R Notebook

Oskar Girardin

R basics

How do we run code ?

We can write R code in files (or scripts) with the `.R` extension. R code does not need to be compiled explicitly (like in C for example) and we do not need to run the script in its entirety (big difference to other programming languages !). To execute a line in RStudio, we select a line (or region) and click on "Run" or press `ctrl/cmd + enter`. To run the whole file, press `ctrl/cmd + shift + enter`. ## Objects ### Creating objects We can create an object by assigning it to a name. You can use the operator `<-` or `=` (as in other programming languages). Note that in R we cannot create new objects, it is not an object-oriented language.

```
x = 2
x <- 2
# These are equivalent
```

Object types

Basics Here are a few examples of different objects that exist in R. Once assigned, these objects appear in the 'Environment' section of RStudio.

```
# numeric
pi = 3.14

# vector
vec = c(1, 2, 0, 2)

# character
name = 'bob'
```

Functions You define functions as follows:

```
func_name = function(arg1, arg2, arg3){
  res = arg1 + arg2 + arg3
  return(res)
}


# example
square = function(x){
  return(x^2)
}
```

You can check the object type using the `class()` function.

```
class(square)
```

```
## [1] "function"
```

```r
class(name)
```

```
## [1] "character"
```

Vectors Vectors and matrices are objects that come with the R base version. We use the function `c()` (from 'concatenate') and `matrix()` to create them and these are some operations that we can do on them.

```r
vec1 = c(3, 1, 2, 4)
vec2 = c(-2, 2, 1, -4)

mat1 = matrix(c(1,2,3,4), nrow = 2, ncol = 2)
mat2 = matrix(c(0, -2, 3, 1), nrow = 2, ncol = 2)
```

```r
vec1[1]; vec1[1:4]
```

Accessing an element (index starts at 1 !!!)

```
## [1] 3
```

```
## [1] 3 1 2 4
```

```r
mat1[1, 2]
```

```
## [1] 3
```

```r
mat1[,2] # second column
```

```
## [1] 3 4
```

```r
vec1 + 2; vec1 * 3; vec1 / 4
```

Adding, multiplying, etc

```
## [1] 5 3 4 6
```

```
## [1]  9  3  6 12
```

```
## [1] 0.75 0.25 0.50 1.00
```

```r
mat1 + 2; mat1 * 4
```

```
##      [,1] [,2]
## [1,]    3    5
## [2,]    4    6
```

```
##      [,1] [,2]
## [1,]    4   12
## [2,]    8   16
```

```r
vec1 %*% vec2 # dot product
```

Vector multiplication

```
##      [,1]
## [1,]  -18
```

```r
vec1 %*% t(vec2) # t() is the transpose
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -6    6    3  -12
## [2,]   -2    2    1   -4
## [3,]   -4    4    2   -8
## [4,]   -8    8    4  -16
```

```
mat1 %*% mat2 # matrix multiplication
```

```
##      [,1] [,2]
## [1,]   -6    6
## [2,]   -8   10
```

```
mat1 * mat2 # element-wise multiplication
```

```
##      [,1] [,2]
## [1,]    0    9
## [2,]   -4    4
```

```
seq(1, 10, by=1)
```

Useful vectors

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(1, 10, by=2)
```

```
## [1] 1 3 5 7 9
```

```
rep(4, 20)
```

```
##  [1] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
```

```
rep(c(1,2), 10)
```

```
##  [1] 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
```

```
square(vec1) # element-wise application of function
```

Combining functions and vectors

```
## [1]  9  1  4 16
```

```
square(mat1) # element-wise application of function
```

```
##      [,1] [,2]
## [1,]    1    9
## [2,]    4   16
```

```
mean(vec1); var(vec1); sd(vec1); max(vec1)
```

Descriptive statistics

```
## [1] 2.5
```

```
## [1] 1.666667
```

```
## [1] 1.290994
```

```
## [1] 4
```

```r
summary(vec1) # Summary statistics
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.75    2.50    2.50    3.25    4.00
```

Probabiblity distributions

R includes functions to get information about distributions (p-values, density values) and to sample from them.

Uniform distribution

`dunif()`: **Density**
`punif()`: **CDF**
`qunif()`: **Quantile**
`runif()`: **Sample**

