H3ABioNet
Pan African Bioinformatics Network for H3Africa

wellcome connecting science

Next Generation Sequencing Bioinformatics Course 2021

Introduction to Linux

Session 2 – Part 2 – AWK

WELLCOME GENOME CAMPUS
CONNECTING SCIENCE
ADVANCED COURSES+ SCIENTIFIC CONFERENCES

AWK

- Scripting language with text processing capabilities for data extraction, comparison, transformation
- Similar to sed, AWK is available on most unix operating systems
- Named after the initials of its inventors Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan
- Used when one wants to extract fields, make comparisons, filter data and general data wrangling

Some features of AWK

- AWK is great as it allows one to work with delimited data
- Similar to sed, it reads in files line by line
- Different to sed, it splits the line into fields – allows for columns
- A lot of data formats in bioinformatics are delimited with a tab (\t) being a common field separator
- AWK has inbuilt functions that allow one to manipulate these fields – unlike sed i.e. allows one to work with columns within a dataset

Basic AWK syntax

- awk - options ‘optional_selection_criteria { action} ’ input_file

- E.g. awk –F “\t” '{ print $1 }' genes.gff

chr1
chr1
chr1
chr2
chr2
chr3
chr4
chr10
chr10
chrX

- The –F flag indicates the field delimiter – in this case a tab

Basic AWK syntax

- Similar to sed, awk prints the output to the screen, if you want to save the output then will need to redirect it to an outfile
- Different to sed – awk has inbuilt variables called $1, $2, $3 …. that map to the fields separated by the \t delimiter when specified
- Usually useful to determine the number of fields a file has first
- E.g. awk '{print NF}' genes.gff
- Number of Fields (NF) is an inbuilt awk variable that is defined each time awk reads in a line

Basic AWK syntax

- E.g. awk '{print NF}' genes.gff

9
10
9
8
10
9
9
9
9
9

- Strange that they are 2 records in line 2 and 5 that have 10 fields, one in line 4 that has 8 fields and the rest have 9 – any thoughts as to why?

Basic AWK syntax

- Let’s look at the records with other records to compare with by using sed to print the first 6 lines of the file:

- sed -n '1,6p' genes.gff

chr1	source1	gene	100	300	0.5	+	0	name=gene1;product=unknown
chr1	source2	gene	1000	1100	0.9	-	0	name=recA;product=RecA protein
chr1	source5	repeat	10000	14000	1	+	.	name=ALU
chr2	source2	gene	10000	1200	0.95	+	0	
chr2	source1	gene	50	900	0.4	-	0	name=gene2;product=gene2 protein
chr3	source1	gene	200	210	0.8	.	0	name=gene3

- Looks like there is a space in fields 2 and 5 between the product name and protein e.g. RecA protein
- The annotation column for record 4 is empty

Basic AWK syntax

- Looks like there is a space in fields 2 and 5 between the product name and protein
- The annotation column for record 4 is empty
- The file is tab separated, in the previous construct we did not tell awk to split the file according to a delimiter: awk '{print NF}' genes.gff
- E.g. awk –F “\t” '{print NF}' genes.gff

9
9
9
8
9
9
9
9
9
9

AWK usage

- awk - options ‘optional_selection_criteria { action} ’ input_file

- Let’s use the optional_selection_criteria to do some filtering on the genes.gff file

- awk -F "\t" '$1 > "chr1" {print $0}' genes.gff

```
chr2    source2  gene     10000  1200  0.95  +  0
chr2    source1  gene     50     900   0.4   -  0
        name=gene2;product=gene2 protein
chr3    source1  gene     200    210   0.8   .  0  name=gene3
chr4    source3  repeat   300    400   1     +  .  name=ALU
chr10   source2  repeat   60     70    0.78  +  .  name=LINE1
chr10   source2  repeat   150    166   0.84  +  .  name=LINE2
chrX    source1  gene     123    456   0.6   +  0
        name=gene4;product=unknown
```

AWK usage

- awk -F "\t" '$1 > "chr1" {print $0}' genes.gff
- awk recognizes mathematical operators such as the greater than sign
- The construct above does two things:
  - Optional_selection_criteria is to use field 1 of the line being read in and check if it is greater then chr1
  - As awk reads in the file line by line, it will print the line ($0) only when the condition is met
- Useful for extracting lines based on a field from a file e.g. all entries for chromosome 2 only
- awk -F "\t" ‘$1 == "chr2" {print $0}' genes.gff

AWK usage

- Also a great way to extract fields from a file and put the input into a new one

- E.g. awk -F "\t" '{print $1,$3,$7}' genes.gff

```
chr1 gene +
chr1 gene -
chr1 repeat +
chr2 gene +
chr2 gene -
chr3 gene .
chr4 repeat +
chr10 repeat +
chr10 repeat +
chrX gene +
```

- Printed out the columns I wanted, and I can send the output to a new file

- Problem - the output does not seem to be tab delimited?

