3. 기본 데이터 형태, 변수 사용

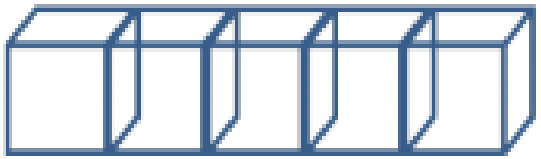
기본 데이터 형태

R코드에서 다루는 Data의 형태는 하기와 같다.

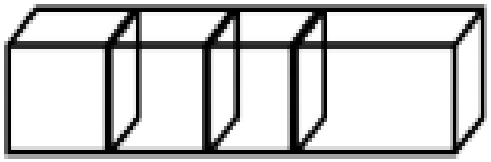
	설명	예제
Vectors	1차원 배열(숫자, 문자, 논리형 Data) Type은 반드시 동일한 형태의 Data를 갖고 있어야 함	Height<-c(58, 59, 60, 61) v<-seq(0, 100, by=25)
Matrices	2차원 배열(숫자, 문자, 논리형 Data) Type은 반드시 동일한 형태의 Data를 갖고 있어야 함	m<-matrix(1:20, nrow=5, ncol=4) dimnames(m)<-list(c('a', 'b', 'c', 'd', 'e'),c('p', 'q', 'r', 's')
Arrays	2차원 이상의 벡터형 데이터	a<-array(1:24, c(2,3,4))
Factors	이산형 데이터에 대한 표현 (명목형, 순서형 데이터)	colors<-c('green', 'red', 'blue') factor(colors)
Data Frames	동일한 길이의 여러 벡터형 데이터를 갖고 있는 리스트	team<-c('Man city', 'Man Utd', 'Arsenal', 'Chelsea') home_wins<-c(14, 10, 10, 9) home_draws<-c(0, 1, 2, 2) league_table<-data.frame(team, home_wins, home_draws)
Lists	각 Item의 명칭을 갖고 있는 Vector Type은 여러 개로 구성될 수 있음	peter<-list(name='peter', age=30, glasses=TRUE)

기본 데이터 형태

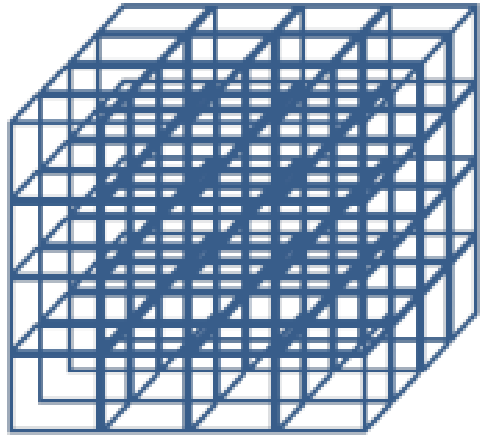
❑ 벡터(동일한 데이터 형)



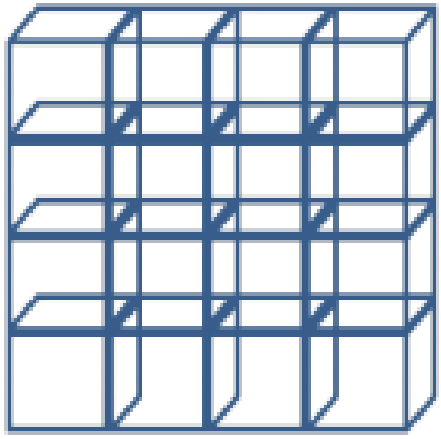
❑ 리스트(다른 데이터 형)



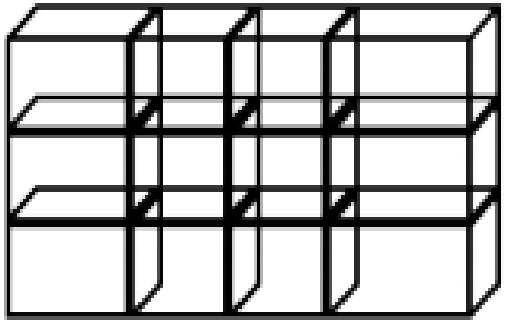
❑ 어레이(동일한 데이터 형)



❑ 매트릭스(동일한 데이터 형)



❑ 데이터프레임(다른 데이터 형)



벡터(vectors)

R 언어에서 가장 많이 사용되고 기본적인 R-Object는 Vectors이다. 여러 개의 데이터가 나열되어 있는 데이터 형태이다. Vectors는 c() 함수를 사용하여 만들 수 있다. 벡터를 만드는 c()함수의 c는 concatenate(붙이다)를 의미한다.

코드

```
# 벡터 생성
apple <- c('red','green',"yellow")
print(apple)

# 벡터변수의 유형 확인하기
print(class(apple))

# 벡터연산
# a<-c(1,3,5) == c(1, 3, 5) + c(100, 100, 100)
a<-c(1, 3, 5)
print(a+100)
```

실행결과

```
[1] "red"   "green"   '' yellow"
[1] "character“
[1] 101  103  105
```

벡터(vectors)

벡터 데이터는 다양한 방법으로 다룰 수 있다.

특정 위치값 제어

```
# 벡터생성
> a<-c(1:5)
# 특정 요소값만 조회
> a[3]
[1] 3
# 특정 요소값만 제외하고 조회
> a[-3]
[1] 1 2 4 5
# 1번~3번까지 요소값만 제외하고 조회
> a[-1:-3]
[1] 4 5
# 2번째에서 4번째까지 요소값만 조회
> a[2:4]
[1] 2 3 4
# 2번째 값을 변경
> a[2]<-6
> a
[1] 1 6 3 4 5
```

신규값 추가

```
# 신규값을 추가한다.
> a<-c(a,7)
> a
[1] 1 6 3 4 5 7
# 벡터길이보다 큰 위치에 신규값 추가하면 NA를 채운후 값을 추가한다.
# Null값이 아닌 NA가 추가된다.
> a[9]<-9
> a
[1]  1  6  3  4  5  7 NA NA  9
# append명령을 이용하여 3번째 값 다음에 10을 추가한다.
> append(a,10,after=3)
 [1]  1  6  3 10  4  5  7 NA NA  9
```

벡터(vectors)

벡터 데이터는 다양한 방법으로 연산을 할 수 있다.

숫자값 연산

```
# 벡터와 벡터 연산하기
> c(1,2,3)+c(4,5,6)
[1] 5 7 9
# 벡터와 값을 연산하기
> c(1,2,3)+1
[1] 2 3 4
```

```
# 숫자와 문자로 되어 있으나
# union연산 결과 벡터내 요소값이 모두 문자로 변경된다.
> union(c('1','2',3),c(4,5,6))
[1] "1" "2" "3" "4" "5" "6"
```

집합연산

```
> var1<-c(1,2,3)
> var2<-c(3,4,5)
> var1-var2
[1] -2 -2 -2
# var1에 있으나 var2에 없는 요소 출력하기
> setdiff(var1,var2)
[1] 1 2
# var1과 var2에 교집합 구하기
> intersect(var2,var1)
[1] 3
# var1과 var2의 합집합 구하기
> union(var1,var2)
[1] 1 2 3 4 5
```

벡터(vectors)

벡터에 연속 or 반복값을 할당하는 방법과 길이를 계산하는 방법은 하기와 같다.

연속 데이터 설정하기

```
# 1에서 5까지의 값을 설정
> a<-seq(1,5);a
[1] 1 2 3 4 5
#2에서 -2까지의 값을 설정
> b<-seq(2,-2);b
[1]  2  1  0 -1 -2
#1에서 10까지 2씩 증가시키면서 값을 설정
> c<-seq(1,10,2);c
[1] 1 3 5 7 9
```

반복 데이터 할당하기, 벡터길이 계산하기

```
#1에서 3까지 값을 2회 반복 설정하기
> d<-rep(1:3,2);d
[1] 1 2 3 1 2 3
#1에서 3까지 값을 각각 2회 반복 설정하기
> e<-rep(1:3,each=2);e
[1] 1 1 2 2 3 3
#벡터 e의 길이 구하기
> length(e)
[1] 6
```

벡터(vectors)

벡터에 특정 문장가 포함되어 있는지를 알아내기 위해서는 %in%을 이용한다.

