Pattern Matching

Given one or more strings, identify those that follow a specific **pattern** instead of a fixed set of characters.

Example:
Given a set of sequences, extract only those that contain this consensus site **TTACNNNAC**

4^3 = **64** fixed consensuses that we need to check for in each sequence that we have

Pattern matching is achieved by using a specific syntax to shape a pattern: a non-univocal String that has several options for every one of its position instead of a fixed character.

Pattern Matching in R

```
Vector <- c("chrI,strand1", "chrII,locus2"))

grep("locus", Vector)
```

pattern Vector of strings

> [1] 2

Indices of elements of the vector that are matched by the pattern

```
Vector[grep("locus", Vector)]
> "chrII,locusn2"
```

Rules of Pattern Building

Symbols

Determine one or more set of characters to be matched

Symbols That match themselves

“A”

Only matches the letter “A”

Symbols That match Groups of other symbols

“\d”

Matches all digits: 1,2,3,4,5,6,7,8,9,0

Operators

Act as modifiers for symbols, Specifying quantitative or positional requirements

“+”

Specifying that the preceding character can be matched 1 or more times

Elementary operators

	char	meaning
Positional	^	beginning of string
	$	end of string
Quantitative	*	match 0 or more times
	+	match 1 or more times
	?	match 0 or 1 times; *or*: shortest match
	{ }	repetition modifier
Alternative	|	alternative
	[ ]	set of characters

Positional operators specify the place where the adjacent character must be,
it must be placed at the very beginning or at the very end of the pattern

Quantitative operators, specify how much of a given character we want,
it must be placed after the character to which they refer

Alternative operators give two or more alternative for a given position in the pattern

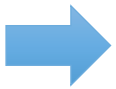
Representing groups of symbols

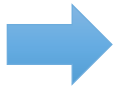
Symbol	Equivalent POSIX	matching
.		Anything
\\d	[:digit:]	Digits
\\D		Non-digits
\\w	[:alnum:]	Alphanumeric characters and _
\\W		Non-alphanumeric, non-_ characters
\\t		Tab
\\r		Carriage return
\\n		Newline
\\s	[:space:]	Whitespace
\\S		Non-whitespace
	[:punct:]	Punctuation

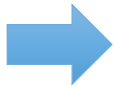
[] and | are also suitable ways of representing multiple options in few symbols

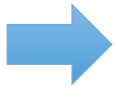
Substituting according to pattern

```
text <- c("arm","leg","head", "foot","hand", "hindleg",
"elbow")
```

```
sub("o", "O", text)
```

```
"arm" "leg" "head" "fOot" "hand" "hindleg" "elbOw"
```

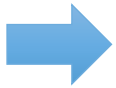
```
gsub("o", "O", text)
```

```
"arm" "leg" "head" "fOOt" "hand" "hindleg" "elbOw"
```

```
gsub("^.", "O", text)
```

```
"Orm" "Oeg" "Oead" "Ooot" "Oand" "Oindleg" "Olbow"
```

```
ChrI:+:1500:2500
ChrVII:-:1450:1600
ChrXIV:-:1450:1700
```


chrI	1500	2500	+
chrVII	1450	1600	-
chrXIV	1450	1700	-

Extracting patterns embedded in strings

ChrI:+:1500:2500		chrI	1500	2500	+
ChrVII:-:1450:1600		chrVII	1450	1600	-
ChrXIV:-:1450:1700		chrXIV	1450	1700	-

Pattern matched between parentheses are stored in special variables that can be Immediately called in the substitution

Groups can be modified (in a rather limited fashion) by adding the prefix \\U or \\L to turn the group into upper or lower case

Determining the position of the match

```
text <- c("arm","leg","head", "foot","hand", "hindleg",
"elbow")
```

```
regexpr("o",text)
```

```
[1] -1 -1 -1 2 -1 -1 4
    attr(,"match.length")
[1] -1 -1 -1 1 -1 -1 1
```

```
regexpr("\w+",text)
```

```
[1] 1 1 1 1 1 1 1
    attr(,"match.length")
[1] 3 3 4 4 4 7 5
```

%in% and which

A combination of the %in% operator and the which function can be used to provide useful and uncomplicated pattern matching between two vectors.

```
stock<-c('car','van')
requests<-c('truck','suv','van','sports','car','waggon','car')

which(requests %in% stock)
[1] 3 5 7

 requests[which(requests %in% stock)]
```

Several condition linked by Boolean operators as arguments of the which function can emulate relational databases retrieval systems.