```r
low = 0
high = 4
x = 2
```

```r
# Density
dunif(x, low, high)
```

```
## [1] 0.25
```

```r
# CDF
punif(x, low, high)
```

```
## [1] 0.5
```

```r
# Quantile
qunif(0.25, low, high)
```

```
## [1] 1
```

```r
# Sample
n = 8
runif(n, low, high)
```

```
## [1] 1.711729 2.286986 2.909712 3.291289 1.674354 3.170645 2.178675 1.585642
```

Normal distribution

`dnorm()`: **Density**
`pnorm()`: **CDF**
`qnorm()`: **Quantile**
`rnorm()`: **Sample**

```r
mu = 0
sd = 1
x = 1
```

```r
# Density
dnorm(x, mu, sd)
```

```
## [1] 0.2419707
```

```r
# CDF
pnorm(0, mu, sd)
```

```
## [1] 0.5
```

```
# Quantile
qnorm(0.95, mu, sd)
```

```
## [1] 1.644854
```

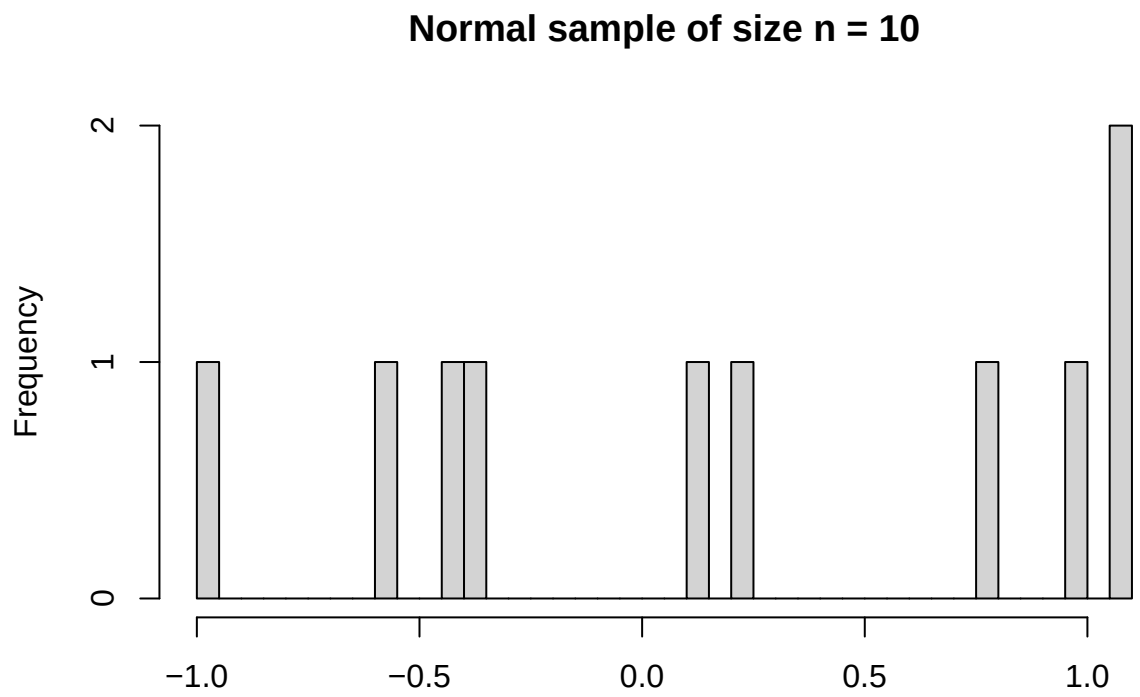
```
# Sample
n = 8
rnorm(n, mu, sd)
```

