Table of Contents

In [181]:

```
# Configuration - please ignore - this should not appear in the slideshow
alias grep='grep --color'
cd
cd workspace/peb_unix_intro/exercises/
# Please do not run this - it is only to format the slides
# if you did, type "unalias less"
alias less="head"
```

Introduction to Unix - awk, sed and Makefiles

Welcome to the Programming for Evolutionary Biology workshop!!

Giovanni M. Dall'Olio. Data Strategy and Design, GSK. March 2020.

Quick link to slides online: https://tinyurl.com/evop-unix (https://tinyurl.com/evop-unix)

All materials available here: https://github.com/dalloliogm/peb_unix_intro/archive/master.zip (https://github.com/dalloliogm/peb_unix_intro/archive/master.zip)

Press space or down key to continue.

Summary of the course today

- Morning first half: Basic Terminal commands, First Login to Linux
- Morning second half: Login to a Remote Unix server, browsing file contents
- Afternoon first half: Finding patterns in a file with grep; piping commands; cut, sort and uniq
- Afternoon second half: awk and sed; makefiles

Working with tabular files: Awk

The `awk` command is the equivalent of Excel/Calc for the command line.

It allows to filter files by specific columns, execute numerical operations, or change to the order of the columns.

Pros of awk vs Excel:

- Very low memory requirements
- Only one line at a time is processed, so `awk` can parse very large files without using resources

Cons of awk:

- Need to define operations before opening the file
- Imagine using an Excel file, without looking at it!
- large I/O usage
- only works on Text files

the GFF3 format for Genomic Annotations

These `awk` exercises are based on GFF3, which is another format to store gene annotations. It's similar to BED, but the column order is different.

The file `genes/human_genome_subset.gff` contains an example:

In [182]:

```
less genes/human_genome_subset.gff
```

```
##gff-version 3
##source-version rtracklayer 1.42.1
##date 2020-02-24
##genome-build .        hg19
chr22   rtracklayer     sequence_feature        24309026        2431
4748    .       +       .       gene_id=100037417;symbol=DDTL;ID=100
037417
chr22   rtracklayer     sequence_feature        22007270        2200
7347    .       +       .       gene_id=100126318;symbol=MIR301B;ID=
100126318
chr22   rtracklayer     sequence_feature        25498384        2550
8659    .       -       .       gene_id=100128531;symbol=LOC10012853
1;ID=100128531
chr22   rtracklayer     sequence_feature        49262582        4929
4198    .       +       .       gene_id=100128946;symbol=LINC01310;I
D=100128946
chr21   rtracklayer     sequence_feature        47247755        4725
6333    .       -       .       gene_id=100129027;symbol=LOC10012902
7;ID=100129027
chr22   rtracklayer     sequence_feature        17517460        1753
9682    .       +       .       gene_id=100130418;symbol=CECR7;ID=10
0130418
```

As you can see it is a tab-separated file.

The **GFF3** (General Feature Format) format specifications are defined here (https://genome.ucsc.edu/FAQ/FAQformat.html#format3), but in short:

- `col1`, `col4` and `col5` contain the chromosome name and genomic coordinates (start and end),
- `col2` describes the tool or resource that generated the annotation,
- `col3` describe the type of feature (e.g. gene, transcript, exon, TF binding site, Histone Acetylation mark, etc...
- `col9` column contains several fields, separated by a semicolon

Basic AWK syntax

The basic AWK syntax is the following:

```
awk 'PATTERN {ACTION}' filename
```

- **PATTERN**: select which rows to print

  `e.g. print only rows containing a specific word or pattern`

- **ACTION**: what to do on each line (e.g. print a specific column, sum them, etc..)

