



UNIVERSITY OF  
**GEORGIA**  
College of Agricultural &  
Environmental Sciences

# Unix commands for data editing

Daniela Lourenco

BLUPF90 TEAM, 02/2023

# Hands on...getting some data

```
cp -r /home/guest002/course/labs/lab1linux_une .
```

```
curl http://nce.ads.uga.edu/wiki/lib/exe/fetch.php?media=lab1Linux_une.zip -o  
lab1linux.zip
```

# 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

`less -S file`

lists *file* line-by-line or page-by-page without wrapping the text

`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
```

# Genomics - huge volume of information

- Example 50kv2 (54609 SNP)
  - For 104 individuals
  - Illumina final report file:
    - 5,679,346 records
    - 302 MB
- Not efficient to read/edit with regular editors (vi, vim, gedit...)

# less command

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

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

```
UGA42014 21002121211111211100100111121102001210220012120101101121111021121121010110111010211101121210011200110101102000111112221
UGA42019 201011202112112010111112012100011012111210021202021102121121211112220210000110102000011022111110011011111011012112111
UGA42029 10111020112111101000011211211001011110220012111112100111212012121112011011102100111000121111021200120102111110111222121
UGA42039 1000201012210220101011120011010201111021110112021111121211211221012111101111211111011010110011211210002112000012211220
UGA42047 200020202222022000000002002200020022002200220022202022002222020222202220220200000200020000002022002220022000200200002222220
UGA42051 2000202022220220000000020022000200220022002200222020220022220202222022202202000002000200000020220022200220002002000112111120
UGA42052 2000202022220220000000020022000200220022002200222020220022220202222022202202000002000200000020220022200220002002000022222110
UGA42056 10210111002121002100011112212001110110110012102112000011111102021012102111101211012000111102020200010212211220100222112
UGA42057 2000202022220220000000020022000200220022002200222020220022220202222022202202000002000200000020220022200220002002000022222110
UGA42061 1000201012210220101011120011010201111021110112021111121211211221012111101111211111011011010110011211210002112000012211220
UGA42085 10101120112102210101000211220111111211221012120112111122112022120221020100112001210011021211021110120011002011011211121
UGA42088 001110100120111020101222111011010200202111110111210110112210211101110202221221102201111001010211110102221110101211121
UGA42094 011211101111101100002212221210101112121102100112101110022110111210200102111211012111012200102120101020122021111122111
UGA42095 2000202022220220000000020022000200220022002200222020220022220202222022202202000002000200000020220022200220002002000112111120
UGA42098 1000201012210220101011120011010201111021110112021111121211211221012111101111211111011010110011211210002112000012211220
UGA42101 01210120101111012001021212102001121121110111101121100001122001121002012011101111022000111111021101010201120210110212012
UGA42108 200020202222022000000002002200020022002200220022202022002222020222202220220200000200020000002022002220022000200200002222220
UGA42109 10111020112111101000011211211001011110220012111112100111212012121112011011102100111000121111021200120102111110201111021
UGA42127 200020202222022000000002002200020022002200220022202022002222020222202220220200000200020000002022002220022000200200002222220
UGA42136 1000201012210220101011120011010201111021110112021111121211211221012111101111211111011010110011211210002112000012211220
UGA42137 10210111002121002100011112212001110110110012102112000011111102021012102111101211012000111102020200010212211220100222112
UGA42138 01210120101111012001021212102001121121110111101121100001122001121002012011101111022000111111021101010201120210110212012
UGA42139 10002010122102201010111200110102011110211101120211111212112112210121111011112111111011010110011211210002112000012211220
UGA42140 10111020112111101000011211211001011110220012111112100111212012121112011011102100111000121111021200120102111110201111021
```

# 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
```



# paste / expand

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

`expand` replaces TAB with spaces

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

```
head file1 file2
```

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

```
==> file1 <==
```

```
1  
2  
3
```

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

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

```
==> file2 <==
```

```
a  
b  
c
```

```
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
```

OR

```
join -1 1 -2 1 <(sort +0 -1 phenotypes.txt) <(sort +0 -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 "pattern with spaces" file1
```

# sed

- Sed is a stream editor

It reads input file and apply commands that match the pattern

- Substitution of a pattern

```
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
```

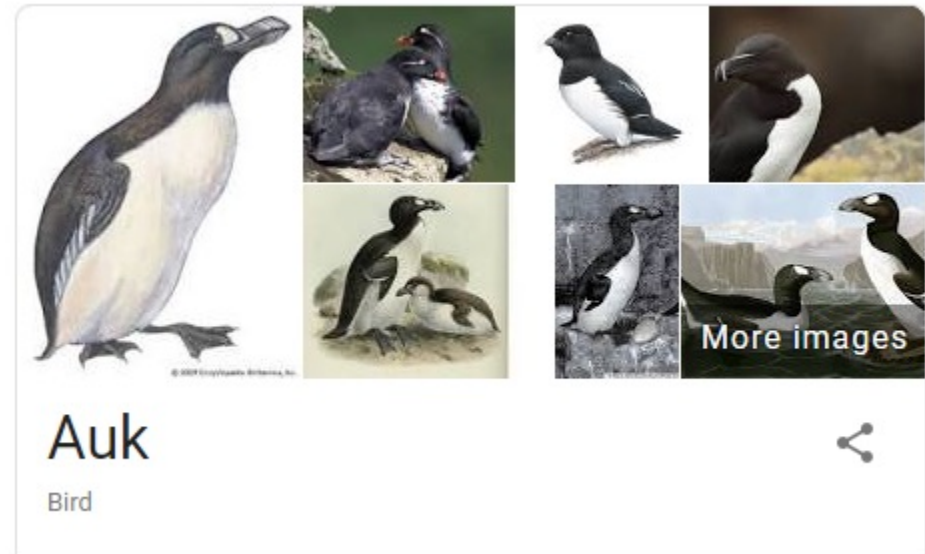
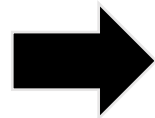
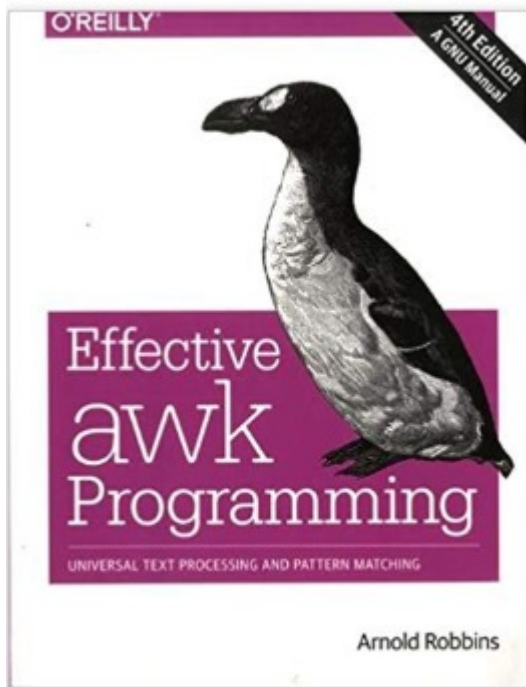
# awk

**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 ($3==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
```

- Process CSV files

```
awk 'BEGIN {FS=","} {print $1,$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
```

file1

1  
2  
3

- 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
```

# 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}' pedigree.txt | sort | uniq -c > s.temp
```

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

# Useful commands for Linux

- Several tutorials on the WEB !!
- unixcombined.pdf from Misztal web site
  - <http://nce.ads.uga.edu/~ignacy/ads8200/unixcombined.pdf>
- Online
  - <https://tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf>