

General

Get Help

```
? <Object/Function>
```

Find the working directory

```
getwd()
```

Setting Working directroy

```
setwd("~/specdata")
```

List files in working dir

```
dir()
```

Load code file into workspace

```
source("file.R")
```

Find the type of an object

```
class(my_vector)
```

List objects in workspace

```
ls()
```

Change page width

```
options(width = 160)
```

Operators

Assignment `var <- <New value>`

Compare two objects `identical( obj1, obj2 )`

Equality `var1 == var2`

Special Values

NA value is **Not Available**

NaN Not a Number

Inf Infinity

T True

F False

Subsetting Vectors

First 10 elements `x[1:10]`

Vector of all NAs `x[is.na(x)]`

Vector of real values `x[!is.na(x)]`

Values greater than 0 `y[y > 0]`

Combine conditions `x[!is.na(x) & x > 0]`

3rd, 5th, 7th elements of x `x[c(3,5,7)]`

All but the 2nd and 10th (neg) `x[c(-2, -10)]` or `x[-c(2,10)]`

Access element by label `vect["bar"]` or `vect[c-label ("foo", "bar")]`

Index vectors come in four different flavors -
- logical vectors, vectors of positive integers, vectors of negative integers, and vectors of character strings

Debugging

`traceback`

Prints function call stack

`debug( <fn> )`

Flags a function for "debug" moifr which allows you to step through execution of a function one line at a time

`browser`

Suspends the execution of a function, and outs the function in debug mode. n-next

`trace`

Allows you to insert debugging code into a function

`recover`

Allows you to modify the error behaviour so that you can browse the function call statck

Vectors

Concatinate function

```
patients <- c("Bill","Gina","Kelly","Sean")
```

Matrices

Help on matrix type

```
? matrix
```

Add dimensions to vector to make a matrix

```
dim(my_vector) <- c(4, 5)
```

View dimesions of a matrix

```
dim(my_matrix)
```

View dimesions of a matrix

```
attributes(my_matrix)
```

Create a matrix. (4x5 containing 1-> 20)

```
my_matrix2 <- matrix( 1:20,4,5 )
```

+ Matrices can only contain a **single class** of data.

+ The first number is the number of rows and the second is the number of columns.

Data Frames

Create a data frame from avector and matrix

```
my_data <- data.f ram e(p ati ent  
rix)
```

Add columns name to data frame

```
colnam es( my_ data) <- cnames _v
```

Select rows based on column value

```
frame[ frame$col == " val ", ]
```

Select columns by position

```
frame[, 1:4] cols 1 to 4
```

<http://www.r-tutor.com/r-introduction/data-frame/data-frame-row-slice>

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Not published yet.

Last updated 13th May, 2016.

Page 1 of 2.

Conversion

To number	<code>as.number(x)</code>
To boolean (logical)	<code>as.logical(x)</code>
To complex number	<code>as.complex()</code>

Reading Data

```
# Create empty data frame
data <- data.frame()

#Read files id is vector of
integers
for ( i in id ){
  infile = sprintf("%03d.csv", directory, i)
  data <- rbind( data, read.csv( infile ) )
}
head(data)
```

IF statement

```
if(<condition>) {
  ## do something
} else {
  ## do something else
}

if(<condition1>) {
  ## do something
} else if(<condition2>) {
  ## do something different
} else {
  ## do something different
}
```

For Statement

```
for(i in 1:10) {
  print(i)
}
```

While Statement

```
count <- 0
while( count < 10) {
  print(count)
  count <- count + 1
}
```

Repeat Statement

```
x0 <- 1
tol <- 1e-8
repeat {
  x1 <- computeEstimate()
  if( abs(x1 - x0) < tol)
  {
    break
  } else {
    x0 <- x1
  }
}
```

next,return

```
for(i in 1:100) {
  if(i <= 20) {
    ## Skip the first
    20 iterations
    next
  }
  ## Do something here
}
```

next is used to skip an iteration of a loop

Loop functions

<code>lapply</code>	Loop over a list and evaluate a function on each element
<code>sapply</code>	Same as <code>lapply</code> but try to simplify the result
<code>tapply</code>	Apply a function over the margins of an array

Loop functions (cont)

<code>mapply</code>	Multivariate version of <code>lapply</code>
<code>apply</code>	Used to evaluate a function (often an anonymous one) over the margins of an array.
<code>rowSums</code>	<code>apply(x,1 ,sum)</code>
<code>rowMeans</code>	<code>apply(x,1 ,mean)</code>
<code>colSums</code>	<code>apply(x,2 ,sum)</code>
<code>colMeans</code>	<code>apply(x,2 ,mean)</code>
<code>x<- list(a = 1:5, b= rnorm(10))</code> <code>lapply(x, mean)</code>	

An anonymous fn for extracting the 1st col of each matrix

```
>lapply(x,function(elt) elt[,1])
```