  `e.g. print only column #1 and #2`

Here is an example awk command:

In [183]:

```
# This prints all the lines on chromosome 21 and with chromosomal position betwe
en 10000 and 20000
# Don't need to type it yet - We'll go through this code in a minute
#
awk '$1 ~ /chr21/ && $4>11223344 && $5<44332211 {print $1, $4, $5, $6}' genes/hu
man_genome_subset.gff | less
```

```
chr21 31661463 31661832 .
chr21 37441940 37498938 .
chr21 31992946 31993169 .
chr21 31962424 31962716 .
chr21 32090843 32091095 .
chr21 34637937 34638565 .
chr21 30968360 31003067 .
chr21 31747612 31747696 .
chr21 14778705 14778781 .
chr21 42539484 42539556 .
```

Awk - the ACTION component

```
awk 'PATTERN {ACTION}' filename
```

The **ACTION** component of an awk command describes what to do with each line.

- Actions must be included in curly brackets { }
- The first column is `$1` ; the second `$2` ; and so on. `$0` to print everything
- The whole `awk` statement must be included in quotes (pref single quotes)

In [184]:

```
# Basic awk command - similar to using cut -f 1,4,5,9
awk '{print $1, $4, $5, $9}' genes/human_genome_subset.gff | head
```

```
##gff-version
##source-version
##date
##genome-build
chr22 24309026 24314748 gene_id=100037417;symbol=DDTL;ID=100037417
chr22 22007270 22007347 gene_id=100126318;symbol=MIR301B;ID=10012631
8
chr22 25498384 25508659 gene_id=100128531;symbol=LOC100128531;ID=100
128531
chr22 49262582 49294198 gene_id=100128946;symbol=LINC01310;ID=100128
946
chr21 47247755 47256333 gene_id=100129027;symbol=LOC100129027;ID=100
129027
chr22 17517460 17539682 gene_id=100130418;symbol=CECR7;ID=100130418
```

Exercise - compare awk with cut

Use `cut` to print columns 1,4,5,9 of the `genes/human_genome_subset.gff` file, and compare it with the `awk` statement

In [185]:

```
echo "Printing columns 1,4,5,9 with awk"
awk '{print $1, $4, $5, $9}' genes/human_genome_subset.gff | head
echo "" # empty line
echo "Printing columns 1,4,5,9 with cut"
cut -f 1,4,5,9  genes/chr8.gff  | head
```

```
Printing columns 1,4,5,9 with awk
##gff-version
##source-version
##date
##genome-build
chr22 24309026 24314748 gene_id=100037417;symbol=DDTL;ID=100037417
chr22 22007270 22007347 gene_id=100126318;symbol=MIR301B;ID=10012631
8
chr22 25498384 25508659 gene_id=100128531;symbol=LOC100128531;ID=100
128531
chr22 49262582 49294198 gene_id=100128946;symbol=LINC01310;ID=100128
946
chr21 47247755 47256333 gene_id=100129027;symbol=LOC100129027;ID=100
129027
chr22 17517460 17539682 gene_id=100130418;symbol=CECR7;ID=100130418

Printing columns 1,4,5,9 with cut
##gff-version 3
##source-version rtracklayer 1.34.1
##date 2017-02-27
##genome-build .
chr8    18248755        18258723        gene_id=10;symbol=NAT2;exerc
1=FALSE;ID=10
chr8    100549014       100549089       gene_id=100126309;symbol=MIR
875;exerc1=FALSE;ID=100126309
chr8    144895127       144895212       gene_id=100126338;symbol=MIR
937;exerc1=FALSE;ID=100126338
chr8    145619364       145619445       gene_id=100126351;symbol=MIR
939;exerc1=FALSE;ID=100126351
chr8    91970706        91997485        gene_id=100127983;symbol=C8o
rf88;exerc1=FALSE;ID=100127983
chr8    74332309        74353753        gene_id=100128126;symbol=STA
U2-AS1;exerc1=FALSE;ID=100128126
```

In [186]:

```
# To skip the headers, use grep with the -v option
# grep -v '#' genes/human_genome_subset.gff |  awk '{print $1, $4, $5, $9}'  |
 head
```

Operations between columns

`awk` allows to do mathematical operations between columns.

For example we can calculate the gene length, from gene start/end.

Coordinates in GFF files are **1-based**, so we need to add 1 to the formula.