숫자값 존재 여부 확인

```
> a<-c(1:5)
> a
[1] 1 2 3 4 5
#값 4가 벡터 a에 존재하는지 확인
> 4 %in% a
[1] TRUE
#값 9가 벡터 a에 존재하는지 확인
> 9 %in% a
[1] FALSE
```

문자값 존재 여부 확인

```
> b<-c("apple","pear","potato","grape","pineapple")
# 문자값 pear가 벡터 b에 존재하는지 확인
>  " pear "  %in% b
[1] TRUE
# 문자일부는 적용되지 않음
> "p" %in% b
[1] FALSE
```

매트릭스(matrix)

2차원의 데이터셋이다. 매트릭스 함수에 벡터를 제공하여 만들 수 있다.

모든 행과 열의 데이터 형태는 동일해야 한다.

코드

```
# 벡터데이터에 행과 열의 개수를 지정하여 매트릭스를 만든다.
# byrow를 이용하여 데이터가 채워지는 방향을 지정한다.
M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)
print(M)
print(class(M))
M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = FALSE)
print(M)

a<- matrix(1:9, nrow=3, ncol=3)
a[1,]
a[,-1]
sum(a); mean(a); median(a[,3]); mean(a[,2])
```

실행결과

```
     [,1] [,2] [,3]
[1,] "a"  "a"  "b"
[2,] "c"  "b"  "a"
[1] "matrix“
     [,1] [,2] [,3]
[1,] "a"  "b"  "b"
[2,] "a"  "c"  "a"

[1] 1 4 7
     [,1] [,2]
[1,]    4    7
[2,]    5    8
[3,]    6    9
[1] 45
[1] 5
[1] 8
[1] 5
```

매트릭스(matrix)

매트릭스 형태의 데이터는 행과 열을 설정하여 조회할 수 있다.

코드

```
> M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)
> M
     [,1] [,2] [,3]
[1,] "a"  "b"  "b"
[2,] "a"  "c"  "a"
# 1열의 모든값을 출력
> M[,1]
[1] "a" "a“
# 1행의 모든값을 출력
> M[1,]
[1] "a" "b" "b"
# 1행과 1열의 값을 출력
> M[1,1]
[1] "a"
```

실행결과

```
     [,1] [,2] [,3]
[1,] "a"  "a"  "b"
[2,] "c"  "b"  "a"
[1] "matrix“
     [,1] [,2] [,3]
[1,] "a"  "b"  "b"
[2,] "a"  "c"  "a"
```

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

```
[1] 45
[1] 5
[1] 8
[1] 5
```

매트릭스(matrix)

매트릭스에서 서브매트릭스를 추출하는 방법은 하기와 같다.

코드

```
> M<-matrix(c(1:12),nrow=3)
> M
> M[c(1:2),c(2:3)]
```

실행결과

```
 [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12


 [,1] [,2]
[1,]    4    7
[2,]    5    8
```

어레이(Arrays)

매트릭스는 2차원에 한정되고 arrays는 어떠한 차원으로도 만들 수 있다.

코드

```
# Create an array.
a <- array(c('green','yellow'),dim = c(3,3,2))
print(a)
```

실행결과

```
, , 1

     [,1]     [,2]     [,3]
[1,] "green"  "yellow" "green"
[2,] "yellow" "green"  "yellow"
[3,] "green"  "yellow" "green"

, , 2

     [,1]     [,2]     [,3]
[1,] "yellow" "green"  "yellow"
[2,] "green"  "yellow" "green"
[3,] "yellow" "green"  "yellow"
```

팩터(Factors)

Factors는 주로 빈도분석에 활용하는 변수형태이다.

vector로 만들어지는 R-object이며, vector에 있는 element들의 고유값(distinct value)을 레이블로 저장한다.

