# R Basiskurs Beispiel (xaringan)

## Descriptive Statistics in R

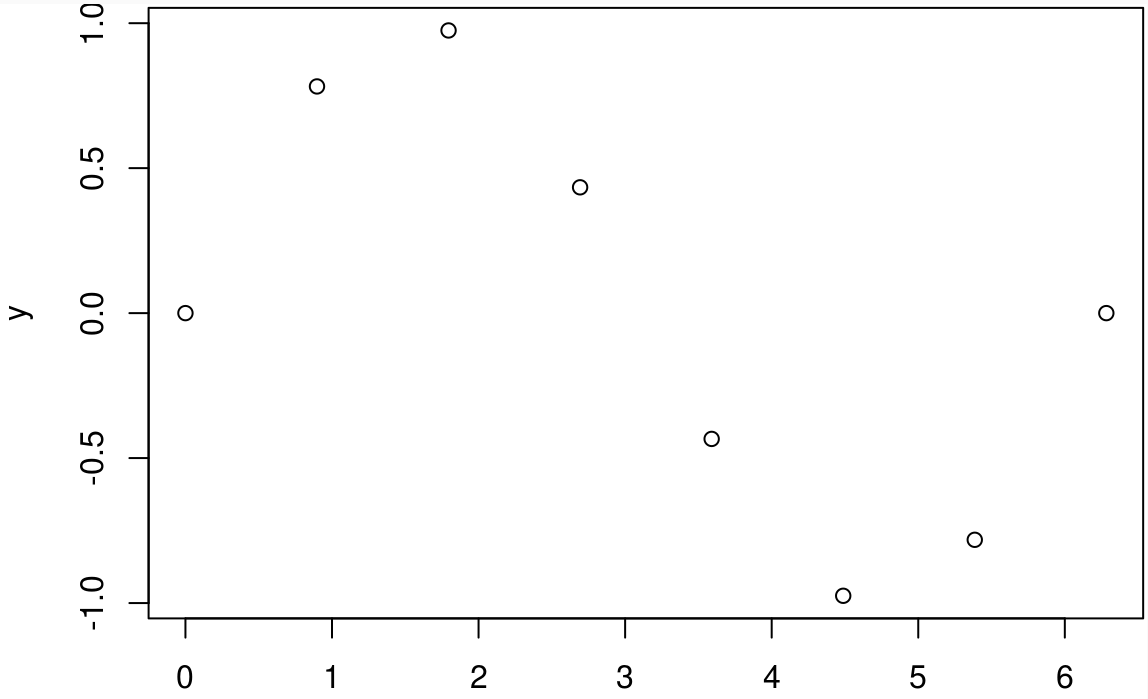
Maximilian Mandl & Daniel Schalk
12. April – 13. April 2018

# Graphics in R



`R` is a powerful tool to generate graphics. The simplest way to produce graphics is to use the `plot` function:

```
par(mar = c(4,4,0.1,0.1))
x = seq(0, 2*pi, length = 8)
y = sin(x)
plot(x, y)
```



# Multiple plots



The `plot` function starts a new graphic device every time it is executed. For multiple plots in one graphic device, the `par()` function with the argument `mfrow` can be used:

```
par(mfrow = c(nrows, ncols))
```

where `nrows` and `ncols` control the dimension of the plotting grid.

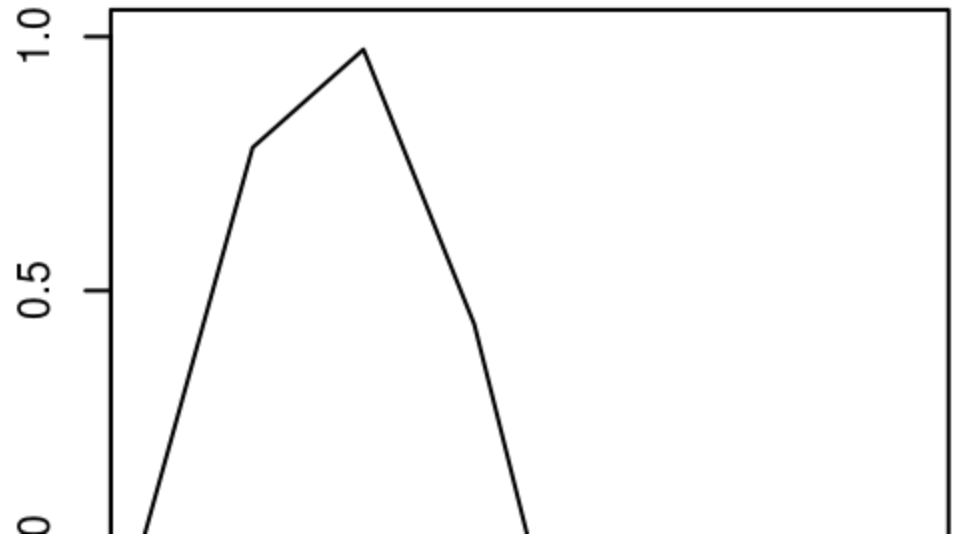
Note: This will affect all following plots!

