Lecture 11:
More RegExp, SED, AWK, GREP

Richard Veale
Fall 2021

Graduate School of Medicine
Kyoto University

RegExp in tools

Tools (sed, awk), process inputs *line by line*.

A line is matched if there is *at least one* match in the line.

Some example input file

```
[riveale@rvrbs19 regex_examples]$ cat richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
this line does not contain it!
richard veale hello
hello richard
hello richard!

There was an empty line before this (and this line does not contain!)
sans vin!
richardrichardrichard
rihcard
[riveale@rvrbs19 regex_examples]$
```

Two examples of grepping…

```
[riveale@rvrbs19 regex_examples]$ grep "richard" richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
richard veale hello
hello richard
hello richard!
richardrichardrichard
```

```
[riveale@rvrbs19 regex_examples]$ grep ".*richard.*" richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
richard veale hello
hello richard
hello richard!
richardrichardrichard
```

Prints every line that contains at least one match.
→ Highlights **every match** (depends on grep version…it's just for convenience)

Two examples of grepping...

```
[riveale@rvrbs19 regex_examples]$ grep "richard" richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
richard veale hello
hello richard
hello richard!
richardrichardrichard
```

```
[riveale@rvrbs19 regex_examples]$ grep ".*richard.*" richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
richard veale hello
hello richard
hello richard!
richardrichardrichard
```

```
[riveale@rvrbs19 regex_examples]$ grep ".*richard" richardonlines.txt
```

What will this do? (what will be matched in red?)

Two examples of grepping...

```
[riveale@rvrbs19 regex_examples]$ grep "richard" richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
richard veale hello
hello richard
hello richard!
richardrichardrichard
```

```
[riveale@rvrbs19 regex_examples]$ grep ".*richard.*" richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
richard veale hello
hello richard
hello richard!
richardrichardrichard
```

```
[riveale@rvrbs19 regex_examples]$ grep ".*richard" richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
richard veale hello
hello richard
```

This is just GREP (printing lines)

We want to use this “match” to change and extract parts of text

```
[riveale@rvrbs19 regex examples]$ sed s/richard/bob/g richardonlines.txt
```

What will this do?

This is just GREP (printing lines)

We want to use this “match” to change and extract parts of text

```
[riveale@rvrbs19 regex_examples]$ sed s/richard/bob/g richardonlines.txt
asdf827199bob83290102
1985 bob what is blah
this line does not contain it!
bob veale hello
hello bob
hello bob!

There was an empty line before this (and this line does not contain!)
sans vin!
bobbobbob
rihcard
```

Replaced every instance of “richard” with “bob”

```
[riveale@rvrbs19 regex examples]$ sed s/.*richard/bob/g richardonlines.txt
```

How about this?

```
[riveale@rvrbs19 regex_examples]$ sed s/.*richard/bob/g richardonlines.txt
bob83290102
bob what is blah
this line does not contain it!
bob veale hello
bob
bob!

There was an empty line before this (and this line does not contain!)
sans vin!
bob
rihcard
```

default sed interprets . (dot) and * (star/asterisk) as their regular expression meaning:

→ wildcard and Kleene star (0 or more repetition)

```
[riveale@rvrbs19 regex_examples]$ sed s/.*richard/bob/g richardonlines.txt
bob83290102
bob what is blah
this line does not contain it!
bob veale hello
bob
bob!

There was an empty line before this (and this line does not contain!)
sans vin!
bob
rihcard
```

It replaced the red parts (the matched/"captured" parts) with bob

```
[riveale@rvrbs19 regex_examples]$ grep ".*richard" richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
richard veale hello
hello richard
hello richard!
richardrichardrichard
```

Original file

```
[riveale@rvrbs19 regex_examples]$ cat richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
this line does not contain it!
richard veale hello
hello richard
hello richard!

There was an empty line before this (and this line does not contain!)
sans vin!
richardrichardrichard
rihcard
[riveale@rvrbs19 regex_examples]$
```

Greedy! (first 2 richard are matched by .*, then final by richard)

Original file → replaced

```
[riveale@rvrbs19 regex_examples]$ cat richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
this line does not contain it!
richard veale hello
hello richard
hello richard!

There was an empty line before this (and this line does not contain!)
sans vin!
richardrichardrichard
rihcard
[riveale@rvrbs19 regex_examples]$
```

```
[riveale@rvrbs19 regex_examples]$ sed s/.*richard/bob/g richardonlines.txt
bob83290102
bob what is blah
this line does not contain it!
bob veale hello
bob
bob!

There was an empty line before this (and this line does not contain!)
sans vin!
bob
rihcard
```

Anti-match (lines not containing?)

```
[riveale@rvrbs19 regex_examples]$ awk '!/richard/' richardonlines.txt
this line does not contain it!

There was an empty line before this (and this line does not contain!)
sans vin!
rihcard
```

! (exclamation mark) → as in many programming languages it means “NOT”

Lines containing neither “richard” NOR “veale”

Note this uses OR (||).