Factors는 factor()함수를 사용하여 만들 수 있으며, nlevels() 함수를 통해 factors의 레벨을 알 수 있다.

코드

```
# 벡터 생성
apple_colors <- c('green','green','yellow','red','red','red','green')

# factor()함수를 이용하여 vector을 factor으로 만들기
factor_apple <- factor(apple_colors)

# factor와 factor의 레벨을 출력하기
print(factor_apple)
print(nlevels(factor_apple))

# factor의 빈도 보여주기
summary(factor_apple)
```

실행결과

```
[1] green  green  yellow red    red    red    green
Levels: green red yellow
[1] 3
green    red yellow
    3      3      1
```

데이터프레임(Dataframe)

데이터프레임은 구조화된 데이터 오브젝트이다. 매트릭스와 다른점은 데이터프레임은 컬럼으로 어떠한 데이터타입이든 넣을 수 있다. 첫 번째 컬럼은 numeric, 두 번째 컬럼은 문자열, 세 번째 컬럼은 논릿값 등 자유롭게 지정할 수 있다. 이것은 같은 길이의 vector들의 list로 볼 수 있다. 데이터 프레임은 data.frame() 함수를 통해 생성할 수 있다.

코드

```
# Create the data frame.

BMI <-  data.frame(
   gender = c("Male", "Male","Female"),
   height = c(152, 171.5, 165),
   weight = c(81,93, 78),
   Age = c(42,38,26)
)
print(BMI)
```

실행결과

```
   gender height weight Age
1   Male  152.0     81  42
2   Male  171.5     93  38
3 Female  165.0     78  26
```

데이터프레임(Dataframe)

행렬을 데이터프레임으로 변환하는 것도 가능하다. 이 때 열이름을 지정해주지 않으면 "X"를 접미사로 하여 X1, X2, 이런 식으로 열 이름이 붙는다. 행 이름은 지정하지 않으면 별도로 생성되지 않으며 표시되지도 않는다.

코드

```
m <- matrix(1:6, nrow=3)
df2 <- data.frame(m)
Df2


df3 <- data.frame(m, row.names=c("r1", "r2", "r3"))
df3
```

실행결과

```
> df2
  X1 X2
1  1  4
2  2  5
3  3  6
```

```
> df3
   X1 X2
r1  1  4
r2  2  5
r3  3  6
```

데이터프레임(Dataframe)

여러 개의 데이터 프레임은 merge()명령을 이용하여 합칠 수가 있다.

코드

```
df1<-data.frame(name=c('apple','grape','pear'),price=c(100,200,300))
df2<-data.frame(name=c('apple','berry','banana'),price=c(400,500,600))
df1
df2
# 데이터가 없는 것도 모두 나오도록
merge(df1,df2,all=T)
```

실행결과

```
 name price
1 apple   100
2 grape   200
3  pear   300
```

```
 name price
1  apple   400
2  berry   500
3 banana   600
```

```
   name price
1  apple   100
2  apple   400
3  grape   200
4   pear   300
5 banana   600
6  berry   500
```

리스트(List)

키, 값의 형태로 데이터를 저장하는 일종의 배열 데이터 형태이다. 서로 다른 데이터 유형을 저장할 수 있다.

코드

```
list1 <- list(name='jenny',
              address='Seoul',
              tel='010-7777-8888',
              pay=500)
list1


#name열만 조회하고 싶은 경우
list1$name


List1[1:2]
```

실행결과

```
$name
[1] "jenny"
$address
[1] "Seoul"
$tel
[1] "010-7777-8888"
$pay
[1] 500
```

```
$name
[1] "jenny"
$address
[1] "Seoul"
```

변수 사용 규칙

R에서 변수를 사용하기 위해서는 하기규칙을 따라야 한다.