```
## [1]  0.4190310  0.4480075  0.4087453  0.1005143  0.4073233 -1.4545736 -1.1229599
## [8] -0.7915450
```

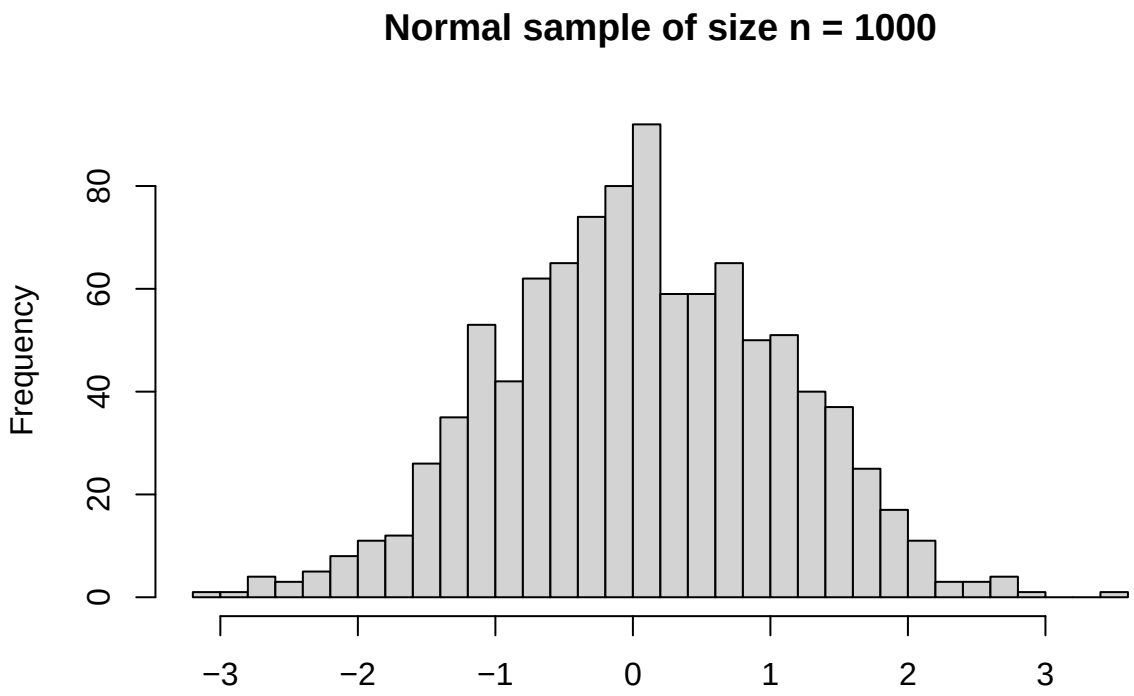
Plots

We can plot points using plotting functions such as: `plot()`, `hist()`, `boxplot()` & `barplot()`

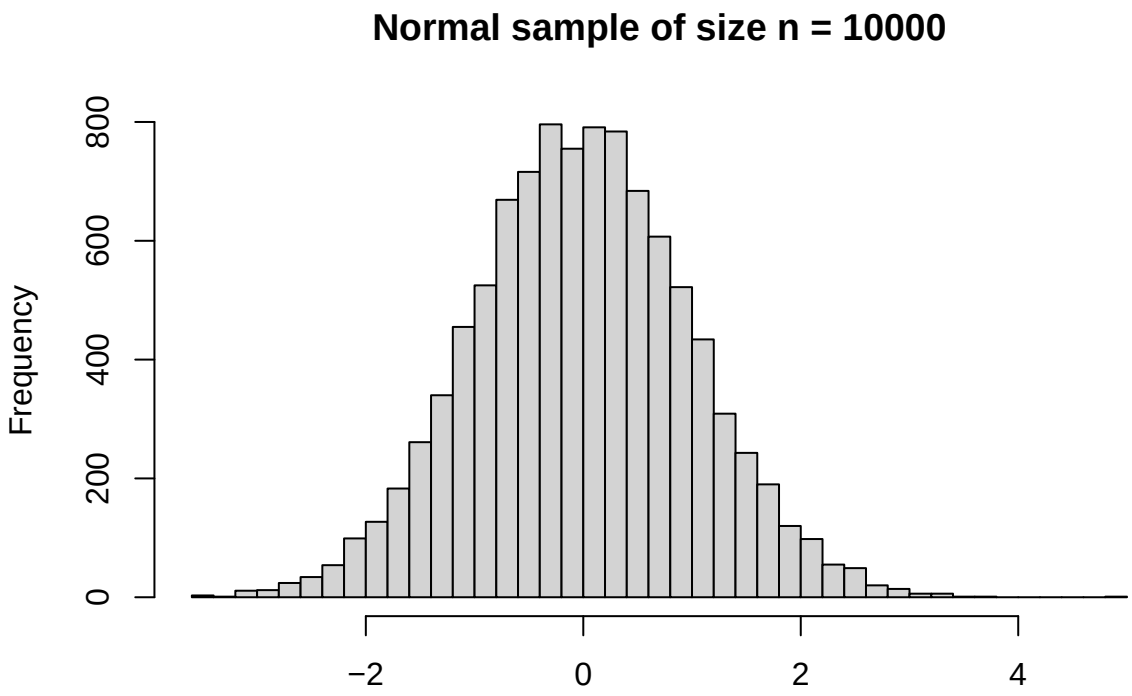
```
n = 10
normal_sample = rnorm(n, mu, sd)
hist(normal_sample, breaks = 40, main=paste('Normal sample of size n =', n), xlab = '')
```



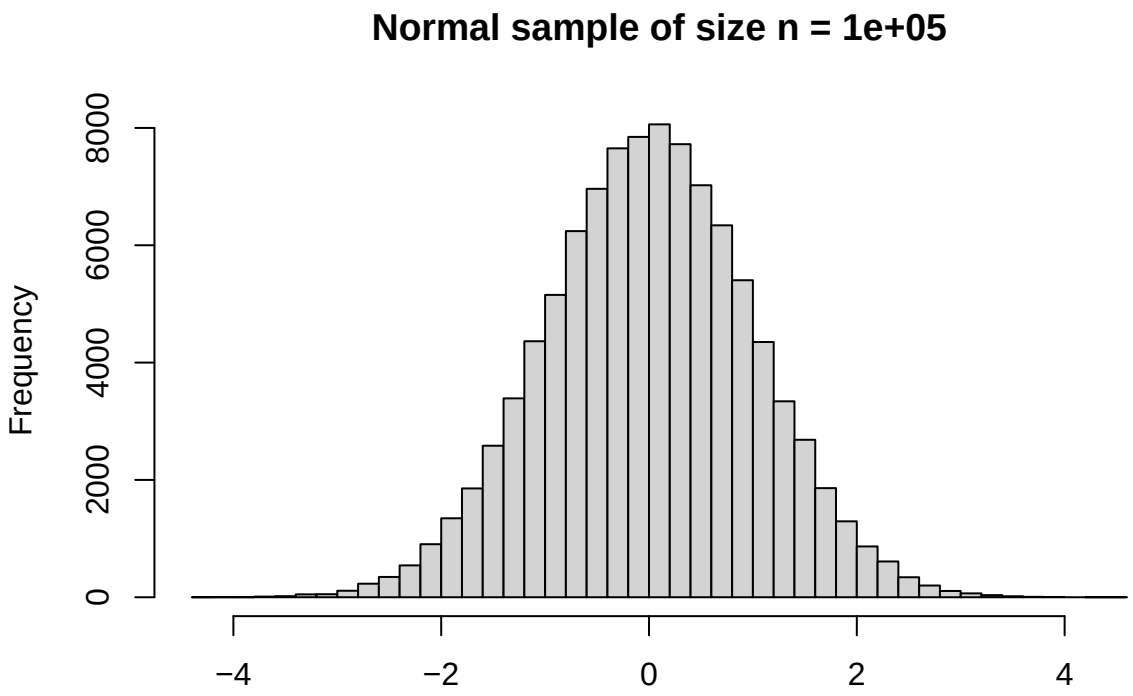

