UNIVERSITY OF GEORGIA
College of Agricultural & Environmental Sciences
Animal Breeding and Genetics Group

Unix commands for data editing

Daniela Lourenco

Ignacio Aguilar

BLUPF90 TEAM – 05/2024

Huge volume of data

- Example: genomic info - 50k v2 (54609 SNP)
  - For 104 individuals
  - Illumina final report file:
    - 5,679,346 records
    - 302 MB

- Not efficient way to read/edit with regular editors (vi, vim, gedit…)

Popular commands

`head file`	prints first 10 lines
`head -20 file`	prints first 20 lines
`tail file`	prints last 10 lines
`less file`	lists *file* line-by-line or page-by-page
`wc -l file`	counts the number of lines
`grep text file`	finds lines that contains text
`cat file1 file2`	concatenates files
`sort`	sorts a file
`cut`	cuts specific columns
`join`	joins lines of two files on specific columns
`paste`	pastes lines of two files
`expand`	replaces TAB with spaces
`uniq`	retains unique lines on a sorted file

head / tail

```
$ head pedigree.txt

UGA42011 UGA41101 UGA34199
UGA42012 UGA41101 UGA38407
UGA42013 UGA41101 UGA39798
UGA42014 UGA41101 UGA37367
UGA42015 UGA41101 UGA40507
UGA42016 UGA41101 UGA34449
UGA42017 UGA41101 UGA37465
UGA42018 UGA41101 UGA40205
UGA42019 UGA41101 UGA37513
UGA42020 UGA41101 UGA34836
```

```
$ head -20 pedigree.txt

$ tail pedigree.txt
```

less

- Allows to view the content of a file and move forward and backward
- For files with long lines use option `-S` (disable line wrapping)

```
$ less -S genotypes.txt
```

```
UGA42014    2100212121111121110010011112110200121022001212010110112111102112112101011011102110112121001120011010110200011111222 1
UGA42019    2010112021121120101111120121000110121112100212020211021211212111122202100001101020000111022111110011011111101101211 2111
UGA42029    1011102011211110100001121121100101111022001211111210011121201212111201101110210011100012111102120012010211111011122 2121
UGA42039    1000201012210220101011120011010201111021110112021111121211211221012111101111211111101101011001121121000211200001221 1220
UGA42047    2000202022220220000000020022000200220022000222020220022220202222022202000000200020000002022002220022000200200002222 2220
UGA42051    2000202022220220000000020022000200220022000222020220022220202222022202000000200020000002022002220022000200200011211 1120
UGA42052    2000202022220220000000020022000200220022000222020220022220202222022202000000200020000002022002220022000200200002222 2110
UGA42056    1021011100212100210001111221200111011011001210211200001111110202101210211110121101200011110202020001021221122010022 2112
UGA42057    2000202022220220000000020022000200220022000222020220022220202222022202000000200020000002022002220022000200200002222 2110
UGA42061    1000201012210220101011120011010201111021110112021111121211211221012111101111211111101101011001121121000211200001221 1220
UGA42085    1010112011210221010100021122011111121122101212011211112211202212022102010011200121001102121102111012001100201101121 1121
UGA42088    0011101001201110201012221110110102002021111101112101110112210211101110202221221102201111100101021111010222110101211 1121
UGA42094    0112111011111101100002212221210101112121102100112101110022110111210200102111211012111012200102120101020122021111112 2111
UGA42095    2000202022220220000000020022000200220022000222020220022220202222022202000000200020000002022002220022000200200011211 1120
UGA42098    1000201012210220101011120011010201111021110112021111121211211221012111101111211111101101011001121121000211200001221 1220
UGA42101    0121012010111101200102121210200112112111011110112110000112200112100201201110111102200011111102110101020112021011021 2012
UGA42108    2000202022220220000000020022000200220022000222020220022220202222022202000000200020000002022002220022000200200002222 2220
UGA42109    1011102011211110100001121121100101111022001211111210011121201212111201101110210011100012111102120012010211111020111 1021
UGA42127    2000202022220220000000020022000200220022000222020220022220202222022202000000200020000002022002220022000200200002222 2220
UGA42136    1000201012210220101011120011010201111021110112021111121211211221012111101111211111101101011001121121000211200001221 1220
UGA42137    1021011100212100210001111221200111011011001210211200001111110202101210211110121101200011110202020001021221122010022 2112
UGA42138    0121012010111101200102121210200112112111011110112110000112200112100201201110111102200011111102110101020112021011021 2012
UGA42139    1000201012210220101011120011010201111021110112021111121211211221012111101111211111101101011001121121000211200001221 1220
UGA42140    1011102011211110100001121121100101111022001211111210011121201212111201101110210011100012111102120012010211111020111 1021
```

Counting lines/characters inside files

- Command **`wc`** counts the number of lines/words/bytes

```
$ wc genotypes.txt
     2024     4048 91108336 genotypes.txt
```

- Number of lines of a file(s)

```
$ wc -l genotypes.txt pedigree.txt
    2024 genotypes.txt
   10000 pedigree.txt
   12024 total
```

