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

Unix commands for data editing

Daniela Lourenco

Ignacio Aguilar

BLUPF90 TEAM – 08/2024

UNIVERSITÀ DEGLI STUDI FIRENZE

Huge volume of data

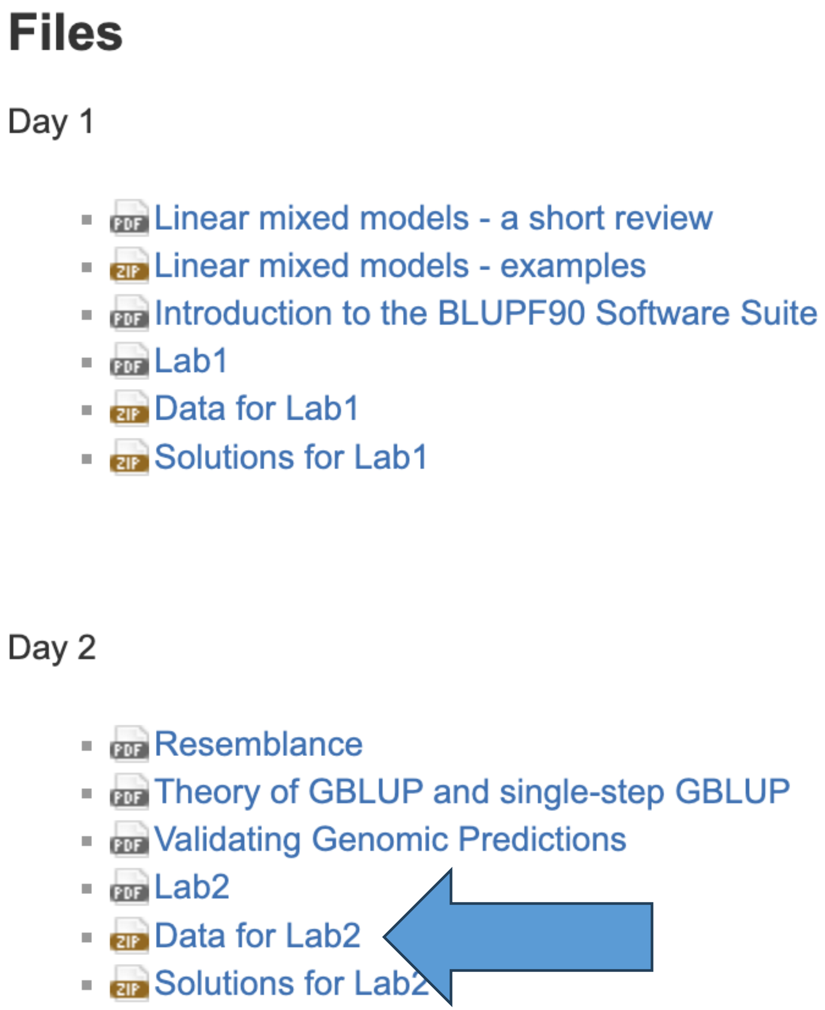
- Example: genomic info - 50k v2 (54609 SNP)
  - For 104 individuals
  - Illumina final report file:
    - 5,679,346 records
    - 302 MB

- 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



Files

Day 1

- Linear mixed models - a short review
- Linear mixed models - examples
- Introduction to the BLUPF90 Software Suite
- Lab1
- Data for Lab1
- Solutions for Lab1

Day 2

- Resemblance
- Theory of GBLUP and single-step GBLUP
- Validating Genomic Predictions
- Lab2
- Data for Lab2
- Solutions for Lab2

