Chapter 2: Errors and Finance

This chapter will show you how to plot values with errors or ranges. The examples in this chapter all use a data file rather than mathematical functions. You should download the file, called "statdata," which should be available in the sample place where you downloaded this book. Alternatively, you may of course use your own data, if you have some that you'd like to work with. This might be more interesting for you; however, the examples here assume that the data columns are in a particular order. We'll be clear about what that order is, so you can either rearrange the format of your files or make small alterations to the example scripts.

Most of the graphs in this chapter will appear to be similar to the basic 2D graphs of the previous chapter, with some extra marks added to the data points. These marks convey additional numbers associated with each value of the independent variable (usually, the x-axis). There can be as many as four additional numbers associated with each point. This would, in a sense, be a six-dimensional graph. In most of our examples, these additional numbers represent the estimated error in a measurement,

or a possible range of values; this plot style is commonly seen in scientific publications. We include financial plots in this chapter as well, because they use similar concepts to display a range of values, and sometimes use the same graphical conventions.

The Data File

We'll use the same file of data for all the examples in this chapter. This file, called "statdata," contains 10 lines. The data represent the function $y = x$, with "random" errors added to both the x and y values. Errors in the positive and negative direction are included separately. Each line of the file therefore contains six numbers, representing the following values:

$$x \qquad y \qquad x^- \qquad x^+ \qquad y^- \qquad y^+$$

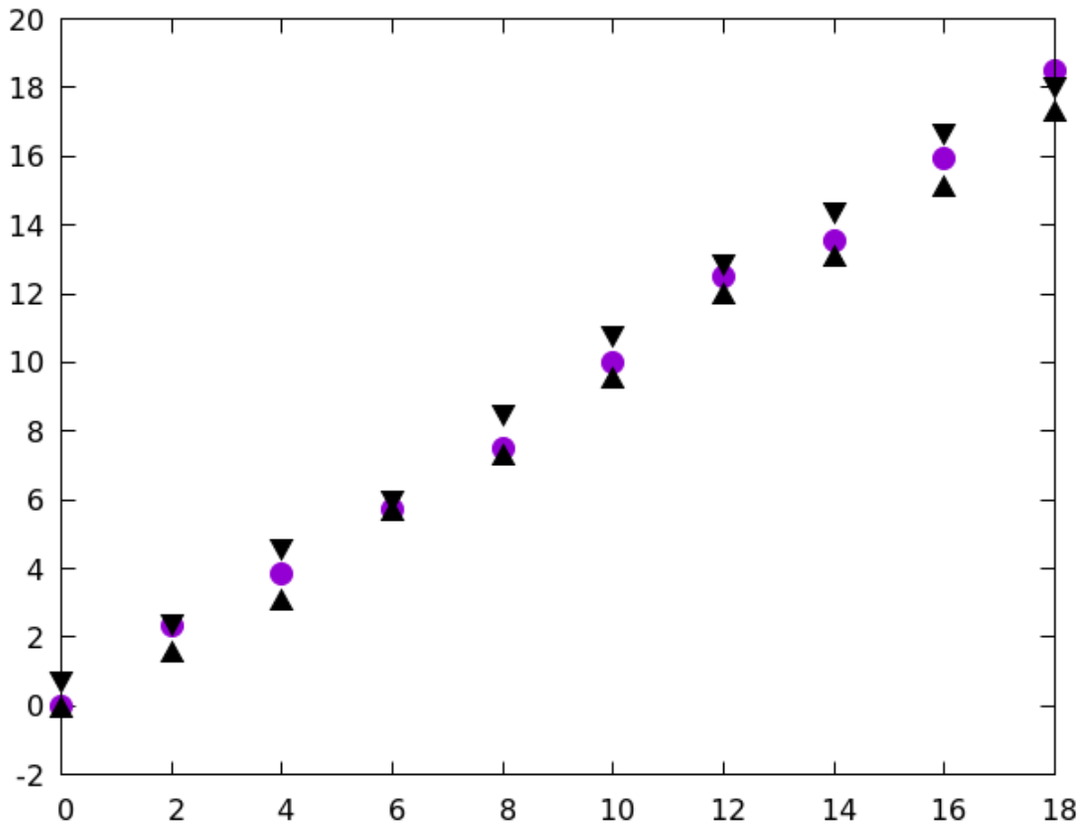
Here y^+ means the value of y plus the estimated error in the positive direction, y^- means the value of y with the estimated error in the negative direction subtracted, etc. (so the positive error itself would be $y^+ - y$, for example). The "errors" are all generated randomly; they are not simply departures from the "theoretical" $y = x$ line. It is more usual in experimental scenarios to have a simple Δy and/or Δx, representing the estimated total error in the measurements, but sometimes we have separate error estimates on the positive and negative side; and since gnuplot can handle these, we include the general case in our data file.

Column Selection

Many of the examples in this chapter use only a few of the six columns of data in the `statdata` file. Also, we will sometimes need to perform some simple arithmetic on the data before handing it off to gnuplot for plotting. All of this can be done simply and easily with the gnuplot `using` keyword. This can be abbreviated to `u`. The `using` command is very important, so we'll introduce it with several examples. The first example shows how to select columns from a data file. We use the `u` abbreviation for `using`; the phrase `u 1:3` tells gnuplot to use the first and third columns, etc. So this script makes three normal, 2D plots, plotting the second, third, and fourth column against the first. The default plot style when plotting from a data file is "points", rather than "lines," which is the default when plotting functions. The script uses the pointtypes (`pt`) 9 and 8 to select triangles for column 9 and 8: this is meant to suggest a range of y values. The script shows one way to plot a set of values and their ranges using the normal plotting commands that we've already learned; many other approaches are possible, limited only by your creativity.

We've illustrated a useful shorthand notation in the script. The empty string (`""`) means the previously mentioned data file: in this case, we are telling gnuplot to plot from `statdata` three times (it starts from the beginning of the file for each plot).

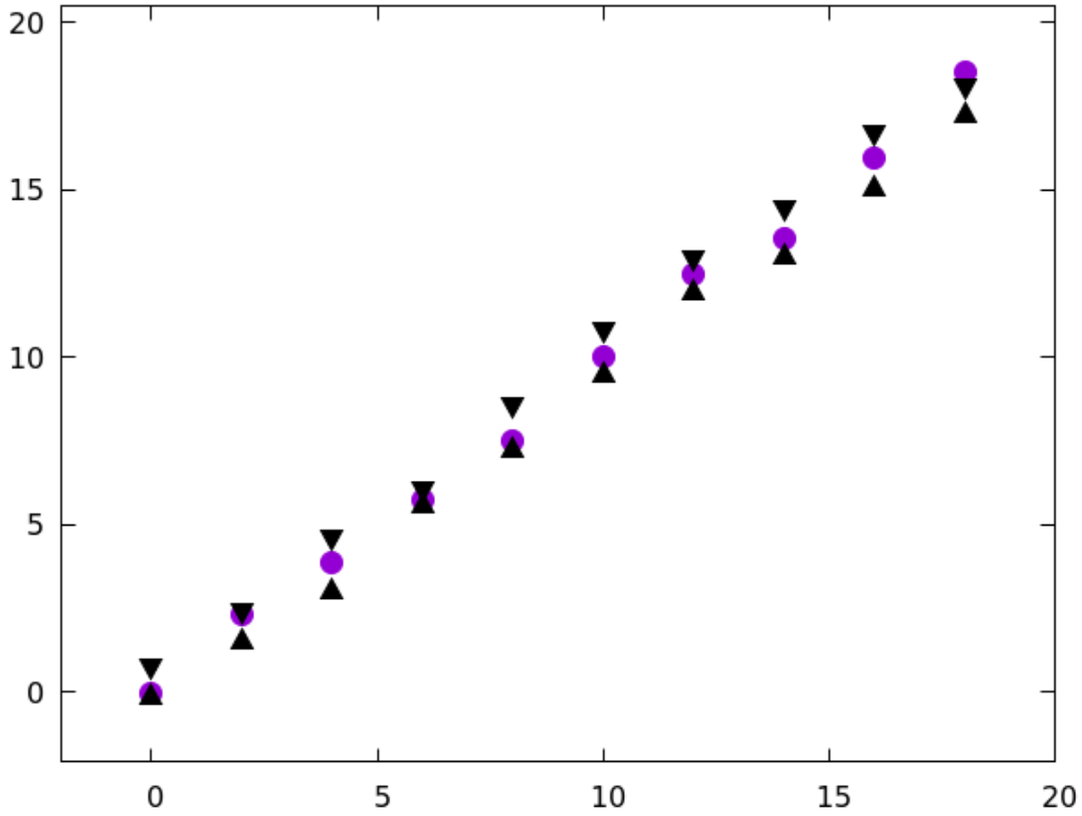
```
unset key
plot "statdata" u 1:2 ps 2 pt 7,\
   "" u 1:3 ps 2 pt 9 lc 8,\
   "" u 1:4 ps 2 lc 8 pt 11
```



Offsets

You might have noticed that the graph in the previous example was a bit crowded. This is because gnuplot set the axis ranges to fit the range of the data, which puts the plot symbols up against the border box. If you want some breathing room, you could manually set the `xrange` and/or the `yrange` to be something larger, but gnuplot has another, slightly more convenient way to do this. As shown below, the `set offset` command expands the axis ranges in the order *left, right, bottom, top*. If you use this, you usually want to use it with the `set auto fix` command, as shown. This prevents gnuplot from extending the range to include the next tic mark when the data values fall between tics.

