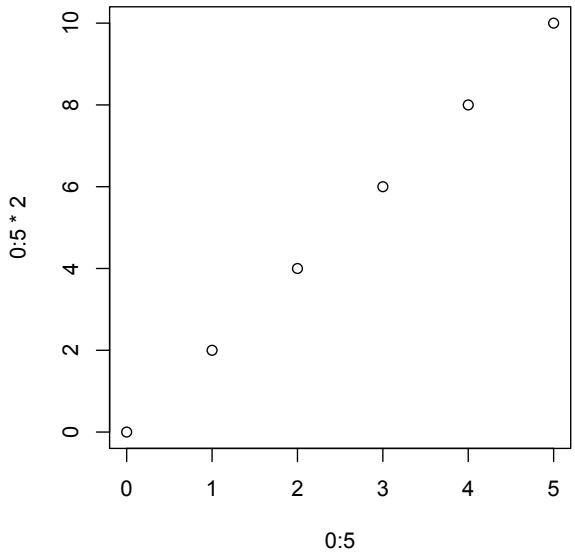
Using the plot command

plot(x,y) – plots a scatterplot with rings for each pair of (x,y)-values (x and y are vectors of the same length)

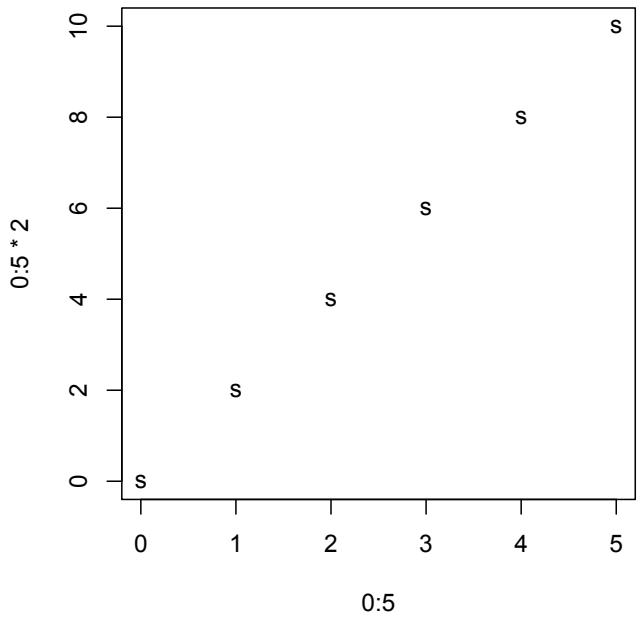
```
> plot(0:5,0:5*2)
```



Plotting character, `pch`

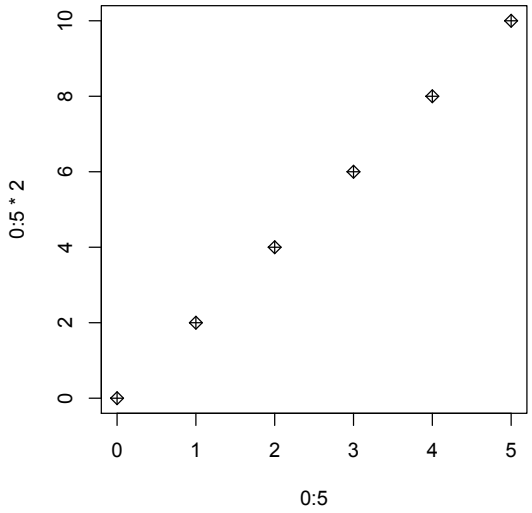
Use something other than rings: change the 'plotting character', `pch`. One can use an actual letter:

```
> plot(0:5,0:5*2,pch='s')
```



There are also numerical codes for more standard symbols. Choose one of these (1 is default):


```
> plot(0:5,0:5*2,pch=9)
```

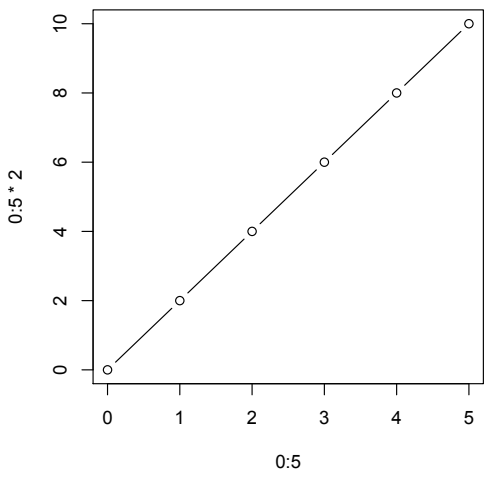


Plot type

To plot lines instead of symbols use the parameter 'type':

`"p"` for **p**oints, [default]
`"l"` for **l**ines,
`"b"` for **b**oth,
`"c"` for the lines part alone of `"b"`,
`"o"` for both '**o**verplotted',
`"h"` for '**h**istogram' like (or 'high-density') vertical lines,
`"s"` for stair **s**teps,
`"S"` for other **s**teps, see 'Details' below,
`"n"` for no plotting.

