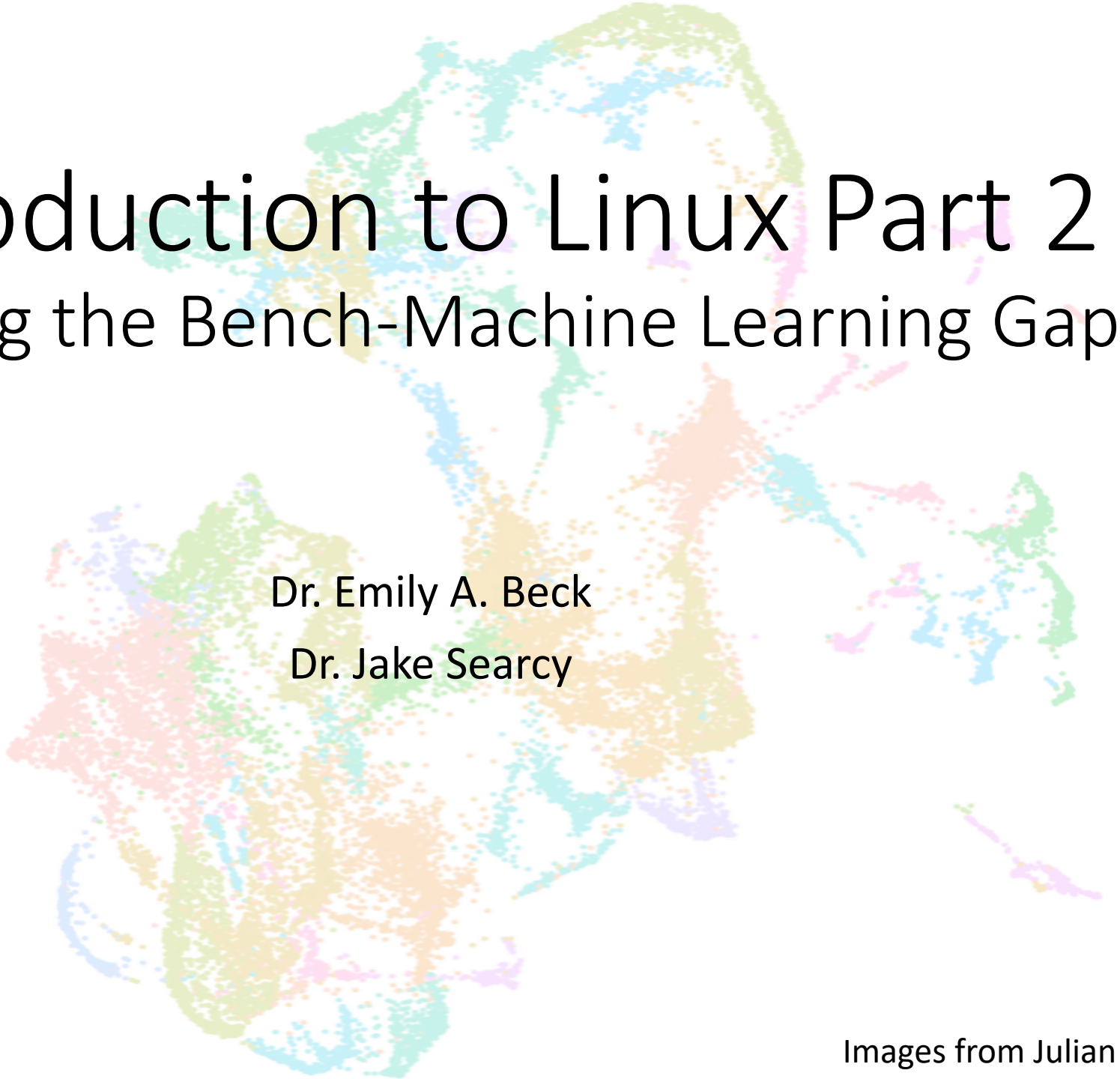
Introduction to Linux Part 2

Bridging the Bench-Machine Learning Gap

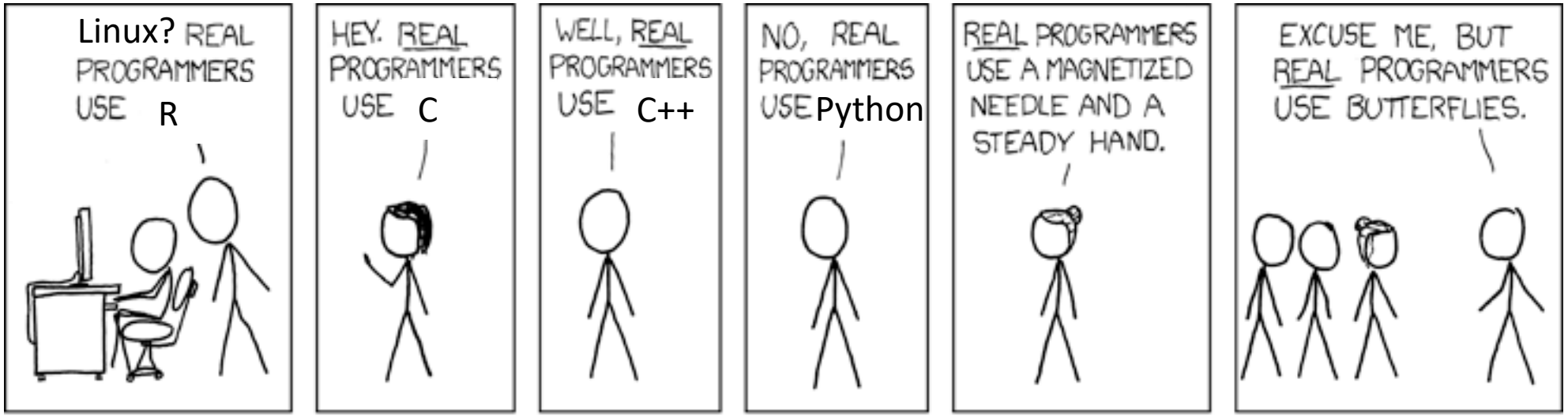
Dr. Emily A. Beck

Dr. Jake Searcy



Images from Julian Catchen-Evomics

There are MANY ways to do a task and MANY tools to help you! Use what works for YOU.



Today we are sticking with Linux and then we will learn alternatives in R later

Learning Objectives

- Understand the value of regular expressions
- Learn to parse and subset files
- Learn to edit fields
- Gain the confidence to learn any text manipulation your heart desires


WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

Regular expressions can be used to optimize searches

Use regular expressions to find patterns:

1. Something made of numbers
2. Something made of letters
3. A certain length
4. Containing special characters
5. Location in a line

Very useful when working with large text files

Useful cheat sheet for regular expressions

Encoding	Modern Equivalent	Pattern Type
`.`		a single character
`.+`		one or more characters
`.*`		zero or more characters
`.?`		Maybe present
`^`		first on the line
`$`		last on the line
`[0-9]`	`\d`	digits
`[a-zA-Z]`	`\w`	letters
`' '`	`\s \t`	space
`{3}`		must be exactly 3 characters long
