# Chapter 3: Histograms and Bar Charts

The title of this chapter refers to two different things that are sometimes confused.

A bar chart is a type of 2D plot uses the heights of a series of vertical (or, sometimes, horizontal) bars, drawn with gaps between them, to indicate the quantities of a small or moderate number of different things, so they can be compared visually. Although it is a type of 2D plot, we discuss it in this chapter rather than in Chapter 1, because it is a special case, and because of the similarity between bar charts and histograms. You are about to see many examples of bar charts, which should make the description above clear; they are also ubiquitous in journalism and advertising.

A histogram is a little different. This visualization shows how a continuous variable is distributed among different ranges of values. The bars in a histogram are plotted abutting each other. When used to visualize a distribution, their width, as well as their height, is significant: each bar's area is proportional to the number of data points that it represents.

For most of the examples in this chapter we use a datafile called "energySources". This is a table of

a handful of countries and the percentage of energy production for each country from several sources. The data is real, and comes from the CIA World Factbook. If you take a look at this datafile, you can see how comments can be added, using the "#" symbol, and how textual labels can be included with the data. You should have downloaded the file from the same place where you downloaded this text.

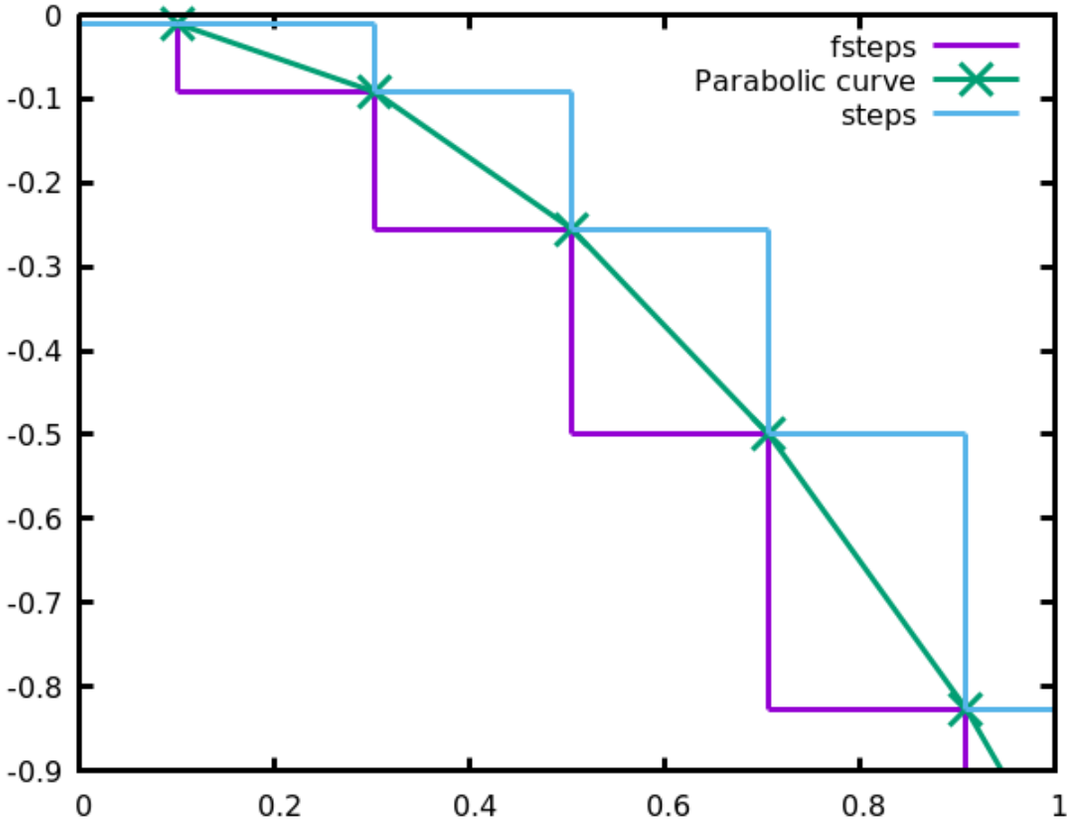
## steps and fsteps

Our first example in this chapter is not actually about histograms, nor about bar charts, but rather about gnuplot's step styles; we're including it here because of some similarity to histogram plots, especially the `histeps` plot in the next example.

The first line in the script shows another way to set a style option, in this case for the `linewidth`. Setting a `termoption` will set a default `lw`, so we won't need to append the phrase to each plot command.

We plot the `parabola.dat` file that we've used before; if you don't have it you can download it from the usual place. The new plot styles are highlighted, and the graph shows the difference between the `steps` and `fsteps` styles; the normal `linespoints` plot is included for reference. As you can see, both styles plot the data with horizontal and vertical line segments; the difference is whether you go down first and then to the right, or right first and then down.

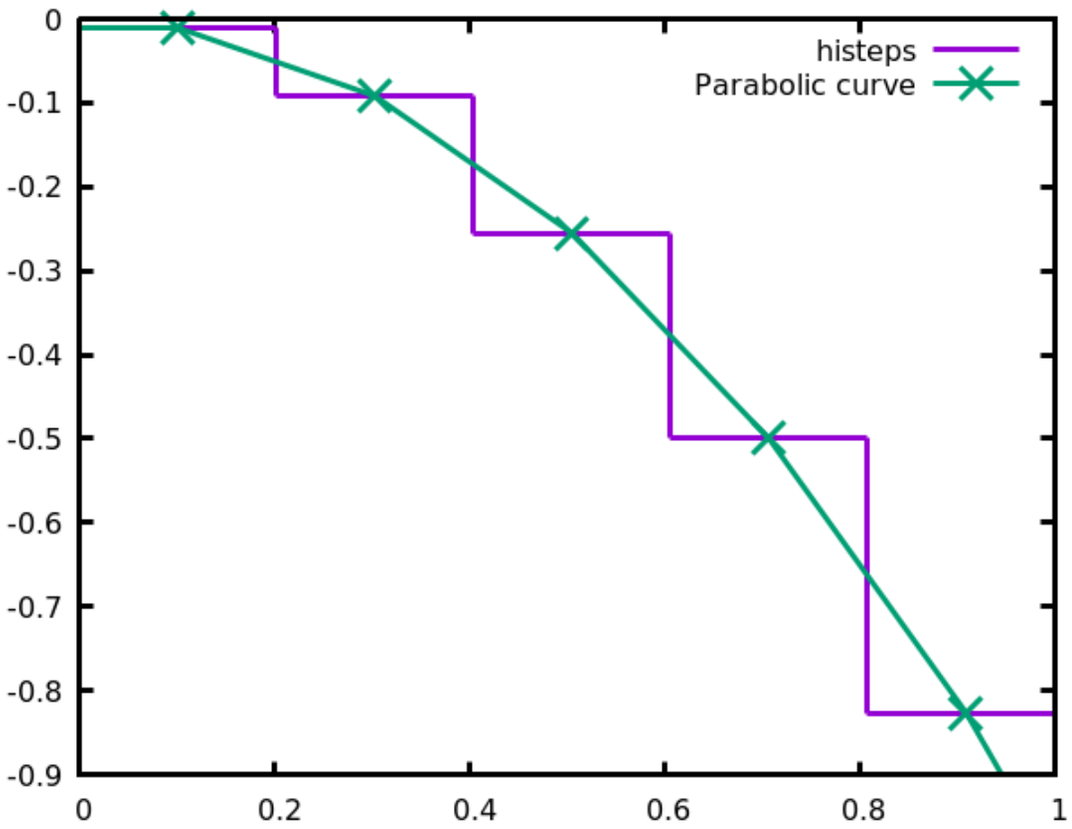
```
set termoption lw 3
set xr [0 : 1]
plot "parabola.dat" with fsteps title "fsteps",\
   "" with linespoints ps 3 title "Parabolic curve",\
   "" with steps title "steps"
```



