Programming for Evolutionary Biology
March 21 – April 6 2014
Leipzig, Germany

Introduction to Unix systems
Part 3: grep, and Unix philosophy

Giovanni Marco Dall'Olio
King's College of London, UK

Schedule

- 9.30 – 11.00: “What is Unix?”
- 11.30 – 12.30: Introducing the terminal
- **14:30 – 16:30: grep & Unix philosophy**
- 17:00 – 18:00: awk, sed, and make

Grep: search files for a pattern

- The command grep allows to search for patterns in a file
- Basic usage:
  - grep “searchword” filename
- Example:
  - grep “ATG” myfasta.fa

Let's go to the exercises folder

- Remember:
  - cd (to go back to the home folder)
  - cd /homes/evopserver/lectures/unix_intro
  - cd exercises
  - ls

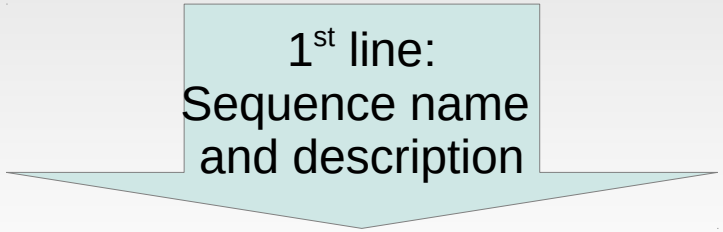
Let's look at the fastq examples

- Go to the folder “unix_intro/exercises/fastq”
  - This time is fastq, not fasta!
- Fastq is a format to represent sequences and quality scores
- To see what is in the files:
  - less sample_1000genomes_fastq.fastq

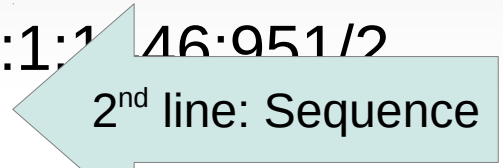
Let's look at the fastq examples

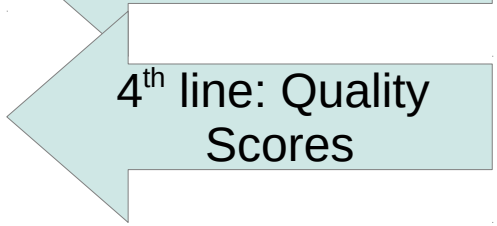
- Go to the folder “unix_intro/exercises/fastq”
  - This time is fastq, not fasta!
- Fastq is a format to represent sequences and quality scores
- To see what is in the files:
  - less sample_1000genomes_fastq.fastq
- Tip: try the less -S option
  - less -S sample_1000genomes_fastq.fastq

The Fastq format



@SRR062634.321 HWI-EAS110_103327062:6:1:1 46:951/2
TGATCATTTGATTAATACTGACATGTAGACAA
+
B5=BD5DAD?:CBDD-DDDDDCDDB+-B:;?A?C





Let's “grep”

- Let's use grep to see if any of the sequences in the fastq folder contains the word “ACTG”
    - grep “ACTG” *
- Note that grep is case sensitive. “ACTG” is different than “actg”

“grep ACTG” *

File Edit View Search Terminal Help

```
[giovanni@evopserver fastq]$  grep "ACTG" *
TGATCATTTGATTAATACTGACATGTAGACAAGAAGAAAAGTATGTTTCATGCTATTTTGA
GTAACTTCCATTTAGAAGCCTACTCCTGAGCACAACATT
CCATGTGGTCCTACGCTATCAGAAGCTATCTTAGGACCTCTCTTGCTTGCATTAAGATTGG
AAAGACAAACTTGAGAAAAAAAATTCAGTTGACTGAGAA
TGGGATTACAAGCGTGAGCCACTGCATCCGGCTGATCACAGCTATTTTAAAGTCCAAGTAG
TTTGATAACTTCAATATCTAAATCACCTGTAAATCTGTT
TGCAATTAAGATTATTGTTTTACTTAATACTGATTCTCTTAAGGATTTTTGTGATACATTT
TAATTTTTTGCACAAATTTCTAAACAATGCACAATAACT
GAACCTGCCTTTGAGAGTTCAGGTTCAAAACACTCTTTCTGTATAATCTGCAAGTGGATAT
TTGGACCACTGGCTGGCCTTCGTTCGACACGGGGATATG
TTTTACATGTTTTTATTGATACGGACTCACTGTTAATTTTTTCATCTTGATATTGGAGTCA
CATTTGGGACTCCAATCCACGGCACACTGGCCCTTCGTT
ACTGAGTATTAAATCTAGACCTTTAATTACACATATTTTTACTTTTCTGTGTGATGAAATA
GGAAGGATGCATGAGGCTTTTTTTTTTTGAGAGGGAGTC
CAAGCTGTACTGCCTTGGCCTCTGAAAGTGCTGGGATTATAGGCATGAGCCACTGCTCACA
GCCGAGGTTGAAAGTATTTTTATGAATGAACAGGAGTGC
CAGGATACAGTCCCCCTCCTGGCTGCTTTCACAGGCTGGTGTTGAGGGTTTGCGGGTTTTT
TTGGGACTGCGTGGTAGTTGGTTATAGGTCCTCCGCTCC
CCTTTTACCAACATCTCTCCTTCCCCTATTCCCCCTTCCCCCCCACTGTCAGAAACTCCAT
TTGTAACACTTTTGAAGTGTATACAAAATATAGATCGGA
TGAGGAACTAAAGTTGATACCCACTTTTTATTAACTAGGCCAAAATGATGACAGTTTAAAG
```

