3. First plots

R provides a wide variety of tools for data visualization. The base R graphics package allows for simple plots, while more advanced visualizations can be created using external libraries like `ggplot2`.

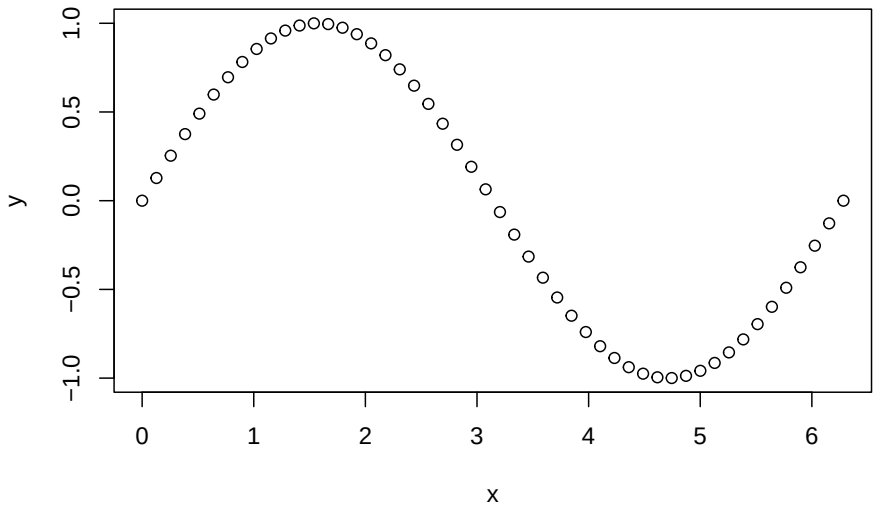
Plotting with the base R is done using the `plot()` function. This function can be used for creating scatter plots, line plots, and more.

3.1 Basic Scatter Plots

If not specified otherwise, the plot function creates a Scatter plot.

```
x <- seq(0,2*pi,length.out=50)
y <- sin(x)

# Basic scatter plot
plot(x, y)
```



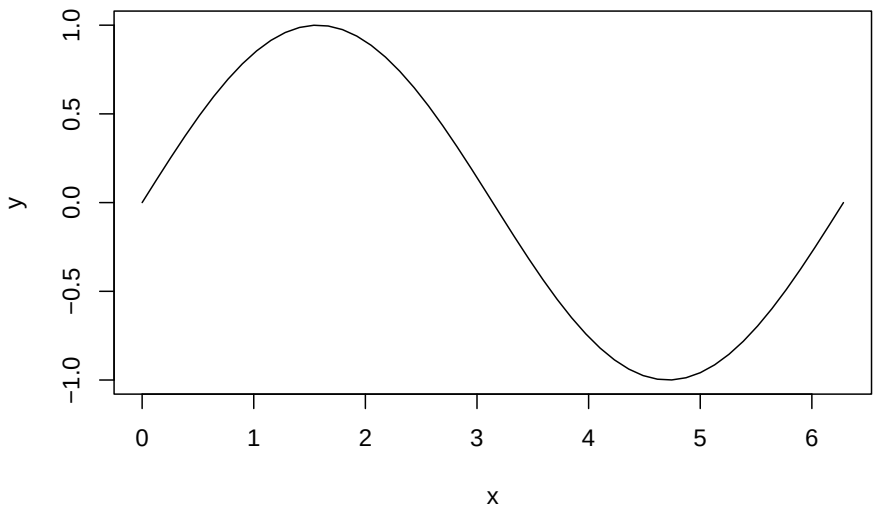
3.2 Other Plot Types

Using the optional argument `type` one can change the plot type, e.g.