## histeps

Here is a third step style: `histeps` also uses horizontal and vertical line segments, but centered on the datapoints, as shown in the graph. The name is intended to refer to histograms, which are similarly centered on the data.

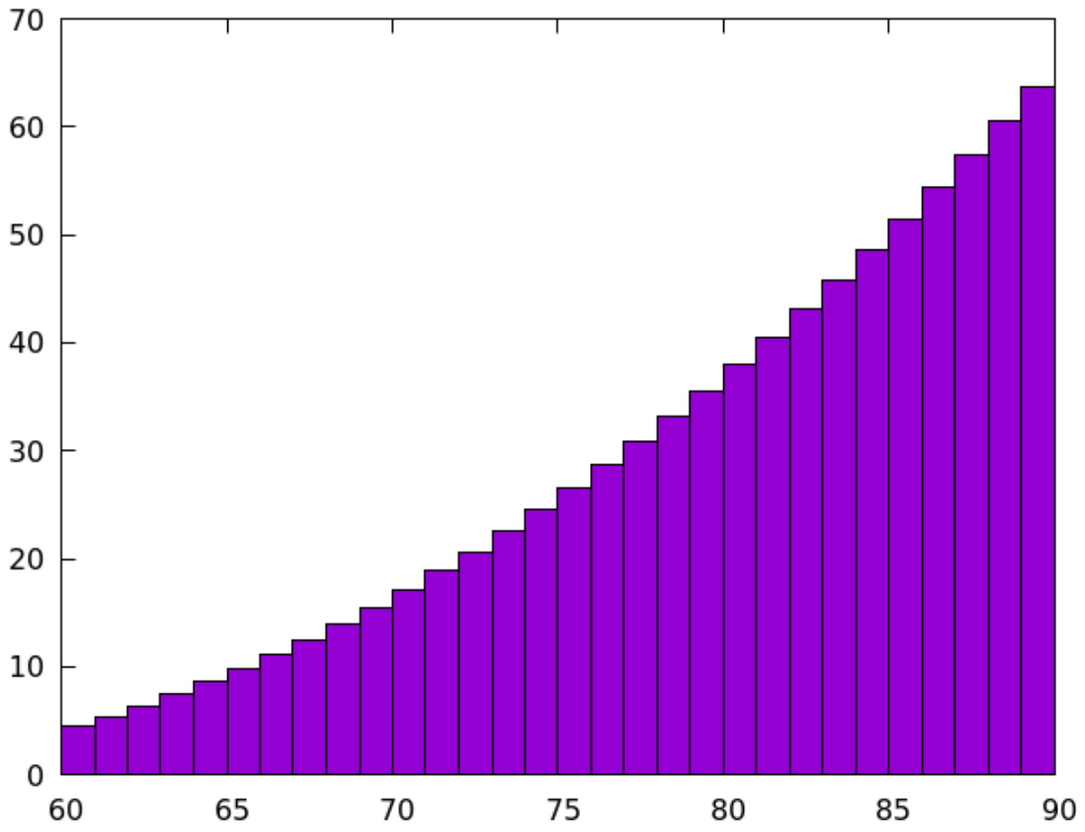
```
set termoption lw 3
set xr [0 : 1]
plot "parabola.dat" with histeps title "histeps",\
   "" with linespoints ps 3 title "Parabolic curve"
```



## Histograms

Here we'll plot a section of our parabolic data as a histogram. The first line in the script tells gnuplot that further commands to plot from a data file should use the histogram style. The second line causes the bars to be filled with a solid color, and to be drawn with a solid black border (the `linetype -1` on most terminals). The third line ensures that the bars are drawn with no gap between them, resulting in a proper histogram plot.

