การใช้ฟังก์ชัน plot( ) ในโปรแกรม R

รูปแบบ

```
plot(x, y, main="title", sub="subtitle",xlab="X-axis label",
     ylab="y-axix label",xlim=c(xmin, xmax),ylim=c(ymin, ymax),…)
```

รูปแบบ

```
title(main="My Title", col.main="red", sub="My Sub-title",
      col.sub="blue",xlab="My X label", ylab="My Y label",
     col.lab="green", cex.lab=0.75)
```

Text can be added to graphs using the **text( )** and **mtext( )** functions. **text( )** places text within the graph while **mtext( )** places text in one of the four margins.

```
text(location, "text to place", pos, ...)
mtext("text to place", side, line=n, ...)
```

Common options are described below.

option	description
location	location can be an **x,y coordinate**. Alternatively, the text can be placed interactively via mouse by specifying location as **locator(1)**.
pos	position relative to location. 1=below, 2=left, 3=above, 4=right. If you specify **pos**, you can specify **offset=** in percent of character width.
side	which margin to place text. 1=bottom, 2=left, 3=top, 4=right. you can specify **line=** to indicate the line in the margin starting with 0 and moving out. you can also specify **adj=0** for left/bottom alignment or **adj=1** for top/right alignment.

Other common options are **cex**, **col,** and **font** (for size, color, and font style respectively).

You can use the **text( )** function (see above) for labeling point as well as for adding other text annotations. Specify location as a set of x, y coordinates and specify the text to place as a vector of labels. The x, y, and label vectors should all be the same length.

```
# Example of labeling points
attach(mtcars)
plot(wt, mpg, main="Milage vs. Car Weight",
   xlab="Weight", ylab="Mileage", pch=18, col="blue")
text(wt, mpg, row.names(mtcars), cex=0.6, pos=4, col="red")
```

You can add mathematically formulas to a graph using TEX-like rules. See **help(plotmath)** for details and examples.

You can create custom axes using the **axis( )** function.

```
axis(side, at=, labels=, pos=, lty=, col=, las=, tck=, ...)
```

where

option	description
side	an integer indicating the side of the graph to draw the axis (1=bottom, 2=left, 3=top, 4=right)
at	a numeric vector indicating where tic marks should be drawn
labels	a character vector of labels to be placed at the tickmarks (if NULL, the *at* values will be used)
pos	the coordinate at which the axis line is to be drawn. (i.e., the value on the other axis where it crosses)
lty	line type
col	the line and tick mark color
las	labels are parallel (=0) or perpendicular(=2) to axis
tck	length of tick mark as fraction of plotting region (negative number is outside graph, positive number is inside, 0 suppresses ticks, 1 creates gridlines) default is -0.01
(...)	other graphical parameters

If you are going to create a custom axis, you should suppress the axis automatically generated by your high level plotting function. The option **axes=FALSE** suppresses both x and y axes. **xaxt="n"** and **yaxt="n"** suppress the x and y axis respectively. Here is a (somewhat overblown) example.

```
# A Silly Axis Example

# specify the data
x <- c(1:10); y <- x; z <- 10/x

# create extra margin room on the right for an axis
par(mar=c(5, 4, 4, 8) + 0.1)

# plot x vs. y
plot(x, y,type="b", pch=21, col="red",
   yaxt="n", lty=3, xlab="", ylab="")
```

```
# add x vs. 1/x
lines(x, z, type="b", pch=22, col="blue", lty=2)

# draw an axis on the left
axis(2, at=x,labels=x, col.axis="red", las=2)

# draw an axis on the right, with smaller text and ticks
axis(4, at=z,labels=round(z,digits=2),
  col.axis="blue", las=2, cex.axis=0.7, tck=-.01)

# add a title for the right axis
mtext("y=1/x", side=4, line=3, cex.lab=1,las=2, col="blue")

# add a main title and bottom and left axis labels
title("An Example of Creative Axes", xlab="X values",
   ylab="Y=X"
```

The **minor.tick( )** function in the **Hmisc** package adds minor tick marks.

```
# Add minor tick marks
library(Hmisc)
minor.tick(nx=n, ny=n, tick.ratio=n)
```

nx is the number of minor tick marks to place between x-axis major tick marks. **ny** does the same for the y-axis. **tick.ratio** is the size of the minor tick mark relative to the major tick mark. The length of the major tick mark is retrieved from **par(**"tck"**)**.