AWK usage

- Problem - the output does not seem to be tab delimited
- To get to the output in \t format, need to change awk's default behaviour – can use the Output Field Separator (OFS)
- E.g. awk -F "\t" 'BEGIN {OFS="\t"} {print $1,$3,$7}' genes.gff

```
chr1    gene    +
chr1    gene    -
chr1    repeat  +
chr2    gene    +
chr2    gene    -
chr3    gene    .
chr4    repeat  +
chr10   repeat  +
chr10   repeat  +
chrX    gene    +
```

AWK usage

- E.g. awk -F "\t" 'BEGIN {OFS="\t"} {print $1,$3,$7}' genes.gff
- BEGIN is an awk variable that tells awk to execute the action in the first set of {} once the first line is read in
- In this case, to set the Output Field Separator variable to be a \t
- awk can also be used to replace every value in a specified field
- E.g. awk -F"\t" 'BEGIN {OFS="\t"} {$2=”H_sapiens"; print $0}' genes.gff

```
chr1    H_sapiens  gene    100    300    0.5    +    0
        name=gene1;product=unknown
chr1    H_sapiens  gene    1000   1100   0.9    -    0
        name=recA;product=RecA protein
chr1    H_sapiens  repeat  10000  14000  1      +    .        name=ALU
chr2    H_sapiens  gene    10000  1200   0.95   +    0
```

AWK usage

- Can combine multiple patterns using the && to mean do if meets criteria 1 “and” criteria 2

- E.g. awk -F"\t" '$1=="chr1" && $3=="gene"' genes.gff

```
chr1    source1    gene    100    300    0.5    +    0
        name=gene1;product=unknown
chr1    source2    gene    1000   1100   0.9    -    0
        name=recA;product=RecA protein
```

- Can also meet criteria 1 “and” criteria 2 “and” criteria 3
- E.g. awk -F"\t" '$1=="chr1" && $3=="gene" && $7=="+"' genes.gff

```
chr1    source1    gene    100    300    0.5    +    0
        name=gene1;product=unknown
```

AWK usage

- E.g. awk -F"\t" '$1=="chr1" && $3=="gene"' genes.gff

```
chr1    source1    gene    100    300    0.5    +    0
        name=gene1;product=unknown
chr1    source2    gene    1000   1100   0.9    -    0
        name=recA;product=RecA protein
```

- Can use the || as an “or” condition to mean do if meets criteria 1 “or” criteria 2

- E.g. awk -F"\t" '$1=="chr1" || $3=="gene"' genes.gff

```
chr1    source1    gene    100    300    0.5    +    0
        name=gene1;product=unknown
chr1    source2    gene    1000   1100   0.9    -    0
        name=recA;product=RecA protein
chr1    source5    repeat  10000  14000  1      +    .    name=ALU
chr2    source2    gene    10000  1200   0.95   +    0
chr2    source1    gene    50     900    0.4    -    0
        name=gene2;product=gene2 protein
chr3    source1    gene    200    210    0.8    .    0    name=gene3
chrX    source1    gene    123    456    0.6    +    0
        name=gene4;product=unknown
```

AWK usage

- One can combine multiple conditions, and filter based on numerical values instead of just strings as we have done

- E.g. awk -F"\t" '$1=="chr1" && $3=="gene" && $4 < 1100' genes.gff

```
chr1    source1    gene    100    300    0.5    +    0
        name=gene1;product=unknown
chr1    source2    gene    1000   1100   0.9    -    0
        name=recA;product=RecA protein
```

AWK basic arithmetic

- As awk recognizes mathematical operators, can use it to preform basic calculations based on some criteria

- E.g. to find the length of repeats in the genes.gff file - awk -F"\t" '$3=="repeat" {print $5 - $4 + 1}' genes.gff

4001

101

11

17

AWK basic arithmetic

- E.g. to find the length of repeats in the genes.gff file - awk -F"\t" '$3=="repeat" {print $5 - $4 **+ 1**}' genes.gff

- The +1 addition is due to the General Feature Format where the sequence numbering starts at 1

(https://www.ensembl.org/info/website/upload/gff.html)

- Different to the BED file format where the sequence numbering starts at 0

(https://m.ensembl.org/info/website/upload/bed.html)

AWK basic arithmetic

- Can use awk to add up the total length of the repeats by using a variable
- E.g. awk -F"\t" 'BEGIN{sum=0} $3=="repeat" {sum = sum + $5 - $4 + 1} END{print sum}' genes.gff → 4130
- A variable called “sum” is set at zero before awk reads in the file
- Each time the line repeat is found, the calculated length of the repeat is added to variable sum
- The END statement tells awk what to do once all the lines in the file have been read – in this instance to print the final value of the variable sum
- Can also use awk’s += operator as a counter e.g. awk -F"\t" 'BEGIN{sum=0} $3=="repeat" {sum =+ $5 - $4 + 1} END{print sum}' genes.gff → 4130

AWK basic arithmetic

- Can use awk to calculate the mean scores of the genes in column 6 of the genes.gff file

- E.g. awk -F"\t" 'BEGIN{sum=0; count=0} $3=="gene" {sum =+ $6; count++} END{print sum/count}' genes.gff → 0.1

- We use a second variable called count is set to zero and adds 1 each time the term gene is matched – this keeps track of the number of matches to gene

- The END statement tells awk divide the total value of sum (0.6) by the number of matches to gene (6) = 0.1

More info and examples on using awk (syntaxes / usage might differ)

- /home/manager/course_data/unix/practical -> unix.pdf
- https://www.tutorialspoint.com/awk/index.htm
- https://bioinformatics.cvr.ac.uk/category/awk/
- https://linuxhint.com/category/awk/
- https://www.shortcutfoo.com/app/dojos/awk/cheatsheet