Getting / cleaning data 2

Using regular expressions

Regular expression patterns

The easiest regular expression patterns are literal text. For example, the regular expression pattern if you're trying to match "Mr" is just "Mr":

```r
ex_names <- c("Braund, Mr. Owen Harris",
              "Cumings, Mrs. John Bradley",
              "Heikkinen, Miss. Laina")
str_extract(ex_names, pattern = "Mr")
```

```
## [1] "Mr" "Mr" NA
```

Regular expression patterns

Regular expression patterns are case sensitive, so you won't match "Mr" with the pattern "mr":

```r
ex_names <- c("Braund, Mr. Owen Harris",
              "Cumings, Mrs. John Bradley",
              "Heikkinen, Miss. Laina")
str_extract(ex_names, pattern = "mr")
```

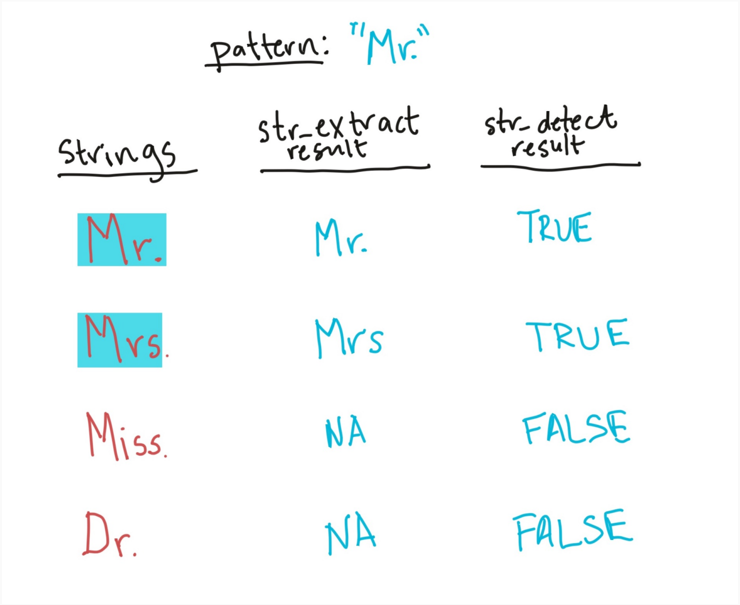
```
## [1] NA NA NA
```

Regular expression patterns

There are a few characters called **metacharacters** that mean something special in regular expression patterns.

To use any of these literally in a regular expression, you need to “protect” them with two backslashes.

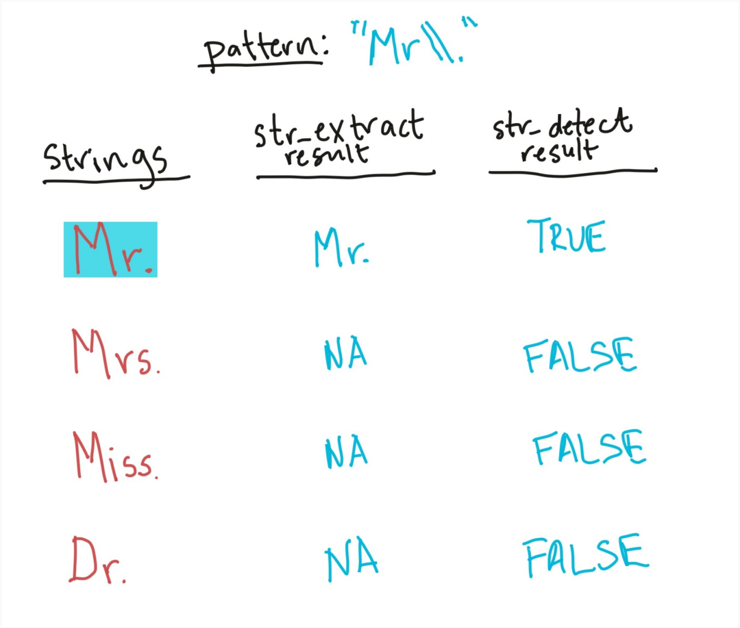
Regular expressions



pattern: "Mr."

