1. verify that the sequence ID column in each of these files is unique – explain your reasoning

Command 1: cut -f2 8\_V-REGION-nt-mutation-statistics\_185\_PC\_130414.txt | wc -l

85613

cut -f2 8\_V-REGION-nt-mutation-statistics\_185\_PC\_130414.txt | uniq -c | wc -l

85613

Explain: command 1 counts the total Id number; command 2 counts the unique Id number. Here, both of the count have the same results. This indicates file 8\_V-REGION-nt-mutation-statistics\_185\_PC\_130414.txt has the unique id number. Run the same commands for file 4\_IMGT-gapped-AA-sequences\_185\_PC\_130414.txt, and also get 85613 id and unique id. File 4- also has unique id.

2. verify that the sequence IDs in the two files are equal and in the same order (hint: man diff) – explain your reasoning

Command.txt > 4\_file.txt

Command2: cut -f2 8\_V-REGION-nt-mutation-statistics\_185\_PC\_130414.txt > 8\_file.txt

Command3: diff 4\_file.txt 8\_file.txt -s

Files 4\_file.txt and 8\_file.txt are identical

Explain: The ids of file-4 and file-8 was exported into 2 separate files: 4\_file.txt and 8\_file.txt. Command 3 compared these two files and output show they are identical.

3. combine the two files (hint: join) into a new file following these rules:

1. only keep rows that have productive sequences

Command1: join -1 2 -2 2 -t \t 4\_IMGT-gapped-AA-sequences\_185\_PC\_130414.txt 8\_V-REGION-nt-mutation-statistics\_185\_PC\_130414.txt | grep ‘^productive’ > join1.txt

Explain: join -1 2 -2 2 join these 2 file based on column 2. ‘\t’ is the delimiter, which keep each column of joined file separated. grep ‘^productive’ find the lines containing ‘productive’

1. output the following fields

Command: cat join1.txt | awk -F'\t' -vOFS='\t' '{ gsub("\\*.\*$", " ", $4) ; gsub("Homsap IGH", " ", $4) ; print }' | awk -F'\t' -vOFS='\t' '{ gsub("\\*.\*$", " ", $5) ; gsub("Homsap IGH", " ", $5) ; print }' | awk -F'\t' -vOFS='\t' '{ gsub("(.\*$", " ", $24) ; print }' | awk -F'\t' -vOFS='\t' '{ gsub("(.\*$", " ", $26) ; print }' > clean\_file1.txt

Explain: awk -F'\t' -vOFS='\t' '{ gsub("\\*.\*$", " ", $5) completed the step ii; awk -F'\t' -vOFS='\t' '{ gsub("(.\*$", " ", $24) complete step iv. Here "(.\*$", " " express find the content in (), and replace it with whitespace.

1. Add a column which is the percentage of mutations (100 \* # mutations / # nucleotides)

Comand: cat clean\_join1.txt | awk ‘{muation\_rate+=100\*$134/$132} END {print muation\_rate /NR}

Explain: I try to run this command, but it looks line there is something run. No results was output.

Reference:

1. Add a column which is the length of the CDR3 (number of amino acids) – hint: see awk, “length”

Answer: I could find out the length by length($24), but I don’t know how to asign it to a new line.

For the last homework, you generated a command to take a list of items and tabulate them into a formatted list of unique items and their counts. Put that into a script, make it executable and then move into a directory that is on your path (like we set up in class). You can test your script on the meds.txt file.

Commond: ./countor meds.txt

Output is like this:

Simvastatin 35

Omeprazole 31

Levothyroxine 27

Metformin 22

Fluticasone 13

Predinisone 9

Lisinopril 7

Alprazolam 5

Albuterol 5

Glipizide 4

Ranitidine 3

Warfarin 2

Humira 2

Stellara 1

Rituxan 1

Explain: The script file is attached, and the output is showed above. It works the same as the comand like.

#!/bin/bash

sort $1 | uniq -c | sed -e 's/^ \*//;s/ /\t/' | \

awk -F$'\t' 'BEGIN {OFS = FS} {print $2, $1}' | \

sed -e 's/:/\t/' | column -t -s$'\t' | sort -nrk2

Next, download immunoglobulin sequence annotation files:

http://repertoire.informatics.uab.edu/info510/exigseq.tar.gz

(note, these will unpack in the current directory).

1. there is a column called “Functionality”. We want to only include rows where the value

in this field is “productive” (caution, some rows have the value “unproductive”)

Command: cat 2\_IMGT-gapped-nt-sequences\_185\_PC\_130414.txt | awk 'NR ==1 || $3 == "productive"'

Explain: Here, I use awk function. NR ==1 makes sure the first row is showed and $3 == "productive" make sure the lines, whose value of column 3 is equal to ‘productive’, are printed.

Reference:

<https://www.shortcutfoo.com/app/dojos/awk/cheatsheet>

1. The v-genes annotations are in a column called “V-GENE and allele”

Command: ~/class3/transposeheader 2\_IMGT-gapped-nt-sequences\_185\_PC\_130414.txt

Output:

1 Sequence number

2 Sequence ID

3 Functionality

4 V-GENE and allele

5 J-GENE and allele

6 D-GENE and allele

7 V-D-J-REGION

8 V-J-REGION

9 V-REGION

10 FR1-IMGT

11 CDR1-IMGT

12 FR2-IMGT

13 CDR2-IMGT

14 FR3-IMGT

15 CDR3-IMGT

16 JUNCTION

17 J-REGION

18 FR4-IMGt

Explain: I used the script file transposeheade to show the column names and its column number. ‘V-GENE and allele’ is located at column 4.