- `type="l"`: Line plot
- `type="b"`: points connected by lines
- `type="o"`: Line plot where observations are drawn as points
- `type="h"`: Histogram-like vertical lines
- `type="S"`: Stair steps (for step functions)

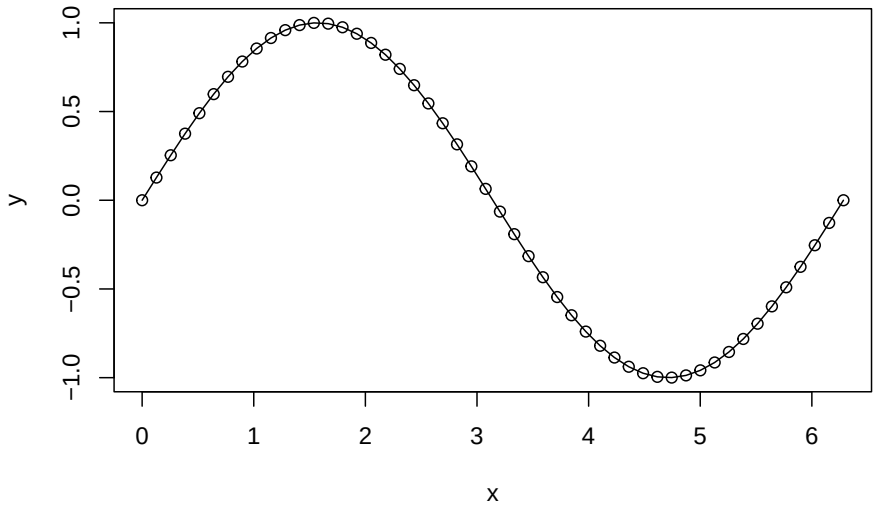
Example 1 (Line Plot)

```
plot(x, y, type="l")
```



