Lecture 10
regular expressions (cont'd)

Course: Practical Bioinformatics (BIOL 4220)
Instructor: Michael Landis
Email: michael.landis@wustl.edu

Lecture 10 outline

Last time: regex intro

This time: even more regex!

regex

- regex capture
- using *sed*
- text replacement

Text replacement

So far, we've used regex to match *search patterns*

Matched text in search patterns is ***captured*** by regex as characters or groups

Captured text may be ***replaced*** in a variety of useful ways (*e.g. with sed*)

sed, stream editor

sed accepts a file or piped text as input, then edits that input line-by-line

Edits can delete lines, insert lines, and/or substitute text

Basic usage:

```
$ sed [options] 'commands' [input_file]
```

sed, commands

sed is most often used to process lines that match a regex ***search pattern*** (*e.g. '/find_me/'*)

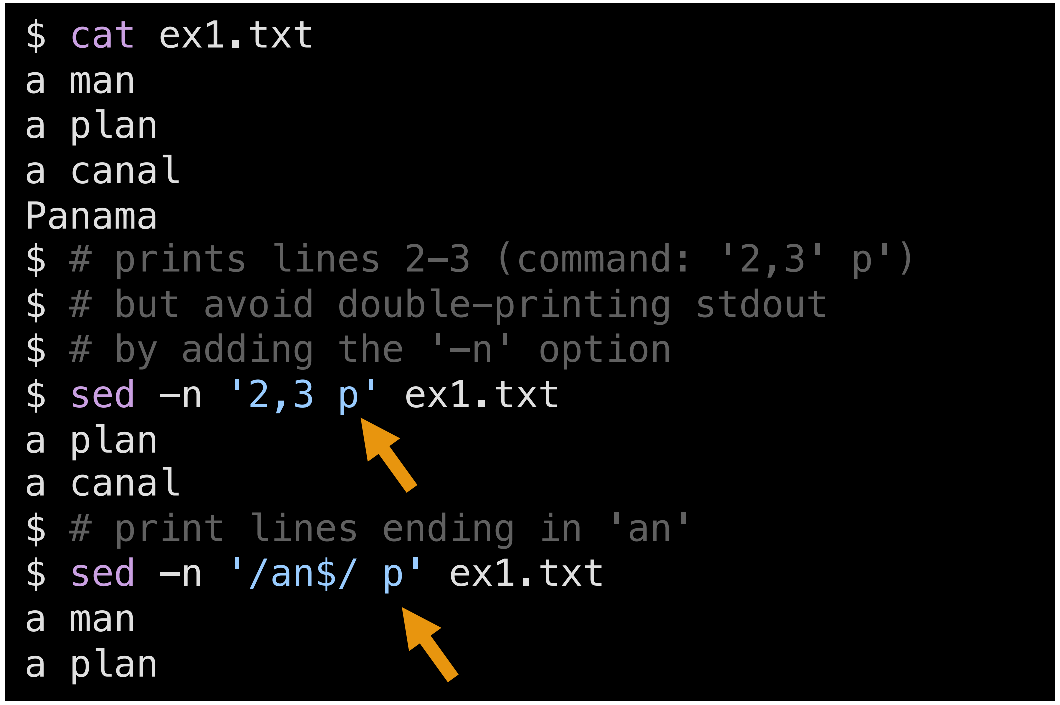
a *sed* ***command*** defines what to do when each matched line is found

Example commands + patterns:

```
/[br]at/ d     # delete lines w/ 'bat' or 'rat'
s/[br]at/cat/g # replace 'bat' or 'rat' with 'cat'
```

sed, print lines

print selected lines from the input text file or stream



```
$ cat ex1.txt
a man
a plan
a canal
Panama
$ # prints lines 2-3 (command: '2,3' p')
$ # but avoid double-printing stdout
$ # by adding the '-n' option
$ sed -n '2,3 p' ex1.txt
a plan
a canal
$ # print lines ending in 'an'
$ sed -n '/an$/ p' ex1.txt
a man
a plan
```

sed, delete lines

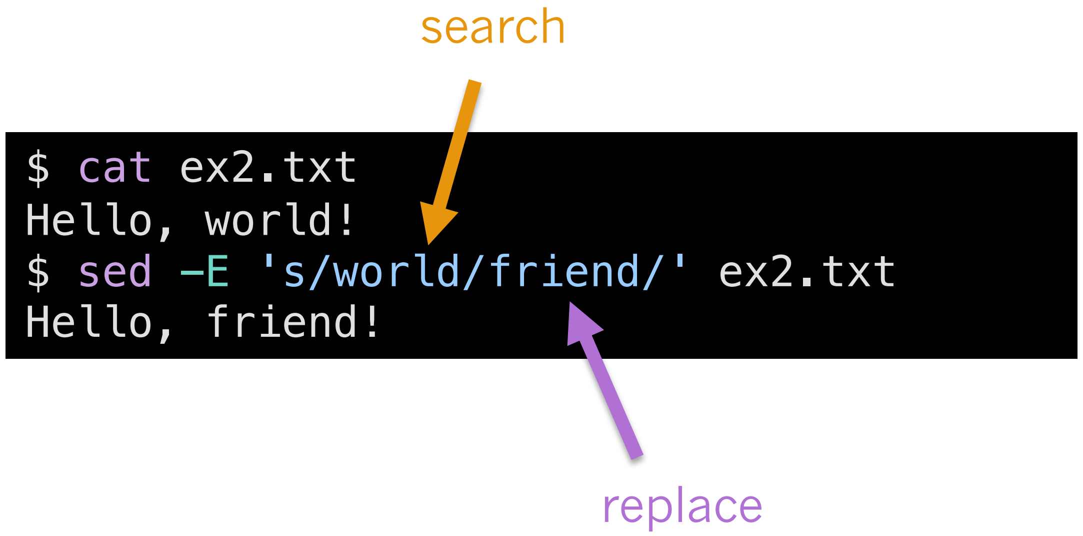
delete selected lines from
the input text file or stream

```
$ cat ex1.txt
a man
a plan
a canal
Panama
$ # delete lines 2–3 (command: '2,3 d')
$ # found in text, and then
$ # print remaining lines to stdout
$ sed '2,3 d' ex1.txt
a man
Panama
$ # delete lines containing 'ana'
$ sed '/ana/ d' ex1.txt
a man
a plan
```

sed, text substitution

The most common use for *sed* is ***text substitution***, where the regex defines a search-and-replace pattern

We'll use the *Extended Regular Expression* grammar features, enabled with *sed -E*

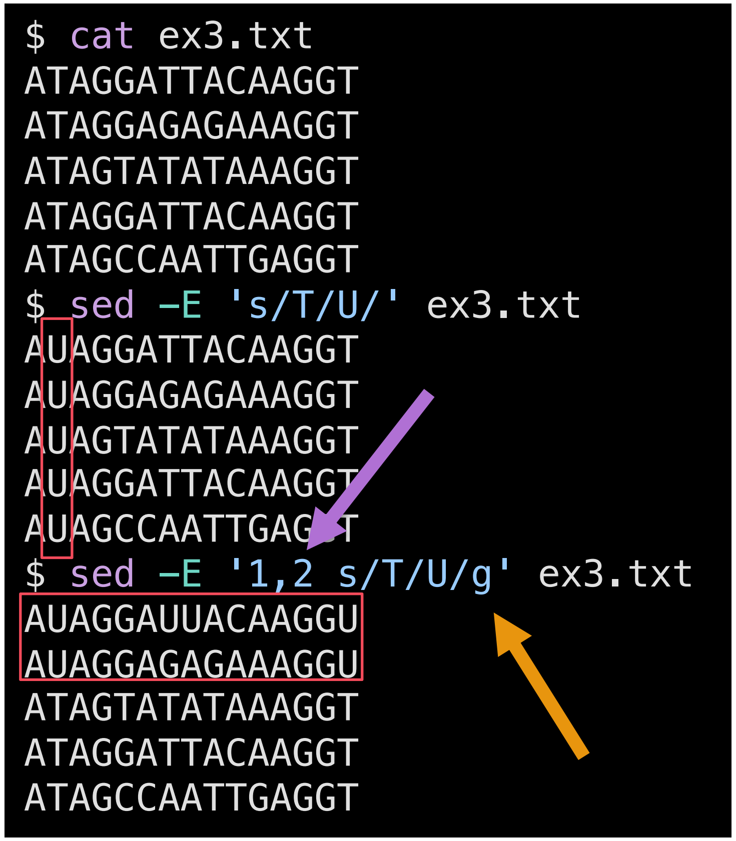


```
$ cat ex2.txt
Hello, world!
$ sed -E 's/world/friend/' ex2.txt
Hello, friend!
```

sed, substitution scope

Search commands match the first occurrence per line (default)