```
n = 1000
normal_sample = rnorm(n, mu, sd)
hist(normal_sample, breaks = 40, main=paste('Normal sample of size n =', n), xlab = '')
```




```
n = 10000
normal_sample = rnorm(n, mu, sd)
hist(normal_sample, breaks = 40, main=paste('Normal sample of size n =', n), xlab = '')
```

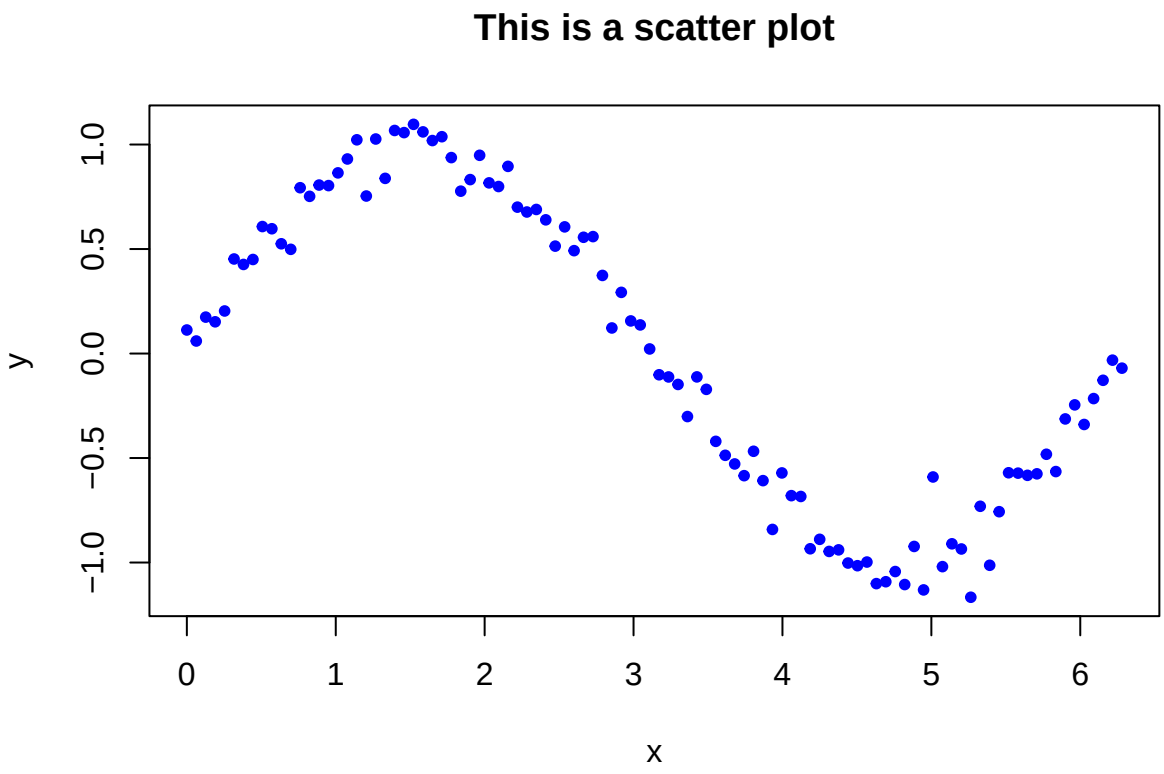



