RegEx 101

Cheng Yan, Chao Huang

Roadmap

- Definition of regular expression
- Basic Syntax
- Advanced Syntax
- Applications in Data Science

What is a regular expression?

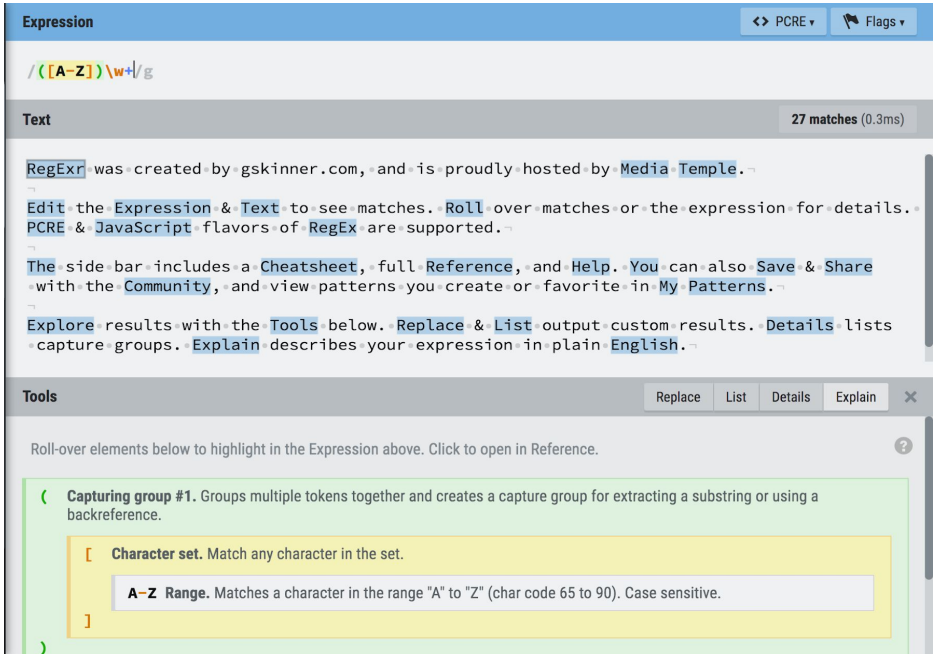
```
[a-zA-Z_\-]+@(([a-zA-Z_\-])+\.)+[a-zA-Z]{2,4}
```

- **Regular expression** (“RegEx”): sequence of char defining search patterns
  - Consist of small patterns
  - Search/Extract/Substitute characters in a string
  - Supported by text editors and command line tools
  - Implemented in almost every modern programming language
  - Powerful but maybe hard to read at first sight

When will we use RegEx?

- String manipulation
  - Renaming files
  - Parsing system log
- Web scraping
  - Extracting email address, telephone number
- Data manipulation
  - Column selection in dplyr
- ...

Useful tools for testing

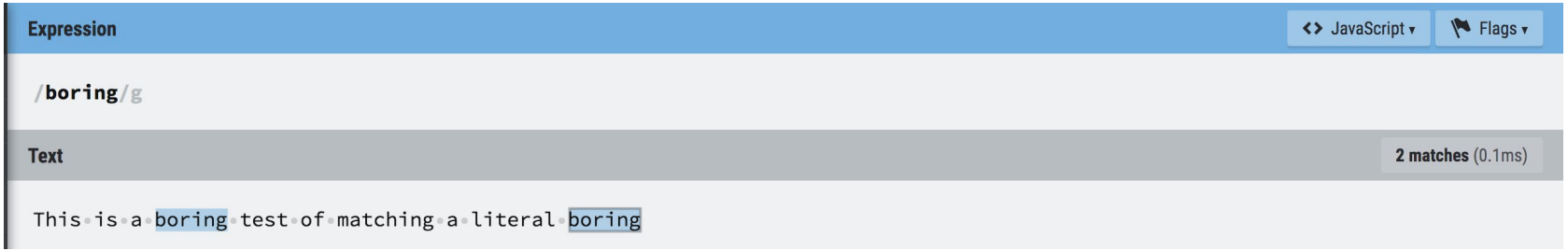


https://regexr.com/

- Great online tools to learn, try and test RegEx
- Syntax may vary a little bit between different implementations