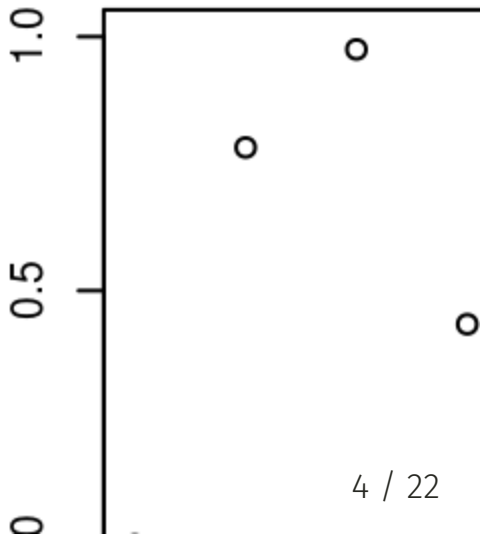
Use `par(mfrow = c(1, 1))` to reset this behavior.

# Graphical parameters

Plots can be customized by changing **graphical parameters**, e.g., the plot type can be specified using the `type` argument:

```
par(mfrow = c(1,3)) # multiple plots (1 row and 3 columns)
plot(x, y, type = "l") # for lines
plot(x, y, type = "p") # for points
plot(x, y, type = "b") # for both
```



# Graphical parameters for labeling



Some important graphical parameters are for labeling are:

- `main` specifies the title of the plot
- `xlab`, `ylab` changes the label of the horizontal/vertical axis

```
par(mfrow = c(1,3)) # multiple plots (1 row and 3 columns)
plot(x, y, main = "sin")
plot(x, y, main = "sin", xlab = "x values")
plot(x, y, main = "sin", xlab = "x values", ylab = "y values")
```

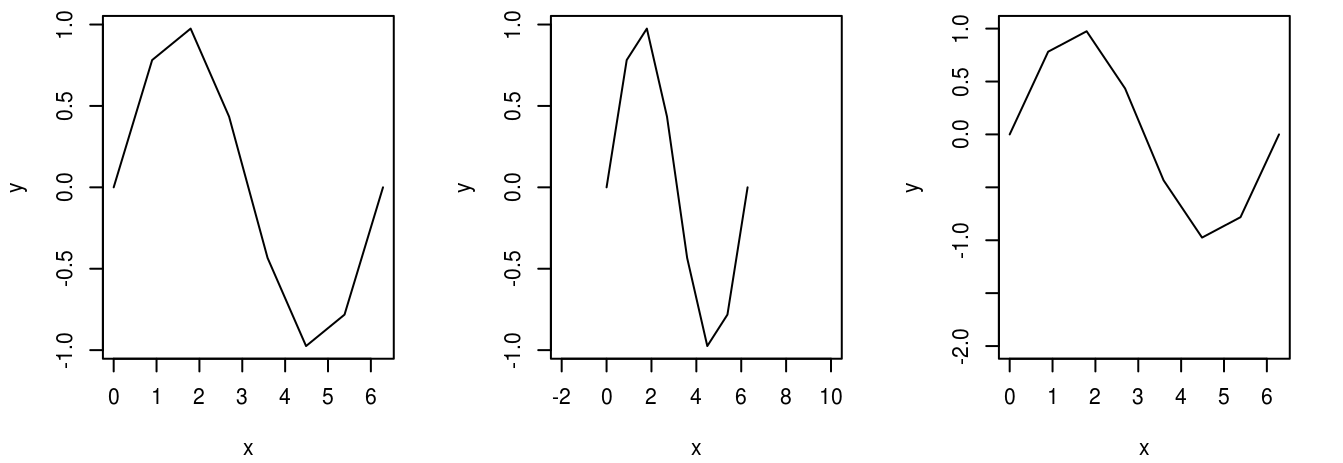
# Other graphical parameters



## `xlim` and `ylim`

The ranges for the x-axis an y-axis coordinates can be changed using `xlim` and `ylim`, respectively:

```
par(mfrow = c(1,3)) # multiple plots (1 row and 3 columns)
plot(x, y, type = "l") # ranges are chosen automatically
plot(x, y, type = "l", xlim = c(-2, 10)) # set ranges for x axis
plot(x, y, type = "l", ylim = c(-2, 1)) # set ranges for y axis
```



# Other graphical parameters



## `lty`

The line type can be specified using the `lty` argument:

```
par(mfrow = c(1,2)) # multiple plots (1 row and 2 columns)
plot(x, y, type = "l", lty = 1)
plot(x, y, type = "l", lty = 2)
```