In [187]:

```
# Calculating gene length
awk '{print $1, $4, $5, $5-$4+1, $9}' genes/human_genome_subset.gff  |  head
```

```
##gff-version   1
##source-version   1
##date   1
##genome-build   1
chr22 24309026 24314748 5723 gene_id=100037417;symbol=DDTL;ID=100037
417
chr22 22007270 22007347 78 gene_id=100126318;symbol=MIR301B;ID=10012
6318
chr22 25498384 25508659 10276 gene_id=100128531;symbol=LOC100128531;
ID=100128531
chr22 49262582 49294198 31617 gene_id=100128946;symbol=LINC01310;ID=
100128946
chr21 47247755 47256333 8579 gene_id=100129027;symbol=LOC100129027;I
D=100129027
chr22 17517460 17539682 22223 gene_id=100130418;symbol=CECR7;ID=1001
30418
```

Exercise: calculating gene center

Use `awk` to calculate the gene center for every row.

In [188]:

```
awk '{print $1, $4, $5, "center:"($5+$4)/2, "length:" $5-$4+1}' genes/human_geno
me_subset.gff  |  head
```

```
##gff-version   center:0 length:1
##source-version   center:0 length:1
##date   center:0 length:1
##genome-build   center:0 length:1
chr22 24309026 24314748 center:24311887 length:5723
chr22 22007270 22007347 center:2.20073e+07 length:78
chr22 25498384 25508659 center:2.55035e+07 length:10276
chr22 49262582 49294198 center:49278390 length:31617
chr21 47247755 47256333 center:47252044 length:8579
chr22 17517460 17539682 center:17528571 length:22223
```

Awk - the PATTERN component

```
awk 'PATTERN {ACTION}' filename
```

The PATTERN component of an awk command specifies filtering conditions.

- Filter rows where a specific column matches a string, or values are higher than a cut-off
- Columns can be referred as `$1`, `$2`, etc
- The whole awk statement must be included in quotes

Filtering by a string

The `awk` statement below will filter all the lines where the first column is `chr22`.

Note the syntax to match a specific string: `$COLUMN_NUMBER ~ /PATTERN/`

In [189]:

```
awk '$1 ~ /chr22/ ' genes/human_genome_subset.gff | head
```

```
chr22   rtracklayer     sequence_feature        24309026        2431
4748    .       +       .       gene_id=100037417;symbol=DDTL;ID=100
037417
chr22   rtracklayer     sequence_feature        22007270        2200
7347    .       +       .       gene_id=100126318;symbol=MIR301B;ID=
100126318
chr22   rtracklayer     sequence_feature        25498384        2550
8659    .       -       .       gene_id=100128531;symbol=LOC10012853
1;ID=100128531
chr22   rtracklayer     sequence_feature        49262582        4929
4198    .       +       .       gene_id=100128946;symbol=LINC01310;I
D=100128946
chr22   rtracklayer     sequence_feature        17517460        1753
9682    .       +       .       gene_id=100130418;symbol=CECR7;ID=10
0130418
chr22   rtracklayer     sequence_feature        17640279        1764
6335    .       +       .       gene_id=100130717;symbol=HDHD5-AS1;I
D=100130717
chr22   rtracklayer     sequence_feature        40428336        4043
2581    .       +       .       gene_id=100130899;symbol=LOC10013089
9;ID=100130899
chr22   rtracklayer     sequence_feature        42486937        4253
2702    .       +       .       gene_id=100132273;symbol=NDUFA6-DT;I
D=100132273
chr22   rtracklayer     sequence_feature        51021455        5102
2355    .       +       .       gene_id=100144603;symbol=CHKB-DT;ID=
100144603
chr22   rtracklayer     sequence_feature        18512151        1852
0734    .       +       .       gene_id=100192420;symbol=LINC01634;I
D=100192420
```

Exercise: How many chromosomes there are in the file?

- Use `awk` to print all the lines where the first column matches `chr`
- Use `wc` to count how many lines there are (see `man wc`)
- Bonus: Use `sort` and `uniq` to find how many unique chromosomes are present (don't need `cut` this time)
- Bonus: Use the `-c` flag in `uniq` to count # of rows per chromosome

In [190]:

```
awk '$1 ~ /chr/ ' genes/human_genome_subset.gff | wc
```

```
   1649   14866  169483
```

In [191]:

```
awk '$1 ~ /chr/ {print $1}' genes/human_genome_subset.gff | sort | uniq -c
```

```
    296 chr21
    535 chr22
    818 chr8
```

Filtering by numerical conditions

`awk` can also filter by numerical conditions.

We can also specify multiple conditions, using `&&` (AND) and `||` (OR).

The following will filter genes on `chr21`, from `11,223,344` to `44,332,211`:

In [192]:

```
awk '$1 ~ /chr21/ && $4>11223344 && $5<44332211' genes/human_genome_subset.gff |
less
```

```
chr21   rtracklayer     sequence_feature        31661463        3166
1832    .       -       .       gene_id=100131902;symbol=KRTAP25-1;I
D=100131902
chr21   rtracklayer     sequence_feature        37441940        3749
8938    .       -       .       gene_id=100133286;symbol=LOC10013328
6;ID=100133286
chr21   rtracklayer     sequence_feature        31992946        3199
3169    .       +       .       gene_id=100151643;symbol=KRTAP20-4;I
D=100151643
chr21   rtracklayer     sequence_feature        31962424        3196
2716    .       -       .       gene_id=100288287;symbol=KRTAP22-2;I
D=100288287
chr21   rtracklayer     sequence_feature        32090843        3209
1095    .       -       .       gene_id=100288323;symbol=KRTAP21-3;I
D=100288323
chr21   rtracklayer     sequence_feature        34637937        3463
8565    .       -       .       gene_id=100288432;symbol=IL10RB-DT;I
D=100288432
chr21   rtracklayer     sequence_feature        30968360        3100
3067    .       +       .       gene_id=100379661;symbol=GRIK1-AS2;I
D=100379661
chr21   rtracklayer     sequence_feature        31747612        3174
7696    .       -       .       gene_id=100422891;symbol=MIR4327;ID=
100422891
chr21   rtracklayer     sequence_feature        14778705        1477
8781    .       -       .       gene_id=100423018;symbol=MIR3156-3;I
D=100423018
chr21   rtracklayer     sequence_feature        42539484        4253
9556    .       +       .       gene_id=100423023;symbol=MIR3197;ID=
100423023
```

Exercise - Find chr8 sub-region

Print all the lines between for chromosome 8, between positions 5,000,000 and 10,000,000 (columns 4 and 5)

- Use the syntax `awk 'PATTERN' filename`
- PATTERN is a condition to filter column `$1 ~ /chr8/`, `$4 > 5000000` and column `$5 < 10000000`
- filename is `genes/human_genome_subset.gff`

In [193]:

```
awk '$1 ~ /chr8/ && $4 > 5000000 && $5 < 10000000 ' genes/human_genome_subset.gf
f  | head
```

```
chr8    rtracklayer     sequence_feature        7143733 7212876 .
-       .       gene_id=100128890;symbol=FAM66B;exerc1=TRUE;ID=10012
8890
chr8    rtracklayer     sequence_feature        7215498 7220490 .
-       .       gene_id=100131980;symbol=ZNF705G;exerc1=TRUE;ID=1001
31980
chr8    rtracklayer     sequence_feature        7812535 7866277 .
+       .       gene_id=100132103;symbol=FAM66E;exerc1=TRUE;ID=10013
2103
chr8    rtracklayer     sequence_feature        7783859 7809935 .
+       / Cows in \
chr8    rtracklayer     sequence_feature        6261077 6264069 .
| the      |
chr8    rtracklayer     sequence_feature        7272385 7274354 .
-       \ Genome!  /
chr8    rtracklayer     sequence_feature        7946463 7946611 .
.         ---------
chr8    rtracklayer     sequence_feature        6602685 6602765 .
+        ||  ^__^
chr8    rtracklayer     sequence_feature        8905955 8906028 .
+        ||  (oo)\\_______
chr8    rtracklayer     sequence_feature        6602689 6602761 .
-            (__)\\       )\\/\
```