Strings	str_extract result	str_detect result
Mr.	Mr.	TRUE
Mrs.	Mrs	TRUE
Miss.	NA	FALSE
Dr.	NA	FALSE

Regular expressions



pattern: "Mr\\."

Strings	str_extract result	str_detect result
Mr.	Mr.	TRUE
Mrs.	NA	FALSE
Miss.	NA	FALSE
Dr.	NA	FALSE

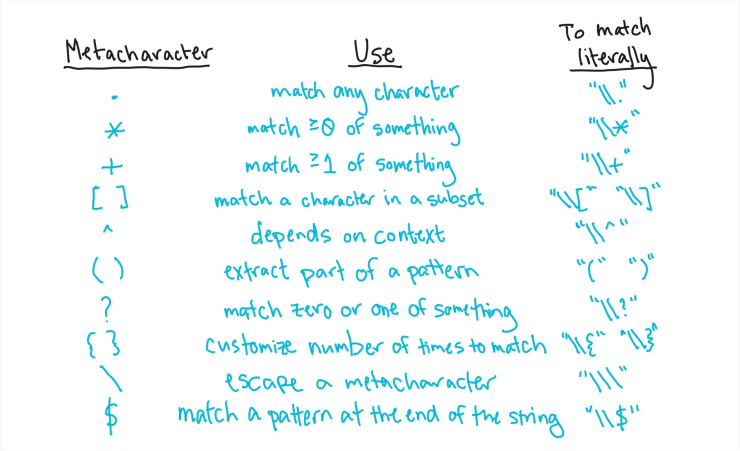
Regular expression patterns

For example, "." is a metacharacter, so to match "Mr.", you need to use the pattern "Mr\\.":

```
ex_names <- c("Braund, Mr. Owen Harris",
              "Cumings, Mrs. John Bradley",
              "Heikkinen, Miss. Laina")
str_extract(ex_names, pattern = "Mr\\.")
```

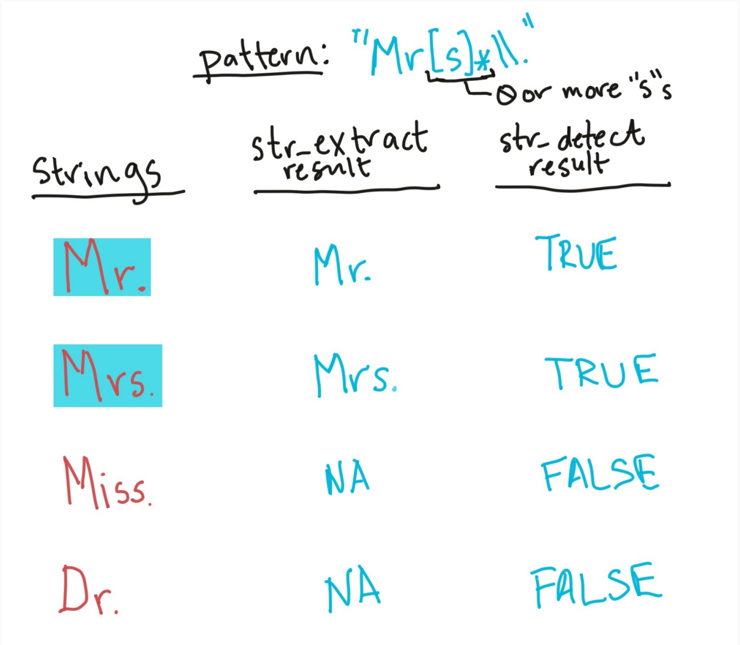
```
## [1] "Mr." NA    NA
```

Regular expression metacharacters



Metacharacter	Use	To match literally
.	match any character	"\\."
*	match ≥ 0 of something	"\*"
+	match ≥ 1 of something	"\\+"
[ ]	match a character in a subset	"\\[" "\\]"
^	depends on context	"\\^"
( )	extract part of a pattern	"(" ")"
?	match zero or one of something	"\\?"
{ }	customize number of times to match	"\\{" "\\}"
\	escape a metacharacter	"\\\"
$	match a pattern at the end of the string	"\\$"