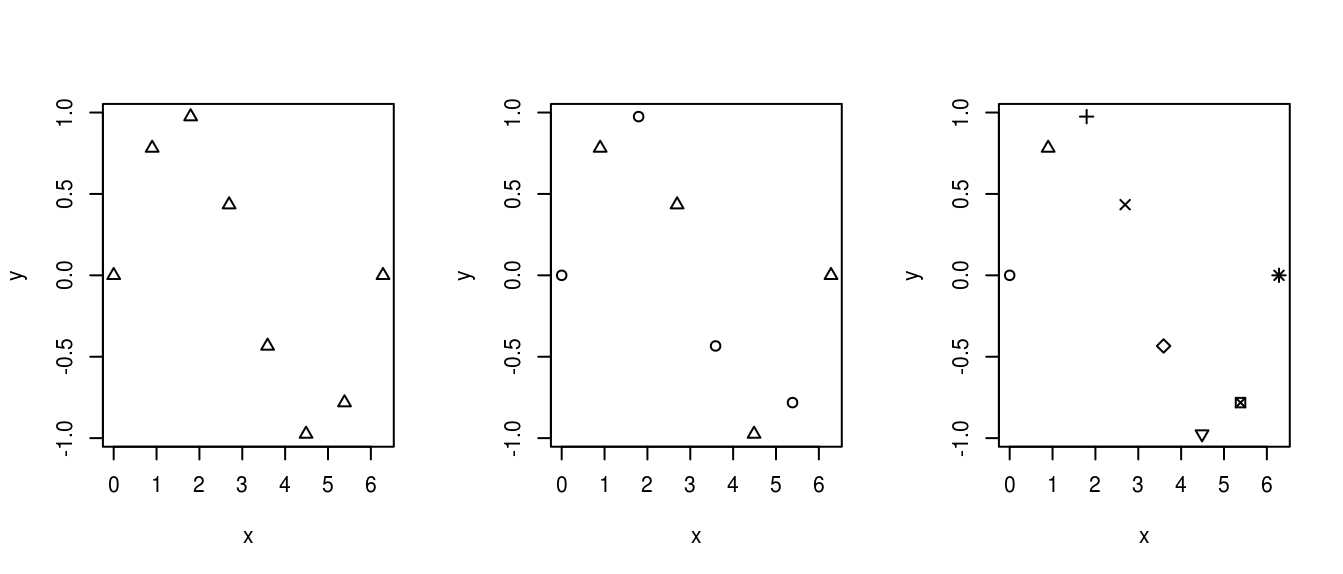
# Other graphical parameters



## pch

The point character can be set globally for all points or for each point separately:

```
par(mfrow = c(1,3)) # multiple plots (1 row and 3 columns)
plot(x, y, pch = 2) # always use the same pch
plot(x, y, pch = 1:2) # alternate the pch for subsequent points
plot(x, y, pch = 1:length(x)) # for each point a different pch
```



# Other graphical parameters



## `col`

Colors can be specified

- using RGB color specification, see `?rgb`,
- using one of the 657 character values included in `colors()`, e.g.

```
head(colors())
```

```
## [1] "white"         "aliceblue"     "antiquewhite"  "antiquewhite1"
## [5] "antiquewhite2" "antiquewhite3"
```

- using an integer that refers to the index of the colors defined in `palette()`, e.g.

```
 palette()
```

```
 ## [1] "black"   "red"     "green3"  "blue"    "cyan"    "magenta" "yellow"
 ## [8] "gray"
```

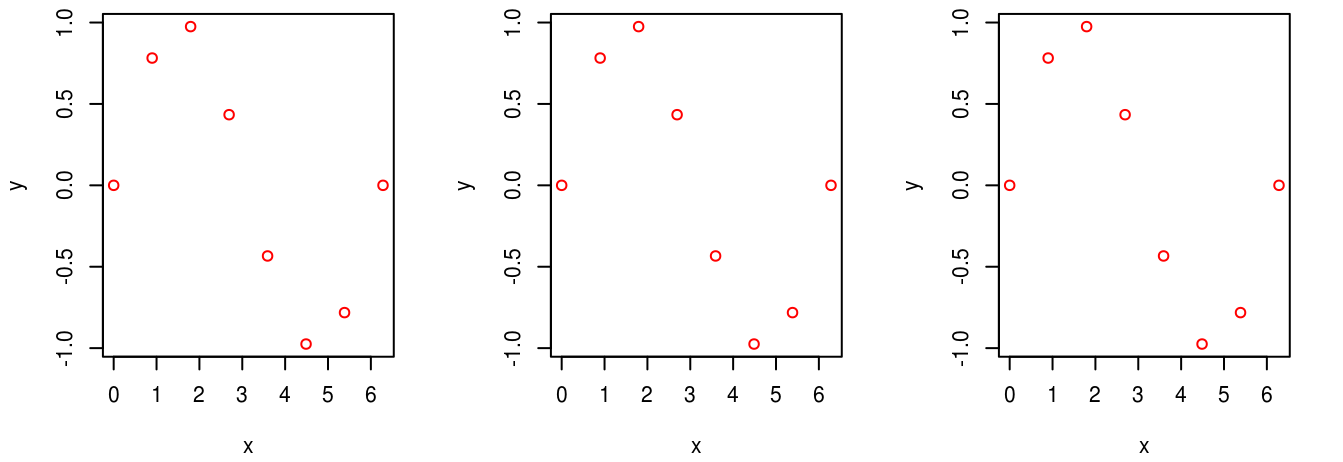
- `col = 1` is equivalent to `col = "black"`
- `col = 2` is equivalent to `col = "red"` ...