변수 사용 규칙

❑ **변수명칭 : 영어/한글 모두 가능하며 문자로 시작한다.**

- 숫자로 시작하지 않도록 한다.
- 예1) var1, hflight, ….
- 예2) 100var, 3hf, …

❑ **변수명칭 : 대소문자를 구분한다.**

- 윈도우내 개발 솔루션과 다른점
- ABC ≠ abc

❑ **변수에 값 설정 방법**

- 방법 : ‘ <- ’, ‘ = ‘
- 추천 : ‘ <- ‘ 사용을 권장한다.
- 예) var1 <- 100
- 참고 : ‘ -> ’으로도 변수 값 설정이 가능하다.
- 예) 100 -> var1

변수 사용 금지어

R에서 변수를 설정할 때 아래와 같은 예약어는 사용하지 않도록 주의한다.

R에서 사용되는 예약어

- break : loop문을 빠져나오는 명령어로 사용
- else : 조건문에서 사용
- FALSE : boolean값으로 사용
- for : 반복문에서 사용
- function : 사용자 함수 정의할 때 사용
- if : 조건문에서 사용
- in : 반복문에서 사용
- Inf : infinite의 약어로 무한대를 의미하는 용도로 사용
- NA : Not Available의 약어로 결측값을 의미하는 용도로 사용
- NaN : Not a Number의 약어로 0/0과 같이 수학적으로 정의되지 않음을 의미하는 용도로 사용
- next : 반복문에서 사용
- NULL : 결측치를 의미하는 용도로 사용
- repeat : 반복문을 의미하는 용도로 사용
- TRUE : boolean값으로 사용
- while : 반복문에서 사용

변수에 값 대입

R에서 변수값은 다양한 방법으로 설정할 수 있다.

연속 변수값 설정

❑ 연속적인 값 대입하기

```
> seq1 <-1:6
> seq1
[1] 1 2 3 4 5 6
```

❑ 문자는 연속적인 값 대응이 안됨

```
> seq2<-"a":"c"
Error in "a":"c" : NA/NaN argument
In addition: Warning messages:
1: NAs introduced by coercion
2: NAs introduced by coercion
```

연속 시간데이터 설정

❑ 시간데이터를 연속으로 대입하기(日 기준)

```
> date1<-seq(from=as.Date('2018-09-01'),to=as.Date('2018-09-10'),by=2)
> date1
[1] "2018-09-01" "2018-09-03" "2018-09-05" "2018-09-07" "2018-09-09"
```

❑ 시간데이터를 연속으로 대입하기(月 기준)

```
> date2<-seq(from=as.Date('2018-09-01'),to=as.Date('2018-12-10'),by='month')
> date2
[1] "2018-09-01" "2018-10-01" "2018-11-01" "2018-12-01"
```

❑ 시간데이터를 연속으로 대입하기(年 기준)

```
> date3<-seq(from=as.Date('2018-09-01'),to=as.Date('2022-12-10'),by='year')
> date3
 [1] "2018-09-01" "2019-09-01" "2020-09-01" "2021-09-01" "2022-09-01"
```

변수목록 조회 및 제거

현재 R에 설정되어 존재하는 변수 목록을 조회하고 삭제하는 방법은 하기와 같다.

변수목록 조회 및 삭제

❑ **objects()**

```
> a<-iris
> b<-"i am korean"
> objects()
[1] "a" "b"
```

❑ **rm(삭제 대상변수명)**

```
> a<-iris
> b<-"i am korean"
> objects()
[1] "a" "b"
> rm(a)
> objects()
[1] "b"
```

모든 변수 삭제

❑ **rm(list=ls())**

```
> a<-iris
> b<-"i am korean"
> objects()
[1] "a" "b"
> rm(list=ls())
> objects()
character(0)
```

변수 연산자

R에서 사용가능한 산술 연산자는 다음과 같다.

산술연산자

❑ 더하기 : +

> 1+2

[1] 3

❑ 빼기 : -

> 2-1

[1] 1

❑ 곱하기 : *

> 3*9

[1] 27

❑ 나누기(실수가능) : /

> 5/2

[1] 2.5

산술연산자

❑ 정수나누기 : %/%

> 5%/%2

[1] 2

❑ 나머지 구하기 : %%

> 5%%2

[1] 1

❑ 승수 구하기 : ^, **

> 2^3

[1] 8

변수값 표시

0의 개수에 따라서 큰 값은 e(지수) 표기법을 적용한다.