Example 2 (Lines and Points)

```
plot(x, y, type="o")
```

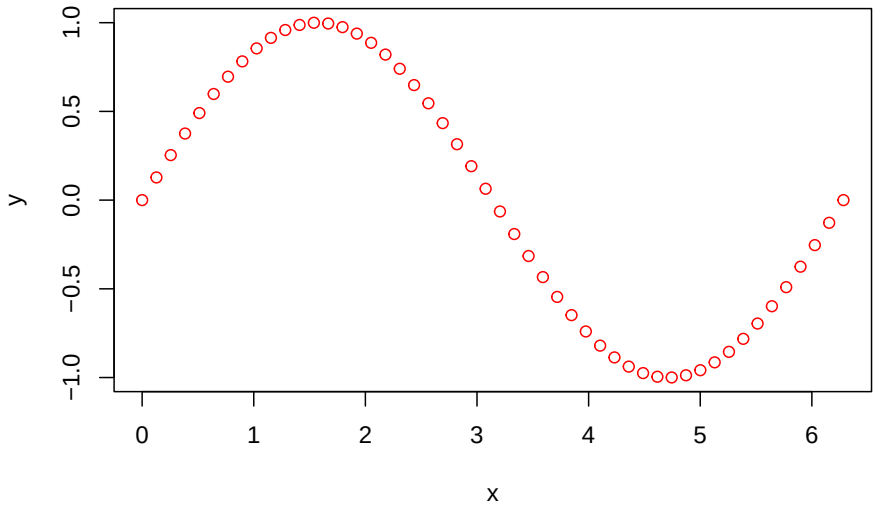


3.2 Colors

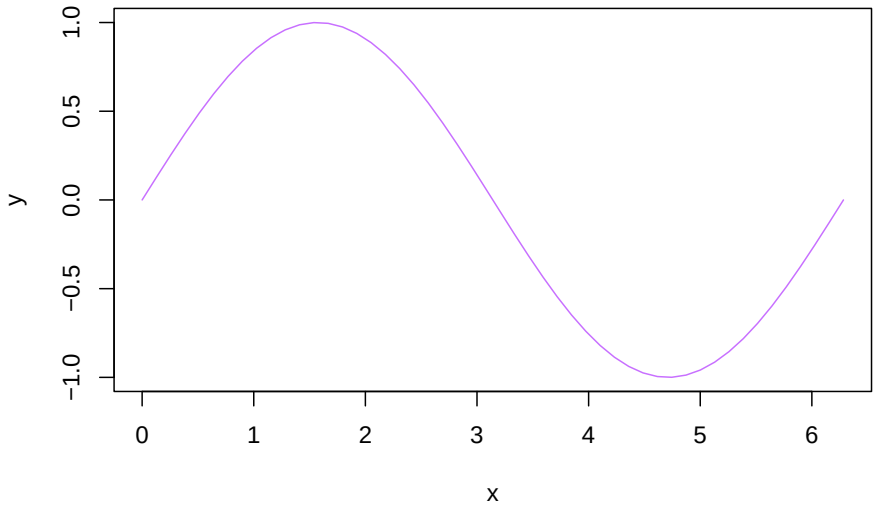
Using the optional `col` argument, the line or character colour can be adapted. The colour can either be specified by its name (Predefined colors) or by using the colours rgb specification (`rgb(red, green, blue, alpha)`)

Examples

```r
plot(x, y, col="red")
```




```r
plot(x, y, type="l", col=rgb(200, 112, 255, maxColorValue = 255))
```



3.3 Point Characters

Instead of using a hollow circle as point character, one can choose another point character (Predefined Point Characters) by specifying the `pch` argument.

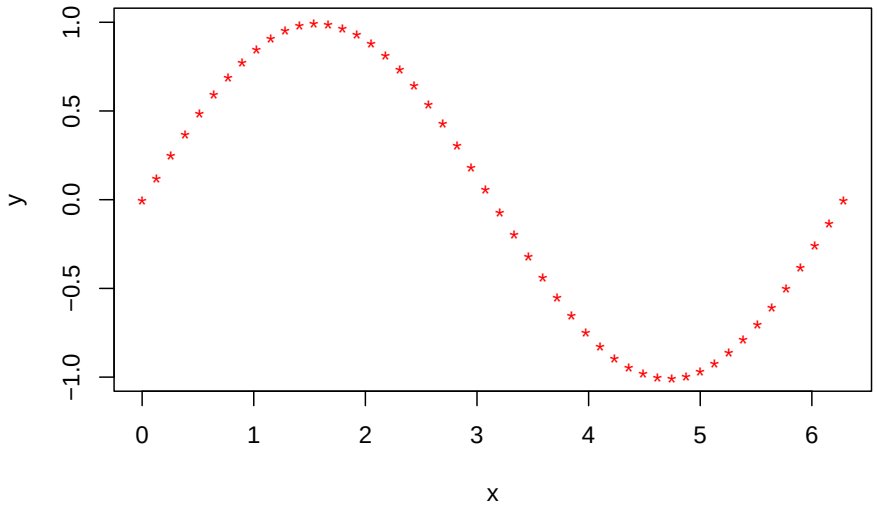
Some common `pch`-values are

- `pch=19`: Solid circle.
- `pch=0`: Square.

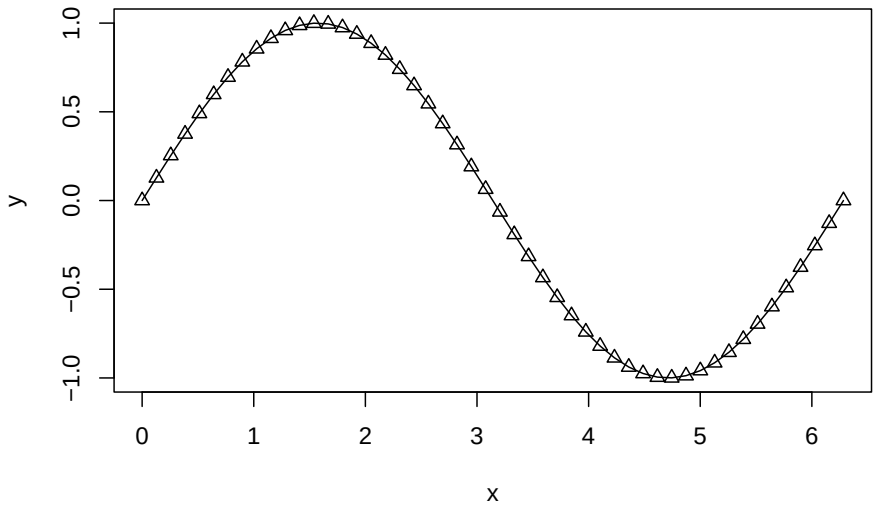
- `pch=1`: Hollow circle.
- `pch=3`: Plus sign.

