NGS Bioinformatics 2021

Practical assignment MEMORANDUM

Module topic: Introduction to the command line
Contact session title: Session 2 command line
Trainer: Sumir Panji and Amel Ghouila

Session 2 – Practical 1 - Sed

Introduction

This practical will cover the use of the stream editor (sed) tool that is present on most unix operating systems to match and replace patterns

Tools used in this session

A unix terminal, unix and sed as on the virtual machine distributed to course participants.

- Log into VM and open a terminal
- Change directory to: cd course_data/unix/
    - mkdir sed
    - cd sed
    - mkdir sed_practical
    - cd sed_practical
- Copy files to use for the sed practical into the sed_practical working directory:
    - cp ../practical/Notebooks/grep/exercises.fasta .
    - cp ../practical/Notebooks/grep/sequences.fasta .
    - cp ../practical/Notebooks/awk/genes.gff .

Please note

- **Hand-in information** please upload your completed assignment to the Vula assignments tab. Take note of the final hand-in date, which will be indicated on Vula.

1) Inspect the file called exercises.fasta using less. How many sequences with fasta headers are in the exercises.fa file?
    a. grep -c ">" exercises.fasta = 1000 sequences

2) How many sequences are in the sequences. fasta file?
    a. grep -c ">" sequences.fasta = 4 sequences

3) Examine the sequences.fasta file – what do you notice
    a. All the sequence characters are the same case? No
    b. The sequence characters are in a mixture of cases? Yes

4) From examining the sequences.fasta file, we know that each fasta header and sequence sit on a new line, with the fasta headers being on odd numbered lines. Let us try and print out the fasta header lines only using sed:
    a. sed -n 'p;n' sequences.fasta

5) What command would you use to print out only the even number lines with the actual sequence? sed -n 'n;p' sequences.fasta

6) Let us change the character cases to make them upper case using sed
    a. sed 's/[a-z]/\U&/' sequences.fasta

7) Not all the characters were changed to upper case – what would I need to add to the sed command to change everything to upper case?
    a. sed 's/[a-z]/\U&/g' sequences.fasta

8) Let us look at the gene.gff file copied to the sed_practical directory
    a. How many entries / lines are there? wc -l genes.gff - 10

9) Let us convert the pattern chrm to Chromosome in the genes.gff file
    a. sed 's/chr/Chromosome/' genes.gff

10) Let us convert the gene.gff file into a comma separated file:
    a. sed 's/\t/,/' genes.gff

11) The command above only worked for the first tab character, we all the tabs to be separated by a comma – we need to use the global flag (g):
    a. sed 's/\t/,/g' genes.gff

12) From the sequences.fasta file, lets add the organism p_falciprium to the start of the fasta headers:
    a. sed 's/^>/>p_falciprium_/' sequences.fasta

13) For the command above, I did not use a global flag, yet all the fasta sequence headers were modified – why?
    a. Sed reads in a file line by line and makes a change to the first instance of the character on the new line – each fasta header starts on a new line

14) Lets change all the characters in the sequences.fasta file to upper case and add p_falciprium to the start of the fasta headers and then convert the upper case fasta headers to lower case using a series of sed pipes:
    a. sed 's/[a-z]/\u&/g' sequences.fasta | sed 's/^>/>p_falciprium_/g' | sed 's/SEQUENCE/sequence_/g'

15) What would I need to do to the above command to send the output into a new file?
    a. sed 's/[a-z]/\u&/g' sequences.fasta | sed 's/^>/>p_falciprium_/g' | sed 's/SEQUENCE/sequence_/g' > formatted_seqences.fasta

Self practical session – not done via recording

16) Using the genes.gff file, print out only lines 1, 3 to 5, 7 and 9:
    a. sed -n '1p; 3,5p; 7p; 9p' genes.gff