```
n = 100000
normal_sample = rnorm(n, mu, sd)
hist(normal_sample, breaks = 40, main=paste('Normal sample of size n =', n), xlab = '')
```



Scatter plot

```
n = 100
x = seq(0, 2*pi, length.out = n)
y = sin(x) + 0.1*rnorm(n, 0, 1)
plot(x, y, col = 'blue', pch = 20, main = 'This is a scatter plot')
```



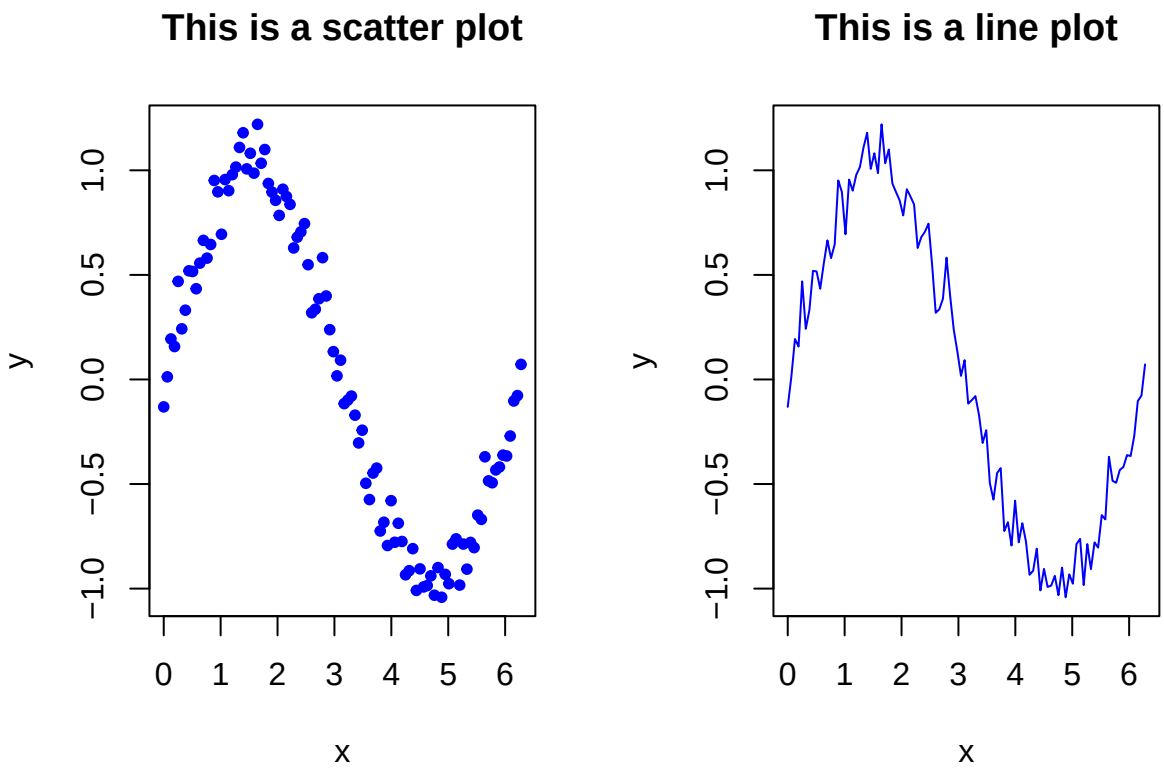
Line plot