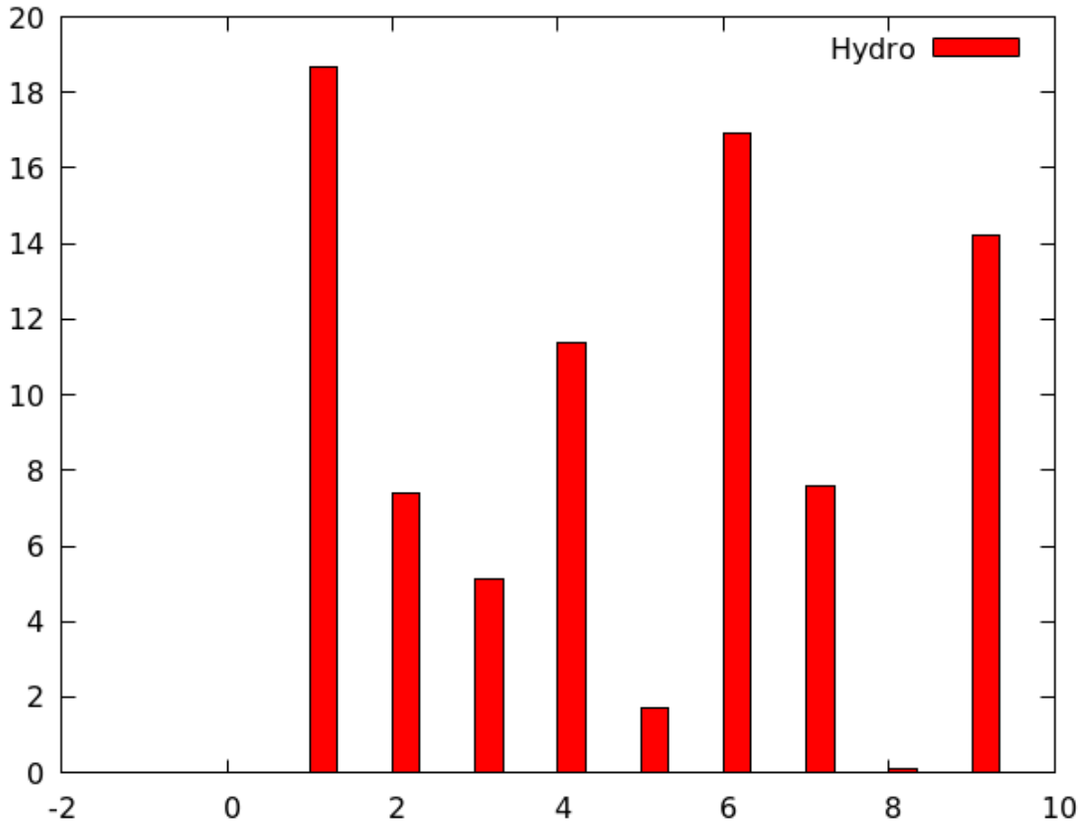
```
set style data histogram
set style fill solid border -1
set style histogram gap 0
set xr [60 : 90]
unset key
plot "parabola.dat"  u (-$2)
```



## Bar Charts

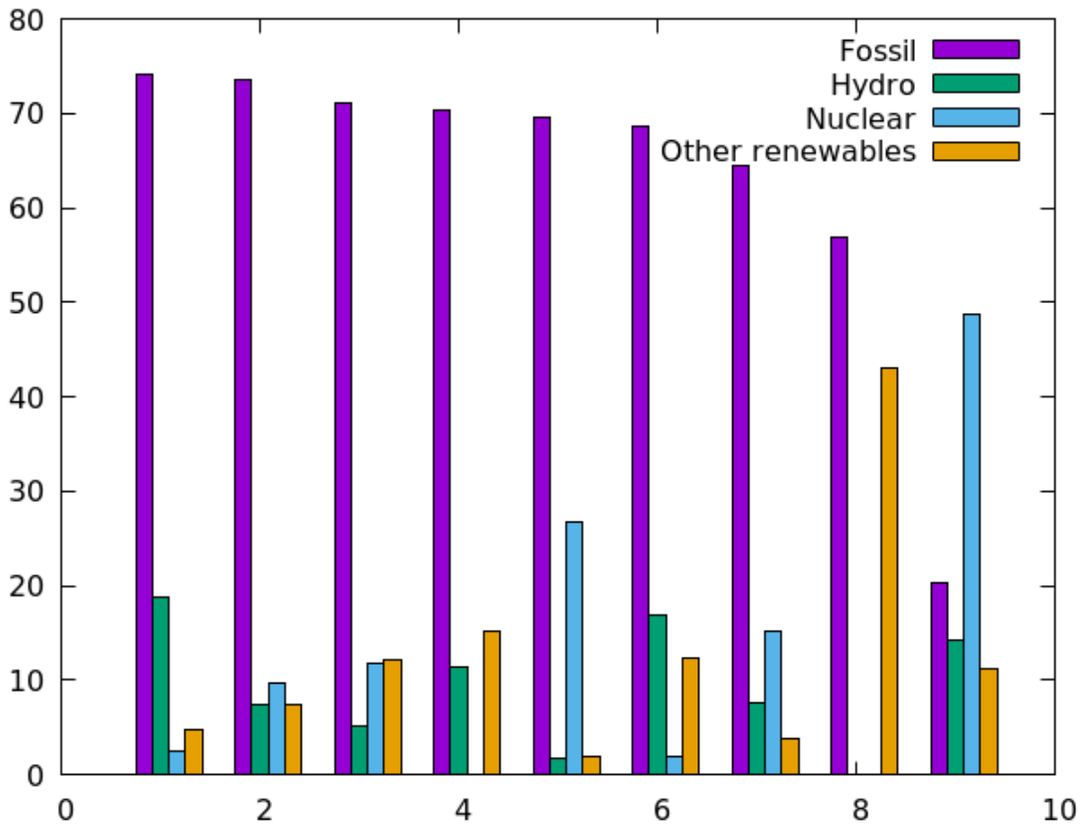
We now turn to simple bar charts. In this and the next handful of examples, we use the provided "energySources" file. We'll often be extracting a subset of the data in this file for plotting, as we do in this first example. Gnuplot uses the "histogram" style for bar charts as well as true histograms. The third line uses commands that we've already seen; the final phrase setting the `linecolor` sets the color of the fill. You can see that gnuplot puts the number of the row on the x-axis; we'll see later how to make this more informative. The default gap between bars (or clusters of bars; see the next example) is the width of two bars.

```
set style data histogram
set style fill solid border -1
plot "energySources" u 3 title "Hydro" lc "red"
```