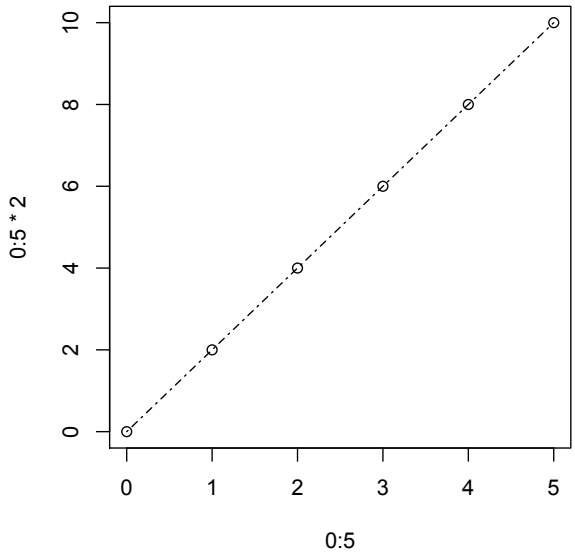
```
> plot(0:5,0:5*2,type='b')
```



Line type, `lty, lwd`

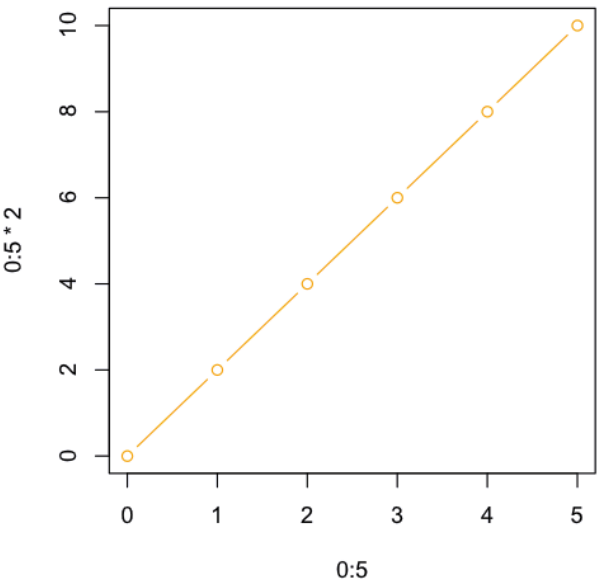
Use the parameter `lty` to change the line type. Line types can either be specified as an integer (0=blank, 1=solid (default), 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash) or as one of the character strings "blank", "solid", "dashed", "dotted", "dotdash", "longdash", or "twodash", where "blank" uses 'invisible lines' (i.e., does not draw them).

```
> plot(0:5,0:5*2,type='o',lty='dotdash')
```



Plot with color, `col`

```
> plot(0:5,0:5*2,type='b',col='orange')
```



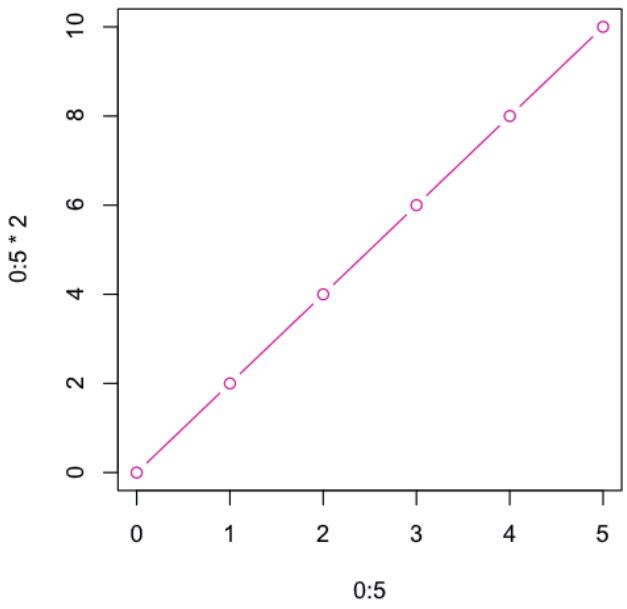
There are hundreds of available colors. Use colors() to see a list:

```
> colors()

 [1] "white"  "aliceblue"  "antiquewhite" "antiquewhite1" "antiquewhite2"
```

```
 [6] "antiquewhite3" "antiquewhite4" "aquamarine"  "aquamarine1"
"aquamarine2"

 [11] "aquamarine3"  "aquamarine4"  "azure"  "azure1"  "azure2"
```