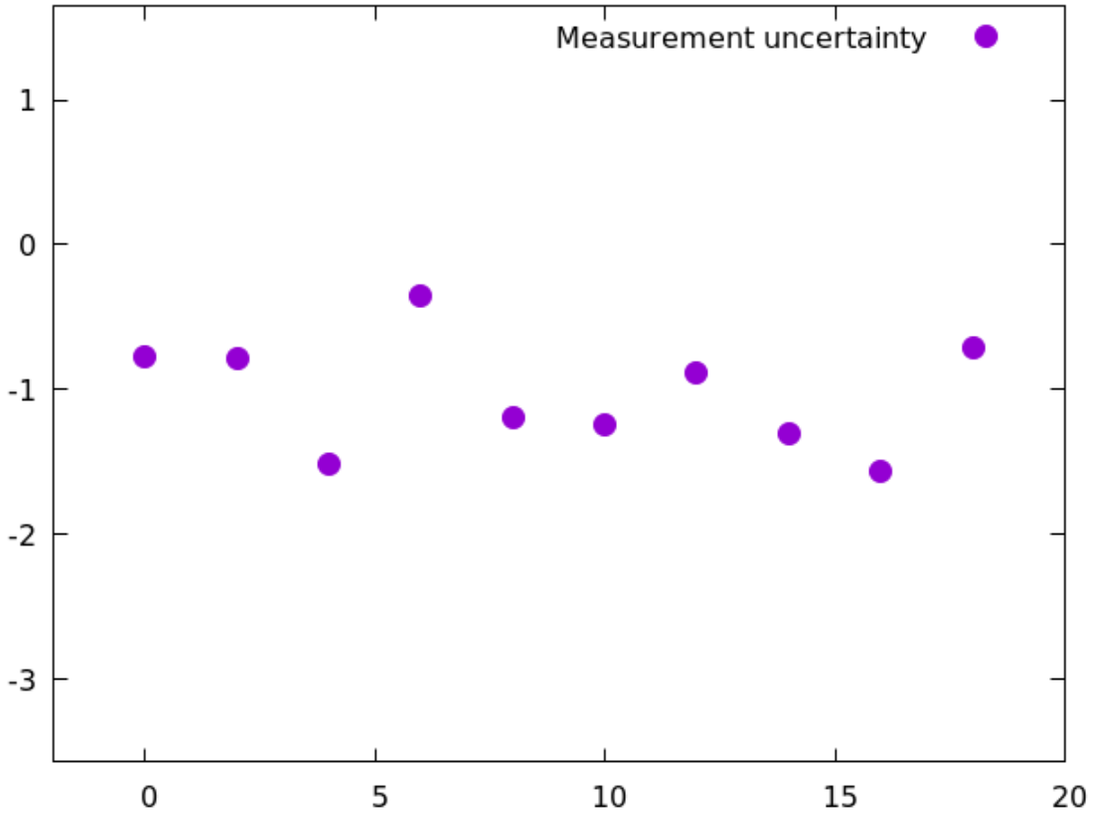
```
unset key
set auto fix
set offsets 2,2,2,2
plot "statdata" u 1:2 ps 2 pt 7,\
   "" u 1:3 ps 2 pt 9 lc 8,\
   "" u 1:4 ps 2 lc 8 pt 11
```



Calculating with Columns

This next example shows how you can calculate with the `using` keyword. Suppose we wanted to plot the numerical range of values of each data point, rather than the data limits, as we did in the previous example. The range is $y^+ - y^-$, which, because of the way our file is organized (see the introduction to this chapter), amounts to column 3 - column 4. Arithmetic is performed with the `using` keyword, inside round brackets. Within an arithmetic expression, columns must be prefixed by a dollar sign, to distinguish them from simple numbers. In the example, the first column for the plot is the first column in the file; the second column for the plot is the result of the expression after the colon.

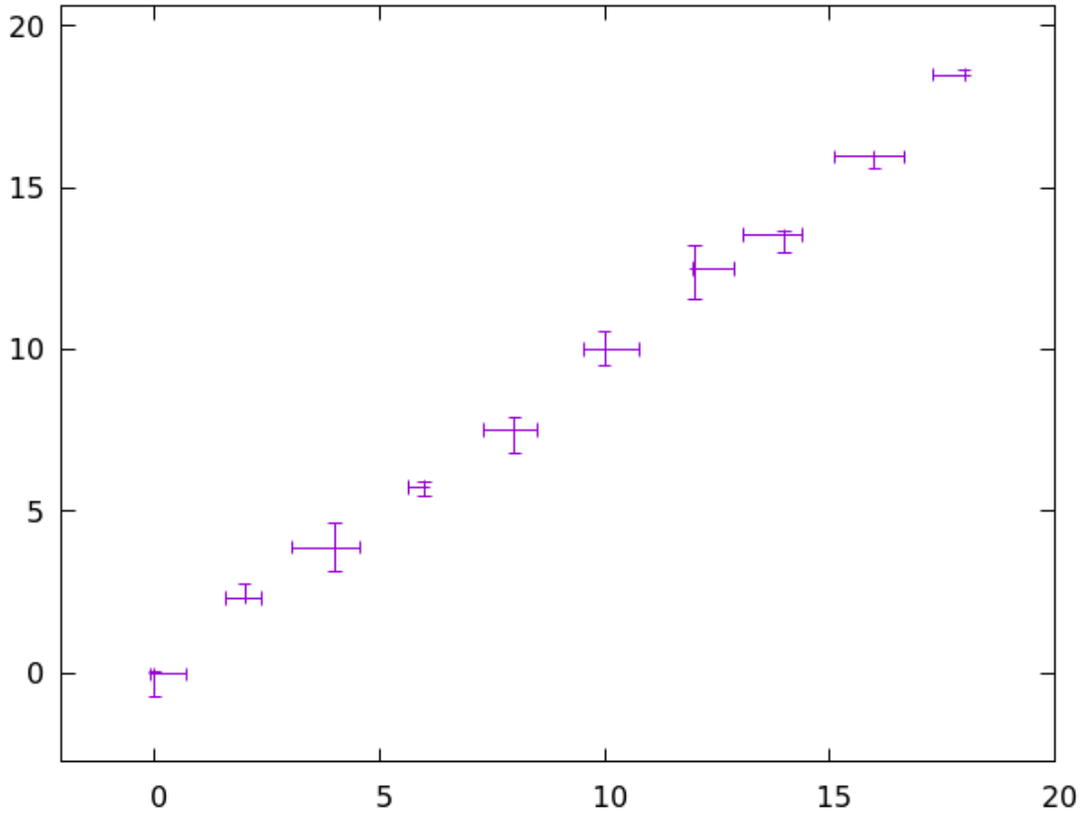
```
set auto fix
set offsets 2,2,2,2
plot "statdata" u 1:($3 - $4) ps 2 pt 7\
   title "Measurement uncertainty"
```