Add reference lines to a graph using the **abline( )** function.

```
abline(h=yvalues, v=xvalues)
```

Other graphical parameters (such as line type, color, and width) can also be specified in the abline( ) function.

```
# add solid horizontal lines at y=1,5,7
abline(h=c(1,5,7))
# add dashed blue verical lines at x = 1,3,5,7,9
abline(v=seq(1,10,2),lty=2,col="blue")
```

Note: You can also use the **grid( )** function to add reference lines.

Add a legend with the **legend()** function.

```
legend(location, title, legend, ...)
```

Common options are described below.

option	description
location	There are several ways to indicate the location of the legend. You can give an **x,y coordinate** for the upper left hand corner of the legend. You can use **locator(1)**, in which case you use the mouse to indicate the location of the legend. You can also use the **keywords** "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "bottomright", or "center". If you use a keyword, you may want to use **inset=** to specify

	an amount to move the legend into the graph (as fraction of plot region).
title	A character string for the legend title (optional)
legend	A character vector with the labels
...	Other options. If the legend labels colored lines, specify **col=**and a vector of colors. If the legend labels point symbols, specify **pch=** and a vector of point symbols. If the legend labels line width or line style, use **lwd=** or **lty=** and a vector of widths or styles. To create colored boxes for the legend (common in bar, box, or pie charts), use **fill=** and a vector of colors.

Other common legend options include **bty** for box type, **bg** for background color, **cex** for size, and **text.col** for text color. Setting **horiz=TRUE** sets the legend horizontally rather than vertically.

```
# Legend Example
attach(mtcars)
boxplot(mpg~cyl, main="Milage by Car Weight",
   yaxt="n", xlab="Milage", horizontal=TRUE,
   col=terrain.colors(3))
legend("topright", inset=.05, title="Number of Cylinders",
   c("4","6","8"), fill=terrain.colors(3), horiz=TRUE)
```

For more on legends, see **help(legend)**. The examples in the help are particularly informative.

Combining Plots

R makes it easy to combine multiple plots into one overall graph, using either the **par( )** or **layout( )** function.

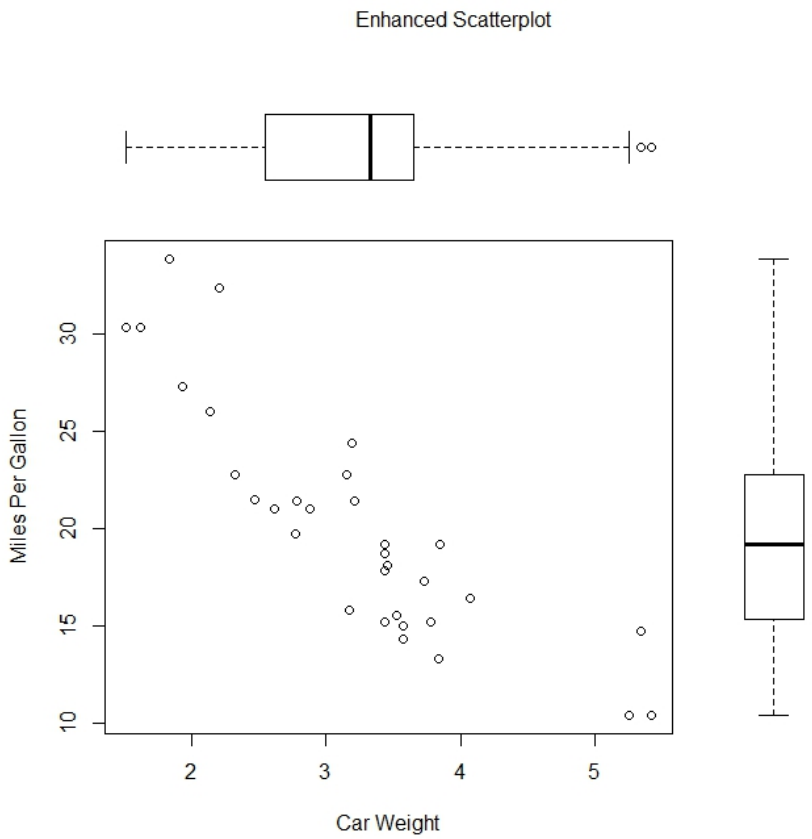
With the **par( )** function, you can include the option **mfrow=c(***nrows*, *ncols***)** to create a matrix of *nrows x ncols* plots that are filled in by row. **mfcol=c(***nrows*, *ncols***)** fills in the matrix by columns.

```
# 4 figures arranged in 2 rows and 2 columns
attach(mtcars)
par(mfrow=c(2,2))
plot(wt,mpg, main="Scatterplot of wt vs. mpg")
plot(wt,disp, main="Scatterplot of wt vs disp")
hist(wt, main="Histogram of wt")
boxplot(wt, main="Boxplot of wt")
```