# Other graphical parameters



## col

```
par(mfrow = c(1,3)) # multiple plots (1 row and 3 columns)
plot(x, y, col = rgb(1,0,0))
plot(x, y, col = "red")
plot(x, y, col = 2)
```



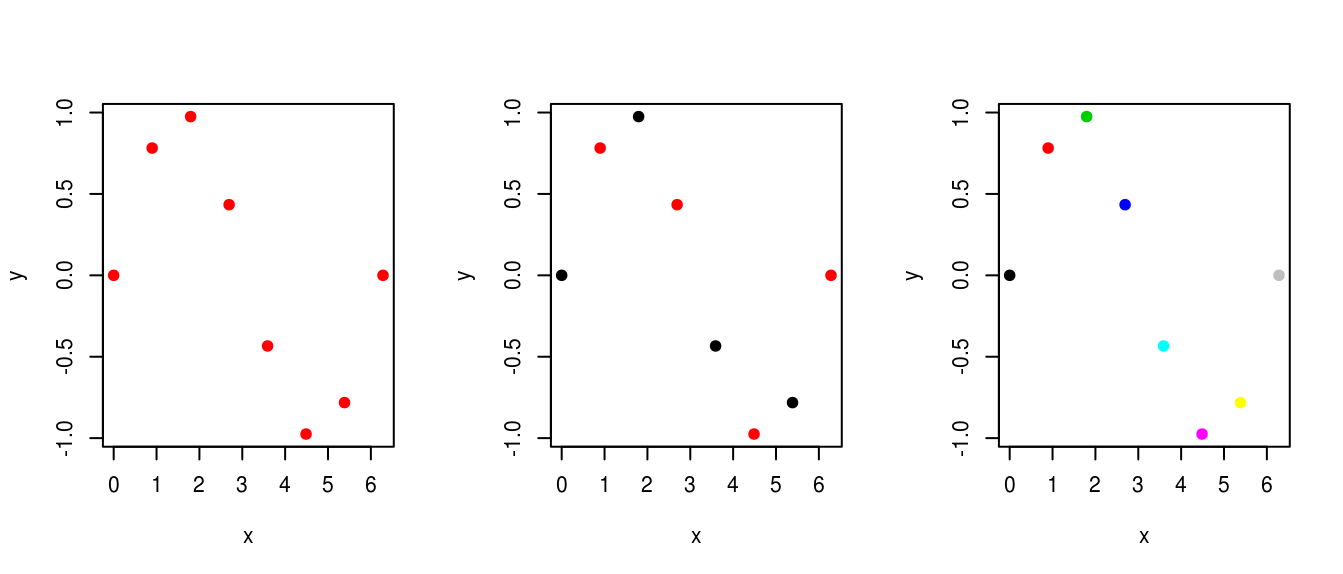
# Other graphical parameters



## `col`

The colors can be set globally for all points/lines or for each point/line separately:

```
par(mfrow = c(1,3)) # multiple plots (1 row and 3 columns)
plot(x, y, pch = 19, col = 2) # always use the same col
plot(x, y, pch = 19, col = 1:2) # alternate the col for subsequent points
plot(x, y, pch = 19, col = 1:length(x)) # for each point a different col
```



# Other graphical parameters

Münchner R Kurse

If all the graphical parameters are specified inside the `plot()` function, they are valid only for the specific plot.

It is possible, though, to set some graphical parameters globally by using the `par()` function (such as the `mfrow` argument for multiple plots).

# Summary



- `main` specifies the title of the plot
- `xlab` and `ylab` change the label of the horizontal and vertical axis, respectively.
- `xlim` and `ylim` set the range of the horizontal and vertical axis.
- `lty` specifies the line type for lines.
- `pch` specifies the point character for points, see `?pch`.
- `col` specifies the color, see also `demo("colors")` or `?grDevices::colors`.

See `?par` for the full list of graphical parameters that can be specified.