17) From the /home/manager/course_data/unix/practical/Notebooks/awk folder, copy the file exercises.bed into the working folder within which you should be in and inspect the .bed

file: /course_data/unix/sed/sed_practical e.g. cp ../../practical/Notebooks/awk/exercises.bed . Inspect the bed file:

a. less exercises.bed

18) How many lines / rows are in the exercises.bed file?
    a. wc -l exercises.bed = 326 lines

19) The program you need the bed file to use as input does not like the – character between values in the different columns apart from the strand column. For questions 19 to 22, pipe the output to less. Using sed, substitute the all the terms contig- with contig_ in the exercises.bed file using the global flag:
    a. sed 's/contig-/contig_/g' exercises.bed | less

20) Using sed, substitute gene- with Gene_ in the exercises.bed file using the global flag
    a. sed 's/gene-/Gene_/g' exercises.bed | less

21) Using sed, substitute the term repeat with REPEAT in the exercises.bed file using the global flag
    a. sed 's/repeat/REPEAT/g' exercises.bed | less

22) Repeat questions 21, 22, and 23, but without using the global flag substitution flag.
    a. sed 's/contig-/contig_/' exercises.bed | less
    b. sed 's/gene-/Gene_/' exercises.bed | less
    c. sed 's/repeat/REPEAT/' exercises.bed | less

23) Is there a difference between using the global substitution flag Yes / No?
    a. No

24) What would the explanation be?
    a. Sed reads in a file pattern by pattern and makes the first substitution of that pattern in the line, these patterns only appear once in each line

25) Combine sed pipes to substitute contig- with contig_ gene- with gene_ and repeat with REPEAT and pipe the output to less to check if the substitutions are working
    a. sed 's/contig-/contig_/g' exercises.bed | sed 's/gene-/gene_/g' | sed 's/repeat/REPEAT/g'

26) Once you are happy that your sed command is working, send the output to a file called formatted_exercises.bed
    a. sed 's/contig-/contig_/g' exercises.bed | sed 's/gene-/gene_/g' | 's/repeat/REPEAT/' > formatted_exercises.bed

Session 2 – Practical 2 - AWK

Introduction

This practical will cover the use of the AWK tool that is present on most unix operating systems to filter and work with bioinformatics files.

Tools used in this session

A unix terminal, unix and awk as on the virtual machine distributed to course participants.

- In the same directory as course_data/unix make a directory called awk and change into this directory. We will copy the relevant files into here that will be used for the practical and assessment.

- First off, check if awk is installed by typing awk and should get some output about options and usage on the screen.

- Copy the files to be used for the practical into the awk directory you created:
  - cp ../practical/Notebooks/awk/exercises.bed .
  - cp ../practical/Notebooks/awk/genes.gff .

- Have a quick look at the files copied into the awk directory, they should be familiar from the sed practical. They both seem to have a table type of structure with rows and columns.

1) Using awk, let us print out the first column of the genes.gff file:
   a. awk '{print $1}' genes.gff

2) Let us try and print out column 9 of the genes.gff file:
   a. awk '{print $9}' genes.gff

3) The output from column 9 seems to missing the protein annotation for the gene product. Why?

4) Let us change the awk command slightly to take into account a default delimiter, a \t in this case
   a. awk -F "\t" '{print $9}' genes.gff

5) It would be nice to know how many columns are in the dataset genes.gff, we can use the awk NF function:
   a. awk '{print NF}' genes.gff

6) I get differences between 8 to 10 columns which is not correct as a general feature format file should have 9 columns that should be split by tabs (see the url below for an explanation of a gff file: https://www.ensembl.org/info/website/upload/gff.html Let's try again to the get the correct number of fields by splitting on the correct delimiter which is tab-seperated:
   a. awk -F "\t" '{print NF}' genes.gff

7) All the rows give the correct number of nine fields bar line 4 of the file. Let's use sed to print out line 4 of the file:
   a. sed -n '4p' genes.gff
      Seems to be missing an annotation column