숫자 크기에 따른 표기법

❑ 작은수의 경우 표기

```
> 50000
[1] 50000
```

❑ 큰수(0이 5개 이상)의 경우는 지수표기 적용

```
> 500000
[1] 5e+05
```

지수 표기법 사용

❑ 지수표기법을 이용한 숫자값 설정

```
> 5e2
[1] 500
> 5e-1
[1] 0.5
```

NA와 NULL의 의미

Na와 NULL의 정확한 의미는 서로 다르다.

NA의 의미

❑ **NA는 값을 모르는 경우를 의미한다.**

- Not Applicable
- Not Available

❑ **sum연산에 NA가 사용되는 경우**

```
> sum(1,3,5)
[1] 9
> sum(1,NA,5)
[1] NA
```

❑ **NA를 빼고 sum연산을 하는 방법**

```
> sum(1,NA,5,na.rm=TRUE)
[1] 6
```

NULL의 의미

❑ **NULL은 값이 없는 경우를 의미한다.**

- 결측치를 의미한다.

❑ **sum, mean연산에 NULL이 사용되는 경우**

```
>#NULL을 제외한 합을 구한다.
> sum(1,NULL,5)
[1] 6

#NULL을 제외한 평균값을 구한다.
> a<-c(1,NULL,5)
> mean(a)
[1] 3
```

R Cheat sheet

Base R
Cheat Sheet

Getting Help

Accessing the help files

?mean
Get help of a particular function.
help.search('weighted mean')
Search the help files for a word or phrase.
help(package = 'dplyr')
Find help for a package.

More about an object

str(iris)
Get a summary of an object's structure.
class(iris)
Find the class an object belongs to.

Using Libraries

install.packages('dplyr')
Download and install a package from CRAN.

library(dplyr)
Load the package into the session, making all its functions available to use.

dplyr::select
Use a particular function from a package.

data(iris)
Load a built-in dataset into the environment.

Working Directory

getwd()
Find the current working directory (where inputs are found and outputs are sent).

setwd('C://file/path')
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

sort(x)
Return x sorted.
rev(x)
Return x reversed.
table(x)
See counts of values.
unique(x)
See unique values.

Selecting Vector Elements

By Position

x[4]	The fourth element.
x[-4]	All but the fourth.
x[2:4]	Elements two to four.
x[-(2:4)]	All elements except two to four.
x[c(1, 5)]	Elements one and five.

By Value

x[x == 10]	Elements which are equal to 10.
x[x < 0]	All elements less than zero.
x[x %in% c(1, 2, 5)]	Elements in the set 1, 2, 5.

Named Vectors

x['apple']	Element with name 'apple'.

Programming

For Loop

```
for (variable in sequence){
   Do something
}
```

Example

```
for (i in 1:4){
   j <- i + 10
   print(j)
}
```

While Loop

```
while (condition){
   Do something
}
```

Example

```
while (i < 5){
   print(i)
   i <- i + 1
}
```

If Statements

```
if (condition){
   Do something
} else {
   Do something different
}
```

Example

```
if (i > 3){
   print('Yes')
} else {
   print('No')
}
```

Functions

```
function_name <- function(var){
   Do something
   return(new_variable)
}
```

Example

```
square <- function(x){
   squared <- x*x
   return(squared)
}
```

Reading and Writing Data

Input	Ouput	Description
df <- read.table('file.txt')	write.table(df, 'file.txt')	Read and write a delimited text file.
df <- read.csv('file.csv')	write.csv(df, 'file.csv')	Read and write a comma separated value file. This is a special case of read.table/ write.table.
load('file.RData')	save(df, file = 'file.Rdata')	Read and write an R data file, a file type special for R.