Errorbars

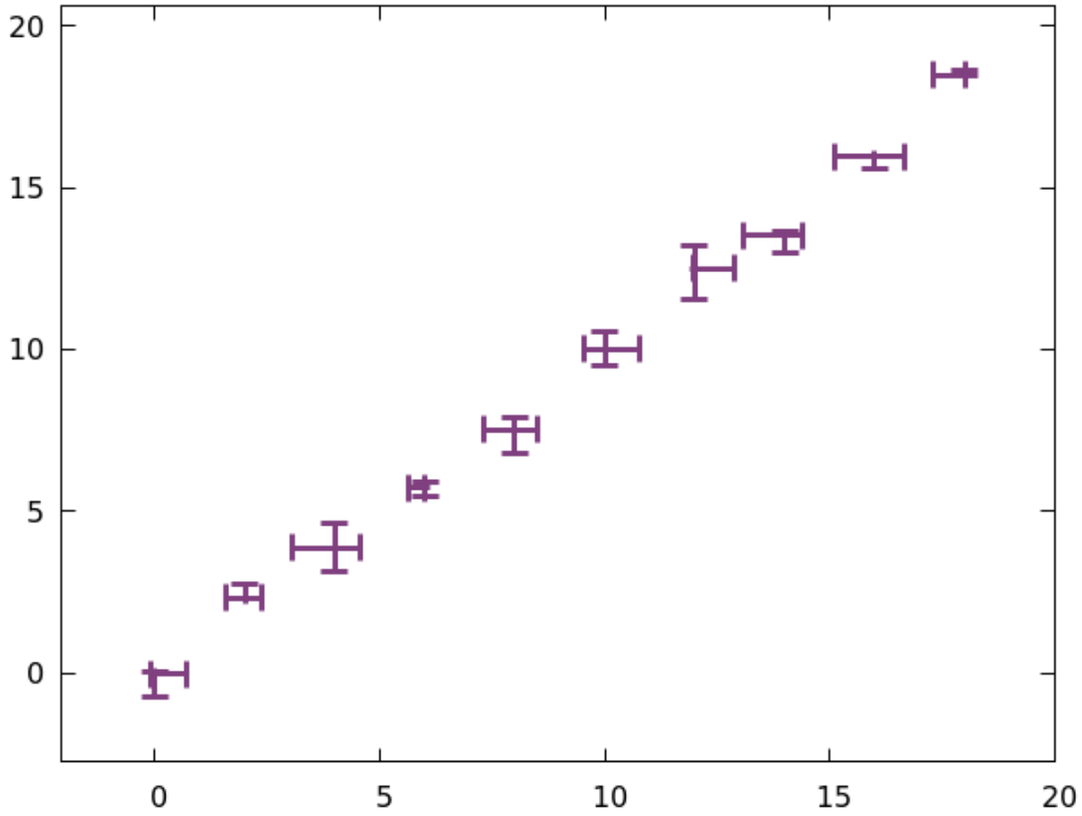
The gnuplot style `xyerrorbars` plots data ranges in the `x` and `y` directions assuming that the data file is organized the way we've set up `statdata`. This style plots line segments covering the range of data and centered on the data values. The ends of the line segments are marked with little perpendicular lines.

```
unset key
set auto fix
set offsets 2,2,2,2
plot "statdata"  with xyerrorbars
```



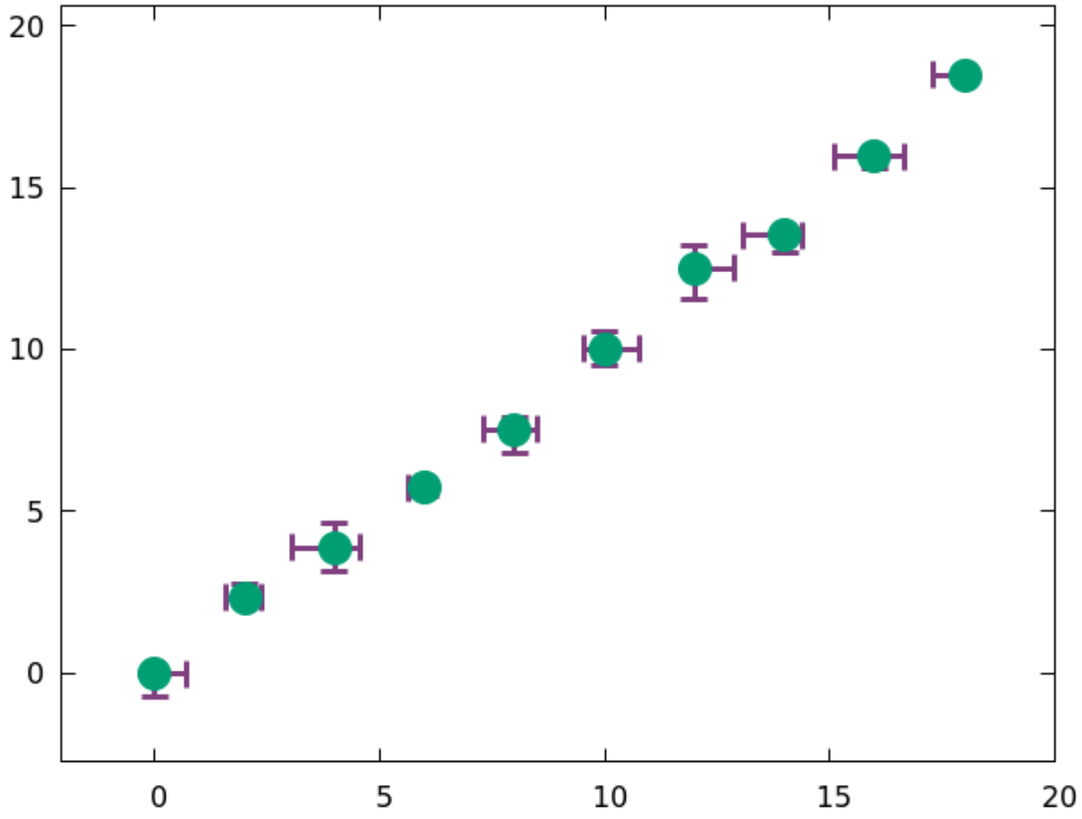
As you can see, the errorbars are drawn quite thin. You will usually want to adjust this style. This is done with the `set errorbars` command, which takes an additional pure numeric argument in addition to the usual arguments for setting thickness, color, and linetype. This extra argument sets the length of the end caps, in arbitrary units; experiment to get the effect that you want. There is either a bug or an odd feature in this setting: the `linewidth` specification has no effect unless the `linetype` is also set. In our example we've overridden the color associated with `lt 1` with a color specification.

```
unset key
set auto fix
set offsets 2,2,2,2
set errorbars lw 3 lt 1 lc rgbcolor("orchid4") 2
plot "statdata"  with xyerrorbars
```



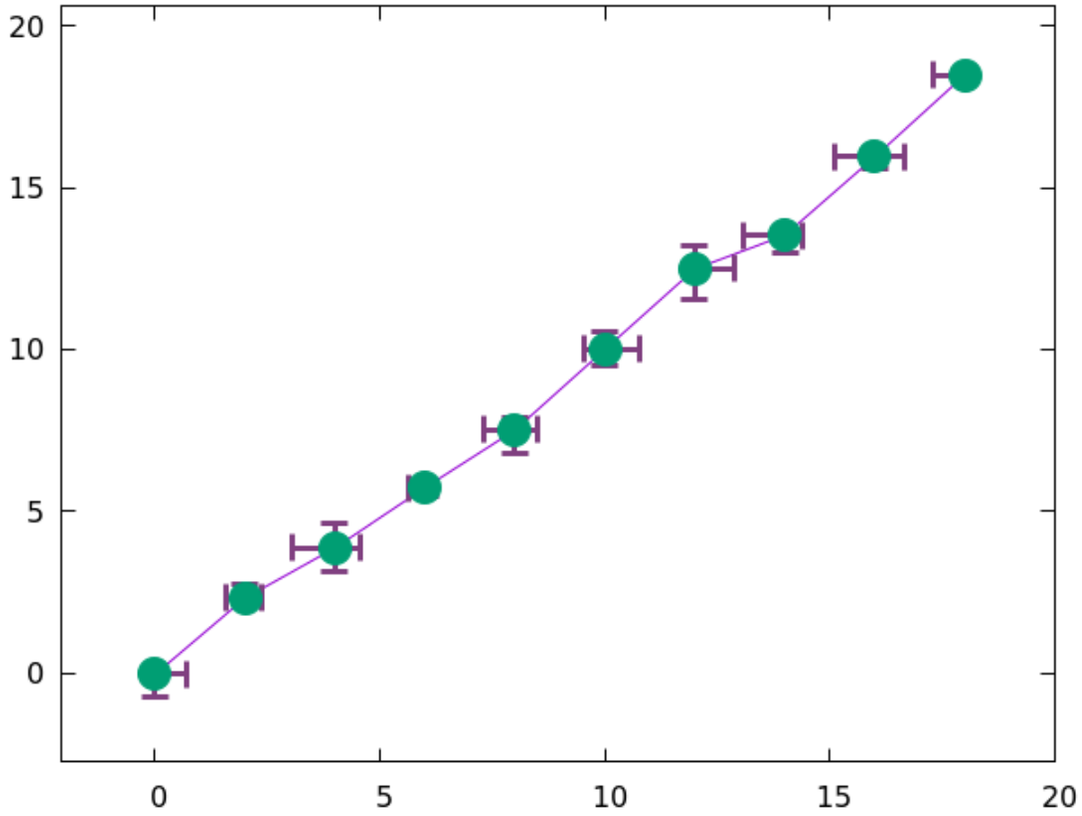
It's more usual to plot the data values conspicuously, with their errors overlaid. This can be accomplished by adding a second plot to the error plots above. Remember that an empty string means to reuse the same date file; if no columns are specified with a `using` command then the first two are used to make a normal 2D plot.

```
unset key
set auto fix
set offsets 2,2,2,2
set errorbars lw 3 lt 1 lc rgbcolor("orchid4") 2
plot "statdata"  with xyerrorbars, "" ps 3 pt 7
```