Examples

```
plot(x, y, col="red", pch="*")
```




```
plot(x, y, type="o", pch=2)
```



3.4 Line types

The line type can be changed using the `lty` argument.

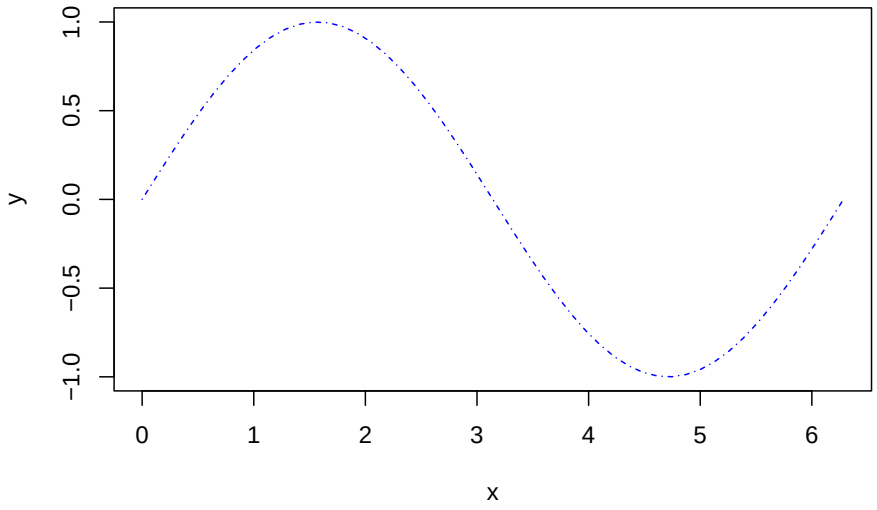
Some common `lty`-values are

- `lty=1`: Solid line.
- `lty=2`: Dashed line.
- `lty=3`: Dotted line.

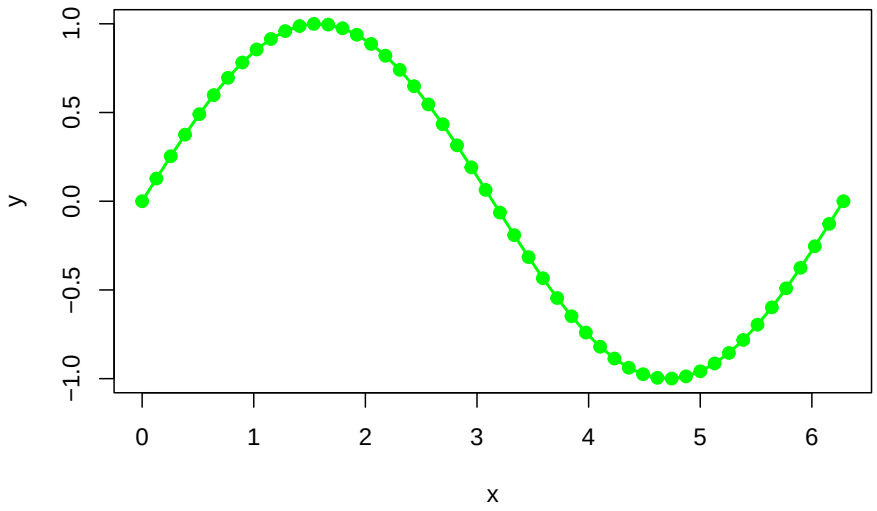
With `lwd` the line width can be specified.