# High-level plots and low-level functions



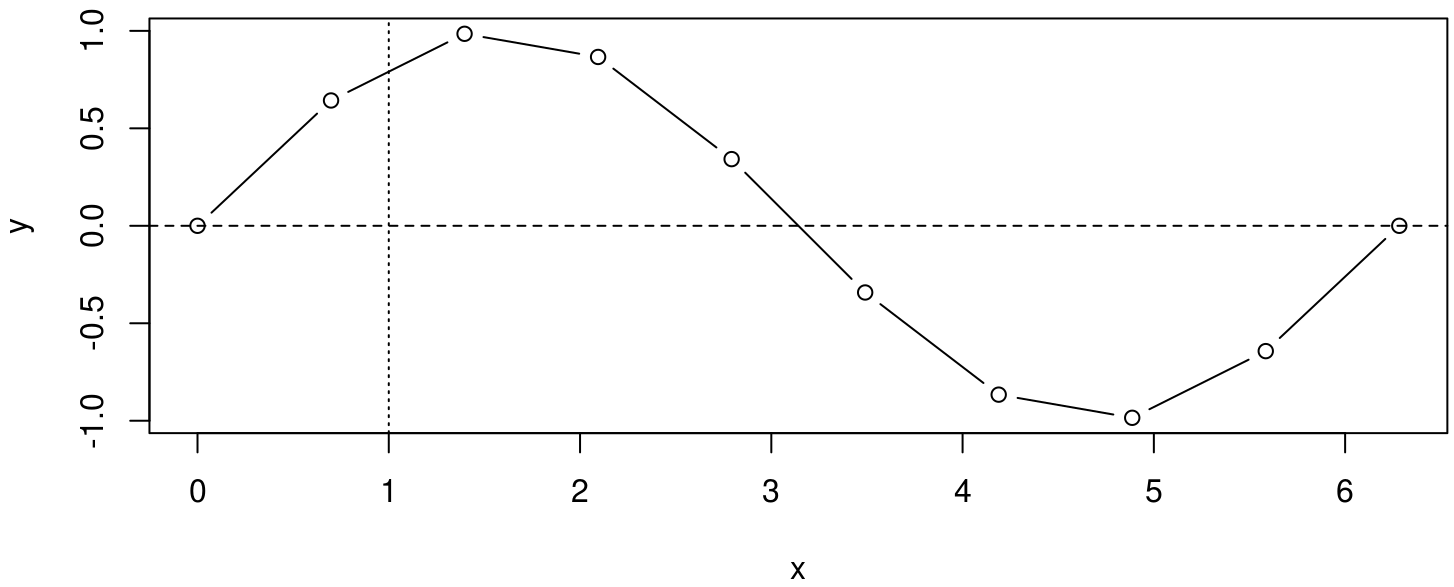
- Besides the `plot` function, there are other high-level plots that always **produce** a new graphic device, e.g.
  - `barplot(x)`: bar plot of values in $x$
  - `hist(x)`: histogram for the values in $x$
  - `pie(x)`: pie chart of values in $x$
  - `boxplot(x, y, ...)`: box plot of vectors $x$, $y$, ...
- Low-level graphical functions **add** elements to an **already existing** graphic device, such as
  - `abline`: add straight lines (e.g., horizontal, vertical).
  - `lines(x, y)`: add lines to a plot, using coordinates $(x, y)$.
  - `points(x, y)`: add points to a plot at coordinates $(x, y)$.
  - `text(x, y, labels)`: add text labels at $(x, y)$.
  - `polygon(x, y)`: draws polygon with vertices at $(x, y)$.
  - `legend(x, y, legend)`: add a legend labels at $(x, y)$.

# Low-level functions: straight lines



The `abline` function adds straight horizontal or vertical lines:

```
plot(x, y, type = "b")
abline(h = 0, lty = 2) # adds a horizontal line
abline(v = 1, lty = 3) # adds a vertical line
```

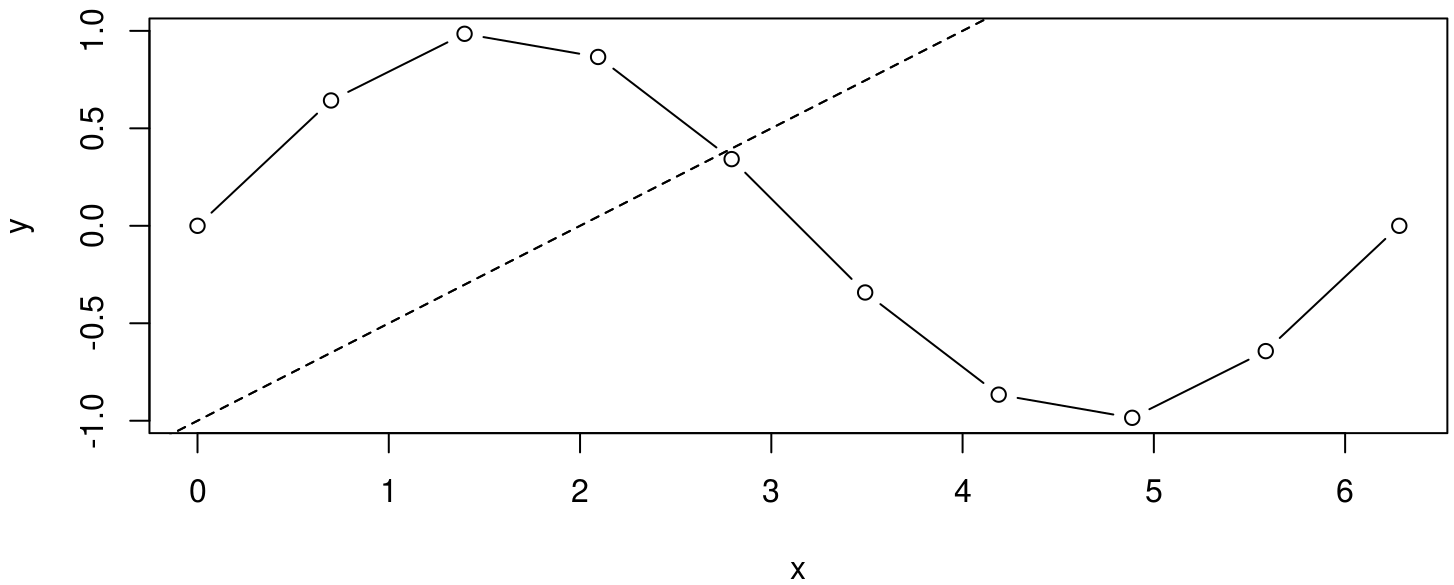


# Low-level functions: straight lines



It is also possible to add straight lines with a given intercept and slope using the `coef` or the `a` and `b` arguments:

```
plot(x, y, type = "b")
abline(coef = c(-1, 0.5), lty = 2) # adds a straight line
abline(a = -1, b = 0.5, lty = 2) # this is equivalent
```