It means, match lines that do NOT contain richard or do not contain veale.

This will be true only when it contains NEITHER.
(i.e. identical to !(richard && veale) →by *De Morgan’s law*)

```
[riveale@rvrbs19 regex_examples]$ awk '!/richard/ || !/veale/' richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
this line does not contain it!
hello richard
hello richard!

There was an empty line before this (and this line does not contain!)
sans vin!
richardrichardrichard
rihcard
```

SED character classes

(remember the first character after mode character s is the seperator…)

What will this do?

```
[riveale@rvrbs19 regex examples]$ sed s:[0-9]:#:g richardonlines.txt
```

SED character classes

(remember the first character after s is the seperator...)

```
[riveale@rvrbs19 regex_examples]$ sed s:[0-9]:#:g richardonlines.txt
asdf######richard########
#### richard what is blah
this line does not contain it!
richard veale hello
hello richard
hello richard!

There was an empty line before this (and this line does not contain!)
sans vin!
richardrichardrichard
rihcard
```

→ Replace any digit (0-9) with the symbol #
→ g (global) at the end made it match all matches in each line

What will this do?

```
[riveale@rvrbs19 regex_examples]$ sed s:[0-9]::g richardonlines.txt
```

What will this do?

```
[riveale@rvrbs19 regex_examples]$ sed s:[0-9]::g richardonlines.txt
asdfrichard
 richard what is blah
this line does not contain it!
richard veale hello
hello richard
hello richard!

There was an empty line before this (and this line does not contain!)
sans vin!
richardrichardrichard
rihcard
```

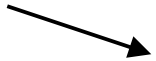
Delete all digits (0-9)

→ replace digits with “” (i.e. empty string)

SED: Getting match (&)

& means “each match”

→ But, needs to be escaped in normal sed… (so \&)

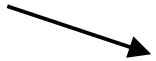


```
[riveale@rvrbs19 regex_examples]$ sed s:[0-9]:\&:g richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
this line does not contain it!
richard veale hello
hello richard
hello richard!

There was an empty line before this (and this line does not contain!)
sans vin!
richardrichardrichard
rihcard
```

SED: Getting match (&)

& means “each match”
Or we can use extended regexp
sed -E
→ so we don’t need to escape &, (,), etc.



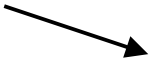
```
[riveale@rvrbs19 regex_examples]$ sed -E "s:[0-9]:&:g" richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
this line does not contain it!
richard veale hello
hello richard
hello richard!

There was an empty line before this (and this line does not contain!)
sans vin!
richardrichardrichard
rihcard
```

Good SED introduction...

https://www.grymoire.com/Unix/Sed.html

SED y/ → symbol replace



```
[riveale@rvrbs19 regex_examples]$ sed "y:01234:abcde:" richardonlines.txt
asdf8c7b99richard8dc9abac
b985 richard what is blah
this line does not contain it!
richard veale hello
hello richard
hello richard!

There was an empty line before this (and this line does not contain!)
sans vin!
richardrichardrichard
rihcard
```

Replace
0 → a
1 → b
2 → c
Etc...

SED – no mode → just matches regex

```
[riveale@rvrbs19 regex_examples]$ sed "/[0-9]/ p" richardonlines.txt
asdf827199richard83290102
asdf827199richard83290102
1985 richard what is blah
1985 richard what is blah
this line does not contain it!
richard veale hello
hello richard
hello richard!

There was an empty line before this (and this line does not contain!)
sans vin!
richardrichardrichard
rihcard
```

```
[riveale@rvrbs19 regex_examples]$ sed -n "/[0-9]/ p" richardonlines.txt
asdf827199richard83290102
1985 richard what is blah
```

With noprint -n
It only prints matches (=grep)

PERL...

PERL

Variables (scalars):

```
my $var = value;
print $var;
```

$_ ← default variable

Arrays:

```
my @var = (value1, value2, value3, …);
print $var[1];
```

PERL

Hashes (dictionaries):

my %var = (key1, value1, key2, value2, …)

My %var = (key1=>value1, key2=>value2, ...)

print $var{“key1”};

PERL (linebreaks)

Weird, but in perl, inside quotes is literal…

```
print “hello
world”;
```

Would print a line break...

PERL (loops)

https://perldoc.perl.org/perlintro#What-is-Perl

```
while (condition) { }

if (condition) { }

elsif (condition) { }

else { }

for ($i=0; $i<$max; $i++) { }

foreach (@array)
```

PERL scope

My makes local scope.

```
#local var
my $var = 2
#global var
$var = 2
```

PERL – inline regexp

if (/foo/) { }

Will execute if $_ contains “foo” (i.e. any match)

if($var =~ /foo/) {}

Will execute if $var contains foo… (any match)

Functions (subroutines)

```
sub func {
#do stuff, access input variables @_[0], @_[1], etc.
# or via shift function (which takes @_ automatically and pops next from front)
# or via my ($var1, $var2, $var3) = @_;
return $result;
}
$var = func(arg1, arg2);
```