Examples

```
plot(x, y, type="l", col="blue", lty=4)
```




```
plot(x, y, type="o", pch=19, lty=1, col="green", lwd=2)
```



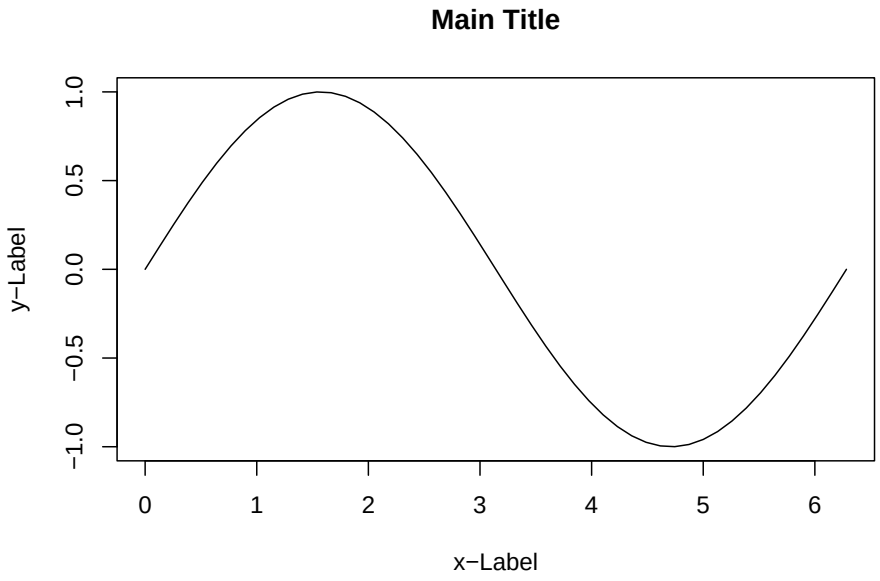
3.5 Axis Labelling and Titles

You can add titles and axis labels using

- `main`: Adds a main title to the plot
- `xlab`: Adds a label to the x-axis
- `ylab`: Adds a label to the y-axis

Example

```
plot(x, y, type="l", main = "Main Title",
     xlab = "x-Label", ylab = "y-Label")
```



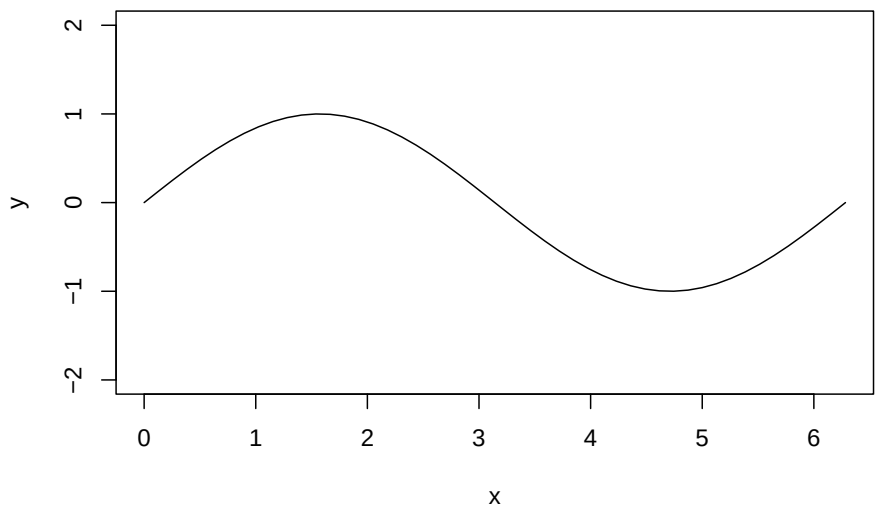
Axis Limits

You set the axis limits using

- `xlim`: range for the x-axis.
- `ylim`: range for the y-axis.