Concatenating files

Put content of file1 and file2 in output_file

```
$ cat file1 file2  > output_file
```

```
==> file1 <==
1
2
3

==> file2 <==
a
b
c

==> output_file <==
1
2
3
a
b
c
```

Add content of file3 to output_file using **>>** redirection

Append content at the end of the file

```
$ cat file3 >> output_file
```

```
==> file3 <==
x
y
z

==> output_file <==
1
2
3
a
b
c
x
y
z
```

expand / paste

`expand` replaces TAB with spaces

`paste` merges files line by line with a TAB delimiter

`paste -d " "` merges files line by line with a space delimiter

```
$ head file1 file2
```

```
==> file1 <==
1
2
3

==> file2 <==
a
b
c
```

```
$ paste file1 file 2 | head
```

```
1   a
2   b
3   c
```

```
$ paste -d " " file1 file 2 | head
```

```
1 a
2 b
3 c
```

sort

- Sorts a file in alphanumeric order
  - specifying which column should be sorted

```
$ sort -k 2,2 file4  > a  or sort +1 -2 file4 > a
$ sort -k 1,1 file4  > b  or sort +0 -1 file4 > b
```

- Sorts a file in numeric order

```
$ sort -nk 2,2 file4  > a  or sort -n +1 -2 file4 > a
$ sort -nk 1,1 file4  > b  or sort -n +0 -1 file4 > b
```

- Sorts a file in reverse numeric order

```
$ sort -nrk 2,2 file4  > a  or sort -nr +1 -2 file4 > a
```

- Sorts based on column 1 then column 2

```
$ sort -k1,1 -k2,2 file4 > ab
```

join

- Merges two files by column 1 in both (they should be sorted)

```
$ join -1 1 -2 1 phenotypes.txt pedigree.txt > new_file
```

- Merges two files by column 1 in both (sorting at the same time)

```
$ join -1 1 -2 1  <(sort -k1,1 phenotypes.txt) <(sort –k1,1 pedigree.txt) > new_file
```

- Merges two files by column 1 but suppresses the joined output lines

```
$ join –v1 phenotypes.txt pedigree.txt > new_file
```

grep

- grep finds patterns within a file and lists all lines that match the pattern

```
$ grep UGA42014 pedigree.txt
```

```
UGA42014 UGA41101 UGA37367
UGA44728 UGA43767 UGA42014
UGA47337 UGA44642 UGA42014
UGA48153 UGA44876 UGA42014
UGA50182 UGA48658 UGA42014
```

- grep -v shows all lines that do not match the pattern

```
$ grep -v UGA pedigree.txt
```

- Pattern with spaces use -e

```
$ grep -e "UGA42014 UGA41101 UGA37367" pedigree.txt
```

sed

- Sed is a stream editor -> it reads input file and applies commands that match the pattern

- Substitution (s) of a pattern globally (g)

```
$ sed ‘s/pattern1/new pattern/g’ file > newfile
$ sed ‘s:pattern1:new pattern:g’ file > newfile
$ sed ‘s:UGA:DL:g’ pedigree.txt > dl.temp
```

- Substitution of a pattern in the same file

```
$ sed -i ‘s/pattern1/new pattern/g’ file
```

- Substitution of a pattern in a specific line (e.g., line 24)

```
$ sed ‘24s/pattern1/new pattern/’ file > newfile
```

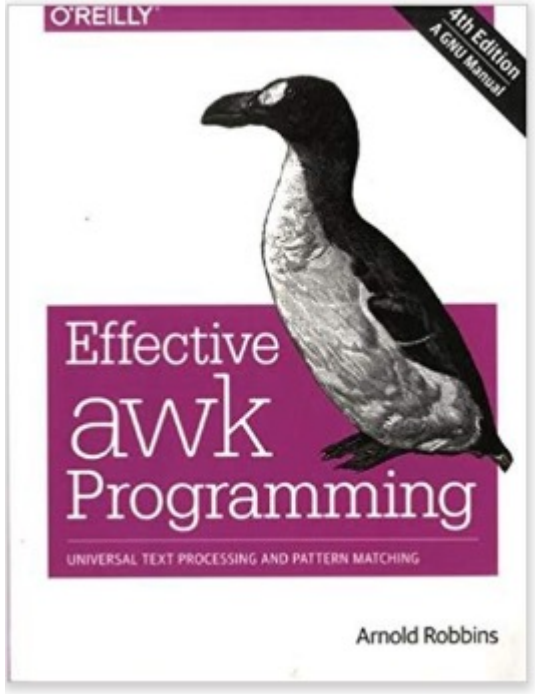
- Deletes lines that contain “pattern to match”