Searches can target a range of rows (*n,m*) or all occurrences (g for global) within each line



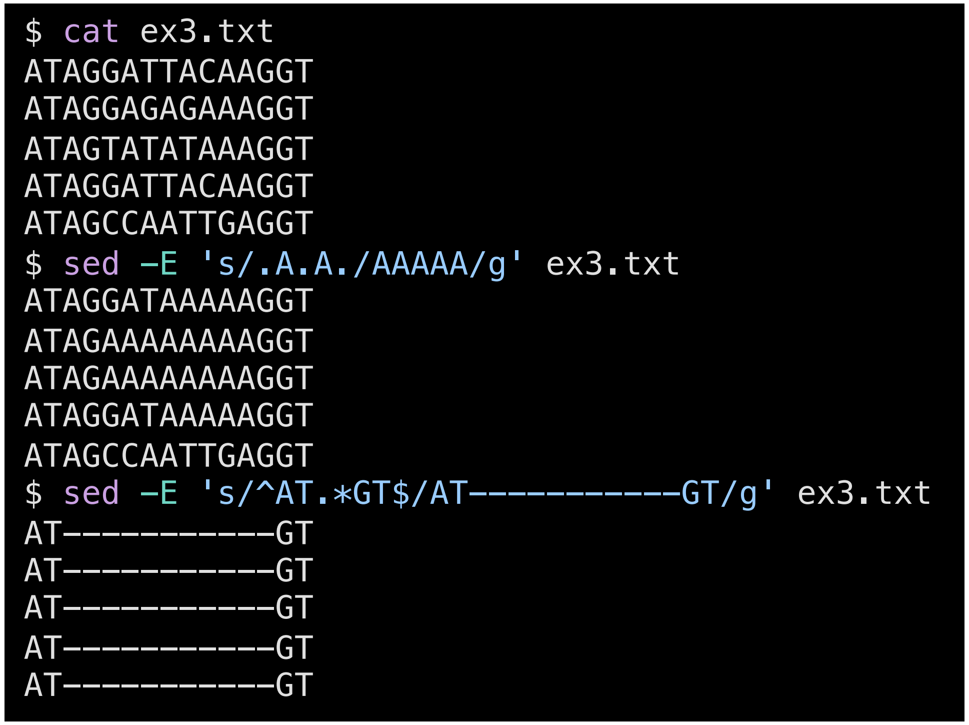
```
$ cat ex3.txt
ATAGGATTACAAGGT
ATAGGAGAGAAAGGT
ATAGTATATAAAGGT
ATAGGATTACAAGGT
ATAGCCAATTGAGGT
$ sed -E 's/T/U/' ex3.txt
AUAGGATTACAAGGT
AUAGGAGAGAAAGGT
AUAGTATATAAAGGT
AUAGGATTACAAGGT
AUAGCCAATTGAGGT
$ sed -E '1,2 s/T/U/g' ex3.txt
AUAGGAUUACAAGGU
AUAGGAGAGAAAGGU
ATAGTATATAAAGGT
ATAGGATTACAAGGT
ATAGCCAATTGAGGT
```