head / tail

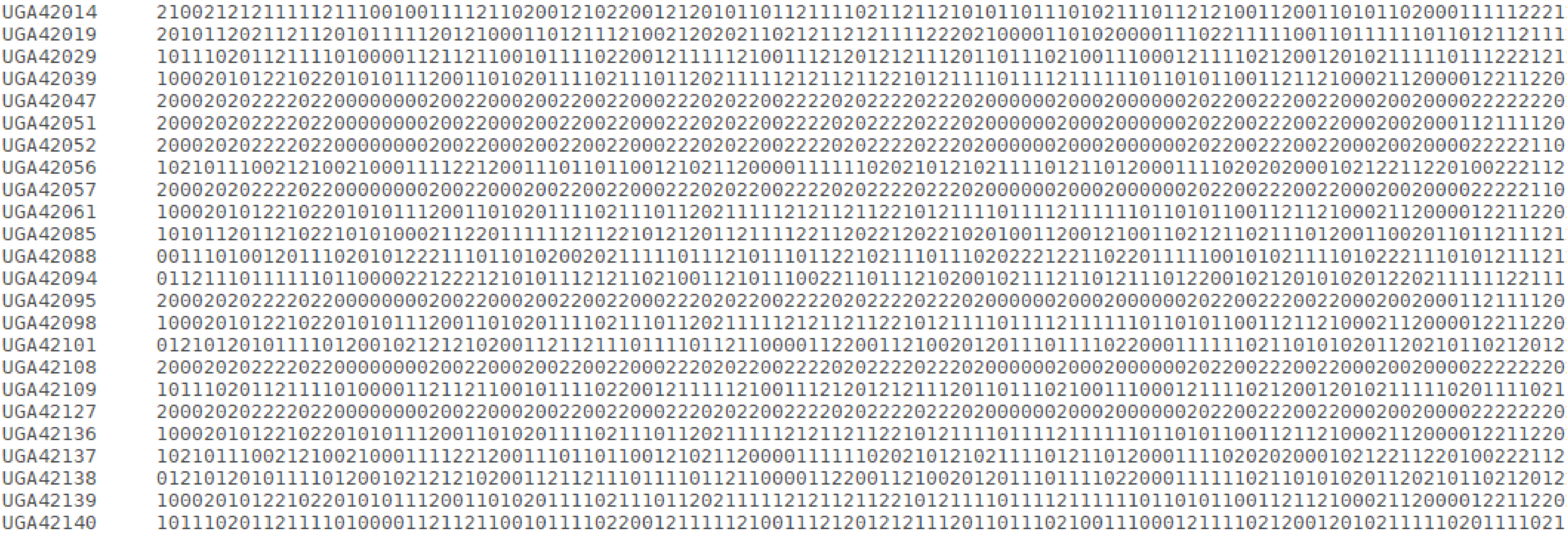
```
$ head pedigree.txt

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    210021212111112111001001111211020012102200121201011011211110211211210101101110102111011212100112001101011020001111122221
UGA42019    201011202112112010111112012100011012111210021202021102121121211112220210000110102000011102211111001101111110110121121111
UGA42029    101110201121111010000112112110010111102200121111121001112120121211120110111021001110001211110212001201021111101111222121
UGA42039    100020101221022010101112001101020111102111011202111112121121122101211110111121111110110101100112112100021120000122112200
UGA42047    200020202222022000000002002200020022002200022202022002222202022220222202000000020002000000020220022200220002002000022222220
UGA42051    200020202222022000000002002200020022002200022202022002222202022220222202000000020002000000020220022200220002002000112111120
UGA42052    200020202222022000000002002200020022002200022202022002222202022220222202000000020002000000020220022200220002002000022222110
UGA42056    102101110021210021000111122120011101101100121021120000111111020210121021111012110120001111020202000102122112201002221112
UGA42057    200020202222022000000002002200020022002200022202022002222202022220222202000000020002000000020220022200220002002000022222110
UGA42061    100020101221022010101112001101020111102111011202111112121121122101211110111121111110110101100112112100021120000122112200
UGA42085    101011201121022101010002112201111112112210121201121111221120221202210201001120012100110212110211101200110020110121111121
UGA42088    001110100120111020101222111011010200202111110111210111011221021110111020222122110220111110010102111101022211101012111121
UGA42094    011211101111110110000221222121010111212110210011210111002211011121020010211121101211101220010212010102012202111111122111
UGA42095    200020202222022000000002002200020022002200022202022002222202022220222202000000020002000000020220022200220002002000112111120
UGA42098    100020101221022010101112001101020111102111011202111112121121122101211110111121111110110101100112112100021120000122112200
UGA42101    012101201011111012001021212102001121121110111101121100001122001121002012011101111022000111111021101010201120210110212012
UGA42108    200020202222022000000002002200020022002200022202022002222202022220222202000000020002000000020220022200220002002000022222220
UGA42109    101110201121111010000112112110010111102200121111121001112120121211120110111021001110001211110212001201021111102011110211
UGA42127    200020202222022000000002002200020022002200022202022002222202022220222202000000020002000000020220022200220002002000022222220
UGA42136    100020101221022010101112001101020111102111011202111112121121122101211110111121111110110101100112112100021120000122112200
UGA42137    102101110021210021000111122120011101101100121021120000111111020210121021111012110120001111020202000102122112201002221112
UGA42138    012101201011111012001021212102001121121110111101121100001122001121002012011101111022000111111021101010201120210110212012
UGA42139    100020101221022010101112001101020111102111011202111112121121122101211110111121111110110101100112112100021120000122112200
UGA42140    101110201121111010000112112110010111102200121111121001112120121211120110111021001110001211110212001201021111102011110211
```

Counting lines/characters inside files

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

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

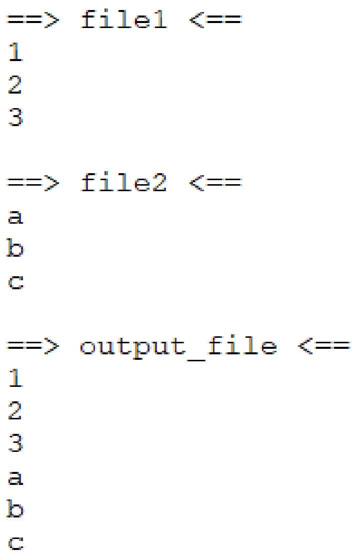
- Number of lines of a file(s)

```
$ 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
```

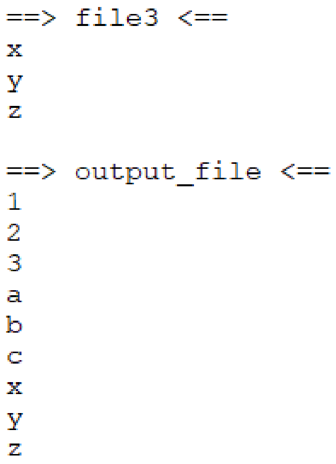


```
==> file1 <==
1
2
3

==> 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 file2 | head

1	a
2	b
3	c
```

```
$ paste -d “ ” file1 file2 | 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
```

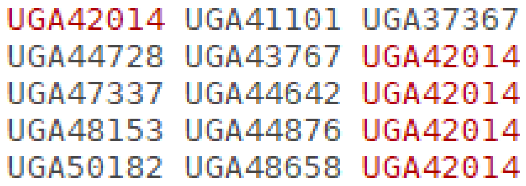
- Merges two files by column 1 but suppresses the joined output lines

```
$ 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
```

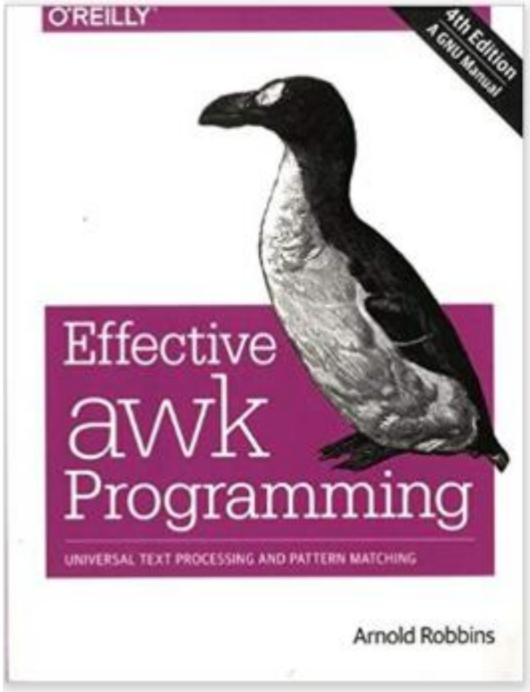
- Deletes lines that contain “pattern to match”

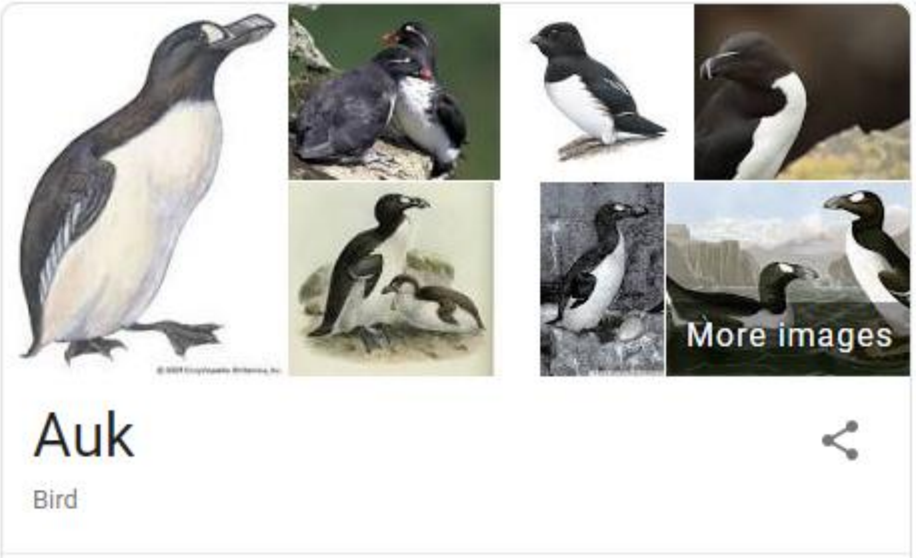
```
$ sed '/pattern to match/d' file
```

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
```

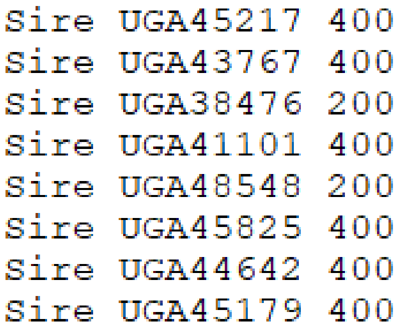
- Process CSV files
- `$ awk 'BEGIN {FS=","} {print $2,$3}' pedigree.txt > ped_out.temp`

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

```
1
2
3
```

- 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
```

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