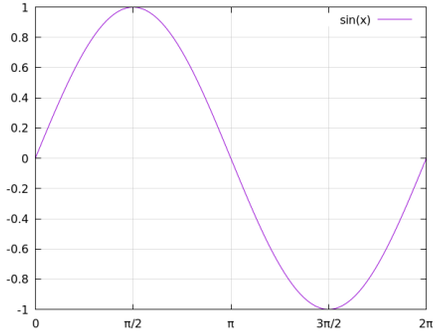


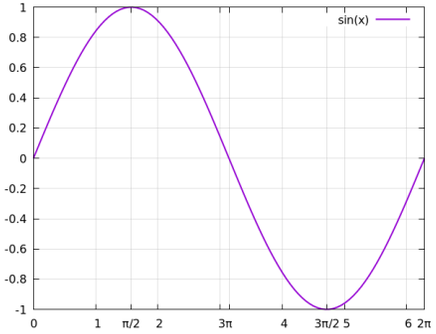


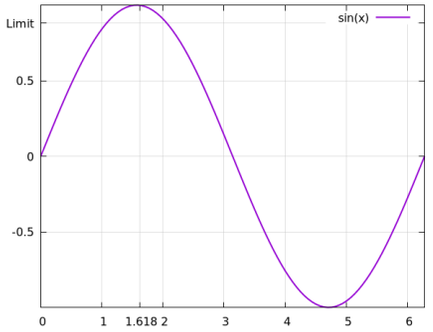


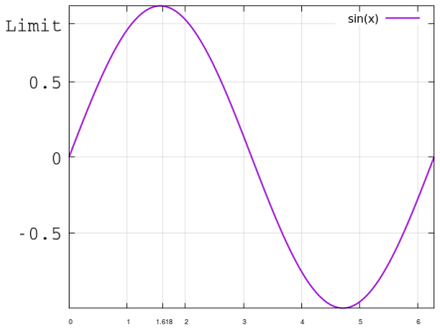


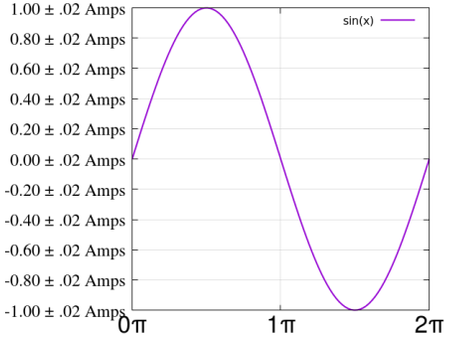


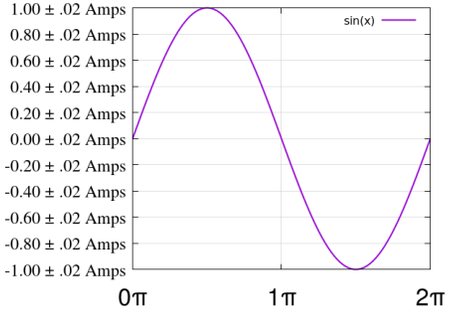


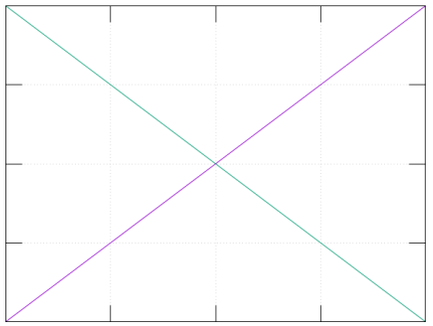


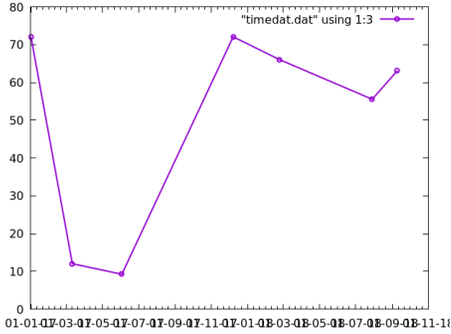


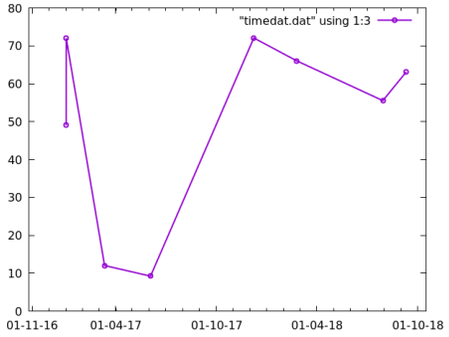


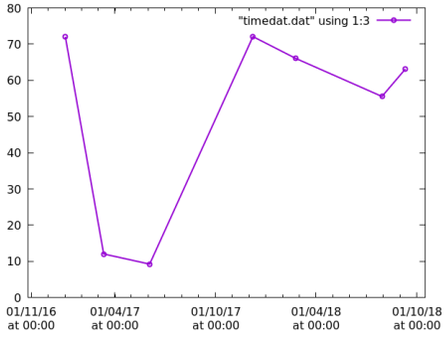


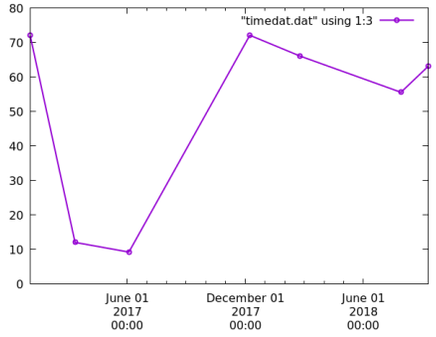


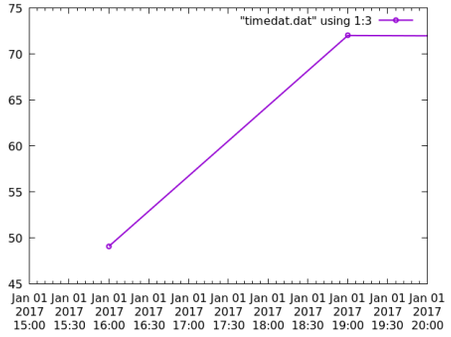


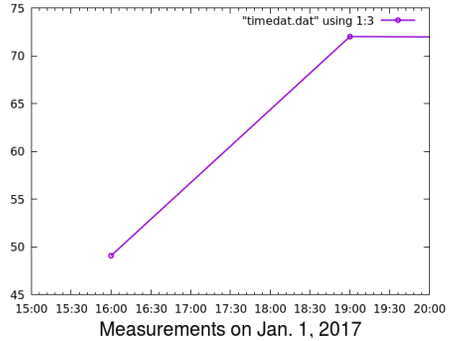


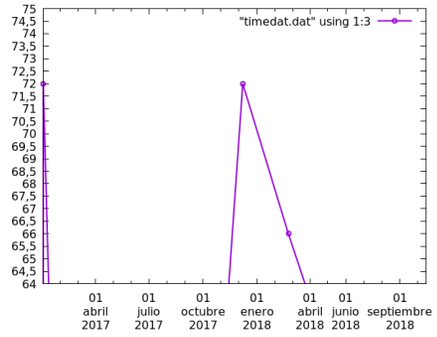

"timedat.dat" using 1:3
80
70
60
50
40
30
20
10
0
01-11-16
01-04-17
01-10-17
01-04-18
01-10-18

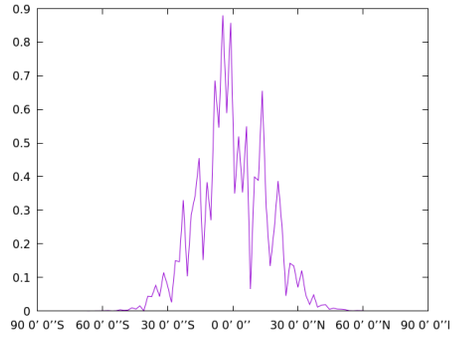

"timedat.dat" using 1:3
80
70
60
50
40
30
20
10
0
01/11/16
at 00:00
01/04/17
at 00:00
01/10/17
at 00:00
01/04/18
at 00:00