`{3,5}`		between 3-5 characters long
`[ACGT]`		a specific set of characters (a class)

How would you represent:

A 7 digit number

A date using convention Month,YYYY

A phone number using convention ###-###-####

Useful cheat sheet for regular expressions

Encoding	Modern Equivalent	Pattern Type
`.`		a single character
`.+`		one or more characters
`.*`		zero or more characters
`.?`		Maybe present
`^`		first on the line
`$`		last on the line
`[0-9]`	`\d`	digits
`[a-zA-Z]`	`\w`	letters
`' '`	`\s \t`	space
`{3}`		must be exactly 3 characters long
`{3,5}`		between 3-5 characters long
`[ACGT]`		a specific set of characters (a class)

How would you represent:

A 7 digit number

A date using convention Month,YYYY

A phone number using convention ###-###-####

TRICKY!! Dashes are special characters. Can you find a workaround?

Useful cheat sheet for regular expressions

Encoding	Modern Equivalent	Pattern Type
.		a single character
.+		one or more characters
.*		zero or more characters
.?		Maybe present
^		first on the line
$		last on the line
[0-9]	\d	digits
[a-zA-Z]	\w	letters
' '	\s \t	space
{3}		must be exactly 3 characters long
{3,5}		between 3-5 characters long
[ACGT]		a specific set of characters (a class)

How would you represent:

A 7 digit number?

.....

