Bianca Ackermann - Unsplash

Regex
(puzzle time!)

Adapted from materials from Drs. Lalejini, Moore, and Bowman

Recall…

Recall…

How many lines contain the string “donut”, ignoring capitalization?

Recall…

How many lines contain the string “donut”, ignoring capitalization?

How many lines contain the string “donut” and the string “coffee”, ignoring capitalization?

Advanced searching

Advanced searching

Regular expression (regex)

Advanced searching

Regular expression (regex)

Defines a ***set*** of strings

Advanced searching

Regular expression (regex)

- Defines a ***set*** of strings
- Basic strings can represent themselves

Advanced searching

Regular expression (regex)

- Defines a ***set*** of strings
- Basic strings can represent themselves
  - e.g., dog is just the string dog

Advanced searching

Regular expression (regex)

- Defines a ***set*** of strings
- Basic strings can represent themselves
  - e.g., dog is just the string dog
- However, we also have *special* characters

Advanced searching

Regular expression (regex)

- Defines a ***set*** of strings
- Basic strings can represent themselves
  - e.g., dog is just the string dog
- However, we also have *special* characters
  - These allow us to match multiple strings

The basics

a matches the string “a”

The basics

a matches the string “a”

a+ matches one or more “a”s

The basics

a	matches the string “a”
a+	matches one or more “a”s
a*	matches zero or more “a”s

The basics

`a` matches the string “a”

`a+` matches one or more “a”s

`a*` matches zero or more “a”s

What does `lo+l` match?

The basics

`a` matches the string “a”

`a+` matches one or more “a”s

`a*` matches zero or more “a”s

What does `lo+l` match?

Where does this differ from `lo*l`?

More basics

What does `(ab)+` match?

More basics

What does `(ab)+` match?

Parentheses can group characters

More basics

What does `(ab)+` match?

Parentheses can group characters

Which ones of these wouldn't match? Why?

"ab" "abab" "" "ba" "abababababab" "aba" "aab"

A common character

What does this command do?

```
cat file_*
```

A common character

What does this command do?

```
cat file_*
```

What if we want that wildcard functionality in regex?

A common character

What does this command do?

```
cat file_*
```

What if we want that wildcard functionality in regex?

. (a dot/period) - matches *any* character

A common character

What does this command do?

```
cat file_*
```

What if we want that wildcard functionality in regex?

. (a dot/period) - matches *any* character

How do we then match the same strings as the command above?

Applying restrictions

Example: what would `(b.d)+` match? `b.+d`?

Applying restrictions

Example: what would `(b.d)+` match? `b.+d`?

What if we want to restrict the wildcard to only match vowels?

Applying restrictions

Example: what would `(b.d)+` match? `b.+d`?

What if we want to restrict the wildcard to only match vowels?

We use character classes `[ ]`

Applying restrictions

Example: what would `(b.d)+` match? `b.+d`?

What if we want to restrict the wildcard to only match vowels?

We use character classes `[ ]`

```
b[aeiou]d
```

Applying restrictions

Example: what would `(b.d)+` match? `b.+d`?

What if we want to restrict the wildcard to only match vowels?

We use character classes `[ ]`

`b[aeiou]d`

How do `(ab)+` and `[ab]+` differ?

Applying restrictions

Example: what would `(b.d)+` match? `b.+d`?

What if we want to restrict the wildcard to only match vowels?

We use character classes `[ ]`

Note we can also use tr-like character classes:

```
[[:digit:]] [a-z]
```

Applying restrictions

Example: what would `(b.d)+` match? `b.+d`?

What if we want to restrict the wildcard to only match vowels?

We use character classes `[ ]`

Note we can also use tr-like character classes:

`[[:digit:]] [a-z]`

We can also invert them: `[^[:digit:]] [^0]`

Matching the forbidden characters

What does `foo.txt` match?

Matching the forbidden characters

What does `foo.txt` match?

What if we want it to only match “foo.txt”?

Matching the forbidden characters

What does `foo.txt` match?

What if we want it to only match “foo.txt”?

```
foo\.txt
```

Recall…

How many lines contain the string “donut” and the string “coffee”, ignoring capitalization?

Recall…

How many lines contain the string “donut” and the string “coffee”, ignoring capitalization?

```
donut|coffee
```

More counts

`a` matches the string “a”

`a+` matches one or more “a”s

`a*` matches zero or more “a”s

More counts

`a`	matches the string “a”
`a+`	matches one or more “a”s
`a*`	matches zero or more “a”s
`a?`	matches zero or one “a”s

More counts

`a` matches the string “a”

`a+` matches one or more “a”s

`a*` matches zero or more “a”s

`a?` matches zero or one “a”s

`a{n}` matches n “a”s

More counts

`a` matches the string “a”

`a+` matches one or more “a”s

`a*` matches zero or more “a”s

`a?` matches zero or one “a”s

`a{n}` matches n “a”s

`a{n,m}` matches between n and m “a”s

Final characters

^a matches “a”s at the beginning of the line

a$ matches “a”s at the end of the line

Some examples

What do these match?

Some examples

What do these match?

```
The (dog|cat) ra+n away$
```

Some examples

What do these match?

```
The (dog|cat) ra+n away$
^bee+s*
```

Some examples

What do these match?

```
The (dog|cat) ra+n away$
^bee+s*
[Ll][ol]{2}[ol]*
```

Some examples

Create a regex to:

Match “ab” “aba” “abb” “abba” “abab” “abbb” “abaa”
and nothing else

Order of operations

	ERE Precedence (from high to low)	
1	Collation-related bracket symbols	[==] [::] [..]
2	Escaped characters	\<special character>
3	Bracket expression	[]
4	Grouping	()
5	Single-character-ERE duplication	* + ? {m,n}
6	Concatenation	
7	Anchoring	^ $
8	Alternation	|

Table from POSIX docs, formatting from StackOverflow

Using regular expressions

We are using extended regular expressions (ERE)

Using regular expressions

We are using extended regular expressions (ERE)

You can use them with `grep` with `-E`:

```
grep -E “ab[ab]{2}” file.txt
```

Grep actually stands for “global regular expression print”

Using regular expressions

We are using extended regular expressions (ERE)

You can use them with `grep` with `-E`:

```
grep -E “ab[ab]{2}” file.txt
```

Grep actually stands for “global regular expression print”

Note that other commands use /pattern/ to denote regex!