Part Three

Substitution with sed

The power of sed

- search and replace (with style)
- extract or delete specific lines
- rearranging text

sed syntax

```
sed [OPTIONS] [operation]
sed [OPTIONS] ‘[ADDRESS(s)[!]] COMMAND[ARGUMENT(S)]’
```

```
sed -r ‘/^ORIGIN/,/^$/ s/[0-9]| //g’ a.txt
```

1) sed accepts lines of input from STDIN
2) **-r** (**-E** on mac) option enables **E**xtended regular expressions
3) run command on lines between ORIGIN and empty line
4) removes numbers and space on addressed lines

sed pseudocode

for each line of input
    remove trailing newline character
    if line matches address
        perform user's commands
    if `-n` option is not set
        append newline and print

sed will not hurt your data

sed reads your data and writes to STDOUT.

STDOUT will pour into your terminal unless redirected to a pipe or file.

Your original file is perfectly safe

sed won’t, but YOU can

NEVER REDIRECT TO ORIGIN

```
# This will overwrite z.txt before any data even
# enters the prog1
cat z.txt | prog1 | prog2 > z.txt
```

Sending input to sed

```
# Pipe data into sed
$ sort a.txt | sed '/hi/'
# Read from a file, ignore STDIN
$ sed '/hi/' a.txt
# Wait for user input
$ sed '/hi/'
```

deletion (d)

The address can be a number or a regular expression:

```
$ sed 'd'          # delete everything
$ sed '1d'         # delete 1st line
$ sed '/here/d'    # delete lines containing 'here'
$ sed '5,10d'      # delete lines 5 to 10
$ sed '10,$d'      # delete lines from 10 on
$ sed '/A/,/B/!d'  # delete lines NOT between A and B
```

print (p)

`-n` option turns off automatic printing of each line

```
$ sed -n '1,10p'          # like head -10
$ sed -n '/this/p'        # like grep this
$ sed -n '/ENTRY/,/^$/p'  # to empty line
$ sed -n '/>/p'           # print only if line starts with '>'
```

Try using the print command without -n, what happens?

What about 'p' without -n?

substitution (s)

syntax: **s**/pattern/replacement/**flags**

```
# replace each line's 1st 'this' with 'that'
sed 's/this/that/'

# use global (g) flag to replace EVERY 'this' with 'that'
sed 's/this/that/g'
```

Substitution flags

```
g         replace every match on line
p         print line (usually use with -n)
I         ignore case
[0-512]   replace nth match
```

substitution examples (1)

```
$ sed ‘/here/,/there/ s/hi/bey/g’
$ sed ‘s/GI://’          # delete first ‘GI:’ string
$ sed -n ‘s/GI://p’      # remove first ‘GI:’; if no GI, no print
$ sed -n ‘s/gi://Ip’     # ignore case
$ sed ‘s/./*/60          # replace 60th char with ‘*’
$ sed ‘s/[0-9]/*/g4      # replace numbers 4th on with ‘*’
```

Regular Expressions (1)

```
.       matches any character except a newline
*       matches 0 or more of the previous char
?       matches 0 or 1 of the precious char
[...]   matches any of the enclosed
[^...] matches everything EXCEPT the enclosed
^       anchors match at the BEGINNING of the line
$       anchors match at the END of the line
\       escapes the following special character
```

Substitution Examples (2)

```
$ sed ‘s/,.*//’          # remove everything after ‘,’
$ sed ‘s/[0-9]/*/g’      # ‘*’ literal in replace
$ sed ‘s/^ *//’          # Remove leading space
$ sed ‘/^$/d’            # Deletes empty lines
$ sed ‘/^\[[0-9]*\]/d’ # e.g. ‘[45] asdf’
$ sed ‘s/.*/(&)/’        # enclose line in parentheses
```

Regular Expressions (2)

`a|b`	matches patterns a OR b
`{x}`	matches x of the previous char
`{x,y}`	matches between x and y of the previous char
`(...)`	captures the enclosed
`\n`	recalls *n*th captured sequence
`+`	matches 1 or more of the previous char

All of these require the **-r** argument (**-E** on mac)

substitution examples (3)

```
# substitute ‘grep’, ‘sed’, or ‘awk’ for ‘perl’
$ sed -r ‘s/grep|sed|awk/perl/g’
# here we escape the special meaning of ‘|’
$ sed -r ‘s/^gi\|([0-9]+).*/\1/’
# remove leading and trailing space
$ sed -r ‘s/^ +| +$//g’
# e.g. replaces ‘123-4567’ with ‘(123)-xxxx’
$ sed -r ‘s/([0-9]{3})-[0-9]{4}/(\1)-xxxx/g’
```

Insert, append, and change

i STR insert line before match

a STR insert line after match

c STR replace line with STR

STR can be any string you write in, if you write nothing, it will be an empty line

Examples

```
# add blank line before ‘>’ match
$ sed ‘/>/i’
# Add a line after a match
$ sed ‘/>/a The previous line had a >’
# blank lines to ‘//’
$ sed ‘/^$/c //’
$ sed ‘$a’
```

Strategies with sed (1)

- Be specific
  - limit the search as much as possible with addresses

```
sed '/^>/ s/Bob/Jerry/g'
```

  - match the context of the thing-to-be replaced when reasonable

```
sed '/^>/ s/gi\|([0-9]+)\|/\1/g'
```

Strategies with sed (2)

- Be general
  - Use regex
  - Avoid hardcoding assumptions

Strategies with sed (3)

- **Extract** one of more words from a line
  - Match entire line, capture desired words

```
sed -r 's/.*([0-9]+).*/\1/'
```

  - Match enough of the context to be unambiguous

```
sed -r 's/.*id=([0-9]+).*/\1/'
```

  - If no context is necessary, just use `grep -o`

```
grep -oE '[0-9]+'
```

Strategies with sed (4)

- Extract multiple line record
  - Look carefully at the data
  - Build and **test** a command matching beginning

```
sed -n ‘/BEGIN/p’
```

  - Build and test one for the end

```
sed -n ‘/END/p’
```

  - Then link them

```
sed -n ‘/BEGIN/,/END/p;
```

Your Turn

Open the **2nd** folder and read the note

Supplemnetal Material

Replacement Patterns

`&`	recalls entire matched text
`\n`	recalls *n*th grouped pattern
`\u`	first char of replacement to uppercase
`\U`	entire replacement to uppercase
`\l`	first char of replacement to lowercase
`\L`	entire replacement to lowercase

Examples

```
$ sed ‘s/.*/\U&/’ # everything to uppercase
```

let’s scream! -> LET’S SCREAM!

```
$ sed ‘s/([A-Z]+)/\l\1/g’
```

hi HOW ARE YOU? -> hi How Are You?

```
$ sed ‘s/help/”&”/’ # help me -> “help” me
```

sed scripts (-f option)

Paste these lines into file `cleanfasta`:

```
    s/^ *| *$//g
    /^>/! s/.*/\U&/
    /^>/! s/\*$//
    /^$/d; /^>/i
    $a
<in> | sed -rf cleanfasta
```

Multiple commands

```
# print lines 2 to END unless match bad
$ sed -n ‘/bad/d; 2,$p’
# Prints from HERE to the line END, delete this
$ sed -n ‘/HERE/,/END/p; s/this//’
```

If you want perl regex ...

1. use **ssed** - sed implementation that supports perl regex, e.g.

```
$ ssed -R ‘s/^\s//’
```

2. actually call perl, e.g.

```
$ perl -pe ‘tr/atgc/tacg/’
```