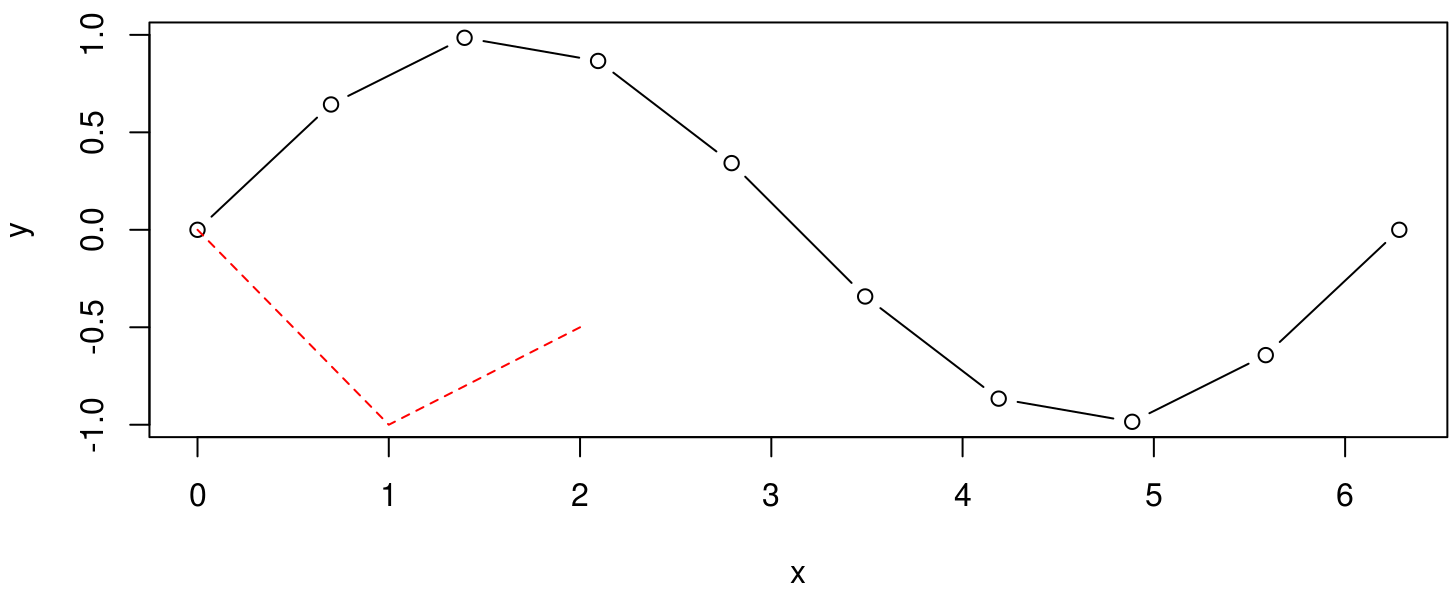


# Low-level functions: lines



The `lines` function connects points at given coordinates:

```
plot(x, y, type = "b")
lines(x = c(0, 1, 2), y = c(0, -1, -0.5), lty = 2, col = 2)
```