```
n = 100
x = seq(0, 2*pi, length.out = n)
y = sin(x) + 0.1*rnorm(n, 0, 1)
plot(x, y, type = 'l', col = 'blue', main = 'This is a line plot')
```

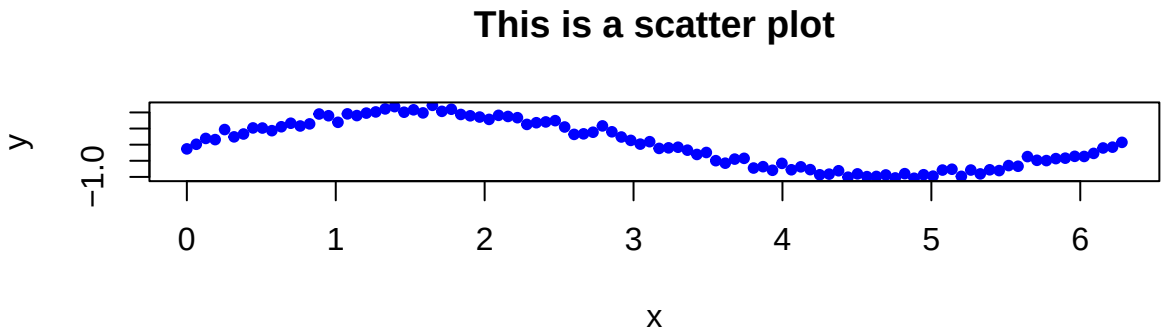
Subfigures

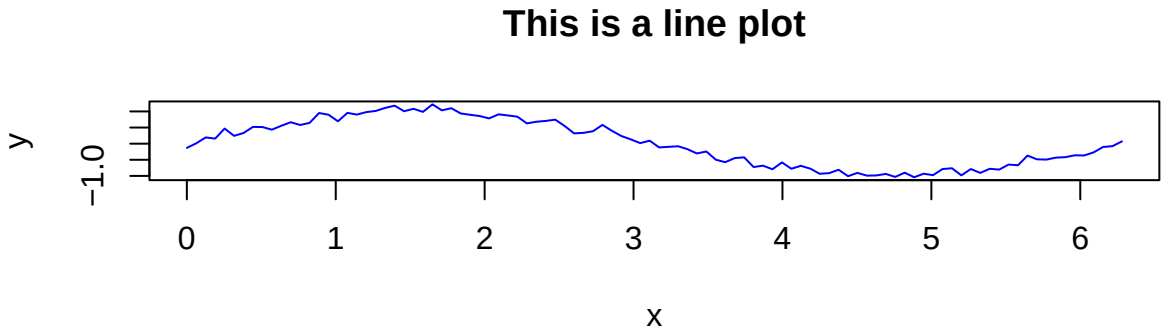
You can create multiple plots in the same figure using the `par()` function.

```
par(mfrow = c(1, 2)) # 1 row and 2 columns
plot(x, y, col = 'blue', pch = 20, main = 'This is a scatter plot')
plot(x, y, type = 'l', col = 'blue', main = 'This is a line plot')
```




```
par(mfrow = c(2, 1)) # 2 rows and 1 column
plot(x, y, col = 'blue', pch = 20, main = 'This is a scatter plot')
plot(x, y, type = 'l', col = 'blue', main = 'This is a line plot')
```





Loops

In R, there are 2 different types of loops: for and while loops.

```
# For-loop
for (i in 1:5) {
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```
# While-loop
i <- 0
while (i <= 4) {
  i <- i + 1
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

If-statements

If-statements are used to run different code based on a condition.

```
# If-statements
condition = TRUE
if (condition) {
 #Code executed when condition is TRUE
} else {
 #Code executed when condition is FALSE
}
```

```
## NULL
```

```
# Example If-statement
num = 3

if (num %% 2 == 0) {
 print(paste(num, 'is even'))
} else if (num %% 2 == 1) {
 print(paste(num, 'is odd'))
} else {
  print(paste(num, 'is not an integer'))
}
```

```
## [1] "3 is odd"
```