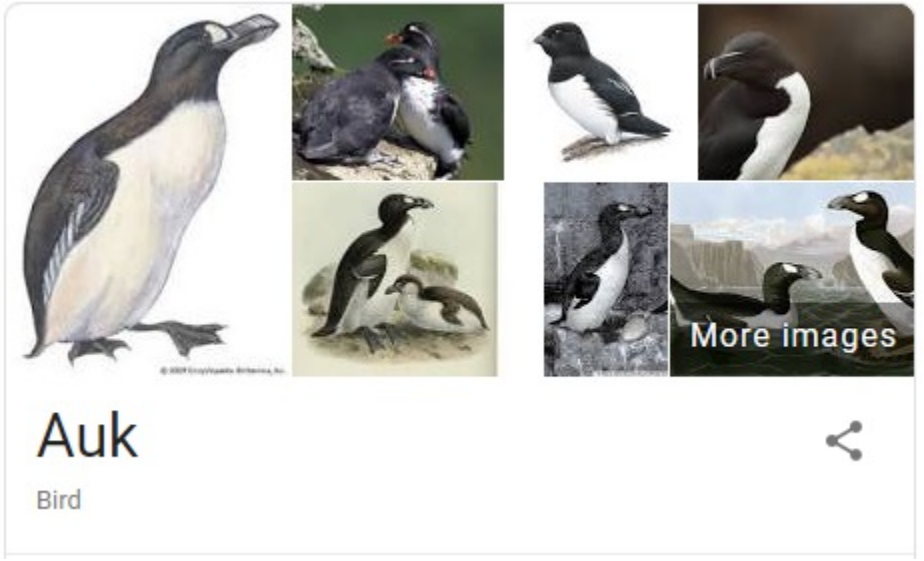
```
$ sed '/pattern to match/d' file
```

awk

AWK is a language for text processing and typically used as a data extraction and reporting tool

Alfred **A**ho
Peter **W**einberger
Brian **K**ernighan





awk

- Interpreted program language, that process data stream of a file line by line
- Very useful and fast command to work with text files
- Can be used as a database query program
  - Selects specific columns or creates new ones
  - Selects specific rows matching some criteria
- Can be used with **if/else** and **for** structures

awk

Implicit variables
NF - number of fields
NR - record number
FS - input field separator
OFS - output field separator

- Print column 1, and last of pedigree file

```
$ awk '{print $1,$NF}' pedigree.txt > anim_dam.temp
```

- Print all columns:

```
$ awk '{print $0}' phenotypes.txt > all_phen.temp
```

- Print column 1 based on occurrence in column 2:

```
$ awk '{if ($2==2) print $1}' phenotypes.txt > fem.temp
```

- Print columns 3 and 4 skipping the first 1000 lines:

```
$ awk '{if (NR>1000) print $3,$4}' phenotypes.txt > part.temp
```

- Print length of column 2 from line 1:

```
$ awk '{if (NR==1) print length($2)}' genotypes.txt
```

- Concatenate effects 2 and 5 and add the new effect to the phenotype file:

```
$ awk '{print $0,$2$5}’ phenotypes.txt > new_phen.txt
```

- Process CSV files
- `$ awk 'BEGIN {FS=","} {print $2,$3}' pedigree.txt > ped_out.temp`

awk hash tables

- Arrays can be indexed by alphanumeric variables in an efficient way

- awk version to count progeny by sire
  - sire id is column 2

```
$ awk '{ sire[$2]+=1} END { for (i in sire)
      {print "Sire " i, sire[i]}}' pedigree.txt
```

```
Sire UGA45217 400
Sire UGA43767 400
Sire UGA38476 200
Sire UGA41101 400
Sire UGA48548 200
Sire UGA45825 400
Sire UGA44642 400
Sire UGA45179 400
```

awk

- awk can be used for pretty much anything related to data processing in Unix

- Sum of elements in column 1

```
$ awk '{ sumf += $1 } END { print sumf}' file1
```

6

- Sum of squares of element in column 1

```
$ awk '{ sumf += $1*$1 } END { print sumf}' file1
```

14

- Average of elements in column 1

```
$ awk '{ sumf += $1 } END { print sumf/NR}' file1
```

2

```
1
2
3
```

uniq

- Command **uniq** lists all unique lines of a file
- Option –c counts the number of times each level occurs in a file

Example: counting progeny by sire in a pedigree file

```
$ awk '$2>0{ print $2}' ped  | sort | uniq –c > s.temp
```

```
$ awk `{ if ($2>0) print $2}' ped  | sort | uniq –c > s.temp
```

cut

- cuts out sections from each line of a file and writes the result to standard output

- Cut the first 3 characters of a line

```
$ cut -c1-3 pedigree.txt > code.txt
```

- Cut the second column of a line

```
$ cut -d " " -f 2 pedigree.txt > code.txt
```

Run in background + Save output

```
$vi blup.sh
#type the following commands inside ai.sh
       #!/bin/bash
       blupf90+ <<AA > blup.log
       renf90.par
       AA
#save and exit
$bash blup.sh &
#can replace bash by sh
```

```
$vi gibbs.sh
#type the following commands inside ai.sh
       #!/bin/bash
       gibbsf90+ <<AA > gibbs.log
       renf90.par
       1000 0
       10
       AA
#save and exit
$bash gibbs.sh &  #can replace bash by sh
```