8) We want to find out how many unique chromosomes are contained in our gene.gff file using awk and unix:
   a. awk -F"\t" '{print $1}' genes.gff | sort -u

9) We want to extract columns 1, 3, 6 and 9 from the genes.gff file while keeping the formatting:
   a. awk -F "\t" '{print $1, $3, $6, $9}' genes.gff

10) Not exactly the output I wanted as not tab-separated, let us use awk's BEGIN and OFS functions to get the output in tab delimited format:
    a. awk -F "\t" 'BEGIN {OFS="\t"} {print $1,$3,$6,$9}' genes.gff

11) Let us extract all genes that map to chromosome 1 within the genes.gff file
    a. awk -F"\t" '$1=="chr1" {print $0}' genes.gff
       we can also send this output to a file e.g. awk -F"\t" '$1=="chr1" {print $0}' genes.gff > chromosome_1_genes.gff

12) Strange, for question 14 I had to use the OFS separator while for question 15 I did not. Why?
    a. For question 15 awk is matching and printing the line as it is, for question 14 awk is printing out the columns using its default delimiter which is a space.

13) Let us filter the genes.gff file to get all entries with chromosome 1 and annotations as genes using the && operator:
    a. awk -F"\t" '$1=="chr1" && $3=="gene"' genes.gff

14) We can print out a specific column using the filtering criteria above e.g. column 9:
    a. awk -F"\t" '$1=="chr1" && $3=="gene" {print $9}' genes.gff

15) Let us pull out all the rows where the column is equal to chromosome 1, or the column 3 is equal to a gene using the || operator:
    a. awk -F"\t" '$1=="chr1" || $3=="gene"' genes.gff

16) Let's modify the previous awk construct to also filter on numerical values:
    a. awk -F"\t" '$1=="chr1" && $3=="gene" && $4 < 1100' genes.gff

17) Let us change the values in the source field to H_sapiens and print out a new gff file using awk:
    a. awk -F"\t" 'BEGIN {OFS="\t"} {$2="H_sapiens"; print $0}' genes.gff

18) Using awk, let us get the length of repeats from genes.gff file keeping in mind the offset:
    a. awk -F"\t" '$3=="repeat" {print $5 - $4 + 1}' genes.gff

19) If using a .bed file, then we would not use the +1 offset:
    a. awk -F"\t" '$4=="repeat" {print $3 - $2}' exercises.bed

20) Let us get the total length of repeats from the genes.gff file:
    a. awk -F"\t" 'BEGIN{sum=0} $3=="repeat" {sum =+ $5 - $4 + 1} END{print sum}' genes.gff

21) Let us calculate the mean of the scores in the gene.gff file using a count as well:
    a. awk -F"\t" 'BEGIN{sum=0; count=0} $3=="gene" {sum =+ $6; count++} END{print sum/count}' genes.gff

Self practical session – not recorded

22) Extract all genes that map to chromosome 2 within the genes.gff file:
    a. awk -F"\t" '$1=="chr2" {print $0}' genes.gff

23) Print out columns 1, 2, 4 and 8 using awk and insert a tab delimiter:

a. awk -F "\t" 'BEGIN {OFS="\t"} {print $1,$2,$4,$8}' genes.gff

24) Sum up the total length of genes in the genes.gff file:
    a. awk -F"\t" 'BEGIN{sum=0} $3=="gene" {sum =+ $5 - $4 + 1} END{print sum}' genes.gff (334)

25) Calculate the mean gene length in the genes.gff file using awk:
    a. awk -F"\t" 'BEGIN{sum=0; count=0} $3=="gene" {sum =+ $5 -$4 +1; count++} END{print sum/count}' genes.gff (55.6667)

26) Find all the entries labelled as gene in column 3 and have a score greater then 0.55 in column 6:
    a. awk -F"\t" '$3=="gene" && $6 > 0.55' genes.gff

Session 2 – Practical 3 – bash and command line tools

Introduction

This practical will cover the use of the bash shell and running some of the bioinformatics programs installed in the VM that will be used later in the course.

