I/O

```
cat(object.to.print.1, object.to.print.2, ...)
paste(object.to.print.1, object.to.print.2, ...)
```

other settings:

`sep=""` *(no spaces between items)*

`"\n"` *(newline)*

```
round(a.double, num.digits)
trunc(a.double)
letters, LETTERS
month.abb, month.name

stop(a.message)


a.variable <- readline()
a.variable <- readline(a.prompt)
a.variable <- as.integer(readline())
```
(or double, or . . .)

Computation – mathematical operations

`+, -, *, /` *(add, subtract, multiply, divide)*
`^` *(powers)*
`%%` *(remainder after division)*
`%*%` *(matrix multiplication)*

`min(a.vector), max(a.vector)` *(consider* `na.rm=TRUE`*)*
`mean(a.vector), median(a.vector)` *(consider* `na.rm=TRUE`*)*

`sum(a.vector), prod(a.vector)`
`cumsum(a.vector)` *(cumulative sum)*
`diff(a.vector)` *(successive differences)*

```
sort(a.vector)
sort(a.vector, decreasing=TRUE)
order(a.vector)
```

Computation – logical operations

`==, >, >=, <, <=, !=` *(comparison)*
`a.vector %in% a.vector` *(member of set)*
`&, |, !` *(element-wise "and," "or," "not")*
`any(a.logical.vector), all(a.logical.vector)`

`is.na(a.vector)` *(test for* `NA` *values)*

```
which(a.logical.vector)
which(a.logical.matrix, arr.ind=TRUE)
which.min(a.vector), which.max(a.vector)
```

```
ifelse(a.logical.vector, vector.1, vector.2)
```

Branching

```
if ( condition ) {
    ...statements...
}
```

```
if ( condition ) {
    ...statements...
} else {
    ...statements...
}
```

compound conditions:

- `&&` ("and")
- `||` ("or")
- `!` ("not")

Repetition

Counter-controlled:

```
for (var in vector) {
    ...statements...
}
```

Condition-controlled:

```
repeat {
    ...statements...
    ...break...
}
```

```
while ( condition ) {
    ...statements...
}
```

Functions

Defining a function:

```
func.name <- function(argument1, argument2, ···) {
    ...statements ...
    return( variable or value )
}

func.name <- function(argument1=default.value1,
                      argument2=default.value2, ···) {
    ...statements ...
    return( variable or value )
}
```

Including another .R file to use a function it has defined:

```
source("nameOfFileWithFunction.R")
```

Calling a function:

```
answer <- func.name(my.var.1, 57, (my.var.2 + 18)/2)
```