Conditions

a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to	is.na(a)	Is missing
a != b	Not equal	a < b	Less than	a <= b	Less than or equal to	is.null(a)	Is null

R Cheat sheet

Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

`as.logical`	`TRUE, FALSE, TRUE`	Boolean values (TRUE or FALSE).
`as.numeric`	`1, 0, 1`	Integers or floating point numbers.
`as.character`	`'1', '0', '1'`	Character strings. Generally preferred to factors.
`as.factor`	`'1', '0', '1', levels: '1', '0'`	Character strings with preset levels. Needed for some statistical models.

Maths Functions

`log(x)`	Natural log.	`sum(x)`	Sum.
`exp(x)`	Exponential.	`mean(x)`	Mean.
`max(x)`	Largest element.	`median(x)`	Median.
`min(x)`	Smallest element.	`quantile(x)`	Percentage quantiles.
`round(x, n)`	Round to n decimal places.	`rank(x)`	Rank of elements.
`signif(x, n)`	Round to n significant figures.	`var(x)`	The variance.
`cor(x, y)`	Correlation.	`sd(x)`	The standard deviation.

Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```

The Environment

`ls()`	List all variables in the environment.
`rm(x)`	Remove x from the environment.
`rm(list = ls())`	Remove all variables from the environment.

You can use the environment panel in RStudio to browse variables in your environment.

Matrixes

`m <- matrix(x, nrow = 3, ncol = 3)`
Create a matrix from x.

`m[2, ]` - Select a row

`m[ , 1]` - Select a column

`m[2, 3]` - Select an element

`t(m)`
Transpose

`m %*% n`
Matrix Multiplication

`solve(m, n)`
Find x in: m * x = n

Lists

`l <- list(x = 1:5, y = c('a', 'b'))`
A list is collection of elements which can be of different types.

`l[[2]]`	`l[1]`	`l$x`	`l['y']`
Second element of l.	New list with only the first element.	Element named x.	New list with only element named y.

Also see the **dplyr** library.

Data Frames

`df <- data.frame(x = 1:3, y = c('a', 'b', 'c'))`
A special case of a list where all elements are the same length.

x	y
1	a
2	b
3	c

List subsetting

`df$x`

`df[[2]]`

Understanding a data frame

`View(df)`	See the full data frame.
`head(df)`	See the first 6 rows.

Matrix subsetting

`df[ , 2]`

`df[2, ]`

`df[2, 2]`

`nrow(df)`
Number of rows.

`ncol(df)`
Number of columns.

`dim(df)`
Number of columns and rows.

`cbind` - Bind columns.

`rbind` - Bind rows.

Strings

Also see the **stringr** library.

`paste(x, y, sep = ' ')`	Join multiple vectors together.
`paste(x, collapse = ' ')`	Join elements of a vector together.
`grep(pattern, x)`	Find regular expression matches in x.
`gsub(pattern, replace, x)`	Replace matches in x with a string.
`toupper(x)`	Convert to uppercase.
`tolower(x)`	Convert to lowercase.
`nchar(x)`	Number of characters in a string.

Factors

`factor(x)`
Turn a vector into a factor. Can set the levels of the factor and the order.

`cut(x, breaks = 4)`
Turn a numeric vector into a factor but 'cutting' into sections.

Statistics

`lm(x ~ y, data=df)`
Linear model.

`glm(x ~ y, data=df)`
Generalised linear model.

`summary`
Get more detailed information out a model.

`t.test(x, y)`
Preform a t-test for difference between means.

`pairwise.t.test`
Preform a t-test for paired data.

`prop.test`
Test for a difference between proportions.

`aov`
Analysis of variance.

Distributions

	Random Variates	Density Function	Cumulative Distribution	Quantile
Normal	`rnorm`	`dnorm`	`pnorm`	`qnorm`
Poison	`rpois`	`dpois`	`ppois`	`qpois`
Binomial	`rbinom`	`dbinom`	`pbinom`	`qbinom`
Uniform	`runif`	`dunif`	`punif`	`qunif`

Plotting

Also see the **ggplot2** library.

`plot(x)`
Values of x in order.

`plot(x, y)`
Values of x against y.

`hist(x)`
Histogram of x.

Dates

See the **lubridate** library.