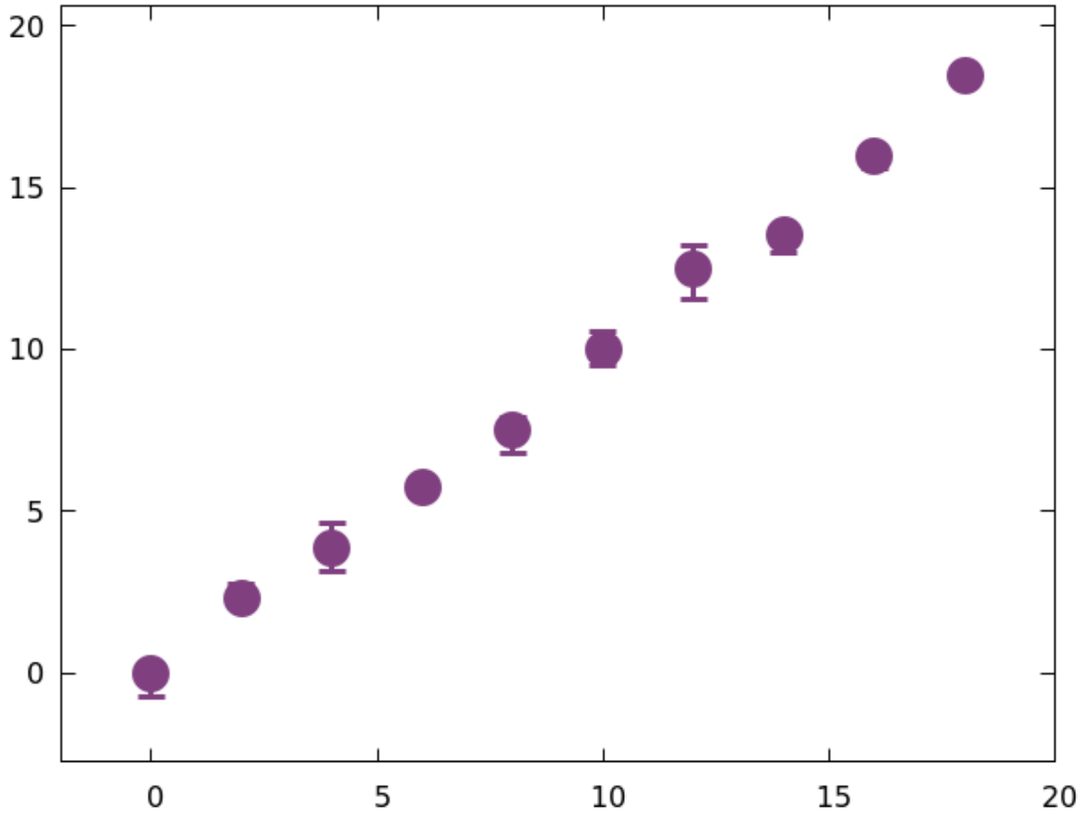
For each of the errorbar styles we cover in this chapter, gnuplot has a related *errorlines* style. We won't give an example of each one, because that would quickly get redundant. But here is one example, to show how it works: `with xyerrorlines` does the same thing as `with xyerrorbars`, but connects the dots with a line.

```
unset key
set auto fix
set offsets 2,2,2,2
set errorbars lw 3 lt 1 lc rgbcolor("orchid4") 2
plot "statdata" with xyerrorlines, "" ps 3 pt 7
```



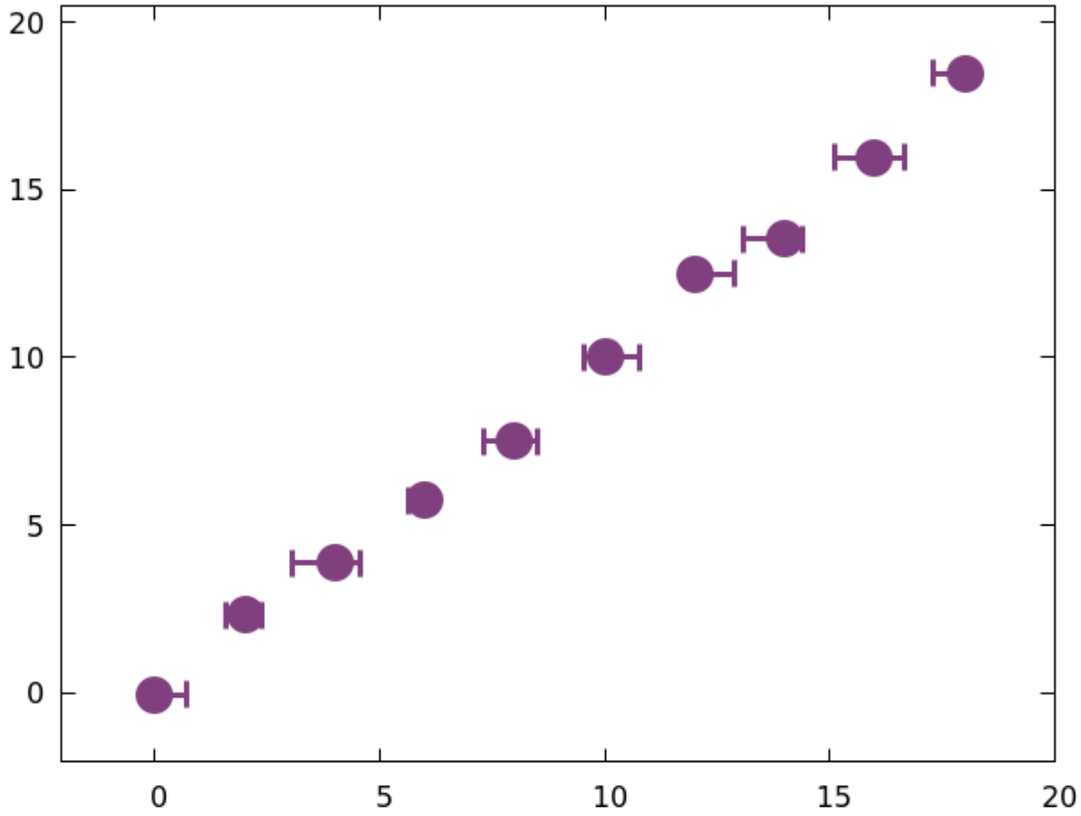
Often we need to plot data with error bars in the y direction, while assuming that the x values are exact. The gnuplot style for this is `with yerrorbars`. When plotting with this style you can supply the data in one of two formats: either $x \quad y \quad \Delta y$ or $x \quad y \quad y_- \quad y^+$. Our data file is in neither format, so we'll employ the `using` directive again:

```
unset key
set auto fix
set offsets 2,2,2,2
set errorbars lw 3 lt 1 lc rgbcolor("orchid4") 2
plot "statdata" u 1:2:5:6 with yerrorbars ps 3 pt 7
```



There is also an `xerrorbars` style, which does what you might expect. Here the data columns must be in the order $x \quad y \quad \Delta x$ or $x \quad y \quad x_- \quad x^+$.