Creating a figure arrangement with fine control

In the following example, two box plots are added to scatterplot to create an enhanced graph.

```
# Add boxplots to a scatterplot
par(fig=c(0,0.8,0,0.8), new=TRUE)
plot(mtcars$wt, mtcars$mpg, xlab="Car Weight",
  ylab="Miles Per Gallon")
par(fig=c(0,0.8,0.55,1), new=TRUE)
boxplot(mtcars$wt, horizontal=TRUE, axes=FALSE)
par(fig=c(0.65,1,0,0.8),new=TRUE)
boxplot(mtcars$mpg, axes=FALSE)
mtext("Enhanced Scatterplot", side=3, outer=TRUE, line=-3)
```



To understand this graph, think of the full graph area as going from (0,0) in the lower left corner to (1,1) in the upper right corner. The format of the **fig=** parameter is a numerical vector of the form c(x1, x2, y1, y2). The first **fig=** sets up the scatterplot going from 0 to 0.8 on the x axis and 0 to 0.8 on the y axis. The top boxplot goes from 0 to 0.8 on the x axis and 0.55 to 1 on the y axis. I chose 0.55 rather than 0.8 so that the top figure will be pulled closer to the scatter plot. The right hand boxplot goes from 0.65 to 1 on the x axis and 0 to 0.8 on the y axis. Again, I chose a value to pull the right hand boxplot closer to the scatterplot. You have to experiment to get it just right. **fig=** starts a new plot, so to add to an existing plot use **new=TRUE**.

You can use this to combine several plots in any arrangement into one graph.

Graphical Parameters

You can customize many features of your graphs (fonts, colors, axes, titles) through graphic options.

One way is to specify these options in through the **par( )** function. If you set parameter values here, the changes will be in effect for the rest of the session or until you change them again. The format is **par(*optionname*=*value*, *optionname*=*value*, ...)**

```
# Set a graphical parameter using par()

par()               # view current settings
opar <- par()       # make a copy of current settings
par(col.lab="red") # red x and y labels
hist(mtcars$mpg)   # create a plot with these new settings
par(opar)           # restore original settings
```

A second way to specify graphical parameters is by providing the ***optionname***=*value*pairs directly to a high level plotting function. In this case, the options are only in effect for that specific graph.

```
# Set a graphical parameter within the plotting function
hist(mtcars$mpg, col.lab="red")
```

See the help for a specific high level plotting function (e.g. plot, hist, boxplot) to determine which graphical parameters can be set this way.

The remainder of this section describes some of the more important graphical parameters that you can set.

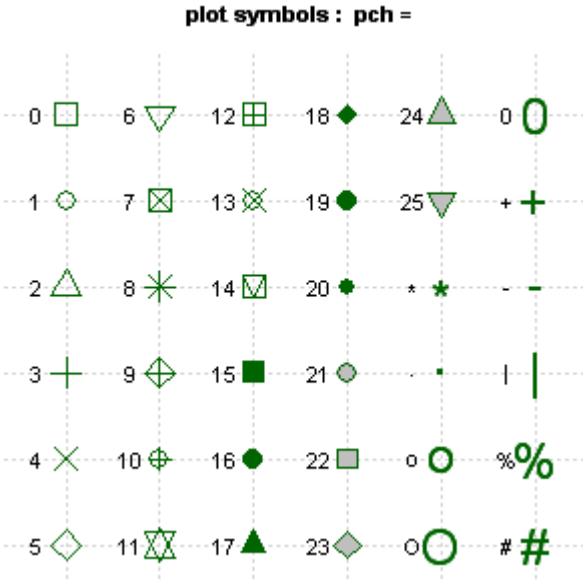
Text and Symbol Size

The following options can be used to control text and symbol size in graphs.

option	description
cex	number indicating the amount by which plotting text and symbols should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, etc.
cex.axis	magnification of axis annotation relative to cex
cex.lab	magnification of x and y labels relative to cex
cex.main	magnification of titles relative to cex
cex.sub	magnification of subtitles relative to cex