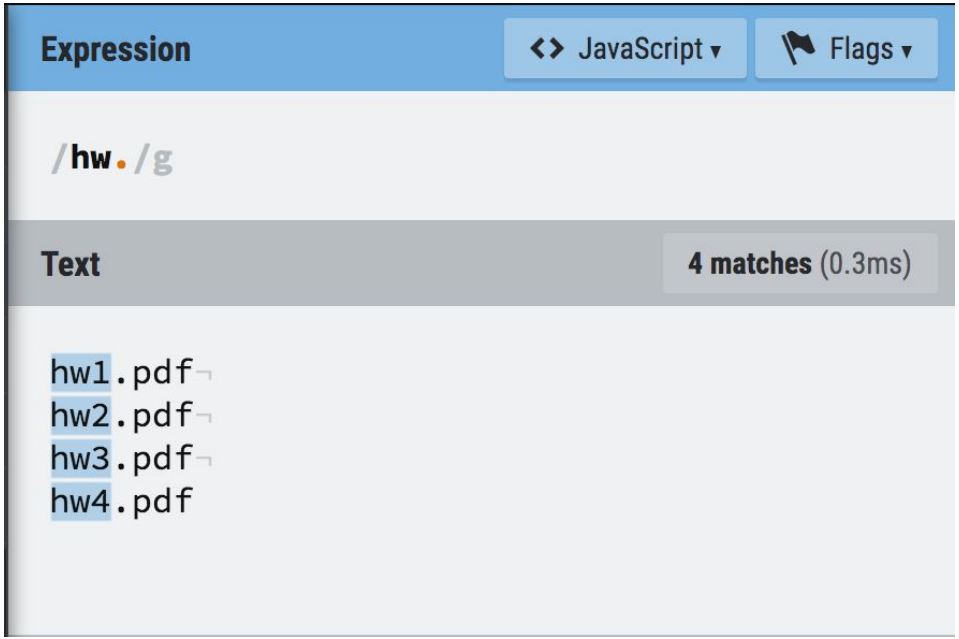
Basic Syntax

Literal Text



- Case sensitive (flag “i” in JavaScript)
- Global match (flag “g” in JavaScript)

Wildcards



. (a dot)

- matches any characters
- Use \. to represent a literal dot

Set of Characters

Expression JavaScript ▾ Flags ▾

```
/[^d][0-9]\.[Jj]pg/g
```

Text **6 matches** (0.3ms)

```
d1.jpg
c1.Jpg
f2.jpg
e3.jpg
k1.jpg
l2.jpg
m3.jpg
```

`[ ]` Match any of character within it, but not matching **all** of them

`[^]` Match any of character **except** those within the brackets

`[-]` Specify ranges, however, `[A-z]` also includes characters like “[” and “^”

Meta Characters

Expression JavaScript ▾ Flags ▾

```
/\w\d\w\d\w\d/g
```

Text **3 matches** (0.3ms)

```
11213
A1C2E3
48075
48237
M1B4F2
90046
H1H2H2
```

`\w` Any alphanumeric character in upper- or lowercase and underscore (same as `[a-zA-Z0-9_]`);
Use `\W` for negation

`\d` Any digit (same as `[0-9]`);
Use `\D` for negation

`\s` Any whitespace character;
Use `\S` for negation

Repeating Matches

Expression JavaScript ▾ Flags ▾

```
/\(?\d{3}\)?-?\d{3}-?\d{4}/g
```

Text **4 matches** (0.2ms)

```
(222)-837-9999
222-837-9999
(222)837-9999
2228379999
222.837.9999
```

`?` Match zero or 1 times

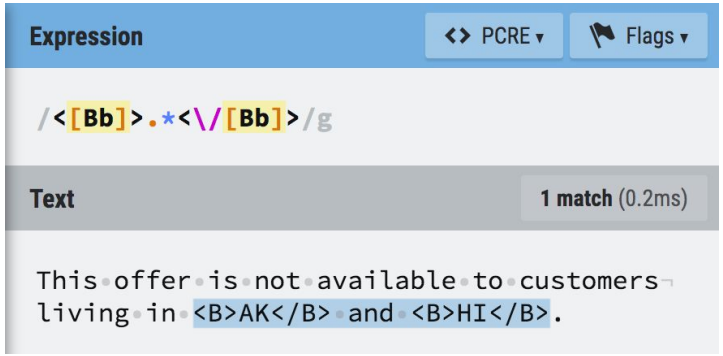
`*` Match arbitrary times (including 0)