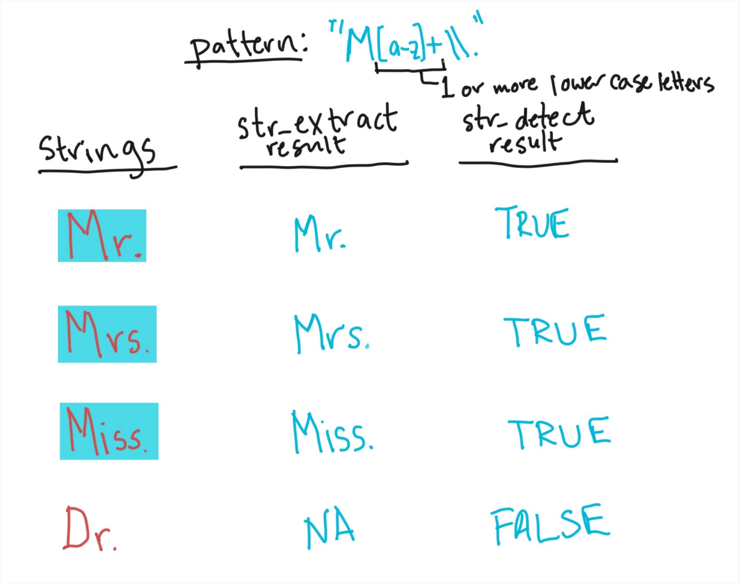
Regular expression patterns



pattern: "Mr$\underbrace{\text{[s]*}}_{\text{0 or more "s"s}}$\\."

Strings	str_extract result	str_detect result
Mr.	Mr.	TRUE
Mrs.	Mrs.	TRUE
Miss.	NA	FALSE
Dr.	NA	FALSE

Regular expression patterns



pattern: "M[a-z]+\\."

1 or more lower case letters

Strings	str_extract result	str_detect result
Mr.	Mr.	TRUE
Mrs.	Mrs.	TRUE
Miss.	Miss.	TRUE
Dr.	NA	FALSE

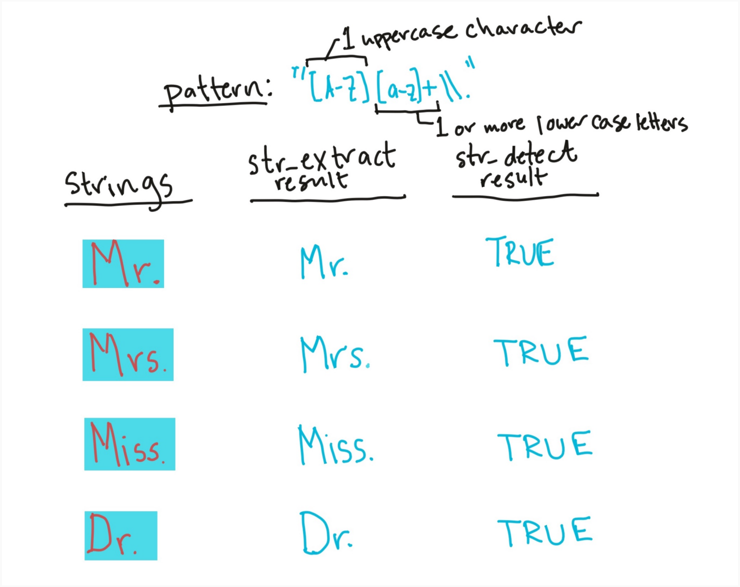
Regular expressions

The last pattern used `[a-z]+` to match one or more lowercase letters. The `[a-z]`is a **character class**.

You can also match digits (`[0-9]`), uppercase letters (`[A-Z]`), just some letters (`[aeiou]`), etc.

You can negate a character class by starting it with `^`. For example, `[^0-9]` will match anything that **isn't** a digit.

Regular expression patterns



1 uppercase character

pattern: "[A-Z][a-z]+\\."

1 or more lower case letters

Strings	str_extract result	str_detect result
Mr.	Mr.	TRUE
Mrs.	Mrs.	TRUE
Miss.	Miss.	TRUE
Dr.	Dr.	TRUE