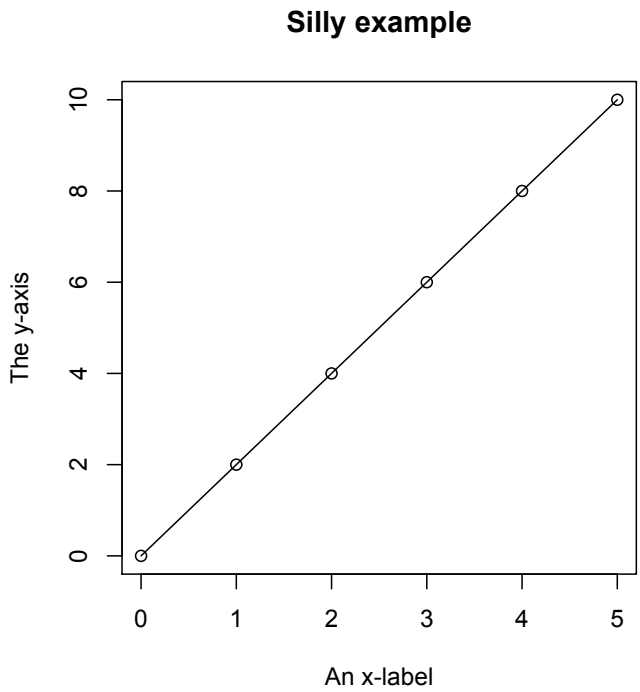
...

```
> plot(0:5,0:5*2,type='b',col='violetred')
```



Add axis labels and a title (`xlab, ylab, main, sub, cex`)

```
plot(0:5,0:5*2,type='o', xlab='An x-label', ylab='The y-axis',
main='Silly example')
```

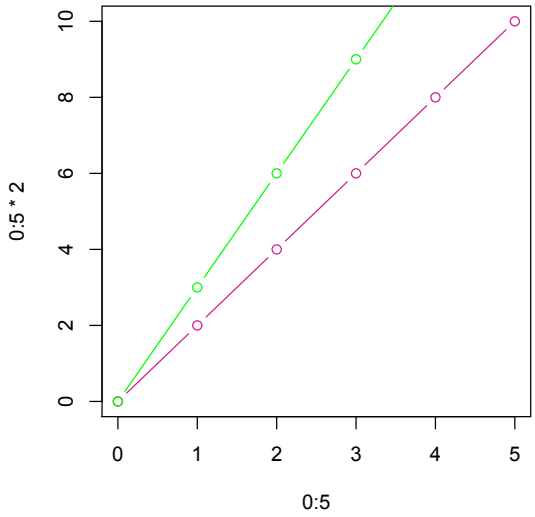


Adding more to an existing plot

Each time you use the plot command, a new graph is created. You can retrieve previous plots from the Quartz menu. Try it!

To add more stuff to an existing plot the commands `lines` and `points` are useful. They work almost exactly like the plot command itself (`lines` draws lines and `points` draws scatterplots).

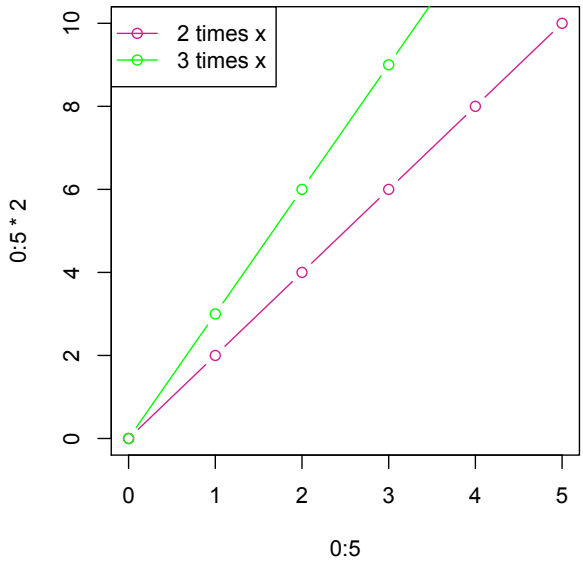
```
> plot(0:5,0:5*2,type='b',col='violetred',lty='solid')

> lines(0:5,0:5*3,type='b',col='green',lty='solid')
```



Add a legend:

The legend needs information on what each line looks like (lty, col and pch), and the accompanying text, typically added as lists or vectors, with one element per legend entry. To add a legend to the graph above use:

```
> legend('topleft', legend=c('2 times x','3 times x'),
lty='solid', pch=1, col=c('violetred','green'))
```



Notice that the legend command has a parameter called legend as well (containg a list of strings). Each parameter is repeated if more values are needed. For instance, both lines above are solid, so `lty='solid'` can be used instead of `lty=c('solid', 'solid')`. I have not yet figured out how to create a legend fully corresponding to `type='b'`.