Plotting Symbols

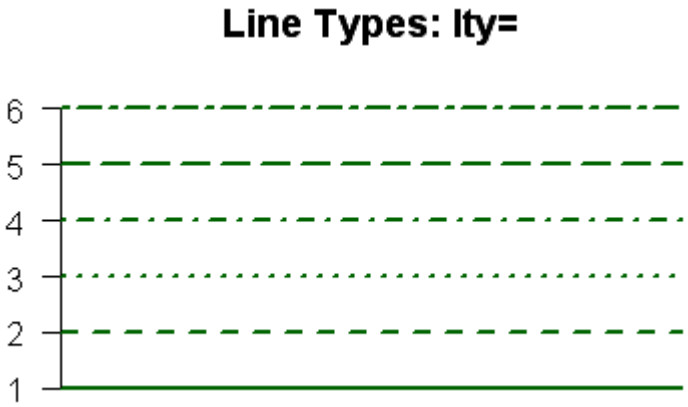
Use the **pch=** option to specify symbols to use when plotting points. For symbols 21 through 25, specify border color (col=) and fill color (bg=).



Lines

You can change lines using the following options. This is particularly useful for reference lines, axes, and fit lines.

option	description
lty	line type. see the chart below.
lwd	line width relative to the default (default=1). 2 is twice as wide.



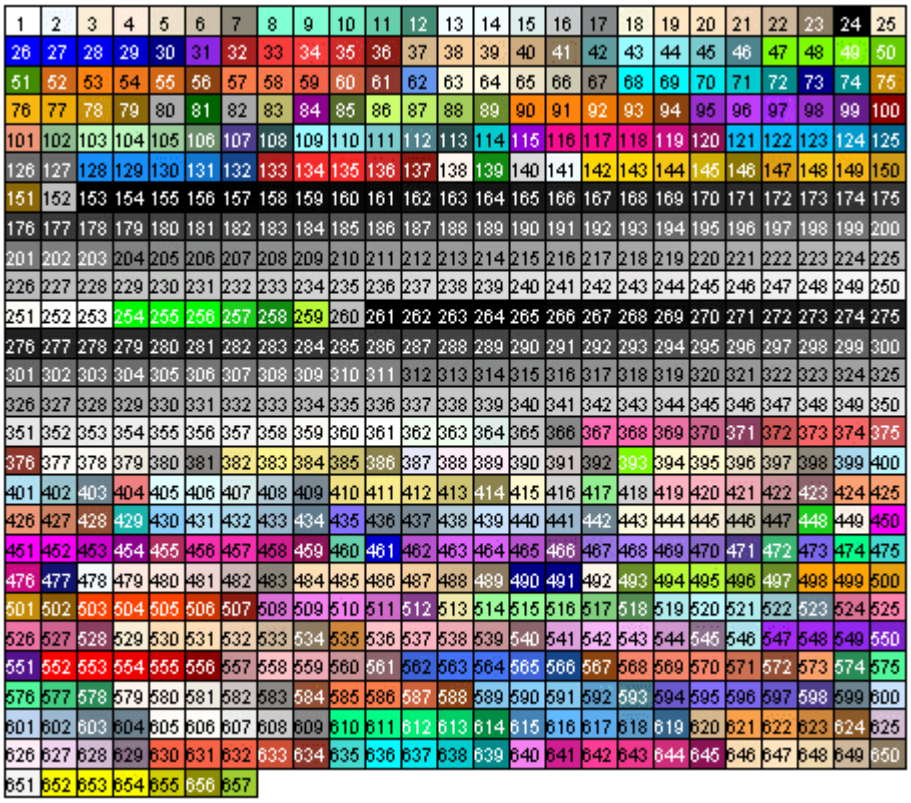
Colors

Options that specify colors include the following.

option	description
col	Default plotting color. Some functions (e.g. lines) accept a vector of values that are recycled.
col.axis	color for axis annotation
col.lab	color for x and y labels
col.main	color for titles
col.sub	color for subtitles
fg	plot foreground color (axes, boxes - also sets col= to same)
bg	plot background color

You can specify colors in R by index, name, hexadecimal, or RGB. For example **col=1**, **col="white"**, and **col="#FFFFFF"** are equivalent.

The following chart was produced with code developed by Earl F. Glynn. See his Color Chart for all the details you would ever need about using colors in R.



You can also create a vector of *n* contiguous colors using the functions **rainbow(*n*)**, **heat.colors(*n*)**, **terrain.colors(*n*)**, **topo.colors(*n*)**, and **cm.colors(*n*)**.

colors() returns all available color names.