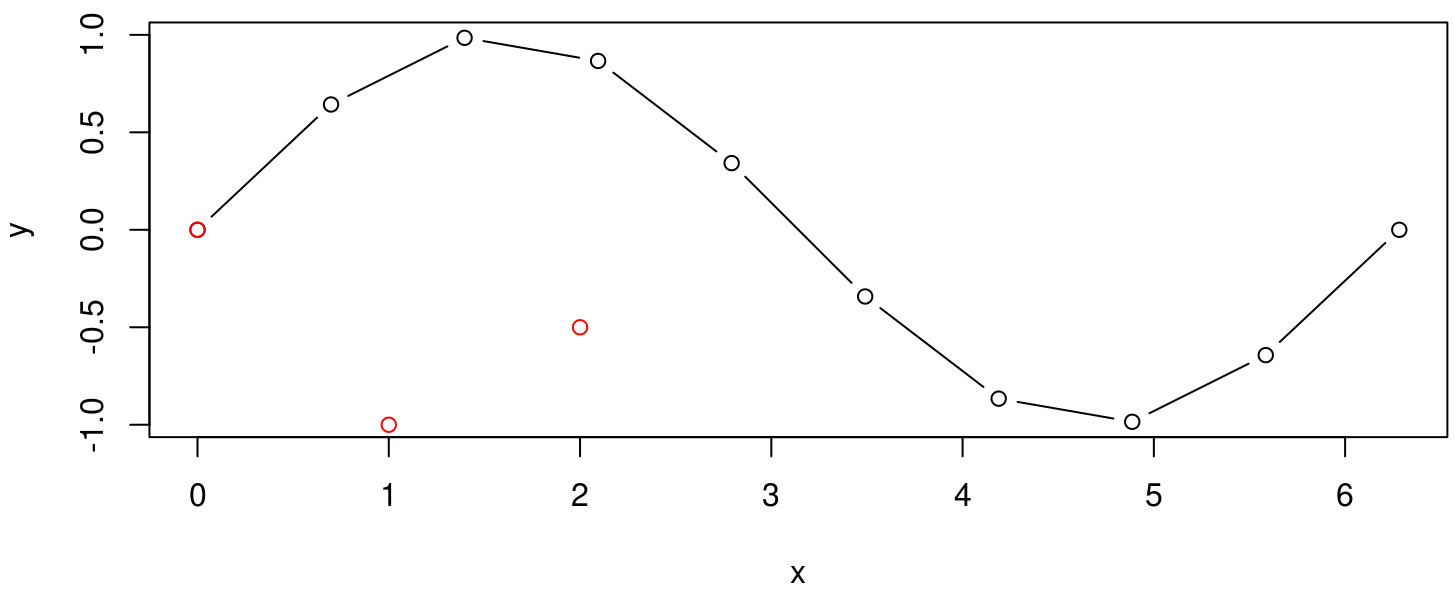


# Low-level functions: points



The `points` function adds points at given coordinates:

```
plot(x, y, type = "b")
points(x = c(0, 1, 2), y = c(0, -1, -0.5), col = 2)
```

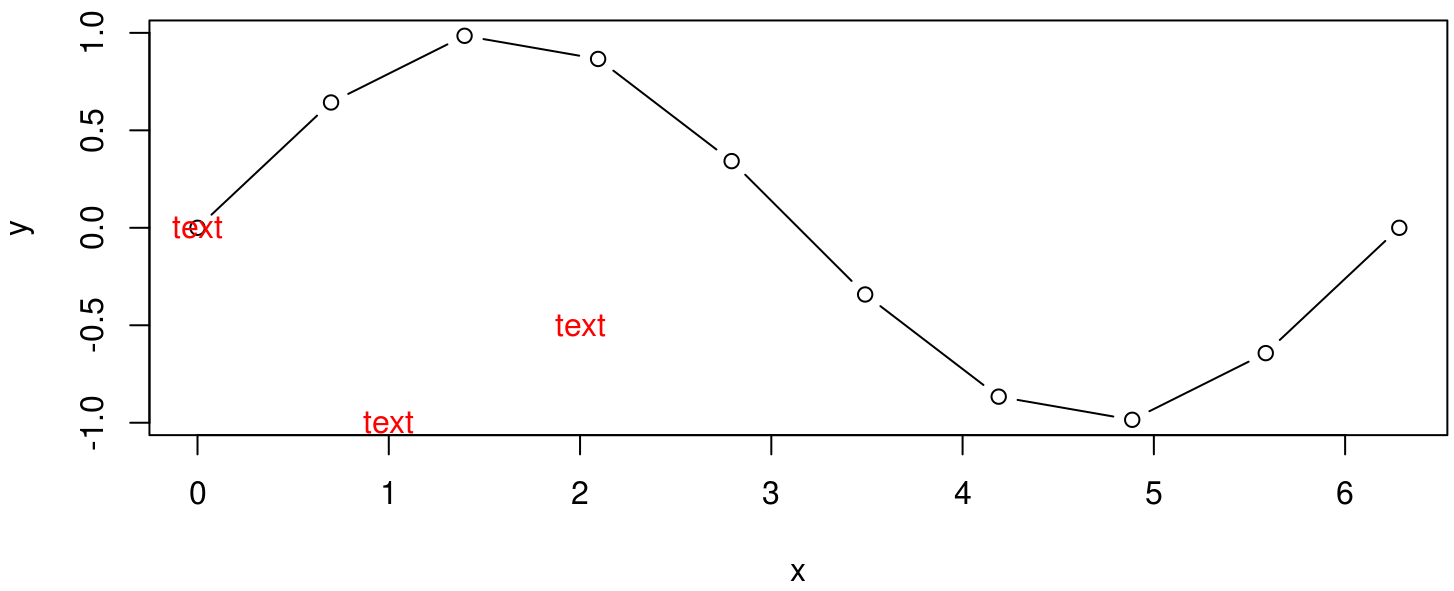


# Low-level functions: text



The `text` function adds a text at given coordinates:

```
plot(x, y, type = "b")
text(x = c(0, 1, 2), y = c(0, -1, -0.5), col = 2, labels = "text")
```