01/10/18
at 00:00


"timedat.dat" using 1:3
80
70
60
50
40
30
20
10
0
June 01
2017
00:00
December 01
2017
00:00
June 01
2018
00:00


"timedat.dat" using 1:3
75
70
65
60
55
50
45
Jan 01 2017 15:00
Jan 01 2017 15:30
Jan 01 2017 16:00
Jan 01 2017 16:30
Jan 01 2017 17:00
Jan 01 2017 17:30
Jan 01 2017 18:00
Jan 01 2017 18:30
Jan 01 2017 19:00
Jan 01 2017 19:30
Jan 01 2017 20:00


"timedat.dat" using 1:3
75
70
65
60
55
50
45
15:00
15:30
16:00
16:30
17:00
17:30
18:00
18:30
19:00
19:30
20:00
Measurements on Jan. 1, 2017


"timedat.dat" using 1:3
75
74,5
74
73,5
73
72,5
72
71,5
71
70,5
70
69,5
69
68,5
68
67,5
67
66,5
66
65,5
65
64,5
64
01 abril 2017
01 julio 2017
01 octubre 2017
01 enero 2018
01 abril 2018
01 junio 2018
01 septiembre 2018


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
90 0′ 0″S
60 0′ 0″S
30 0′ 0″S
0 0′ 0″
30 0′ 0″N
60 0′ 0″N
90 0′ 0″N

Index