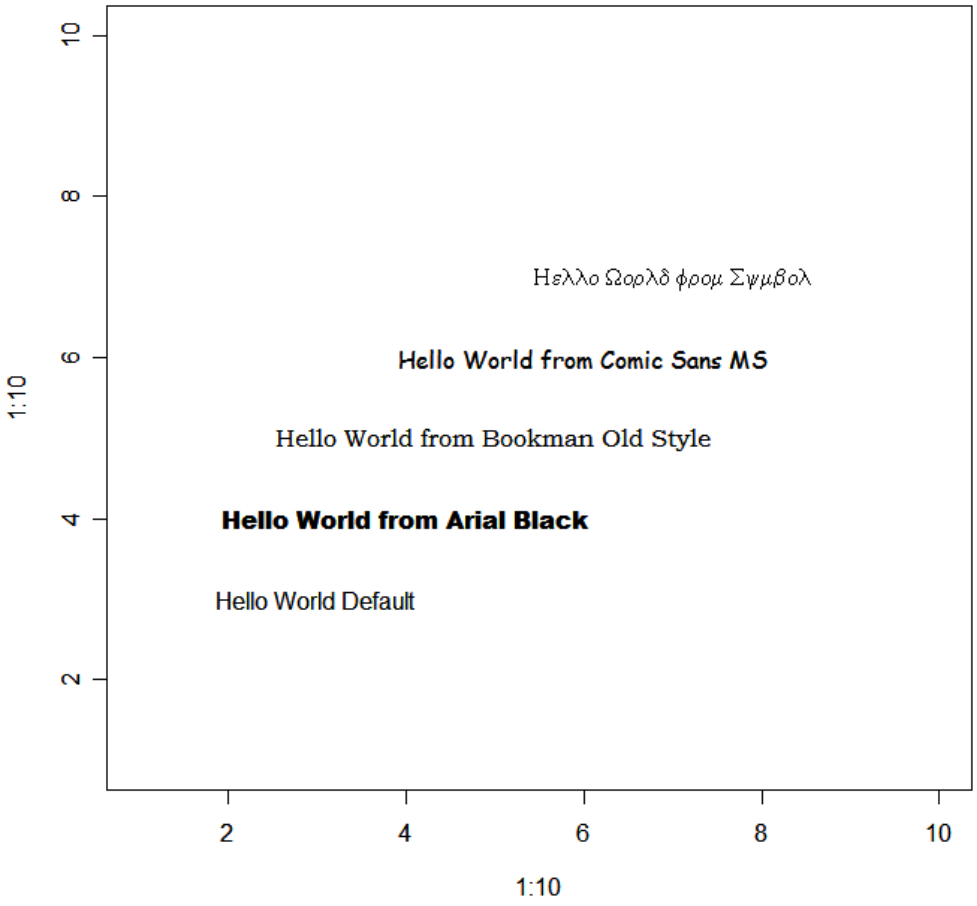
Fonts

You can easily set font size and style, but font family is a bit more complicated.

option	description
font	Integer specifying font to use for text. 1=plain, 2=bold, 3=italic, 4=bold italic, 5=symbol
font.axis	font for axis annotation
font.lab	font for x and y labels
font.main	font for titles
font.sub	font for subtitles
ps	font point size (roughly 1/72 inch) text size=ps*cex
family	font family for drawing text. Standard values are "serif", "sans", "mono", "symbol". Mapping is device dependent.

In windows, mono is mapped to "TT Courier New", serif is mapped to"TT Times New Roman", sans is mapped to "TT Arial", mono is mapped to "TT Courier New", and symbol is mapped to "TT Symbol" (TT=True Type). You can add your own mappings.

```
# Type family examples - creating new mappings
plot(1:10,1:10,type="n")
windowsFonts(
  A=windowsFont("Arial Black"),
  B=windowsFont("Bookman Old Style"),
  C=windowsFont("Comic Sans MS"),
  D=windowsFont("Symbol")
)
text(3,3,"Hello World Default")
text(4,4,family="A","Hello World from Arial Black")
text(5,5,family="B","Hello World from Bookman Old Style")
text(6,6,family="C","Hello World from Comic Sans MS")
text(7,7,family="D", "Hello World from Symbol")
```



Margins and Graph Size

You can control the margin size using the following parameters.

option	description
mar	numerical vector indicating margin size c(bottom, left, top, right) in lines. default = c(5, 4, 4, 2) + 0.1
mai	numerical vector indicating margin size c(bottom, left, top, right) in inches
pin	plot dimensions (width, height) in inches

For complete information on margins, see Earl F. Glynn's margin tutorial.

Going Further

See **help(par)** for more information on graphical parameters. The customization of plotting axes and text annotations are covered next section.