The location of the legend (the first parameter) can be one of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center". One can also specify exact coordinates. See legend documentation for more options (`> ?legend`).

Several plots in one window

Use the `par( )` function with the `mfrow` option to tell R to subdivide (all the following) figure window into a set of subplots, and the fill them in row by row.

```
par( mfrow = c(2,3) ) # specify a 2-by-3 plot layout
```

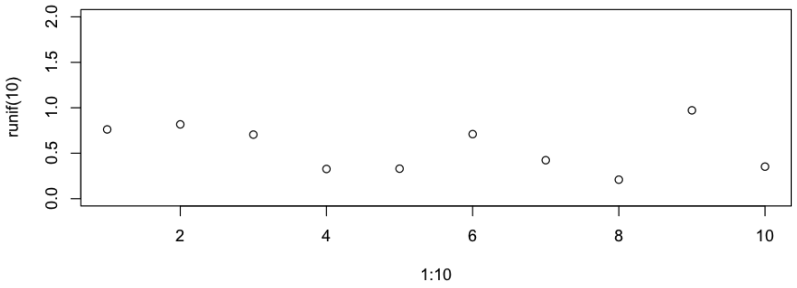
Using `mfcol` instead of `mfrow` fills in the layout by columns.

Other parameters

There are plenty of parameters that can be used to change the looks of a plot. Some useful ones are:

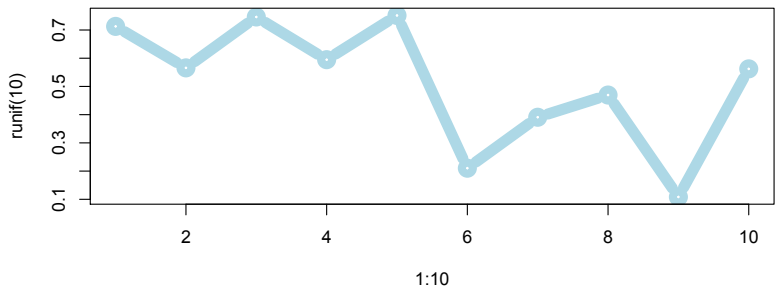
`xlim, ylim`: sets the limits of the x- and y-axis, respectively. The value is a vector with the lower and upper limit of the corresponding axis. Example:

```
plot(1:10, runif(10) , ylim=c(0,2))
```



`lwd`: changes the thickness of lines. Default is 1. Example:

```
plot(1:10, runif(10) , type='b', lwd=10, col='lightblue')
```

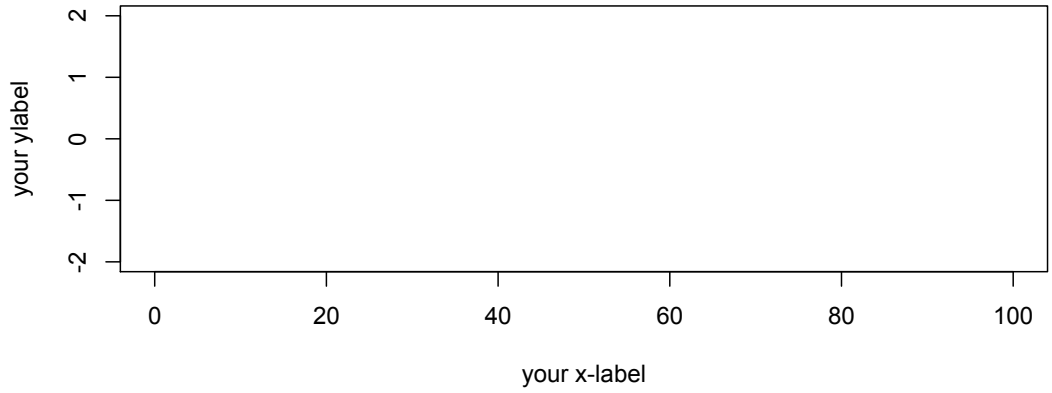


See the help system for further information (?plot, ?plot.default, ?points, ?lines, ?par)

