Functions

- We have already encountered quite a functions in R, e.g. mean(x), sd(x), matrix(x,…), sort(x).
- Functions have a name and a list of arguments or input objects.
- For example, the argument to the function mean() is usually a vector x.
- Functions also have a list of output objects, i.e. objects that are returned once the function has been run.
- A function must be written and loaded into *R* before it can be used.

Functions are typically written if we need to compute the same thing for several data sets and what we want to calculate is not already implemented in the commercial software yet, as would be the case in many areas of research.

Argument Matching (Named arguments and defaults)

As first noted in Generating regular sequences, if arguments to called functions are given in the "name=object" form, they may be given in any order. Furthermore the argument sequence may begin in the unnamed, positional form, and specify named arguments after the positional arguments.

Thus if there is a function fun1 defined by

```
> fun1 <- function(data, data.frame, graph, limit) {
    …
    # [function body omitted]
    …
    }
```

then the function may be invoked in several ways, for example

```
> ans <- fun1(d, df, TRUE, 20)
> ans <- fun1(d, df, graph=TRUE, limit=20)
> ans <- fun1(data=d, limit=20, graph=TRUE, ata.frame=df)
```

are all equivalent.

In many cases arguments can be given commonly appropriate default values, in which case they may be omitted altogether from the call when the defaults are appropriate. For example, if fun1 were defined as

```
> fun1 <- function(data, data.frame, graph=TRUE,  limit=20) {
... }
```

Then it could be called as

```
> ans <- fun1(d, df)
```

which is now equivalent to the three cases above, or as

```
> ans <- fun1(d, df, limit=10)
```

which changes one of the defaults.

It is important to note that defaults may be arbitrary expressions, even involving other arguments to the same function; they are not restricted to be constants as in our simple example here.

Writing Basic Functions

A simple function can be constructed as follows:

```
function_name <- function(arg1, arg2, ...){
commands
output
}
```

You decide on the name of the function. The function command shows R that you are writing a function. Inside the parenthesis you outline the input objects required and decide what to call them.

- Default settings can be specified , but you do not need to use them.
- The commands occur inside the { }.
- The name of whatever output you want goes at the end of the function.
- Comments lines (usually a description of what the function does is placed at the beginning) are denoted by #.

Let's construct a function "*powfunc*" that computes X^Y for a specified value of X and optionally Y. If Y is not specified, the default value is 2 (i.e. X^2)

The resulting value from the function is send back to the main script using the command return().

```
powfunc = function(X,pow=2){

return(X^pow)

}

powfunc(4)

# answer [1] 16

powfunc(4,3)

# answer [1] 64
```

A set of such functions can be constructed as a ***package***, and submitted to CRAN.