Math and Numbers in Figure Text

The following `matplotlib` commands place text on a figure. (The tutorial on plotting shows where they appear.)

`xlabel`	label to the horizontal axis
`ylabel`	label to the vertical axis
`title`	add a title just above the axes
`suptitle`	add a title farther above the axes
`text`	add text at an arbitrary location in data coordinates
`figtext`	add text at an arbitrary location in relative coordinates
`annotate`	add an annotation, with optional arrow

It is often useful to include mathematical expressions and formatted numbers in the text.

Math in Figure Text

Mathematical expressions can be included in the text that any of the figure text commands produce by using TeX typesetting. In TeX, each symbol is represented by a string starting with a backslash (`\`). For example, "`\theta`" stands for the Greek letter theta (θ). In ordinary strings, backslashes get interpreted in ways that would interfere with TeX typesetting. For example, "`\t`" and "`\n`" are ordinarily interpreted as a tab and a linefeed, respectively. The first example would cause problems when trying to typeset a theta symbol. To get around this difficulty, a string can be preceded by an "`r`" so that it is treated as a raw string. Compare the output of the following `print` statements.

```
print 'This is \t an \n ordinary string'
print r'This is \t a \n raw string'
```

In a string sent to one of the `matplotlib` commands, text in between a pair of dollar signs (`$`) is interpreted as a mathematical expression typeset in TeX. For example, the following command would place the label "θ" on the horizontal axis.

```
xlabel(r'$\theta$')
```

Suppose that you want to indicate that the angle θ is in radians. The following command would result in "$\theta(radians)$", which doesn't look very good. Since everything between the dollar signs is treated as a math, the text is italicized and the space is ignored.

```
xlabel(r'$\theta (radains)$')
```

Regular text and math text can be combined within the same string as shown below. This command would label the horizontal axis "θ (radians)", which looks much better. The text "radians" is not italicized and the space is included.

```
xlabel(r'$\theta$ (radians)')
```

Alternatively, the TeX command "`\rm`" can be used to make the font Roman for the plain text. Curly brackets must surround the text that you want to be affected, including the space. This method tends to produce the best results because all of the text has a consistent size.

```
xlabel(r'$\theta \rm{ (radians)}$')
```

Typesetting simple mathematical expressions in the TeX format is straightforward. The lower case and upper case Greek letters have fairly obvious names (for example, "`\delta`" is δ and "`\Delta`" is Δ). You can look up other common symbols (such as "`\inf`" is ∞) at the website given at the end of this tutorial. Subscripts and superscripts are made with "_" and "^" as shown below.

```
r'$\alpha_i = \beta^2$'
```
$\alpha_i = \beta^2$

If you want multiple characters to appear in a subscript or superscript, they can be grouped in curly brackets. Notice that a symbol can have both a subscript and a superscript. The subscript of *B* is plain text because it is preceded by "`\rm`", which changes the font to Roman.

```
r'$A_{ij}^3 < B_{\rm{max}}$'
```
$A_{ij}^3 < B_{\rm max}$

Common functions such as "sin" have shortcuts ("`\sin`" in this case) that typeset them in a Roman font. The result of the second command below is better than that of the first.

```
r'$sin\theta$'
```
$sin\theta$

```
r'$\sin\theta$'
```
$\sin\theta$

The following example shows how to make a fraction.

```
r'$C = \frac{3}{4}$'
```
$C = \frac{3}{4}$

Preceding brackets with "`\left`" and "`\right`" automatically makes them the right size for what's inside. The following examples show how this can be useful.

```
r'$(\frac{5}{3})$'
```
$(\frac{5}{3})$

```
r'$\left(\frac{5}{3}\right)$'
```
$\left(\frac{5}{3}\right)$

Another frequently used TeX command is "`\sqrt`" for a square root symbol.

```
r'$\sqrt{xy} = 3$'
```
$\sqrt{xy} = 3$

You can look up other symbols at the website given below. This brief introduction should be enough to get you started, but is not intended to be complete.

Numbers in Figure Text

It is also useful to include numbers that are calculated by a program in figure text. Suppose that a program fits data to find a time (`t`) is 5.43 and its uncertainty (`sigmat`) is 0.21. If you're plotting the data, you might want to put the text "$t = 5.4 \pm 0.2$ s" on the figure. You could use either of the following lines (the location might need to be adjusted), but you'd have to change the numbers by hand if the data changed. The second example with TeX formatting would look better.

```
figtext(0.5,0.5, 't = 5.4 +/- 0.2 s')
figtext(0.5,0.5, r'$t = 5.4 \pm 0.2 \rm{ s}$')
```

It would be better to use the variables `t` and `sigmat` in the command as shown below. In this example, each copy of "`%3.1f`" in the string is a format. The string is followed by a percent symbol (`%`) and a list of variables to be formatted. The number of formats and variables must match.

```
figtext(0.5,0.5, 't = %3.1f +/- %3.1f s' % (t,sigmat))
```

The form of the formatting in a string is "`%(width).(precision)(specifier)`", where `width` specifies the maximum number of digits, `precision` specifies the number of digits after the decimal point, and the possibilities for `specifier` are shown below. For integer formatting, the `precision` argument is ignored if you give it. For scientific notation and floating point formatting, the `width` argument is optional.

Specifier	Meaning	Example Format	Output for −34.5678
`i`	signed integer	`%5i`	`-34`
`e`	scientific notation	`%5.4e`	`-3.4568e+001`
`f`	floating point	`%5.2f`	`-34.57`

The formatting of the numerical output and TeX formatting can be combined (this is a single command that wraps onto a second line).

```
figtext(0.5,0.5, r'$t = %3.1f \pm %3.1f \rm{ s}$' %
        (t, sigmat) )
```

Additional documentation is available at:

http://matplotlib.sourceforge.net/users/mathtext.html