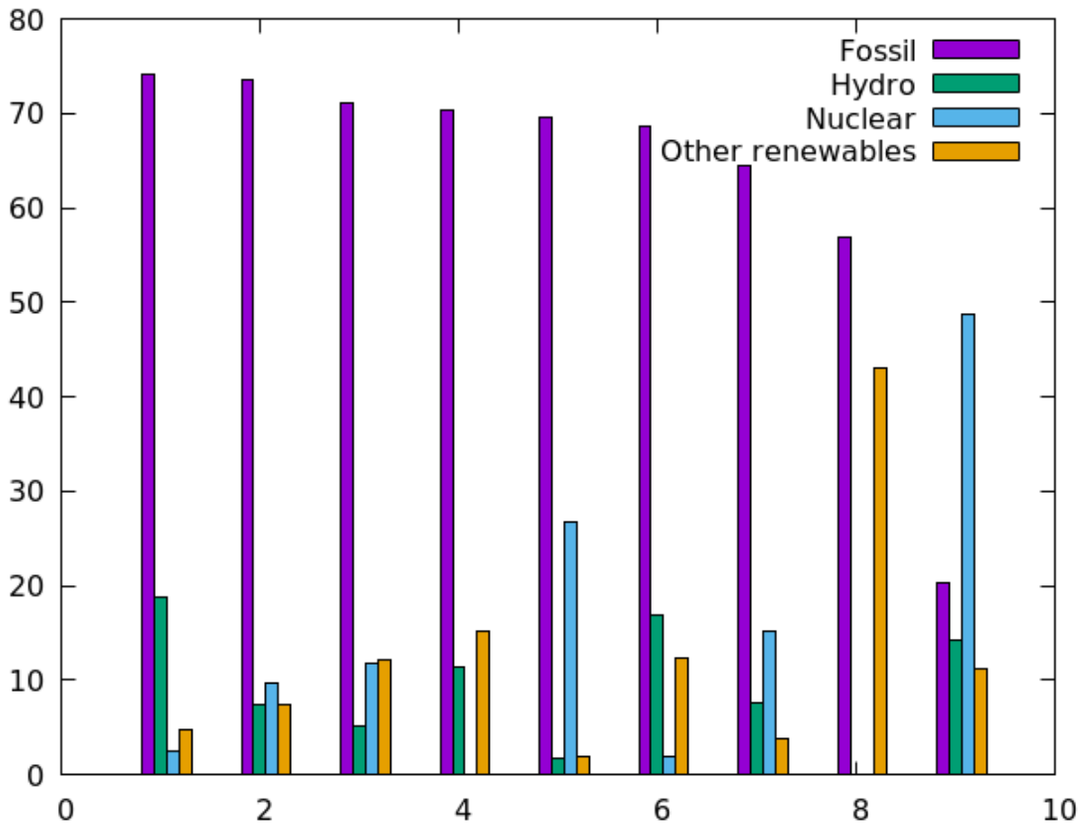
The previous example extracted the third column from the file. If you want to plot more than one category at a time, simply select more columns. Gnuplot will create groups of bars for you, grouping by row in the datafile. Here we plot all the columns. As you can see from the previous example, gnuplot sometimes does a poor job of deciding on the `xrange`, so we'll correct that here.

```
set style data histogram
set style fill solid border -1
set xr [0 : 10]
plot "energySources" u 2 title "Fossil",\
   "" u 3 title "Hydro",\
   "" u 4 title "Nuclear",\
   "" u 5 title "Other renewables"
```



You can adjust the spacing between the groups of bars (called “clusters” in gnuplot terminology) with another command, shown below. The default style is a gap of 2, which leaves a space equal to the width of two bars. As you increase the gap between clusters, the bars are made thinner if necessary to fit everything on the plot.

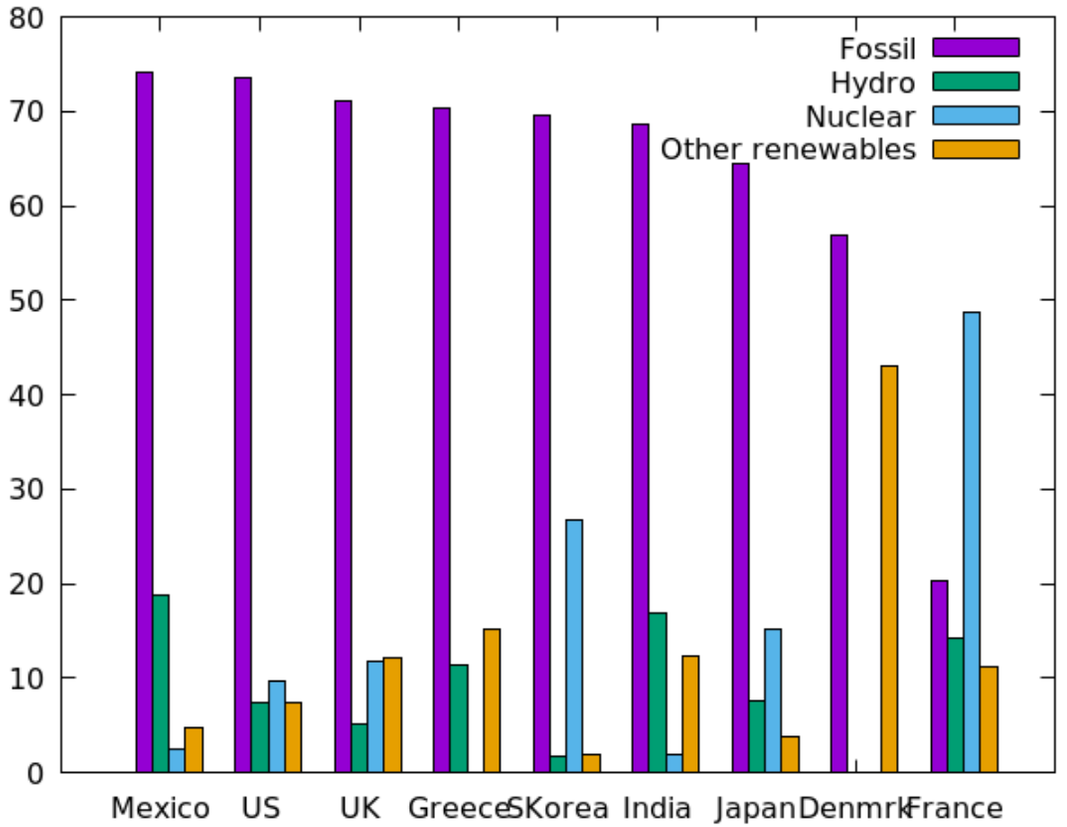
```
set style data histogram
set style fill solid border -1
set style histogram cluster gap 4
set xr [0 : 10]
plot "energySources" u 2 title "Fossil",\
   "" u 3 title "Hydro",\
   "" u 4 title "Nuclear",\
   "" u 5 title "Other renewables"
```