Tools used in this session

A unix terminal, a text editor, files and programs installed on the virtual machine distributed to course participants.

1) Let's try and run some bioinformatics programs, type: vcftools. What version of vcftools is installed?
    a. 0.1.16

2) Type: bedtools, what version of bedtools is installed?
    a. v2.30.0

3) Type: bcftools annotate myfile. What error message is one getting (copy and paste it below)?
    a. [E::hts_open_format] Failed to open file "myfile" : No such file or directory
    Failed to read from myfile: No such file or directory

4) What does this error message mean?
    a. The program cannot find myfile as it does not exist

5) Let's look at the bash scripts that are present in the directory using the less command, copy and paste the output of the script below:
    a. less hello.sh
    b. echo Hello World!

6) Let's try and run the hello.sh script, what is the correct command to run the hello.sh script?
    a. hello.sh – command not found
    b. bash hello.sh
    c. ./hello.sh

7) Let's add the scripts directory temporarily to our $PATH variable so I can run the script anywhere else within this terminal session. Type echo $PATH and copy and paste the output below:

/home/manager/miniconda/bin:/home/manager/miniconda/condabin:/home/manager/salmon/bin:/home/manager/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

8) To add the path to the scripts directory, let's first get the current path to the scripts directory, type pwd and copy and paste the output below:
    a. /home/manager/course_data/unix/practical/Notebooks/advanced_bash/scripts

9) Let us add the full path of the directory to our $PATH variable: "export PATH=$PATH:/home/manager/course_data/unix/practical/Notebooks/advanced_bash/scripts". Type echo $PATH and copy and paste the output below:

    a. /home/manager/miniconda/bin:/home/manager/miniconda/condabin:/home/manager/salmon/bin:/home/manager/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin**:/home/manager/course_data/unix/practical/Notebooks/advanced_bash/scripts**

10) Let's try and run the hello.sh script again by typing hello.sh. What is the output (copy and paste it below):
    a. Hello World!

11) Let's move up two directories and see if we can still run the hello.sh script: cd ../../ . What is the output (copy and paste it below):
    a. Hello World!

12) Let's try running another script in the directory called options_example.2.sh and copy and paste the output below:
    a. filename is:

    First lines of file are:
    head: option requires an argument -- 'n'
    Try 'head --help' for more information. options_example.2.sh

13) Try running the options_example.3.sh with the number 3 after the script without an input file to see what errors one gets. Copy and paste the output below:
    a. usage: options_example.3.sh filename number_of_lines

    Prints the filename, and the given first number of lines of the file

14) Move back into the advanced_bash/scripts directory and try running the options_example.3.sh script with test_file and the number 2 after that. Copy and paste the output below:
    a. filename is: test_file

    First 2 lines of file test_file are:
    test file line 1

test file line 2

15) Let's quickly examine the files in the loop_files directory: using ls and cat e.g cat loop_files/file.* How many files are there in the directory?

5

16) Let's run a for loop on the files in the loop_file directory. What does the second for loop do?

for variable in loop_files/*; do wc -l $variable; done
for filename in loop_files/*; do sed 's/file/FILE/g' $filename; done

a. It matches the term file in each of the files in the loop_directory and converts them to uppercase

17) We can make permanent changes to the file using a sed option (not recommended unless you use the backup flag as well). Do you see any new files in loop_directory? What extension do they have?

cat cat loop_files/file.*
for filename in loop_files/*; do sed -i.bak 's/file/FILE/g' $filename; done
cat cat loop_files/file.*

a. .bak

For the remainder of the practical, go through the different bash scripts to understand and test the different lines of code added in the different versions by commenting them out using a text editor and a # at the beginning of the line (don't forget to save the file before running the various scripts).