[0-9]{7}

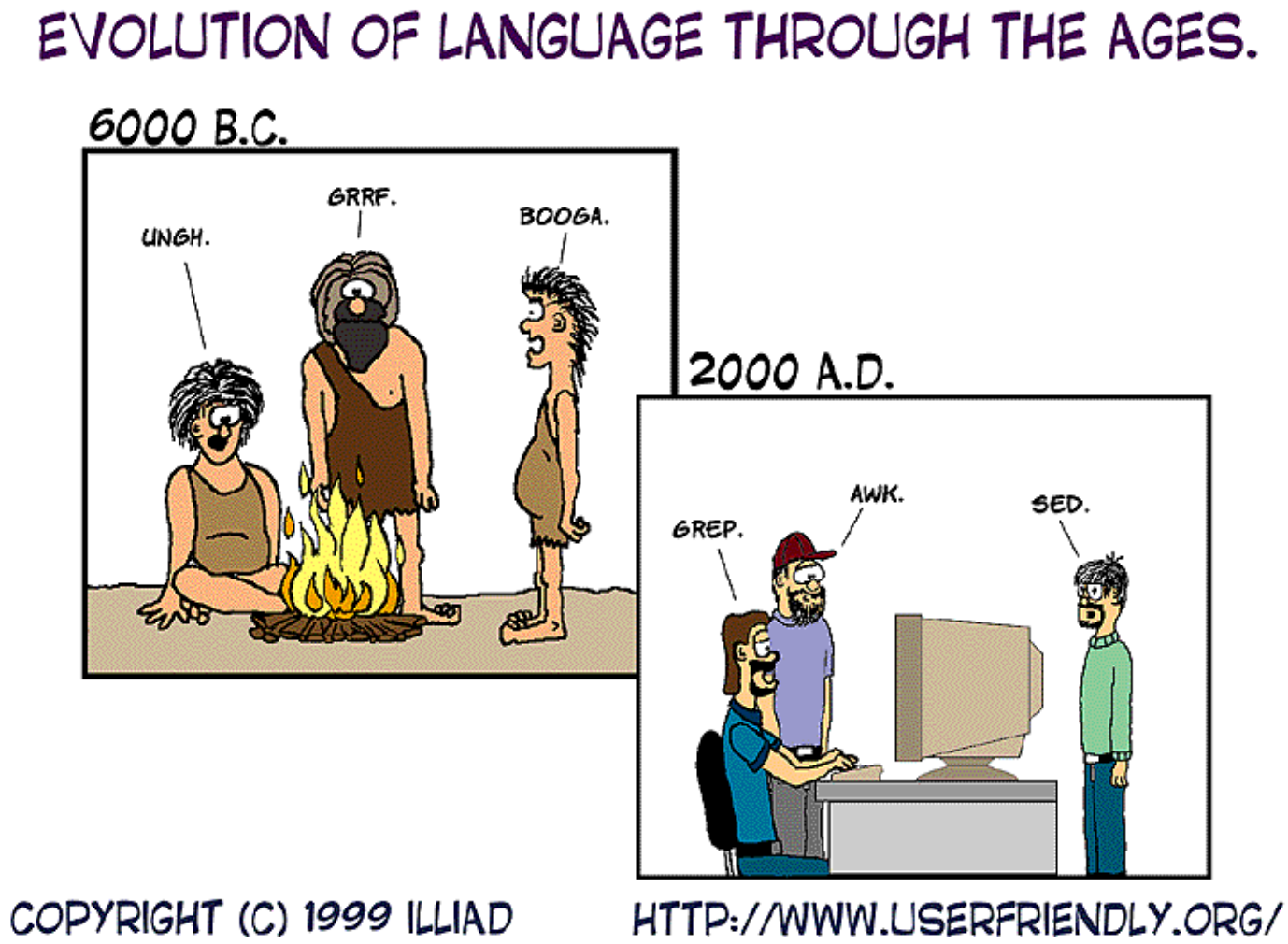
A date using convention Month,YYYY

[a-zA-Z]+,[0-9]{4}

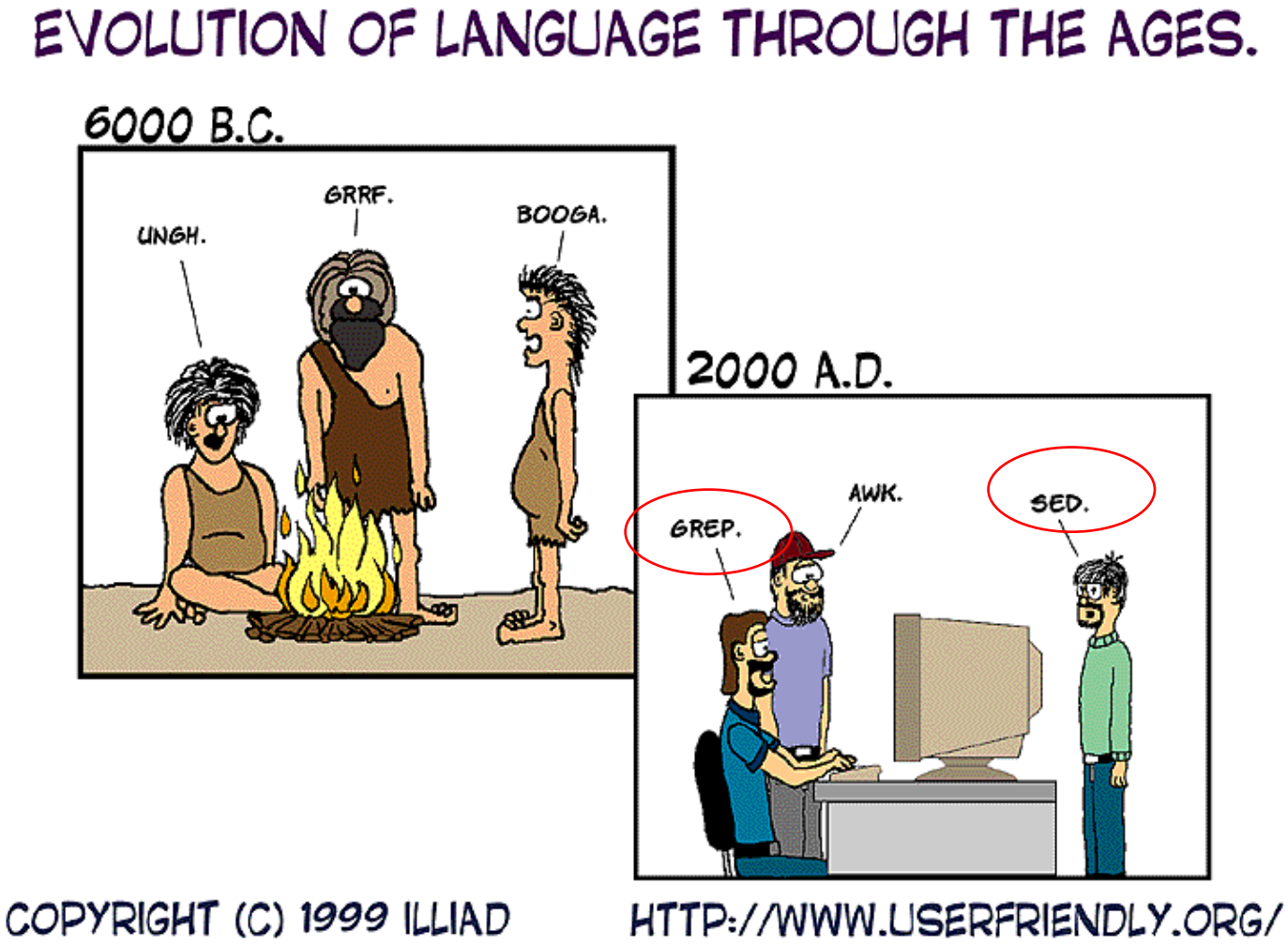
A phone number using convention ###-###-####

[0-9]{3}\-[0-9]{3}\-[0-9]4

Grep/Sed/Awk are three powerful tools to accomplish most common text manipulations in Linux



Grep/Sed/Awk are three powerful tools to accomplish most common text manipulations in Linux



Grep: A highly useful search tool

grep is the simplest of these three commands and differs from awk and sed in that it cannot be used to add, modify, or delete text.

Similar to the **search** function crtl-f that many are familiar with.

It can be used to **print** or **count** the desired text using various flags.

Useful grep flags:

grep -c (count)
grep -v (invert/everything but)
grep -n (show line numbers)
grep -i (ignore case)

How to use:

function -flag term <file>
grep -c ATGC sequence.fasta
Count the number of times "ATGC" appears in my sequence file