## xticlabels

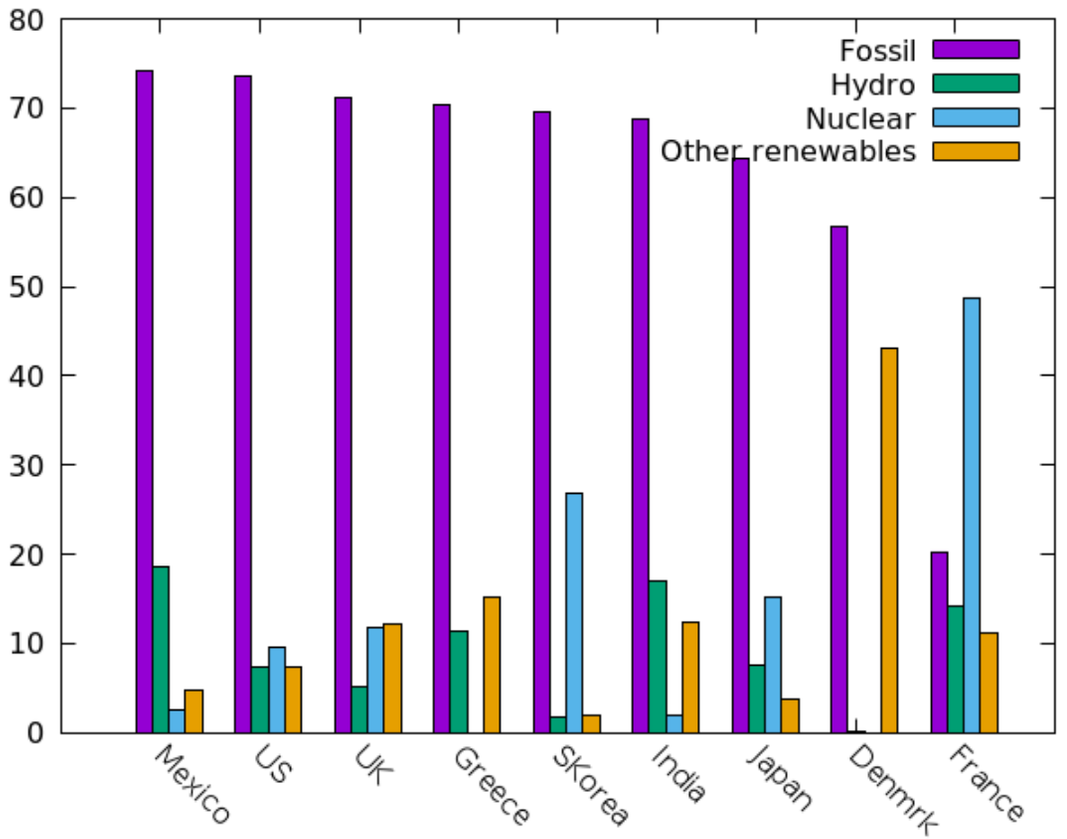
So far the labels on the x-axis in our bar charts have not been very informative. The row numbers don't tell us anything without a legend. Fortunately, the country names that go with the rows are included in the datafile, and gnuplot has a way to use them: the `xticlabels` command, abbreviated `xtic`. This function can take an integer argument, which tells it which column to pluck the labels from. Since this is a column selection, it's included as part of the `using` command. The country names are in column 1, so we do this:

```
set style data histogram
set style fill solid border -1
set xr [0 : 10]
plot "energySources" u 2 title "Fossil",\
   "" u 3 title "Hydro",\
   "" u 4 title "Nuclear",\
   "" u 5:xtic(1) title "Other renewables"
```



You probably noticed that the labels on the x-axis in the previous example were too close together, and even overlapped slightly. Gnuplot will not reduce the font size or take any other measures to fix this: it's up to you. You could make everything fit by using a smaller font size, but this is not ideal. In a later chapter we'll learn all about how to deal with labels and text on plots, but, since this is such a common problem with bar charts, we'll permit ourselves a preview here. Observe the highlighted command in the script below, and the effect it has on the labels:

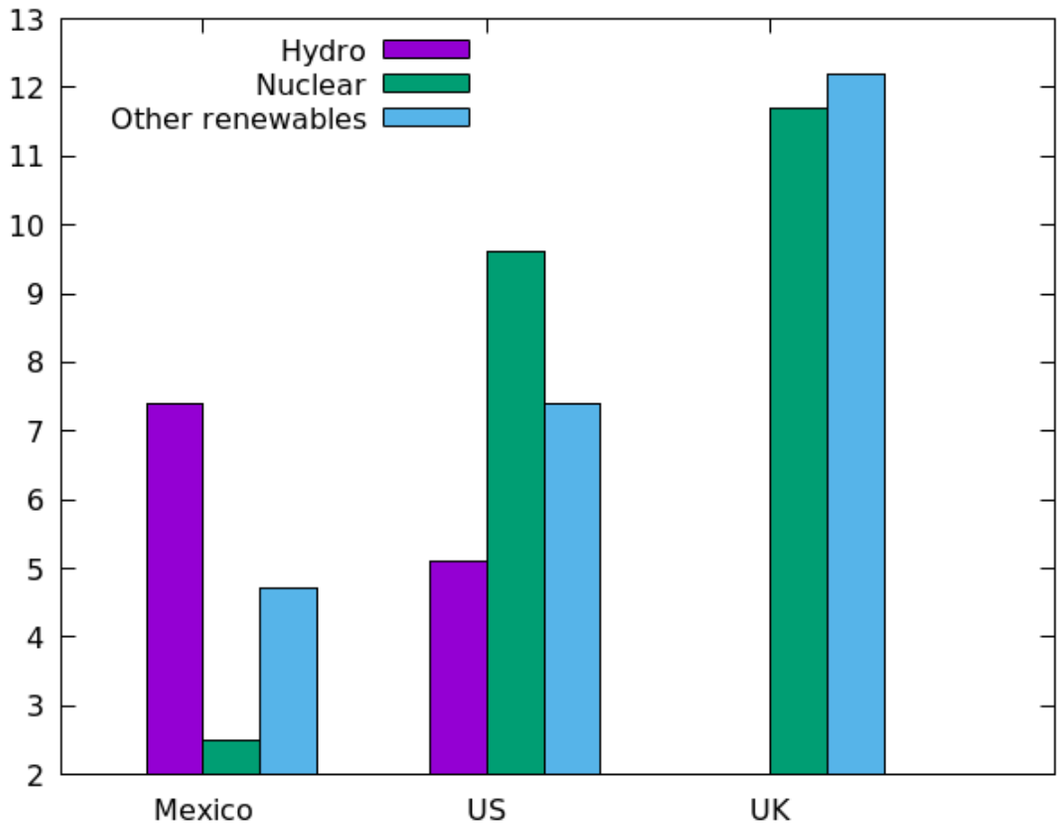
```
set style data histogram
set style fill solid border -1
set xr [0 : 10]
set xtic rotate by -45
plot "energySources" u 2 title "Fossil",\
    "" u 3 title "Hydro",\
    "" u 4 title "Nuclear",\
    "" u 5:xtic(1) title "Other renewables"
```