sed, substitution anatomy

Substitution commands have the general format:

```
n,m s/find/replace/y
```

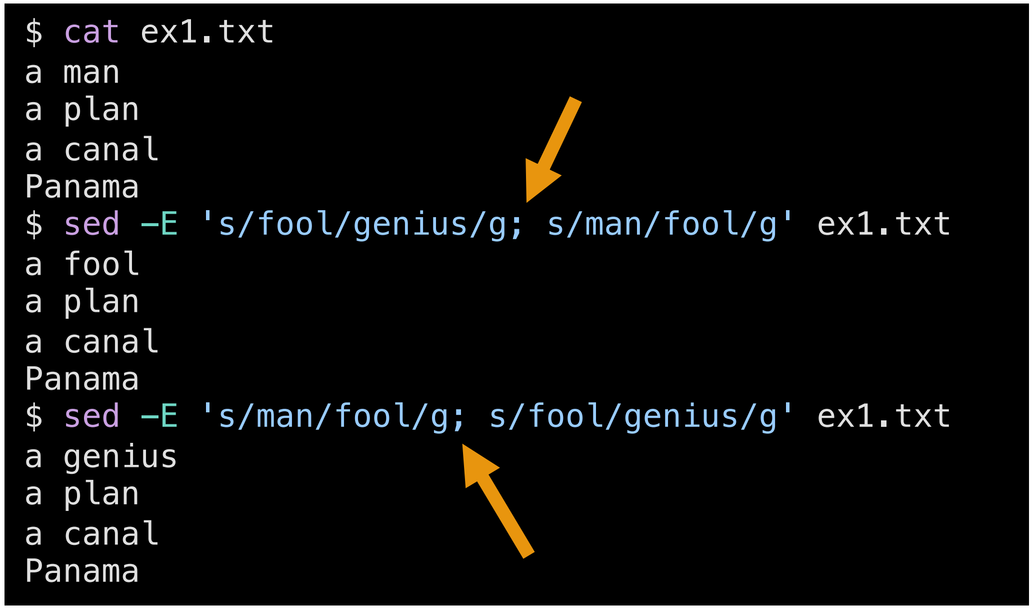
sed search patterns support regex (like *grep -P*)



```
$ cat ex3.txt
ATAGGATTACAAGGT
ATAGGAGAGAAAGGT
ATAGTATATAAAGGT
ATAGGATTACAAGGT
ATAGCCAATTGAGGT
$ sed -E 's/.A.A./AAAAA/g' ex3.txt
ATAGGATAAAAAGGT
ATAGAAAAAAAAGGT
ATAGAAAAAAAAGGT
ATAGGATAAAAAGGT
ATAGCCAATTGAGGT
$ sed -E 's/^AT.*GT$/AT-----------GT/g' ex3.txt
AT-----------GT
AT-----------GT
AT-----------GT
AT-----------GT
AT-----------GT
```

sed, multiple commands

sed can execute ***multiple commands*** in order,
e.g. when commands are separated by semicolons

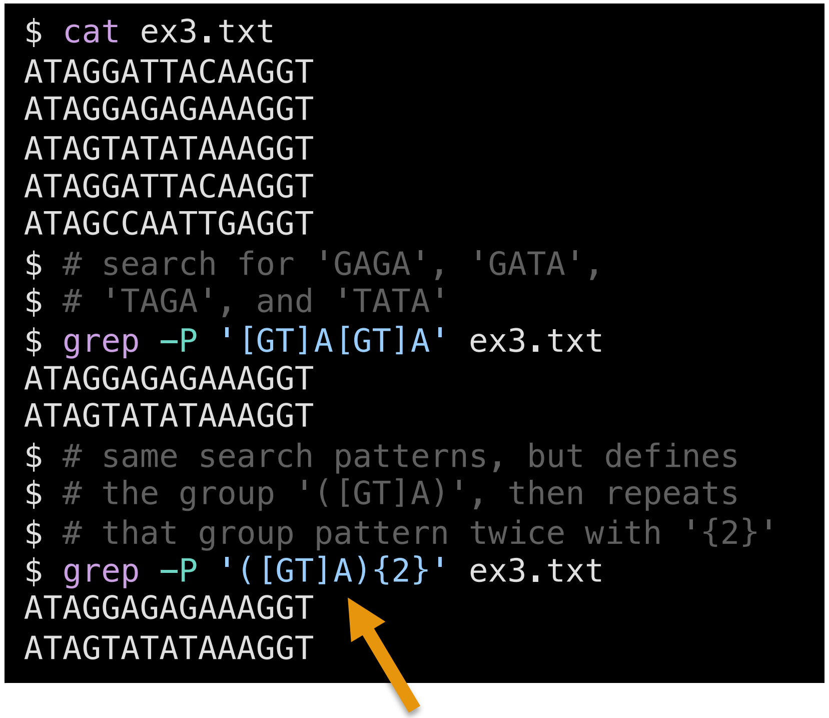


```
$ cat ex1.txt
a man
a plan
a canal
Panama
$ sed -E 's/fool/genius/g; s/man/fool/g' ex1.txt
a fool
a plan
a canal
Panama
$ sed -E 's/man/fool/g; s/fool/genius/g' ex1.txt
a genius
a plan
a canal
Panama
```