How can we use grep to obtain data about a fasta file.

```
>sample1
GGCAGATTCCCCCTAGACCCGCCCGCACCATGGTCAGGCATGCCCCTCCTCATCGCT
GGGCACAGCCCAGAGGGTATAAACAGTGCTGGAGGCTGGCGGGGCAGGCCAGCTGAG
TCCTGAGCAGCAGCCCAGCGCAGCCACCGAGACACCATGAGAGCCCTCACACTCCTC
GCCCTATTGGCCCTGGCCGCACTTTGCATCGCTGGCCAGGCAGGTGAGTGCCCCCAC
CTCCCCTCAGGCCGCATTGCAGTGGGGGCTGAGAGGAGGAAGCACCATGGCCCACCT
CTTCTCACCCCTTTG
>sample2
CCACTGCACTCACCGCACCCGGCCAATTTTTGTGTTTTTTAGTAGAGACTAAATACCA
TATAGTGAACACCTAAGACGGGGGGCCTTGGATCCAGGGCGATTCAGAGGGCCCCGG
TCGGAGCTGTCGGAGATTGAGCGCGCGGTCCCGGGATCTCCGACGAGGCCCTGGA
CCCCCGGGCGGCGAAGCTGCGGCGCGGCGCCCCCTGGAGGCCGCGGGACCCCTGGCC
GGTCCGCGCAGGCGCAGCGGGGTCGCAGGGCGCGGCGGGTTCCAGCGCGGGGATGGC
GCTGTCCGCGGAGGA
```

How many sequences are in my file? grep –c “>” sequence.txt

How many times does a particular sequence appear? grep –c “target” sequence.txt

Interactive Exercise 1: Back to the Penguins!

Useful grep flags:

grep -c (count)
grep -v (invert/everything but)
grep -n (show line numbers)
grep -i (ignore case)

more	head	tail	cat
view a text file one screen full at a time	view the top 15 lines of a file	view the last 15 lines of a file	spit the whole file at once
space-bar: scroll q: quit	-n num controls the number of lines	-n num controls the number of lines	

Use grep and other tools in your toolkit to do the following:

(1)How many **of each** species of penguin are in the file penguins.csv?
(2)How many male penguins are there? (Be careful!)
(3)How many penguins are NOT from Biscoe?

grep –c Adelie penguins.csv
grep –c Gentoo penguins.csv
grep –c Chindstrap penguins.csv

grep –c male penguins.csv
grep –c female penguins.csv
subtract

grep –c Biscoe penguins.csv
All – Biscoe count

Sed: The favorite child

sed can be used for text transformation on a specific term in a file

We are still following the basic formula of:
function -flag term <file>

sed uses a special structure encoded using slashes to denote different parts of the *term* following this formula for search and replace

‘s/pattern/replace/’

There are MANY variations on this function that allow you to do pretty much any manipulation you can think of (print, delete, change case, designate specific lines etc.

Sed: The favorite child

sed can be used for text transformation on a specific term in a file

We are still following the basic formula of:
function -flag term <file>

sed uses a special structure encoded using slashes to denote different parts of the *term* following this formula

‘s/pattern/replace/’

Today we will learn the basics so you are able to tackle new challenges on your own.

‘s/pattern/replace’

“s” indicates you are using the substitution function for editing
”pattern” is the search pattern you want to edit
“replace” is what you want to replace your pattern with

Example:
I want to replace “offive” with “office” in my file.txt
sed ‘s/offive/office’ file.txt

I think I made this mistake multiple times! Can I fix it everywhere?
Add “g” meaning global change
sed ‘s/offive/office/g’ file.txt

WARNING: sed will overwrite your existing file!

Good Data Practices would suggest that we want to keep intermediate files as we manipulate our data. How can we do this?

Whenever I am writing new code I like to create output files along the way so I can view them to make sure my code is doing what I think it is doing.

sed 's/offive/office/g' file.txt > file_edited.txt

Using "> new_file_name" writes a new file with the changes I made

-n ‘;/pattern/p’

sed can also be a useful search and print function

Example: I have a large csv file and I want to make subsets of this file based on a specific term.

-n:

flag that disables automatic printing because we only want to print a part of the file

;

is a special character indicating the end of a function. In this case we aren’t doing anything special so the first segment is blank except the semicolon

Interactive Exercise 2: Cleaning up after messy penguins! **Use: penguin_mess.csv**

(1) Replace all the spaces in the file with underscores.

```
sed 's/ /_/g' penguins_messy.csv > penguins_messy_nospaces.csv
```

(2) Replace all instances of "none" with "NA"

```
sed 's/none/NA/g' penguins_messy.csv > penguins_messy_nospaces.csv
```

(3) Make a new file for each species

```
sed -n '1p;/Adelie/p' penguins.csv > penguins_AdelieONLY.csv
sed -n '1p;/Chinstrap/p' penguins.csv > penguins_ChinstrapONLY.csv
sed -n '1p;/Gentoo/p' penguins.csv > penguins_GentooONLY.csv
```

Tip! Copying headers is often important.

Use
sed -n ‘1p;/pattern/p

This prints the first row in a file then perform the search and print function


Homework