STRING MANIPULATION WITH STRINGR

Regular expressions

Regular expressions

- A language for describing patterns

```
^.[\d]+
```

- "the start of the string, followed by any single character, followed by one or more digits"

Regular expressions as a pattern argument

```
> str_detect(c("R2-D2", "C-3P0"), pattern = "^.\\d+")
[1]  TRUE FALSE

> START %R%
    ANY_CHAR %R%
    one_or_more(DGT)

<regex> ^.[\d]+
```

rebus

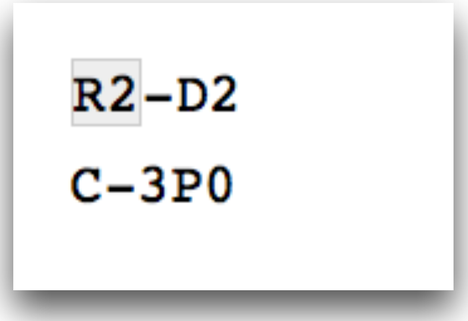
```
START %R%
    ANY_CHAR %R%
    one_or_more(DGT)
```

Regular expression

```
^.[\d]+
```

Regular expressions as a pattern argument

```
> str_detect(c("R2-D2", "C-3P0"),
    pattern = START %R%
              ANY_CHAR %R%
              one_or_more(DGT))
[1]  TRUE FALSE

> str_view(c("R2-D2", "C-3P0"),
             pattern = START %R%
                       ANY_CHAR %R%
                       one_or_more(DGT))
```



In HTML viewer

STRING MANIPULATION WITH STRINGR

Let’s practice!

STRING MANIPULATION WITH STRINGR

More regular expressions

Regular expression review

Pattern	Regular Expression	`rebus`
Start of string	^	START
End of string	$	END
Any single character	.	ANY_CHAR

Alternation

(dog|cat)

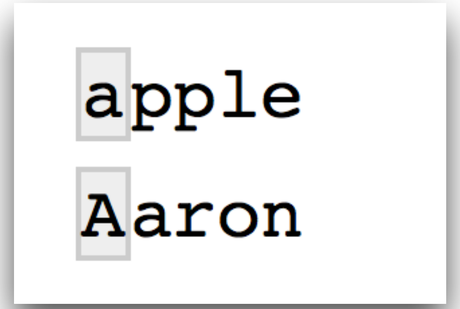
```
> or("dog", "cat")
<regex> (?:dog|cat)

> str_view(c("kittycat", "doggone"),
    pattern = or("dog", "cat"))
```

kittycat

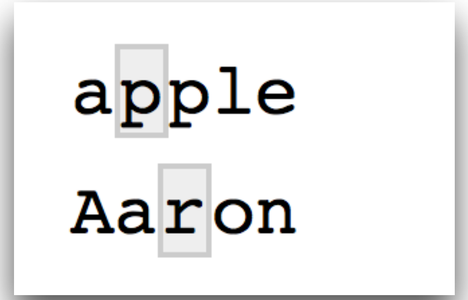
doggone

Character classes

```
> char_class("Aa")
<regex> [Aa]

> str_view(c("apple", "Aaron"),
    pattern = char_class("Aa"))
```



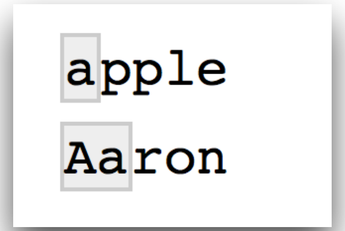

```
> negated_char_class("Aa")
<regex> [^Aa]

> str_view(c("apple", "Aaron"),
    pattern = negated_char_class("Aa"))
```



Repetition

Pattern	Regular Expression	`rebus`
Optional	`?`	`optional()`
Zero or more	`*`	`zero_or_more()`
One or more	`+`	`one_or_more()`
Between n and m times	`{n}{m}`	`repeated()`

```
> str_view(c("apple", "Aaron"),
    pattern = one_or_more("Aa"))
```



STRING MANIPULATION WITH STRINGR

Let’s practice!

STRING MANIPULATION WITH STRINGR

Shortcuts

Ranges in character classes

```
> DOLLAR %R% char_class("0123456789")
<regex> \$[0123456789]

> char_class("0-9")
<regex> [0-9]

> char_class("a-z")
<regex> [a-z]

> char_class("A-Z")
<regex> [A-Z]
```

A digit

A lower case letter

An upper case letter

Shortcuts

```
> DGT
<regex> \d
```

A digit

```
> WRD
<regex> \w
```

A word character

```
> SPC
<regex> \s
```

A whitespace character

```
> char_class("0-9")
<regex> [0-9]
```

```
> char_class("a-zA-z0-9_")
<regex> [a-zA-z0-9_]
```

National Electronic Injury Surveillance System (NEISS)

- `neiss` package https://github.com/hadley/neiss
- Injuries reported in ER of random sample of hospitals

```
19YOM-SHOULDER STRAIN-WAS TACKLED WHILE
PLAYING FOOTBALL W/ FRIENDS
```

19 year old male

STRING MANIPULATION WITH STRINGR

Let’s practice!