Example

```
plot(x, y, type="l", xlim = c(0,2*pi), ylim=c(-2,2))
```

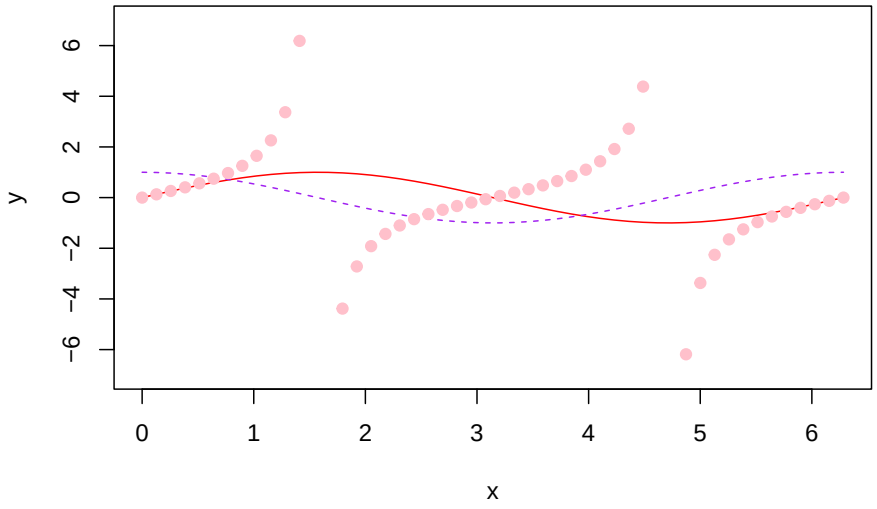


3.6 Multiple lines

If you would like to display multiple functions in the same plot, then the typical approach is to first use `plot()` for the initial function and then use `lines()` or `points()` to overlay the additional functions or points.

Example

```
z <- cos(x)
tan <- tan(x)
plot(x, y, type="l", col="red", ylim = c(-7,7))
lines(x,z, lty=2, col="purple")
points(x, tan, pch=19, col="pink")
```



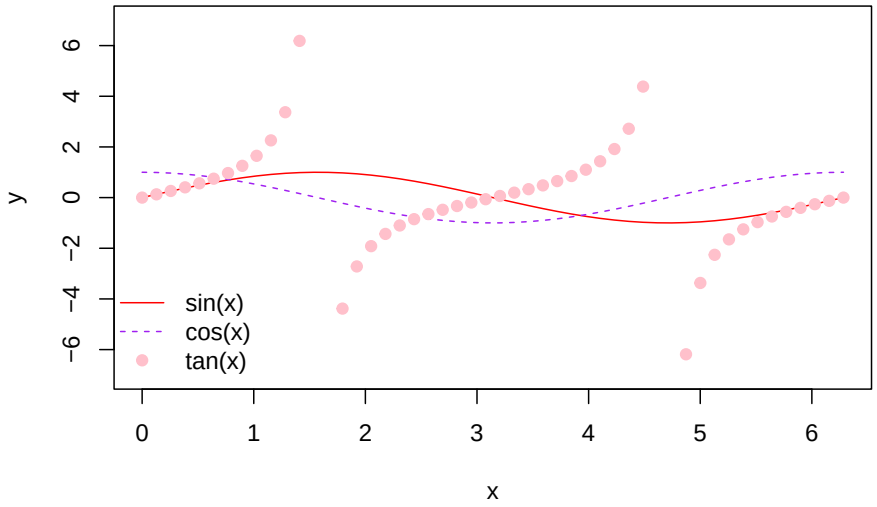
3.6 Legends

As soon as you display multiple functions in the same plot it is helpful to additionally add a legend to your plot.