Group patterns, *a(bc)*

Use parentheses to define ***groups*** in the search pattern – *e.g. a(bc) defines bc as a group*

Like character patterns, group patterns can be modified with wildcards, repeats, *etc.*

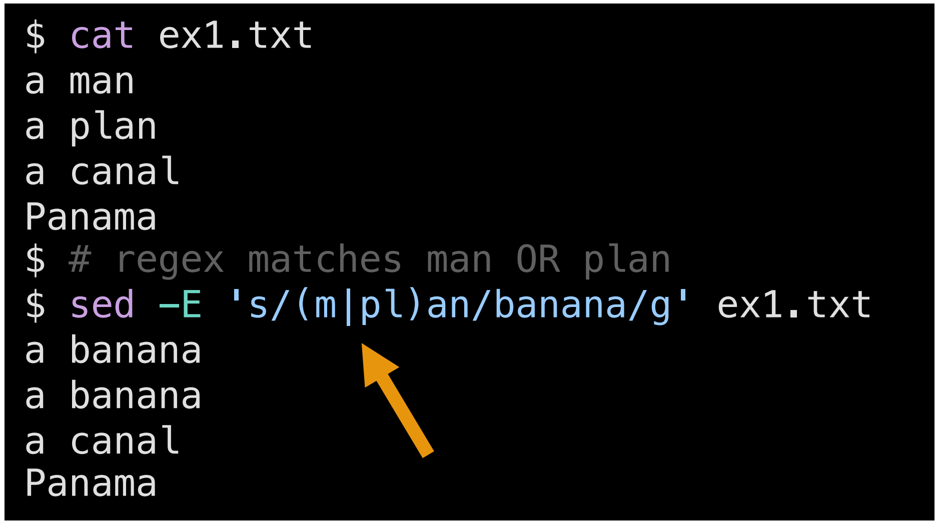


```
$ cat ex3.txt
ATAGGATTACAAGGT
ATAGGAGAGAAAGGT
ATAGTATATAAAGGT
ATAGGATTACAAGGT
ATAGCCAATTGAGGT
$ # search for 'GAGA', 'GATA',
$ # 'TAGA', and 'TATA'
$ grep -P '[GT]A[GT]A' ex3.txt
ATAGGAGAGAAAGGT
ATAGTATATAAAGGT
$ # same search patterns, but defines
$ # the group '([GT]A)', then repeats
$ # that group pattern twice with '{2}'
$ grep -P '([GT]A){2}' ex3.txt
ATAGGAGAGAAAGGT
ATAGTATATAAAGGT
```

Either-or, *a(bc|de)*

This regex will match text following either the pattern *abc* or *ade*

Analogous to character sets *[ab]*, except it matches against the entire pattern instead of single characters

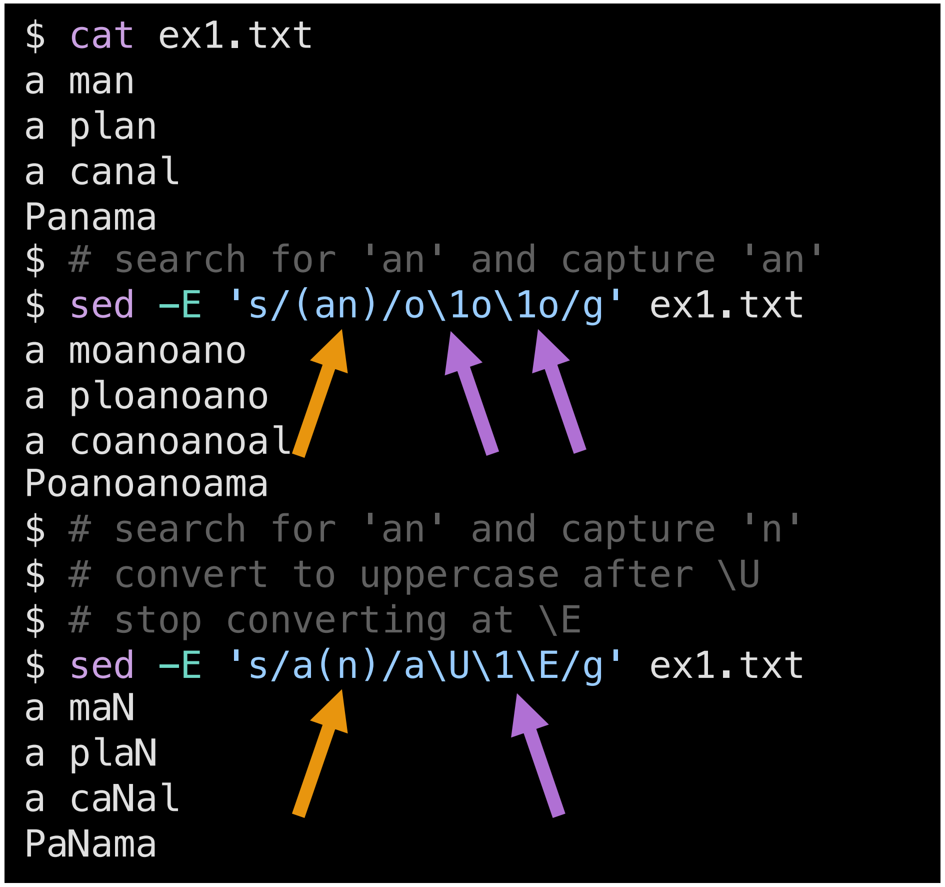


```
$ cat ex1.txt
a man
a plan
a canal
Panama
$ # regex matches man OR plan
$ sed -E 's/(m|pl)an/banana/g' ex1.txt
a banana
a banana
a canal
Panama
```