Exercise - final awk test

- Find the line in `genes/human_genome_subset.gff` for the `POU5F1B` gene
- Calculate the length of the gene, using columns `$4` and `$5` . This will give you a number.
- Do another `awk` search to find the line of genes/chr8.gff where column `$9` contains such number

In [194]:

```
awk '$9 ~ /POU5F1B/ {print "gene length of POU5F1B is " $5-$4+1}' genes/human_ge
nome_subset.gff
```

```
gene length of POU5F1B is 1585
```

In [195]:

```
awk '$9 ~ /gene_id=1585/ {print $0}' genes/human_genome_subset.gff
```

```
chr8    Correct This_is_the_correct_answer      143991975       1439
99259   .       -       .       gene_id=1585; Correct, this is the r
ight answer
```

(skip)

Splitting a file in multiple files, according to value of a column

The example.bed file contains lines belonging to different chromosomes (column 1).

We can split these in separate files, using the following syntax:

In [196]:

```
#(skip)

#awk '{print>$1".txt"}' example.bed # doesn't work on MacBooc
awk '{print>$1}' genes/example.bed
```

Extracting gene symbols from a string in awk

How to extract the gene IDs and Symbols from the GFF file?

```
gene_id=10;symbol=NAT2;exerc1=FALSE;ID=10
```

We need to concatenate two awk commands, using a different Field Separator for each (`-F` flag).

In [216]:

```
awk '$1 ~ /chr/ {print $1, $4, $5, $9}' genes/human_genome_subset.gff |
  awk -F';' '{print $1, $2}' |
  head
```

```
chr22 24309026 24314748 gene_id=100037417 symbol=DDTL
chr22 22007270 22007347 gene_id=100126318 symbol=MIR301B
chr22 25498384 25508659 gene_id=100128531 symbol=LOC100128531
chr22 49262582 49294198 gene_id=100128946 symbol=LINC01310
chr21 47247755 47256333 gene_id=100129027 symbol=LOC100129027
chr22 17517460 17539682 gene_id=100130418 symbol=CECR7
chr22 17640279 17646335 gene_id=100130717 symbol=HDHD5-AS1
chr22 40428336 40432581 gene_id=100130899 symbol=LOC100130899
chr21 31661463 31661832 gene_id=100131902 symbol=KRTAP25-1
chr22 42486937 42532702 gene_id=100132273 symbol=NDUFA6-DT
```

Editing Text: sed

The Unix command `sed` is a streaming Text Editor for the command line.

It allows to modify text in a file and remove/add rows on text files

Pros of sed vs Word

- `sed` can modify large files without using much memory
- Don't need to keep the full file contents in memory

Cons of sed

- You need to define the operations before opening the contents
- Imagine editing a file without looking at it!

Basic sed syntax

The `sed` syntax is:

```
sed "OPERATION" FILENAME
```

Today we are going to see only the basic usage of `sed`, which is Find&Replace

The syntax is:

```
sed "s/FIND/REPLACE/" FILENAME
```

Find & Replace with sed

This will replace all the lower case instances of `chr` with upper case `CHR`:

In [207]:

```
sed "s/chr/CHR/" genes/human_genome_subset.gff | head
```

```
##gff-version 3
##source-version rtracklayer 1.42.1
##date 2020-02-24
##genome-build .        hg19
CHR22   rtracklayer     sequence_feature        24309026        2431
4748    .       +       .       gene_id=100037417;symbol=DDTL;ID=100
037417
CHR22   rtracklayer     sequence_feature        22007270        2200
7347    .       +       .       gene_id=100126318;symbol=MIR301B;ID=
100126318
CHR22   rtracklayer     sequence_feature        25498384        2550
8659    .       -       .       gene_id=100128531;symbol=LOC10012853
1;ID=100128531
CHR22   rtracklayer     sequence_feature        49262582        4929
4198    .       +       .       gene_id=100128946;symbol=LINC01310;I
D=100128946
CHR21   rtracklayer     sequence_feature        47247755        4725
6333    .       -       .       gene_id=100129027;symbol=LOC10012902
7;ID=100129027
CHR22   rtracklayer     sequence_feature        17517460        1753
9682    .       +       .       gene_id=100130418;symbol=CECR7;ID=10
0130418
```