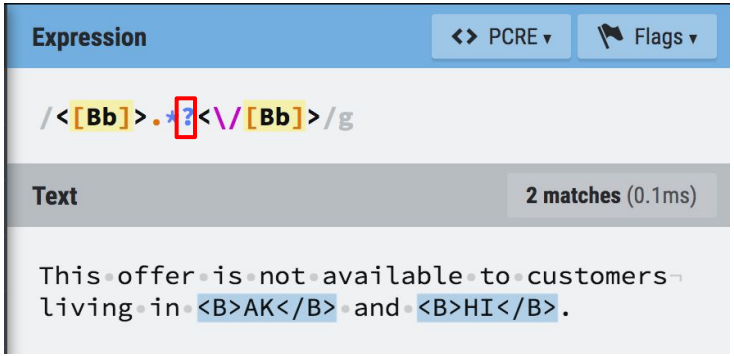
`+` Match one or more

`{min, max}`

- `{min,}` ***min*** times or more
- `{,max}` up to ***max*** times
- `{num}` ***num*** times exactly

Greedy or Lazy?





- Append **?** to the end of repeat matches with no upper bounds, e.g., **{n,}?**, ***?**, **+?**
- Default setting is greedy matching

Position Matching

Expression JavaScript ▾ Flags ▾

```
/\B\-\B/g
```

Text **1 match** (0.1ms)

```
Please enter the nine-digit id as it
appears on your color - coded pass-key.
```

Expression JavaScript ▾ Flags ▾

```
/\b\-\b/g
```

Text **2 matches** (0.2ms)

```
Please enter the nine-digit id as it
appears on your color - coded pass-key.
```

`\b` Matching positions between `\w` and `\W` (word boundaries)

`\B` Matching any positions **except** those between `\w` and `\W`

`^` Matching the start of a string

`$` Matching the end of a string

Advanced Syntax

Capturing Group

1. Group 0
2. Group 1.. \num

Regex : (\d{3}-){2}\d{4}

Group1: (\d{3}-)

Regex : (\d{3}-)(\d{3}-)\d{4}

Group1: (\d{3}-) Group2: (\d{3}-)

Regex: (\d{3}-)\1\d{4}

Look Around

Four types of look around

1. Positive look ahead ?=
2. Negative look ahead ?!
3. Positive look behind ?<=
4. Negative look behind ?<!

Application in EDAV

Exercise 2, Question 1(e)

Problem:

Wrap a long string to several lines with (approximately) same length

Solution with RegEx:

- Match a blank character (“\s”) after at least “length” characters (indicates a “look behind”)
- For look behind sub-match, use lazy mode to match as few characters as possible
- “Reset” after each match (\K, keep out match so far)
- gsub(RegEx, '\n', string, perl=TRUE)

Exercise 2, Question 1(e)

Example (length=40):

Expression PCRE ▾ Flags ▾

```
/(.{40,}?)\K\s+/g
```

Text **12 matches** (0.3ms)

```
RegExr was created by gskinner.com, and is proudly hosted by Media Temple. Edit the
Expression & Text to see matches. Roll over matches or the expression for details. PCRE &
JavaScript flavors of RegEx are supported. The side bar includes a Cheatsheet, full
Reference, and Help. You can also Save & Share with the Community, and view patterns you
create or favorite in My Patterns. Explore results with the Tools below. Replace & List
output custom results. Details lists capture groups. Explain describes your expression in
plain English.
```

Note: Use PCRE engine, which is also the engine for RegEx in R

Reference

- https://courses.cs.washington.edu/courses/cse341/10au/lectures/slides/28-regular-expressions.ppt
- Forta, Ben. *Sams teach yourself regular expressions in 10 minutes*. Sams Publishing, 2004