Creating an empty plot

Sometimes it is necessary, or at least convenient, to create an empty plot first and add lines and other elements later. An empty plot can be created with a plot command, just make sure to specify the axis boundaries and make the type='n' (no plot):

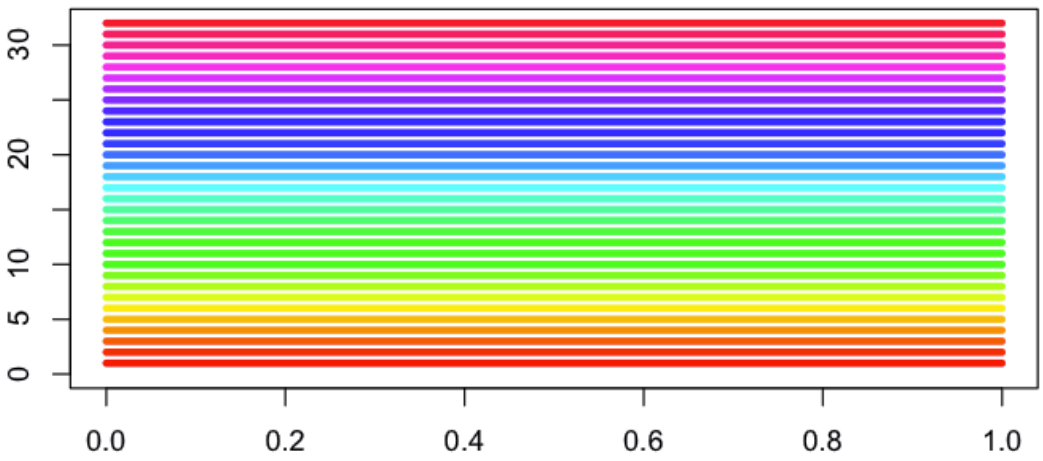
```
plot( NA, type='n', xlim=c(0,100), ylim=c(-2,2), xlab='your x-label', ylab='your ylabel' )
```



Palettes

Palettes is a handy way of drawing lines (or other objects) with different colors. Try this:

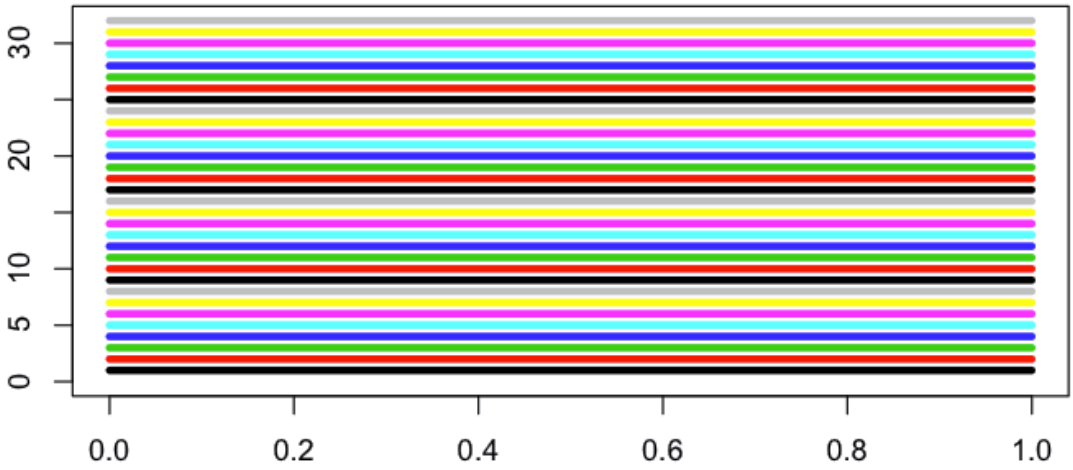
```
n <- 32
palette(rainbow(n))
plot(NA,type='n',xlim=c(0,1),ylim=c(0,n),xlab='',ylab='')
for (i in 1:n) {
      lines( c(0,1), c(i,i), col=i, lwd=4)
}
```



The `palette()` command creates a palette of, in this case, 32 colors from the rainbow palette. They can later be used as integer values, 1-32, of the `col` parameter.

Set the palette back to its default using `palette('default')`. The default palette (on my Mac) rotates between 8 colors. Try

```
n <- 32
palette("default")
plot(NA,type='n',xlim=c(0,1),ylim=c(0,n),xlab='',ylab='')
for (i in 1:n) {
      lines( c(0,1), c(i,i), col=i, lwd=4)
}
```



Use `?rainbow` to find other palettes besides `rainbow`.