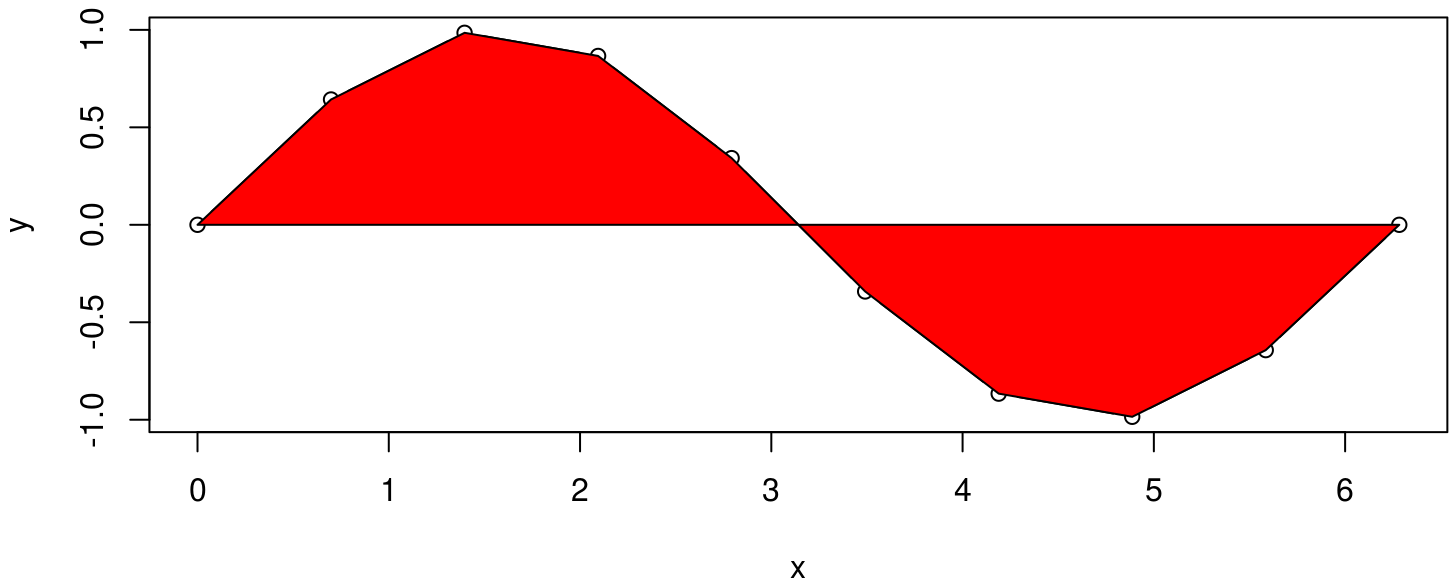


# Low-level functions: polygon



The `polygon` function draws a closed geometric shape which can be filled with a color:

```
plot(x, y, type = "b")
polygon(x, y, col = 2)
```

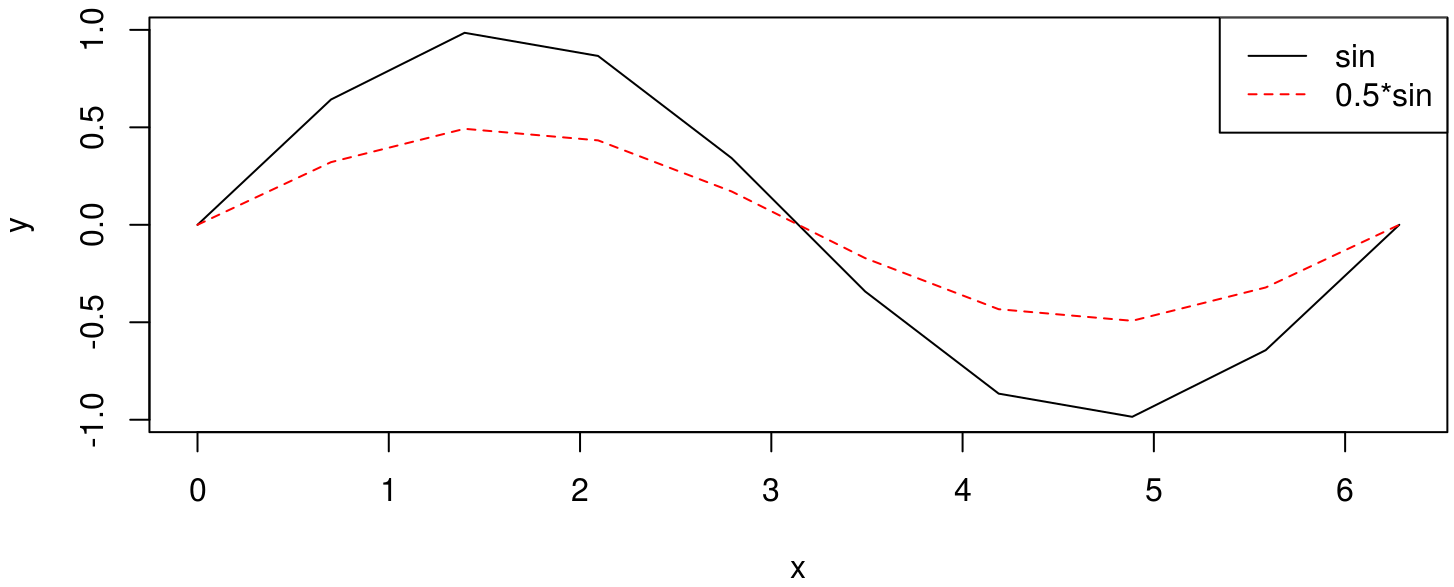


# Low-level functions: legend



The `legend` function adds a legend to the plot:

```
plot(x, y, type = "l")
lines(x, y*0.5, lty = 2, col = 2)
legend("topright", legend = c("sin", "0.5*sin"),
  col = c(1, 2), lty = c(1, 2))
```



# Low-level functions: legend



Different line and point types can be used in the `legend`:

```
plot(x, y, type = "b")
lines(x, y*0.5, lty = 2, col = 2)
points(x, y*0.25, pch = 4, col = 3)
legend("topright", legend = c("sin", "0.5*sin", "0.25*sin"),
  col = c(1, 2, 3), pch = c(1, NA, 4), lty = c(1, 2, NA))
```