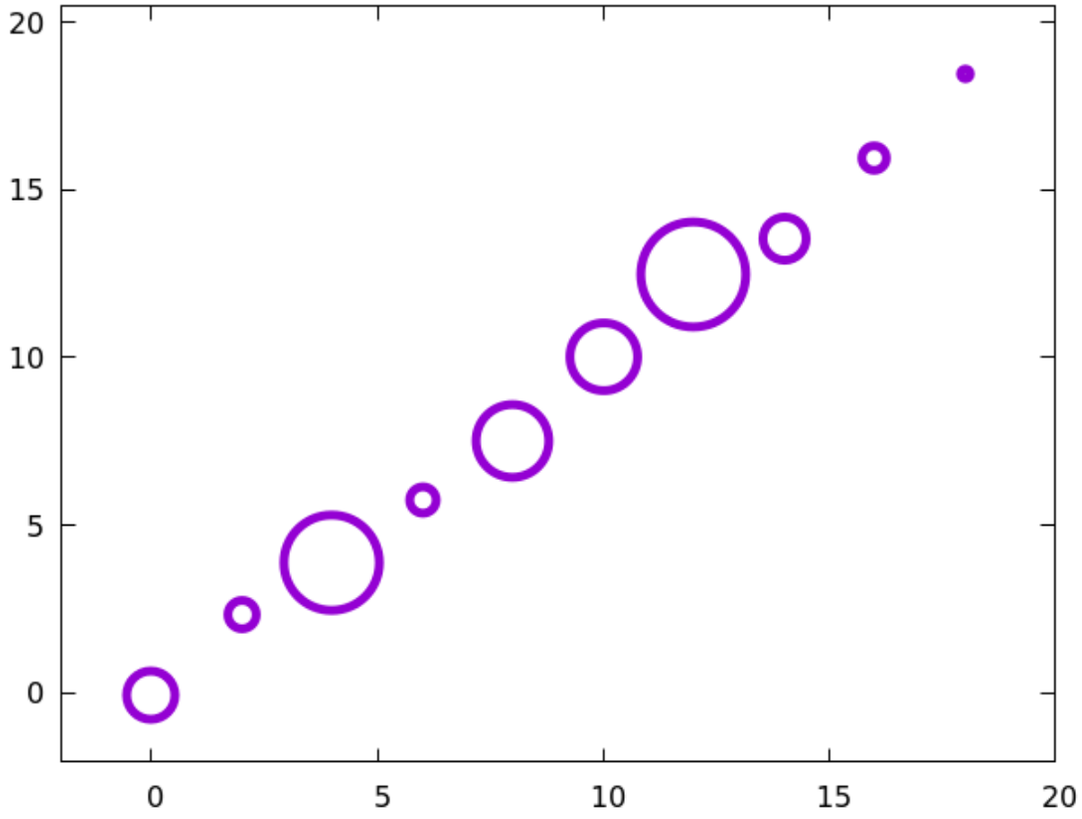
```
unset key
set auto fix
set offsets 2,2,2,2
set errorbars lw 3 lt 1 lc rgbcolor("orchid4") 2
plot "statdata" u 1:2:3:4 with xerrorbars ps 3 pt 7
```



"var"

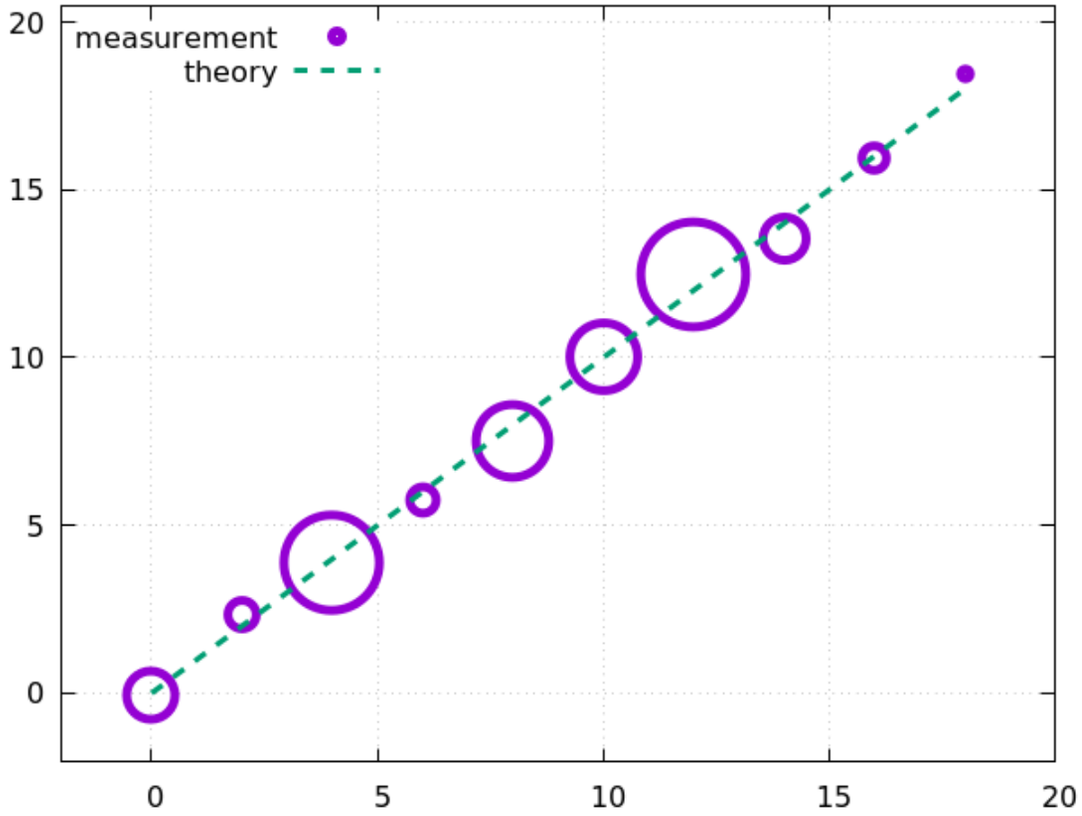
The properties of the lines or points that make up your graph (the ones that you set using the `set ps`, etc. commands) can be controlled by the data itself, using gnuplot's `var` command. This opens up a tremendous array of possibilities. Since this chapter is about visualizing errors, or ranges of values, here is a script that plots the data with circles of varying sizes, where the size of each circle shows the y-error of its corresponding data point. The y-error is calculated by subtracting y^- from y^+; the multiplier 6 was arrived at through trial and error (the argument to `with pointsize` is simply a multiplier applied to the default size for the terminal in use). The result of this arithmetic is supplied as a virtual third column, which is picked up by the `var` command, highlighted below:

```
unset key
set auto fix
set offsets 2,2,2,2
plot "statdata" u 1:2:(6*($6-$5)) pt 6 lw 5 ps var
```



Since the data in the file `statdata` was generated by applying some random noise to the line $y = x$, let's complete the previous example by treating the line as if it were a "theory" and the data points as of they were "measurements."

```
set auto fix
set offsets 2,2,2,2
set grid
set key top left
plot "statdata" u 1:2:(6*($6-$5)) pt 6 lw 5 ps var title "m
   x lw 3 dt "-" title "theory"
```

