1 VECTORS AND MATRICES

First we download and install R Studio. I explain a bit the layout and we start writing code. We create variables.
Scalar:

```r
1  t <- 5
```

Vectors:

```r
numbers <- scan() # keyboard input
numbers <- c(0,1,2,3,4,5) # generating a vector manually
manynumbers <- c(numbers, numbers, numbers) # c can combine vectors as well
numbers <- 0:5 # create sequence in steps of one
time <- seq(from=0, to=5, by=0.5) # more general sequences
help(seq) # other options of seq
time <- seq(from=0, to=1, along.with=numbers) # the same number of elements as numbers
time + numbers # elementwise addition
time * numbers # elementwise multiplication
time % * % numbers # scalar product
sum(time*numbers) # equivalently
random <- rnorm(5) # gaussian random samples
length(random) # length of a vector
```

Matrices:

```r
M <- matrix(data=c(9,2,3,4,5,6), ncol=3) # manually generate a matrix
# useful for debugging
dim(M)
class(M)
print(M)
```
str(M)
Mtransp <- t(M) # transpose of a matrix
M %*% Mtransp # matrix product
Mtransp %*% M # as well
t <- rnorm(3)
M %*% t # matrix-vector product
# talk about dimensions! best define t as 1-dim matrix
t <- matrix(data=t, ncol=1)
M*Mtransp # will not work
M*M # elementwise product
M%*%t # be very careful about this!!! t will get up-scaled to the size of M.

# avoid using this if possible

Indexing:
numbers[3] # third element of numbers
numbers[-3] # all other elements of numbers
M[2,3] #
numbers[c(1,4,5)] # index by integer array
numbers[1:3] # useful
index <- (1:5 %% 2) == 1 # odd numbers
numbers(index) # logical indexing
numbers(index) <- 2
M[,1] # rows and columns
M[,2]
M[2,]
M[c(1,2), c(1,3)] # submatrix, this works with all indexing tricks above

Functions and flow control: File -> New -> R Script. Save as myfunc.r. We show how to run/source.

doUBLify <- function(x){
  return(x*2)
}

source
doUBLify(5)
doUBLify <- function(x, optional = FALSE){
  if (optional == TRUE){
    cat(x, "says hello!", "\n")
  }
  return(2*x)
}
t <- doUBLify(5, optional=TRUE)
t <- doUBLify(1:5, optional=TRUE)
for (i in 1:5) {
  doUBLify(i, optional=TRUE)
}
Plotting:

```r
supSeq <- function(x) {
  n <- length(x)
  res <- vector()
  for (i in 1:n) {
    s <- max(x[i:n])
    res <- c(res, s)
    cat("supremum after pos: ", i, " is: ", s, "\n")
  }
  return(res)
}

t <- seq(0, 10*2*pi, by=0.1)
myseq <- sin(t)*exp(-0.05*t)
sups <- supSeq(myseq)
plot(t, myseq, type='l')
lines(t, sups, col='red')
```