Exercise - Find&Replace with sed

Starting from the query from the last awk exercise:

```
 awk '$1 ~ /chr/ {print $1, $4, $5, $9}' genes/human_genome_subset.gff |
awk -F';' '{print $1, $2}'
```

Add a `sed` statement to replace `gene_id=` and `symbol=` with empty strings

In [215]:

```
awk '$1 ~ /chr/ {print $1, $4, $5, $9}' genes/human_genome_subset.gff |
  awk -F';' '{print $1, $2}' |
  sed "s/gene_id=//" |
  sed "s/symbol=//" | head
```

```
chr22 24309026 24314748 100037417 DDTL
chr22 22007270 22007347 100126318 MIR301B
chr22 25498384 25508659 100128531 LOC100128531
chr22 49262582 49294198 100128946 LINC01310
chr21 47247755 47256333 100129027 LOC100129027
chr22 17517460 17539682 100130418 CECR7
chr22 17640279 17646335 100130717 HDHD5-AS1
chr22 40428336 40432581 100130899 LOC100130899
chr21 31661463 31661832 100131902 KRTAP25-1
chr22 42486937 42532702 100132273 NDUFA6-DT
```

Makefiles

The main `peb_unix_intro` contains a file called `Makefile` .

Let's have a look at its contents:

In [218]:

```
cd
cd workspace/peb_unix_intro  # Adjust this path to your system
head Makefile
```

```
# This is a Makefile, which will be explained later in the course.
# Please don't look at it yet :-)

explain: explain_text
	@echo Try running "make explain" to read an explanation
publish: slides_bash commit
	echo "convert the slides to pdf, commit, and push to github"
	git push
```

Defining pipelines with Makefiles

`Makefiles` are files describing pipelines of shell commands.

Today you have typed many Unix commands, with different options and target. Would you remember what you typed and which options you have used?

Writing them into a `Makefile` would make it easier to reproduce them.

A Makefile is a collection of "rules".

Each of these rules follows this basic syntax is:

```
target: prerequisites
    commands to execute
```

As you can see in the Makefile included, most of the rules allow to regenerate the exercise files, or to execute some commands without having to type them everytime.

For example, the rule "testrule" is associated to two echo commands.

How to run Makefile rules

To execute a rule in the Makefile, simply type:

```
make [name of the rule]
```

For example:

In [199]:

```
make testrule
```

```
echo this is a Makefile rule
this is a Makefile rule
echo You can associate it to as many commands you want
You can associate it to as many commands you want
```

The program "make" will automatically detect any file named "Makefile" in the current directory, and execute any rule with the specific name.

Exercise - run your make rule

- Look at the contents of the Makefile
- Find the rule needed to complete this first Make exercise
- run it with `make RULENAME`

In [219]:

```
make make_exercise
```

```
Run this rule to complete the first Make exercise
Make allows to save pipelines of Unix commands, and quickly re-execu
te them
```

Exercise - write your own Make rule

- Edit the Makefile, using nano or other editors
- create a new rule called `new_rule`
- Associate a `echo` statement to it, copying the other rules
- Note: you need a TAB in the second line of the rule
- run the rule using `make`

What have you learned today?

- The Unix command line is fun! (hopefully)
- There is an incredible number of many Unix Commands to deal with text files
- Unix commands are specialized to do single task, and are meant to be combined together with the pipe operator

Dinner time!!

In [205]:

```
cd ~/workspace/peb_unix_intro/exercises
```