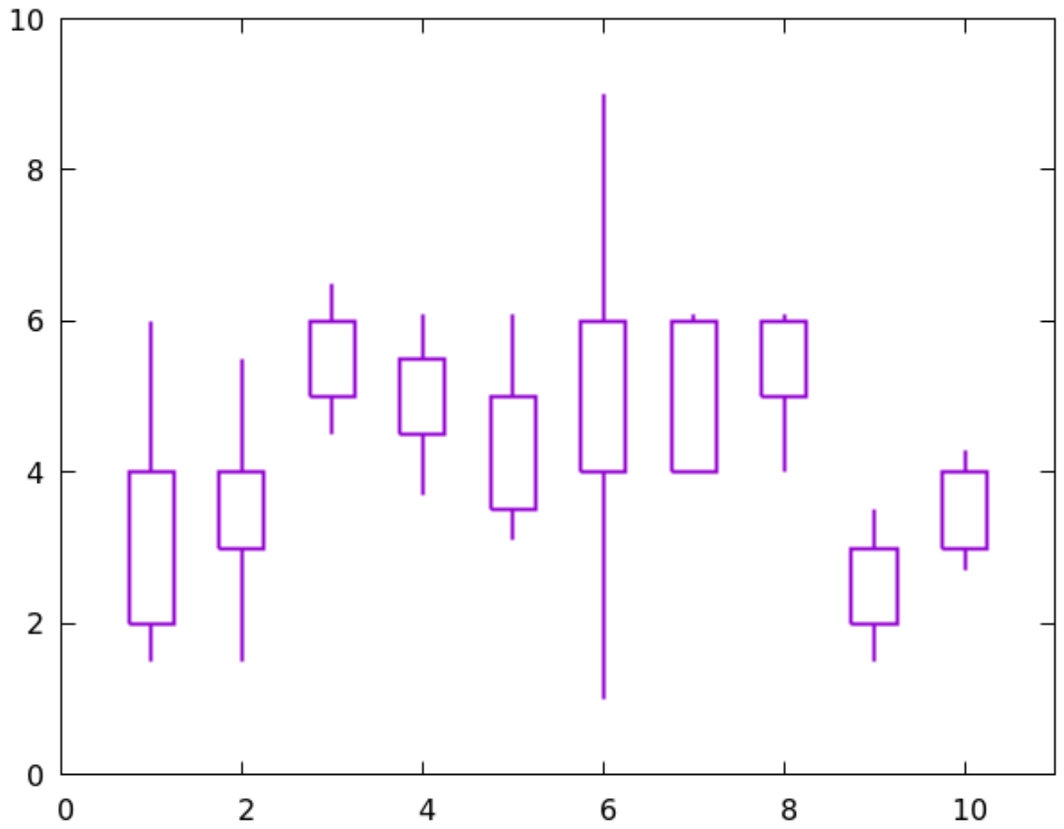


Whisker Plots

Another convention for visualizing data ranges is a "whisker plot", also known in the statistics world as a "box and whisker plot" a boxplot, or a candlestick plot. In gnuplot this type of plot is created by saying `with candle`. The statistical whisker plot is a series of symbols, each one showing the mean value of a set of measurements, the width of the central part of the measurements' or population's distribution, and the extent of the remainder of the distribution excluding the "outliers" (the outliers themselves are sometimes shown as dots, but we won't use that style here). See a statistics textbook for definitions and use of these concepts. This type of plot is also sometimes used for financial price data rather than the finance plot that we'll show later in this chapter. We will avoid the specialized jargon of statistics and further discussion of the uses of these plots, but the statisticians among our readers know why they're here.

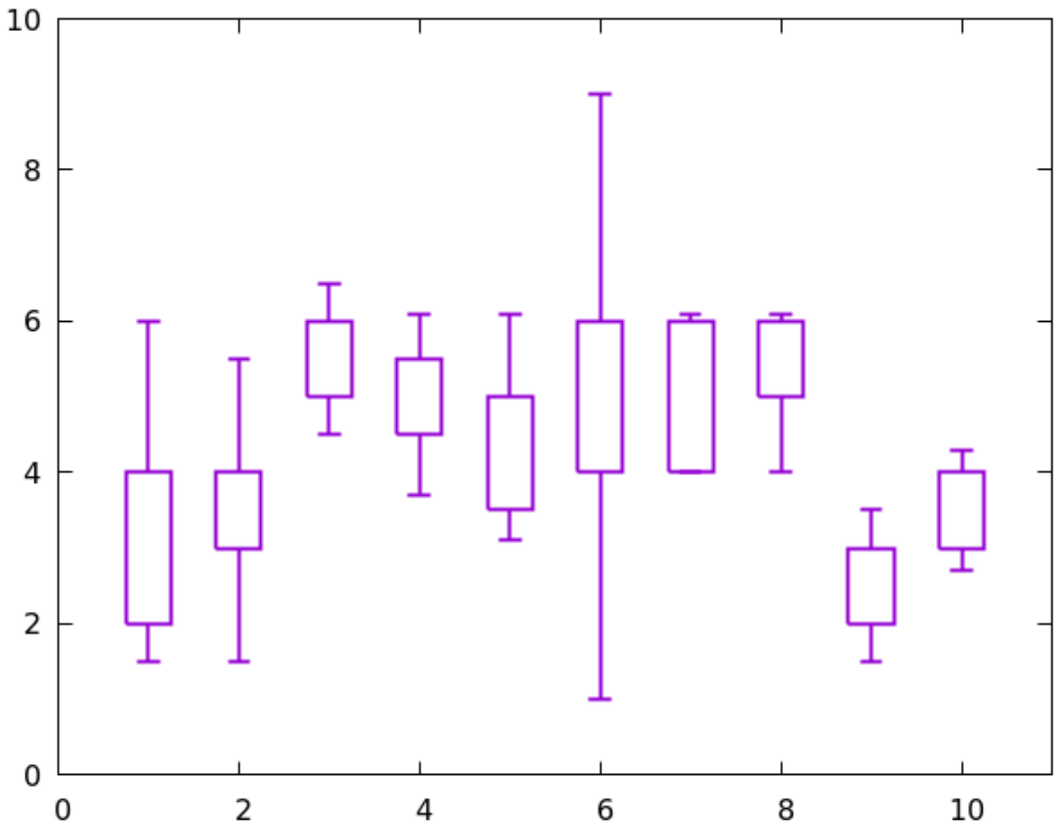
Our file `statdata` is not ideal for illustrating whisker plots, so we've included another data file called `candles`. The column order used by gnuplot's candlestick style is `x box_min whisker_min whisker_high box_high`; rarely is a data file in this format, and our file `candles` is no exception, so we'll turn to the `u` command to repurpose the data columns. The `candles` file contains the columns `x whisker_min box_min data_mean box_high whisker_high`. Note, for example, that the mean values are not even included in the candlestick plot; if desired, they must be overlayed with a second plot command. Below is a basic example of the candlestick style. The boxwidth controls the width of the candlesticks.

```
unset key
set auto fix
set offsets 1,1,1,1
set boxwidth .5
plot "candles" u 1:3:2:6:5 with candle lw 2
```



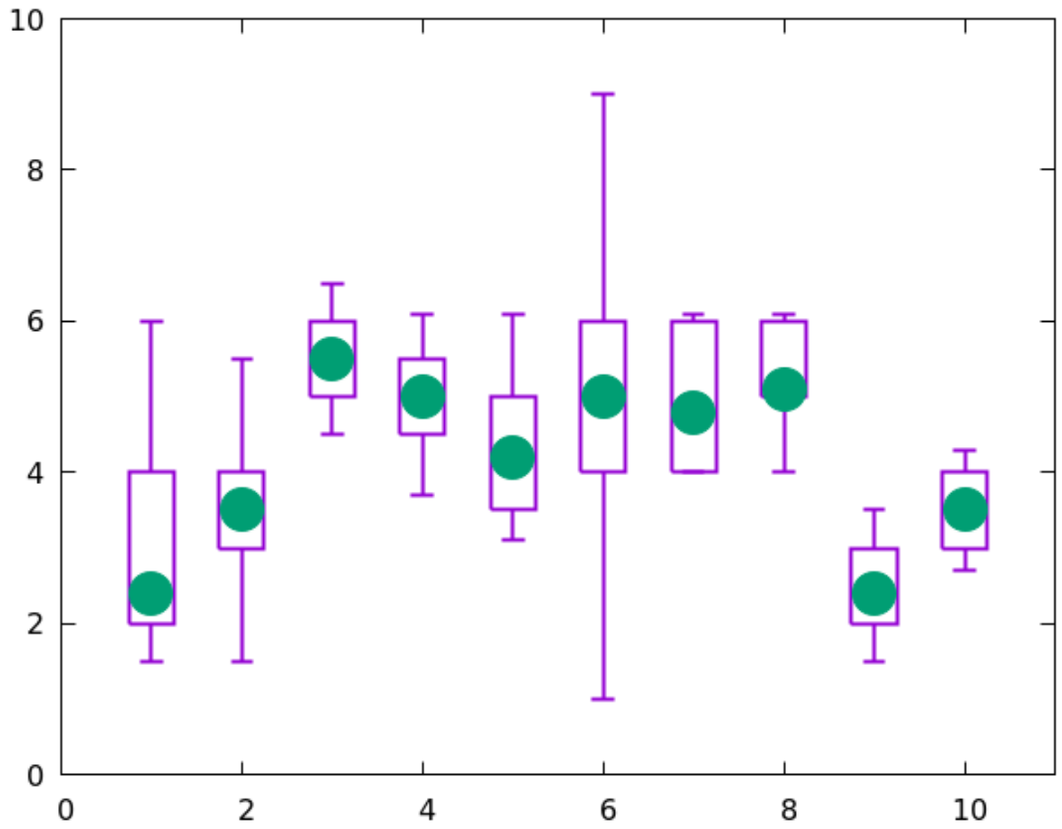
It's common in statistical plots to put little end-caps on the whiskers. Here is the previous script with the command to do that added; the numerical argument is the width of the Whiskers as a fraction of the width of the box:

```
unset key
set auto fix
set offsets 1,1,1,1
set boxwidth .5
plot "candles" u 1:3:2:6:5 with candle lw 2 whisker 0.5
```