Capture & backreference

Any text matched within a group is ***captured***

Captured test may be inserted into the ***replace pattern*** using a ***backreference*** variable (\0, \1, ...)

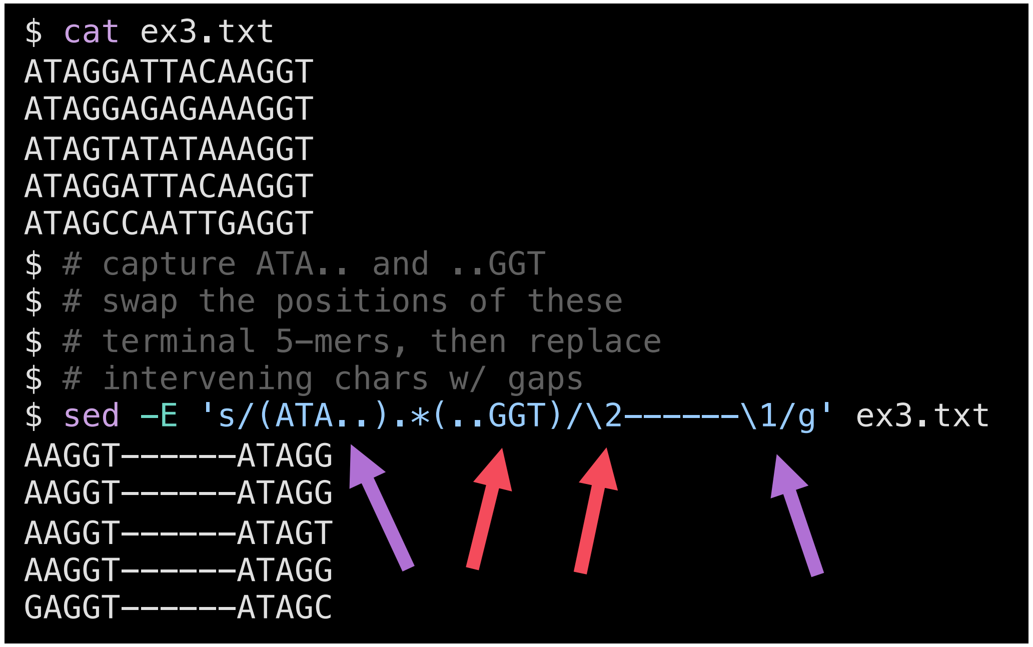


```
$ cat ex1.txt
a man
a plan
a canal
Panama
$ # search for 'an' and capture 'an'
$ sed -E 's/(an)/o\1o\1o/g' ex1.txt
a moanoano
a ploanoano
a coanoanoal
Poanoanoama
$ # search for 'an' and capture 'n'
$ # convert to uppercase after \U
$ # stop converting at \E
$ sed -E 's/a(n)/a\U\1\E/g' ex1.txt
a maN
a plaN
a caNal
PaNama
```

Multiple groups, *(a)bc(d)*

Each set of parentheses defines a different group

Captured patterns are *backreferenced* by numbered variables, named by their order of capture (\0, \1, \2)

