The R Book

Chapter 2: Essentials of the R Language
Session 7

Character Strings

character strings are defined by double quotation marks:

```
>a<-"abc"
>b<-"123"
```

Numbers can be characters, but characters cannot be numbers.

```
> as.numeric(a)
[1] NA
Message d'avis :
NAs introduits lors de la conversion automatique

> as.numeric(b)
[1] 123
```

Character Strings

Distinction:
- length of a character object (a vector)
- numbers of characters in the strings

```
> pets<-c("cat","dog","gerbil","terrapin")
> length(pets)
[1] 4

> nchar(pets)
[1] 3 3 6 8
```

strings are not factors

```
> class(pets)
[1] "character"
> is.factor(pets)
[1] FALSE
```

Character Strings

strings are not factors

```
> class(pets)
[1] "character"
> is.factor(pets)
[1] FALSE
```

However, if part of a dataframe, character variables coerced to act as factors:

```
> df<-data.frame(pets)
> is.factor(df$pets)
[1] TRUE
> df
      pets
1      cat
2      dog
3   gerbil
4 terrapin
```

Character Strings

built-in vectors in R that contain the 26 letters of the alphabet

```
> letters
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"

> LETTERS
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
[20] "T" "U" "V" "W" "X" "Y" "Z"
```

which number in the alphabet the letter n

```
> which(letters=="n")
[1] 14
```

to suppress the quotes

```
> noquote(letters)
 [1] a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Character Strings

concatenating strings

```
> c(a,b)
[1] "abc" "123"
```

joining strings

```
> paste(a,b,sep="")
[1] "abc123"
```

sep="", means that the two character strings are to be pasted together

Combinations of paste

```
> d<-c(a,b,"new")
> e<-paste(d,"a longer phrase containing blanks")
> e
[1] "abc a longer phrase containing blanks"
[2] "123 a longer phrase containing blanks"
[3] "new a longer phrase containing blanks"
```

Character Strings

Extracting parts of strings

```
> phrase<-"the quick brown fox jumps over the lazy dog"

> q<-character(20)
> q
 [1] "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""

> for (i in 1:20) q[i]<- substr(phrase,1,i)

> q
 [1] "t"                   "th"                  "the"
 [4] "the "                "the q"               "the qu"
 [7] "the qui"             "the quic"            "the quick"
[10] "the quick "          "the quick b"         "the quick br"
[13] "the quick bro"       "the quick brow"      "the quick brown"
[16] "the quick brown "    "the quick brown f"   "the quick brown fo"
[19] "the quick brown fox" "the quick brown fox "
```

vector with 20 “blank” entries

loops through phrase and assigns q

split up a character string into individual characters with strsplit()

```
> phrase
[1] "the quick brown fox jumps over the lazy dog"
```

```
> strsplit(phrase,split=character(0))
```
any character

```
[[1]]
 [1] "t" "h" "e" " " "q" "u" "i" "c" "k" " " "b" "r" "o" "w" "n" " " "f" "o" "x"
[20] " " "j" "u" "m" "p" "s" " " "o" "v" "e" "r" " " "t" "h" "e" " " "l" "a" "z"
[39] "y" " " "d" "o" "g"
```

Count occurances with the table function :

```
> table(strsplit(phrase,split=character(0)))

  a b c d e f g h i j k l m n o p q r s t u v w x y z
8 1 1 1 1 3 1 1 2 1 1 1 1 1 1 4 1 1 2 1 2 2 1 1 1 1 1
```

split up a character string into individual characters with strsplit()

Count occurances with the table function :

```
> table(strsplit(phrase,split=character(0)))

  a b c d e f g h i j k l m n o p q r s t u v w x y z
8 1 1 1 1 3 1 1 2 1 1 1 1 1 1 4 1 1 2 1 2 2 1 1 1 1 1
```

counting the number of words in a phrase

```
> words<-1+table(strsplit(phrase,split=character(0)))[1]
> words
9
```

extract characters between the first and second occurrences of ‘the’

```
> strsplit(phrase,"the")[[1]] [2]
[1] " quick brown fox jumps over "
```

strsplit output is a list …, second element printed

Count number of characters

```
> nchar(strsplit(phrase,"the")[[1]] [2])
[1] 28
```

Change to upper/lower case

```
> toupper(phrase)
[1] "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"
> tolower(toupper(phrase))
[1] "the quick brown fox jumps over the lazy dog"
```

The match Function

Where do the values in the second vector appear in the first vector?

```
> first<-c(5,8,3,5,3,6,4,4,2,8,8,8,4,4,6)
> second<-c(8,6,4,2)

> match(first,second)
 [1] NA  1 NA NA NA  2  3  3  4  1  1  1  3  3  2
```

produces a vector of subscripts :
matches values between « first » and « second » and assigns subscripts

drug A for patients in the first vector that were identified in the second vector, drug B to all the others

```
> drug<-c("A","B")
> drug[1+is.na(match(first,second))]
 [1] "B" "A" "B" "B" "B" "A" "A" "A" "A" "A" "A" "A"
"A" "A" "A"
```

Writing functions in R

objects that carry out operations on arguments

Syntax :
function (argument list) body

keyword function → indicates to R that you want to create a function.

argument list → comma-separated list of formal arguments.
- symbol (i.e. a variable name such as x or y),
- statement of the form symbol = expression (e.g. pch=16)

body → any valid R expression / set of R expressions.
- Generally, expressions are contained in curly brackets { },

Arithmetic mean of a single sample

```
> arithmetic.mean<-function(x) sum(x)/length(x)
> y<-c(3,3,4,5,5)
> arithmetic.mean(y)
[1] 4
```

Median of a single sample

```
> sort(y)[ceiling(length(y)/2)]
```
[1] 4 but what if y has even numbers of values ?

```
> med<-function(x) {
+ odd.even<-length(x)%%2
+ if (odd.even == 0) (sort(x)[length(x)/2]+sort(x)[1+ length(x)/2])/2
+ else sort(x)[ceiling(length(x)/2)]
+ }
> a<-c(1,2,3,4)
> med(a)
[1] 2.5
```