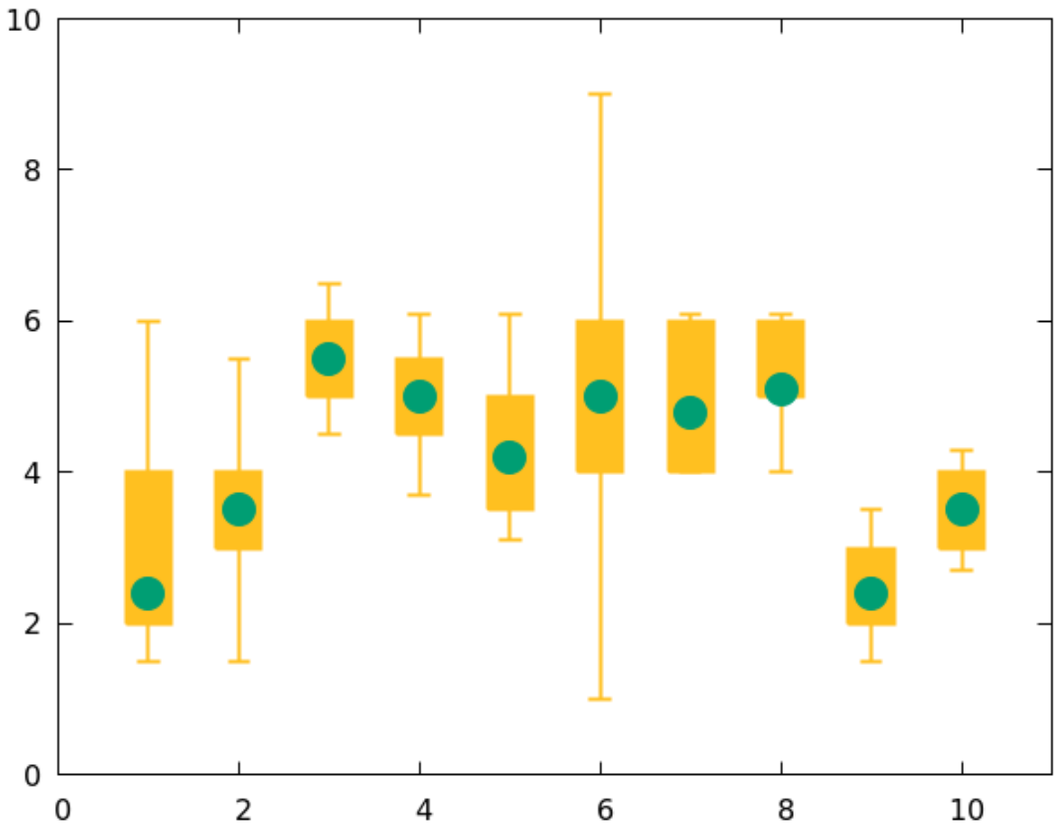
As we mentioned above, the candlesticks don't include the actual data means. Here we add them with a second plot command.

```
unset key
set auto fix
set offsets 1,1,1,1
set boxwidth .5
plot "candles" u 1:3:2:6:5 with candle lw 2 whisker 0.5,\
   "" u 1:4 pt 7 ps 4
```



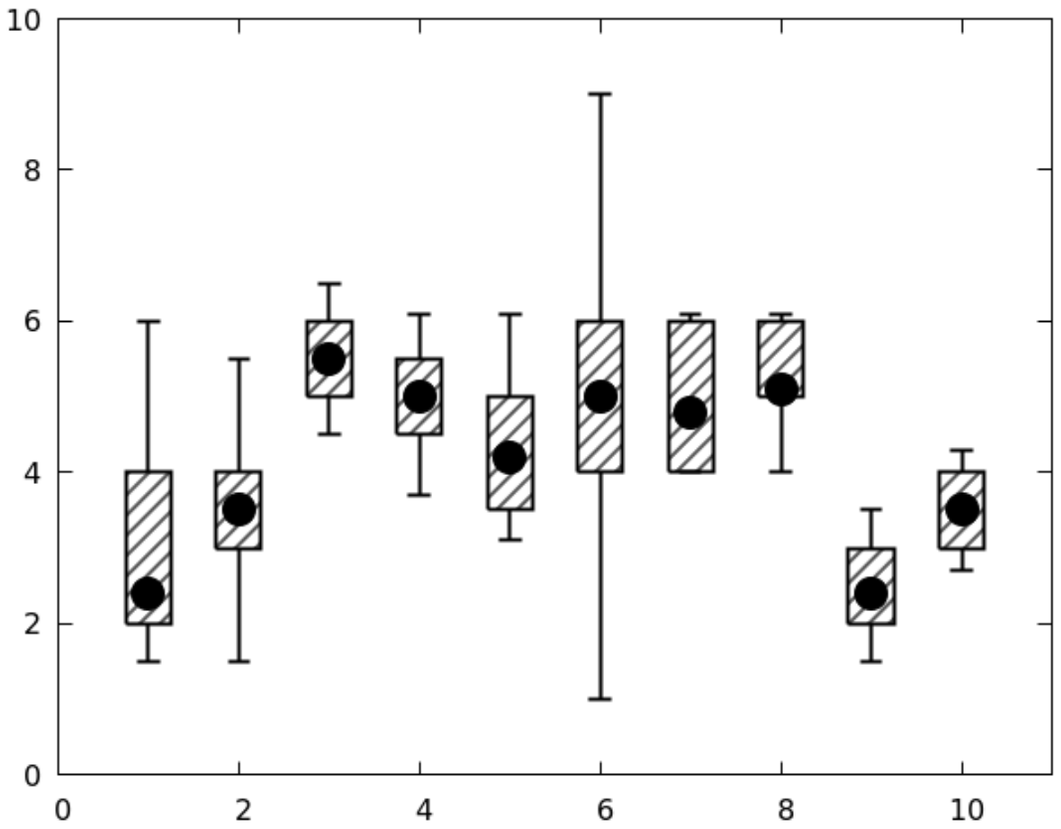
The default style for candlesticks is to draw an empty box. When this style us used for market price data, the columns are interpreted as `date open low high close`, and if the closing price is lower than the opening price, the candlestick box is filled in. However, if a solid or patterned fill style is set for the boxes, this will be used for all the candlesticks in all cases. Here is our candlestick plot with solid filled boxes:

```
unset key
set auto fix
set offsets 1,1,1,1
set boxwidth .5
plot "candles" u 1:3:2:6:5 with candle lw 2 whisker 0.5\
   fs solid fc rgbcolor("goldenrod"),\
   "" u 1:4 pt 7 ps 3
```



You can also fill the candlesticks with a pattern, which is especially useful for monochrome publication:

```
set monochrome
unset key
set auto fix
set offsets 1,1,1,1
set boxwidth .5
plot "candles" u 1:3:2:6:5 with candle lw 2 whisker 0.5\
   fs pattern 5, "" u 1:4 pt 7 ps 3
```