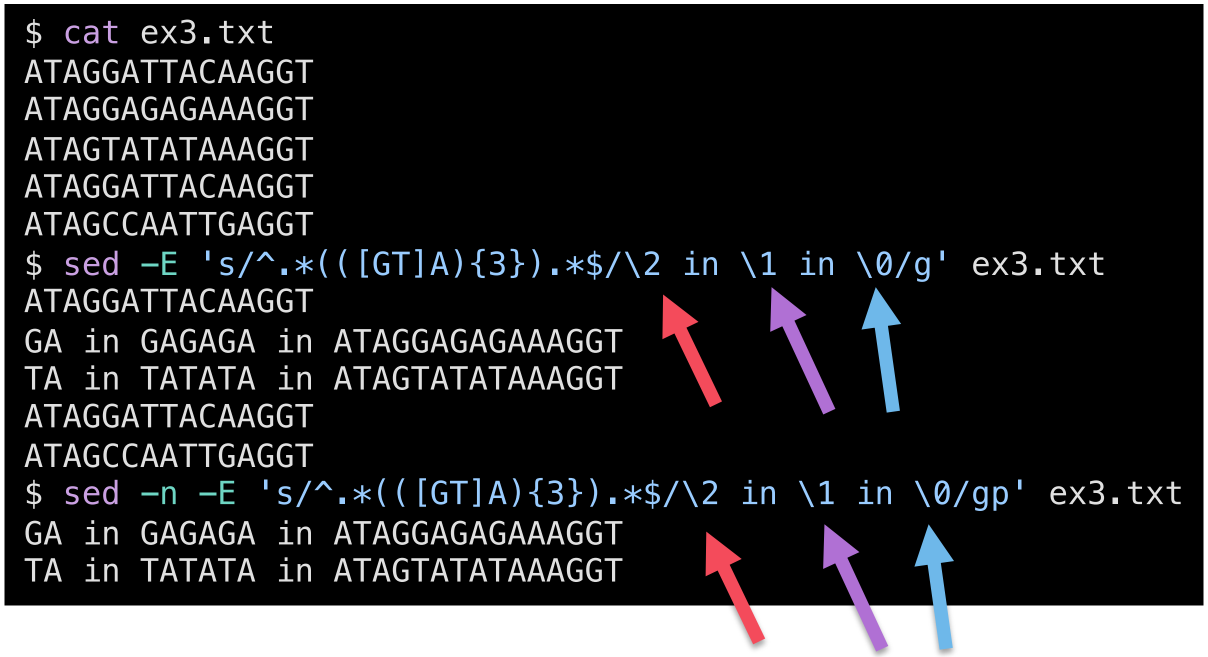


```
$ cat ex3.txt
ATAGGATTACAAGGT
ATAGGAGAGAAAGGT
ATAGTATATAAAGGT
ATAGGATTACAAGGT
ATAGCCAATTGAGGT
$ # capture ATA.. and ..GGT
$ # swap the positions of these
$ # terminal 5-mers, then replace
$ # intervening chars w/ gaps
$ sed -E 's/(ATA..).*(..GGT)/\2------\1/g' ex3.txt
AAGGT------ATAGG
AAGGT------ATAGG
AAGGT------ATAGT
AAGGT------ATAGG
GAGGT------ATAGC
```

Nested groups, *ab(c(d))*

Group that contain other groups are ***nested groups***

Backreferenced variables are numbered from out-to-in, from left-to-right (\0, \1, \2)



```
$ cat ex3.txt
ATAGGATTACAAGGT
ATAGGAGAGAAAGGT
ATAGTATATAAAGGT
ATAGGATTACAAGGT
ATAGCCAATTGAGGT
$ sed -E 's/^.*(([GT]A){3}).*$/\2 in \1 in \0/g' ex3.txt
ATAGGATTACAAGGT
GA in GAGAGA in ATAGGAGAGAAAGGT
TA in TATATA in ATAGTATATAAAGGT
ATAGGATTACAAGGT
ATAGCCAATTGAGGT
$ sed -n -E 's/^.*(([GT]A){3}).*$/\2 in \1 in \0/gp' ex3.txt
GA in GAGAGA in ATAGGAGAGAAAGGT
TA in TATATA in ATAGTATATAAAGGT
```

Overview for Lab 10