Example

```
# draw plot
plot(x, y, type="l", col="red", ylim = c(-7,7))
lines(x,z, lty=2, col="purple")
points(x, tan, pch=19, col="pink")

# add legend
legend("bottomleft", legend = c("sin(x)", "cos(x)", "tan(x)"), bty = "n",
       col = c("red", "purple", "pink"),
       lty = c(1, 2, NA), pch = c(NA, NA, 19))
```



3.6 Multiple Plots Side by Side

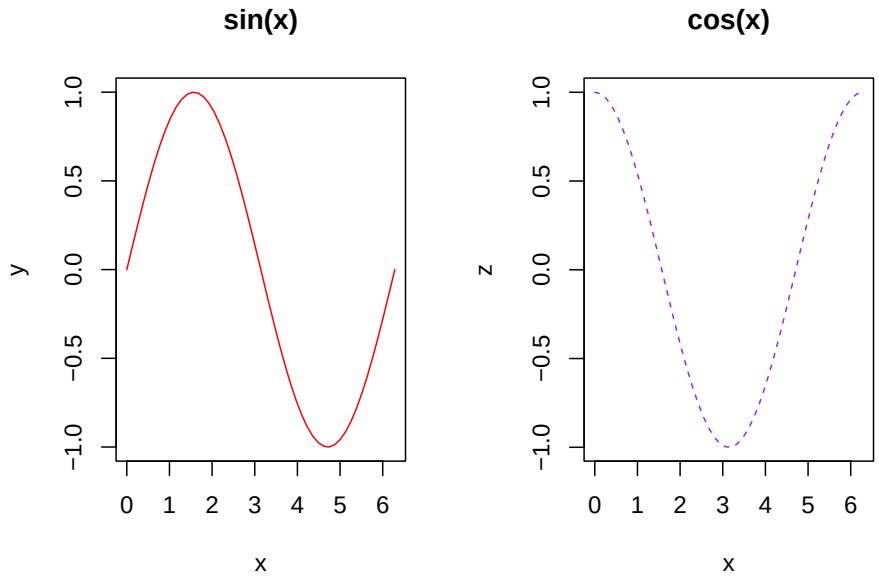
If you want more control over the plotting area, use `par()` to adjust the margins, layout, and other graphical parameters before plotting. Here, we only look at the option to have multiple plots displayed in a grid.

Example

```
# Set up the plotting area with multiple panels (1 row, 2 column)
par(mfrow = c(1, 2))

# draw 1st plot
plot(x, y, type="l", col="red", main="sin(x)")

# draw 2nd plot
plot(x,z, type="l", lty=2, col="purple", main="cos(x)")
```




```
# reset plotting area
par(mfrow = c(1, 1))
```

Exercise:

Draw the exponential function exp(x) for $-2 < x < 2$, and draw in the same plot the natural logarithm $log(y)$ for suitable $y > 0$. Your plot should highlight the fact that $log()$ is the inverse function of $exp()$, i.e. the graph of $log()$ is just the graph of $exp()$ mirrored at the main diagonal ($x = y$). When producing the first plot, you can specify the range of the x and y axis with the optional arguments xlim and ylim, respectively. Learn about the function abline() and use it to add axes and the main diagonal.

Solution

```
x <- seq(-4.5,4.5,0.001)
y <- exp(x)
#range(y)
plot(c(-4.5,4.5),c(0,0),type="l",col="green",
     xlab='x',ylab='f(x)',
     xlim=c(-4,4),ylim=c(-4,4),
     main="Exponential function and natural logarithm")
abline(v=0,col='green')
abline(a=0,b=1,col='magenta')
lines(x,y,lwd=2)
lines(y,x,lwd=2,col='blue')
legend(x='topleft',
       lty=1,
       lwd=2,
       legend=c('f(x) = exp(x)','f(x) = ln(x)'),
       col=c('black','blue'))
```