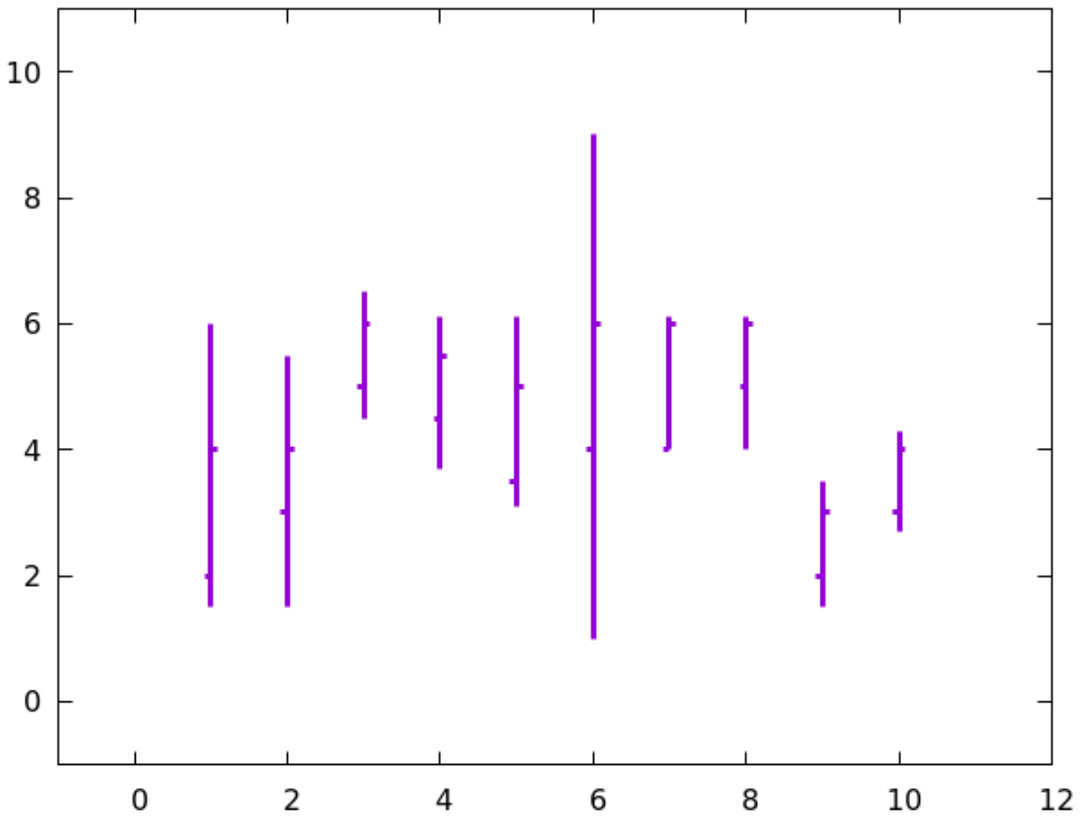


Financebars

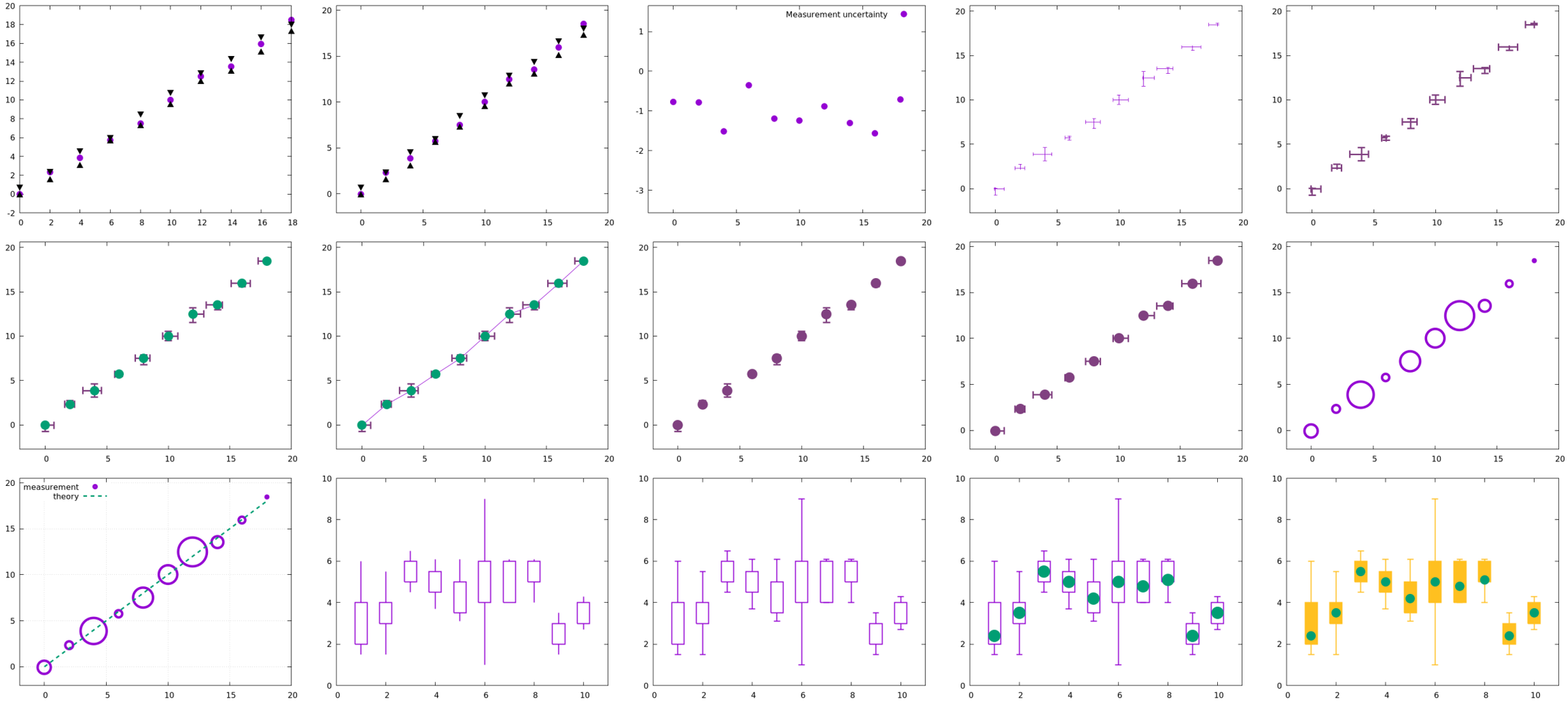
The same information as in our first and second candlestick plots (without the mean data values) can be plotted using gnuplot's `with financebars` style. This is another, more common, convention for plotting market price fluctuations. This style does not distinguish between price increases or decreases. It shows the high and low prices by the extent of a vertical line, and the opening and closing prices with small horizontal tics attached to the line. The width of these tics (the default is almost invisible) is set with the `set bars` command.

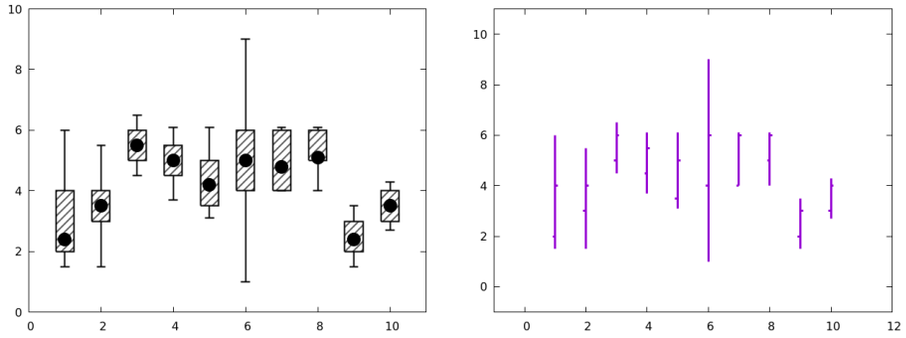
Note that it would be more usual for the values on the horizontal axis to be dates or date/times, but we'll defer plotting with times until we've had a chance to introduce that topic.

```
unset key
set auto fix
set offsets 2,2,2,2
set bars 2
plot "candles" u 1:3:2:6:5 with financebars lw 3
```



Index of Plots




10
8
6
4
2
0
0
2
4
6
8
10

10
8
6
4
2
0
0
2
4
6
8
10
12