## The `every` Command

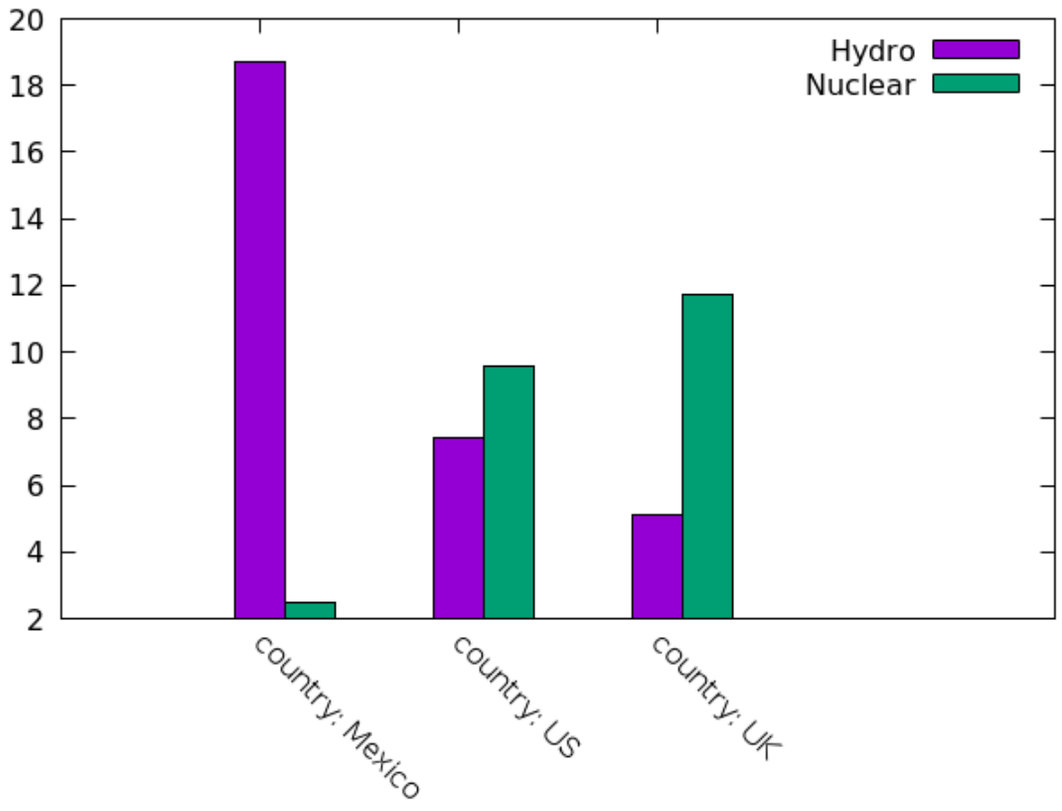
The `using` command is used for selecting columns, as we've seen by now many times. If you want to limit the plot to only certain rows, instead of plotting every row in the datafile, use the `every` command. The syntax is `every ::a::b`, where `a` is the row where you wish to begin, and `b` is the ending row. The row numbering starts at 0. In our case, row zero contains the energy source labels, so we start at 1. The colons can be replaced by numbers for selecting data *blocks* (a subject for a later chapter) and for skipping rows. For all the details in one place, type `help every` at the gnuplot prompt.

```
set key left top
set style data histogram
set style fill solid border -1
set xr[0.5 : 4]
plot "energySources" u 3 every ::1::3 title "Hydro",\
   "" u 4 every ::0::3 title "Nuclear",\
   "" u 5:xtic(1) every ::0::3 title "Other renewables"
```



Up above we learned about `xticlabels`. The integer argument to this function (say, `n`) is treated as an abbreviation of `stringcolumn(n)`, where `stringcolumn` is a function that reads the column specified in its argument as a series of strings (rather than numbers). In fact, `xticlabels` takes a string argument. We can use gnuplot's string concatenation operator, which is the period (full-stop), `.`, to build up a more elaborate label:

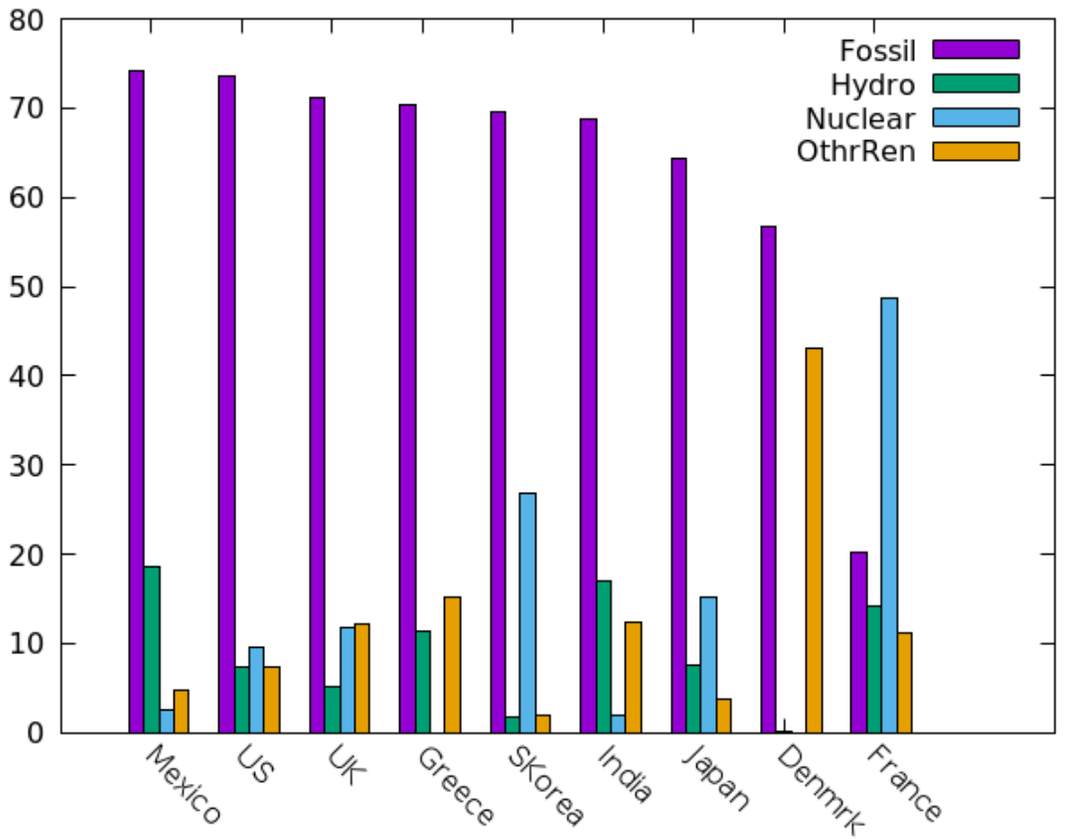
```
set style data histogram
set style fill solid border -1
set xtic rotate by -45
set xr[-1 : 4]
plot "energySources" u 3 every ::1::3 title "Hydro",\
   "" u 4:xtic("country: " . stringcolumn(1))\
   every ::1::3 title "Nuclear"
```