1. In some rows, there are more than one v-gene. Only use the first, and strip the “Homsap IGH” from the beginning and “F” from the end. For example, if the value of this field for a particular row is:

Homsap IGHV1-69\*01 F, or Homsap IGHV1-69\*12 F or Homsap IGHV1-69D\*01 F

then set up a substitution so that you only get the part in red:

V1-69

If you have only one, clean up so that:

Homsap IGHV5-51\*01 F

becomes:

V5-51

A single regular expression substitution should work in both cases. Again, in the case of multiple annotations in one row (like the first example) use the first one.

Command: cat 2\_IMGT-gapped-nt-sequences\_185\_PC\_130414.txt | awk 'NR ==1 || $3 == "productive"' | awk -F'\t' -vOFS='\t' '{ gsub("\\*.\*$", " ", $4) ; gsub("Homsap IGH", " ", $4) ; print }' | cut -f4 | head

Output:

V-GENE and allele

V1-69

V1-18

V1-18

V5-51

V1-18

V1-69

V1-2

V1-2

V1-2

Explain: Here I use the ‘gsub’ function of ‘awk’ to do the replacement. "\\*.\*$" matched all the characters after \*, and replaced it with whites place ; Homsap IGH matched all the characters before V, and replaced it with whites place. $4 indicated that this replacement only happen in the column 4. Here, order to display the result in a easy way, I use | cut -f4 | head to display only the first 10 lines of column 4.

Reference :

<https://unix.stackexchange.com/questions/40767/replacing-one-test-string-with-another-in-particular-column>

<https://www.runoob.com/linux/linux-comm-sed.html>

1. At this point, you should have a list of cleaned up v-gene annotations for the productive sequences. Produce a list of occurrences of these v-genes using your tabulate function.

Command: cat 2\_IMGT-gapped-nt-sequences\_185\_PC\_130414.txt | awk 'NR ==1 || $3 == "productive"' | awk -F'\t' -vOFS='\t' '{ gsub("\\*.\*$", " ", $4) ; gsub("Homsap IGH", " ", $4) ; print }' | cut -f4 | sort | uniq -c | head -n -1

Output:

3286 V1-18

935 V1-2

8 V1-24

1 V1-3

3 V1-45

767 V1-46

46 V1-58

4020 V1-69

98 V1-8

20 V2-5

14 V2-70

520 V3-11

166 V3-13

189 V3-15

87 V3-20

886 V3-21

1 V3-22

6786 V3-23

361 V3-30

2943 V3-33

20 V3-43

41 V3-43D

886 V3-48

1524 V3-49

1172 V3-53

208 V3-64

78 V3-64D

238 V3-66

75 V3-69-1

28243 V3-7

7 V3-71

203 V3-72

81 V3-73

2926 V3-74

900 V3-9

10 V3-NL1

16 V4-28

53 V4-30-2

1 V4-30-4

1 V4-31

997 V4-34

933 V4-38-2

11731 V4-39

2370 V4-4

5623 V4-59

1672 V4-61

826 V5-10-1

965 V5-51

1. 6-1

Explain: | cut -f4 | sort | uniq -c displayed the numbers of each element of column 4. The last line is the header line. head -n -1 removed the last line.

Reference:

<https://stackoverflow.com/questions/4881930/remove-the-last-line-from-a-file-in-bash>

1. Do the same for the file: 2\_IMGT\_gapped-nt-sequences\_185\_SM\_130414.txt (memory B cells)

Command: cat 2\_IMGT-gapped-nt-sequences\_185\_PC\_130414.txt | awk 'NR ==1 || $3 == "productive"' | awk -F'\t' -vOFS='\t' '{ gsub("\\*.\*$", " ", $4) ; gsub("Homsap IGH", " ", $4) ; print }' | awk -F'\t' -vOFS='\t' '{ gsub("\\*.\*$", " ", $5) ; gsub("Homsap IGH", " ", $5) ; print }' | cut -f 4,5,3 | head | column -t

Output:

Functionality V-GENE and allele J-GENE and allele

productive V1-69 J4

productive V1-18 J4

productive V1-18 J2

productive V5-51 J6

productive V1-18 J4

productive V1-69 J4

productive V1-2 J5

productive V1-2 J4

productive V1-2 J3

Explain: Re-run the previous command for the new file. cut -f 4,5,3 | head | column -t displayed the first 10 lines of column 3, 4, 5

1. EXTRA CREDIT: There is a column called “J-GENE and allele”. Clean up this column as well so that, for example

Homsap IGHJ2\*01 F becomes J2

tabulate occurrences of V-J recombinations – for each row (that is productive), find the recombination such as V4-34 :: J4 (I put in that separator “ :: “) and tabulate those.

Command:

cat 2\_IMGT-gapped-nt-sequences\_185\_PC\_130414.txt | awk 'NR ==1 || $3 == "productive"' | awk -F'\t' -vOFS='\t' '{ gsub("\\*.\*$", " ", $4) ; gsub("Homsap IGH", " ", $4) ; print }' | awk -F'\t' -vOFS='\t' '{ gsub("\\*.\*$", " ", $5) ; gsub("Homsap IGH", " ", $5) ; print }' | awk '{print $4"::"$5}' | head

Output:

V1-69::J4

V1-18::J4

V1-18::J2

V5-51::J6

V1-18::J4

V1-69::J4

V1-2::J5

V1-2::J4

V1-2::J3

Explain： awk '{print $4"::"$5}' | displayed the combined column. The "::" in the double citation mark is the delimiter in the new column.

Reference: <https://stackoverflow.com/questions/34775697/concatenate-2-columns-within-a-single-txt-file-and-separate-by-using-awk>