## Automatic Titles

Since our datafile contains titles, it would be nice if gnuplot could read those and use them, so that we would not be required to append a `title` command to each plot command. Gnuplot can do this, in two ways. To read the titles automatically for each `plot` command, use the new command highlighted in the script below, where `col` is an abbreviation for `columnheader`. If, instead, you want to read the titles from the datafile, but only for some of the data, you can append a `title col` to the individual plot commands for which you want the title picked up.

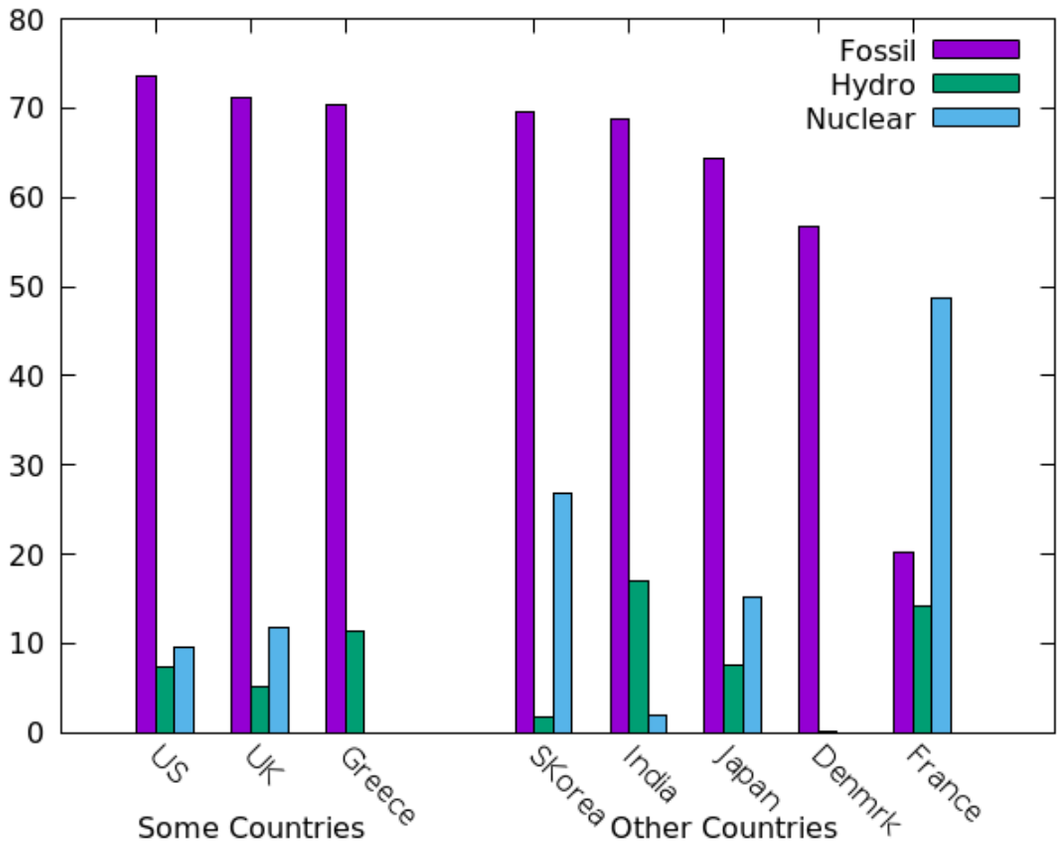
```
set style data histogram
set style fill solid border -1
set key autotitle col
set xr [-1 : 10]
set xtic rotate by -45
plot "energySources" u 2,\
   "" u 3,\
   "" u 4 ,\
   "" u 5:xtic(1)
```



## The `newhistogram` Command: Grouping Clusters

Sometimes, grouping the bars in a bar chart into clusters is just not complicated enough. When you feel the need to group your clusters themselves into bigger groups, that's when you reach for the `newhistogram` command. The `newhistogram` keyword is followed by a title for the grouping, and, usually, a `linetype` or `linecolor` specification to reset the color sequence. You must turn off titles for all bars after the first group, to avoid repeating the titles in the legend, if you are using one. An example should show you what it's all about:

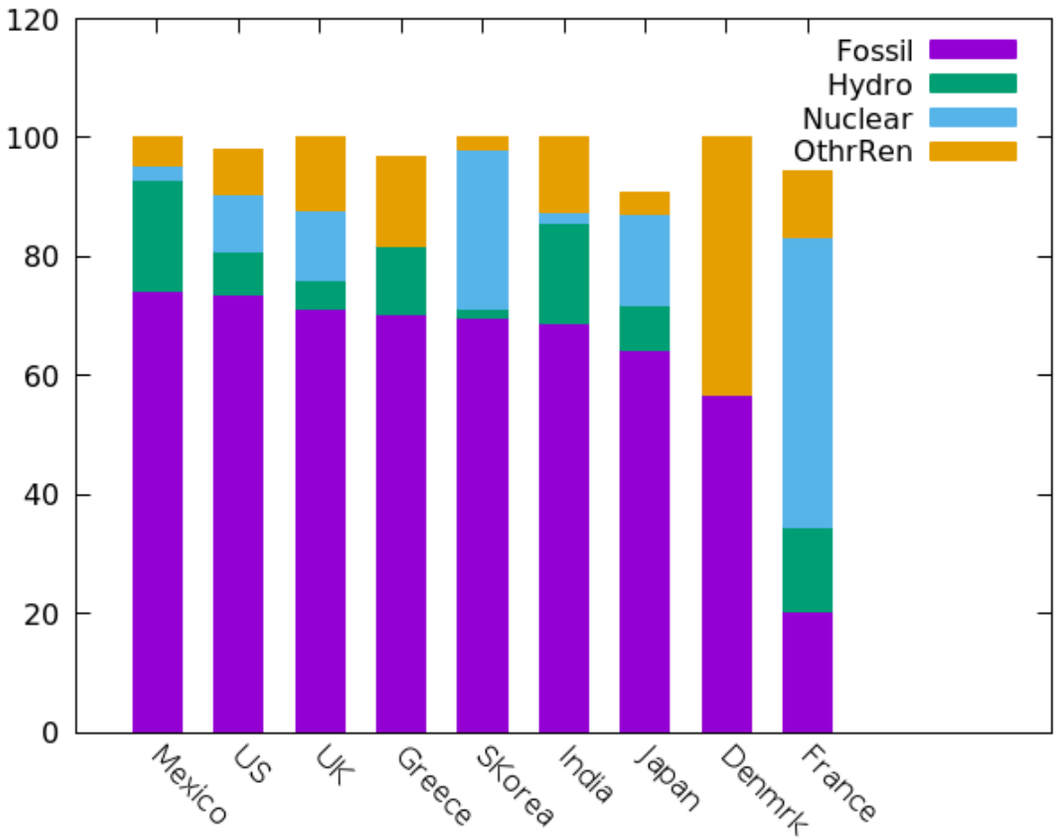
```
set style data histogram
set style fill solid border -1
set key autotitle col
set xr [-1 : 9.5]
set xtic rotate by -45
plot newhistogram "Some Countries" lt 1,\
   "energySources" u 2:xtic(1) every ::1::3 ,\
   "" u 3 every ::1::3,\
   "" u 4 every ::1::3,\
   newhistogram "Other Countries" lt 1,\
   "" u 2:xtic(1) every ::4::8 notitle,\
   "" u 3 every ::4::8 notitle,\
   "" u 4 every ::4::8 notitle
```



## Stacked Bar Charts

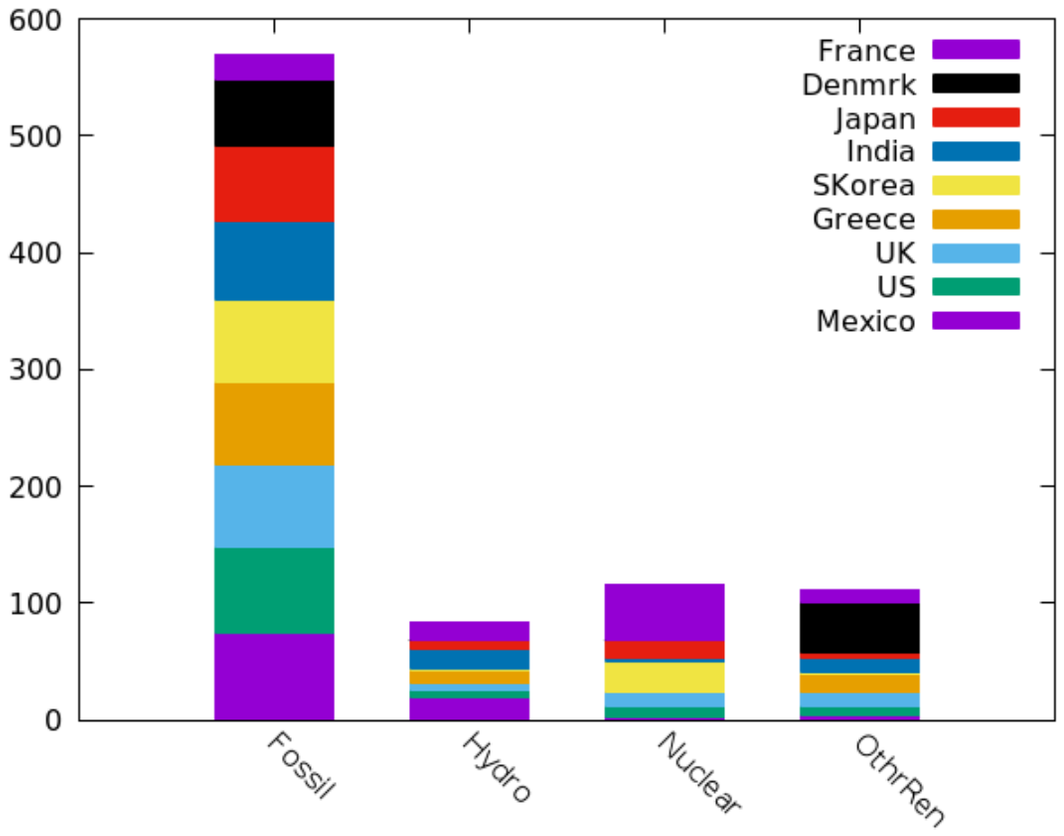
There is another way to display the type of data that we've been dealing with, that sometimes makes it easier to compare quantities. Instead of placing clusters of bars next to each other, each bar can be made of a stack of smaller bars. This results in a compact representation of the data that's easier for the eye to take in quickly. A simple example should make it clear how to achieve this familiar type of bar chart. The example script uses one additional new command, to reduce the `boxwidth` in order to allow a gap between the bars; for zero gap, such as when you are making a real histogram, you can set the `boxwidth` to 1.

```
set style data histogram
set style fill solid
set style histogram rowstacked
set boxwidth 0.6
set key autotitle col
set xr [-1 : 11]
set xtic rotate by -45
plot "energySources" u 2, "" u 3:xtic(1),\
   "" u 4, "" u 5
```



Gnuplot can stack the data the other way around, without having to alter the datafile. In other words, you can choose to make a stack of countries, and plot them against energy source on the x-axis: a transpose of the rows and columns in the datafile. To accomplish this, merely change the `rowstacked` keyword to `columnstacked`, and take care to select titles from the appropriate places. The titling is accomplished with two commands used here for the first time. The `key(1)` command tells gnuplot to take the title displayed in the key from column 1. The `title col` command, which we mentioned above, takes the column title, used on the x-axis, from the column header read from the datafile.

```
set style data histogram
set style fill solid
set style histogram columnstacked
set boxwidth 0.6
set xr [-1 : 4]
set xtic rotate by -45
plot "energySources" u 2:key(1) title col,\
    "" u 3 title col,\
    "" u 4 title col,\
    "" u 5 title col
```



# Index of Plots