Grep options

- Let's have a look at the man page for grep
  - man grep

The grep man page

File Edit View Search Terminal Help

GREP(1) GREP(1)

NAME

grep, egrep, fgrep - print lines matching a pattern

SYNOPSIS

grep [OPTIONS] PATTERN [FILE...]
grep [OPTIONS] [**-e** PATTERN | **-f** FILE] [FILE...]

DESCRIPTION

grep searches the named input FILEs (or standard input if no files are named, or if a single hyphen-minus (**-**) is given as file name) for lines containing a match to the given PATTERN. By default, **grep** prints the matching lines.

In addition, two variant programs **egrep** and **fgrep** are available. **egrep** is the same as **grep -E**. **fgrep** is the same as **grep -F**. Direct invocation as either **egrep** or **fgrep** is deprecated, but is provided to allow historical applications that rely on them to run unmodified.

Manual page grep(1) line 1 (press h for help or q to quit)

Case sensitivity

- Most command line tools are case sensitives
- This means that “ACTG” and “actg” are not considered as the same sequence
- We can use the -i option to ignore the case:
    - grep -i “actg” *

grep -B

- We have used grep to get all the sequences that have “ACTG” in our files
- However, the name of the sequences are stored in the line above the sequence
- We can use the -B option to print the line above each match

Retrieve all the sequences that do <not> match

- The grep **-v** option prints only the lines that do <not> match the query
- grep -v “ACTG” → returns the exact opposite results than the previous query

Short exercise

- Which parameter can be used to count the number of matching lines, instead of printing them? (hint: see the man page)
- Which parameter can be used to print the lines below? (the opposite of the -B option)
- How many sequences match the pattern “AAATTTC” in the sample vcf file?

Hint: saving results to a file

- It may be useful to save the results of a grep search to a file
- To do this, you can use the “>” operator
- Example:
    - grep ACGTG sample_vcf.vcf > results.txt

Regular Expressions

- Regular Expressions are a way to specify a string that matches different set of characters
- They will be explained better in the Perl course
- For now, let's see some basic examples

Simple grep regular expressions

- The symbol “.” matches any character
- Example:
  - grep 'AAA..GGG' matches all the sequences that start with a “AAA”, have any two other characters, and end with a “GGG”

Simple grep regular expressions

- We can use the brackets to specify a set of characters
- For example, “[ACG]” will match any A, C or G
- Let's try it:
    - grep AAA[ACG].GGG sample_1000genomes_fastq.fastq

Exercises

- Search the following sequences in the file sample_1000genomes_fastq.fastq :
  - ACTGAAT
  - ACC followed by any 3 characters, then CC
  - AAA followed by any C or G

More grep exercises

- grep is an important command, so let's see other examples

Let's look at genbank files

- Genbank is a format for storing annotation on sequences, used by the Genbank database
- You should be in the fastq folder. To go to the genbank folder:
    - cd ..
    - ls
    - cd genbank

The genbank format

- Seems a nice format for grepping!

```
LOCUS       NM_000589                921 bp    mRNA    linear   PRI 26-FEB-2012
DEFINITION  Homo sapiens interleukin 4 (IL4), transcript variant 1, mRNA.
ACCESSION   NM_000589
VERSION     NM_000589.2  GI:27477090
KEYWORDS    .
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
            Catarrhini; Hominidae; Homo.
REFERENCE   1  (bases 1 to 921)
  AUTHORS   Panzer,M., Sitte,S., Wirth,S., Drexler,I., Sparwasser,T. and
            Voehringer,D.
  TITLE     Rapid in vivo conversion of effector T cells into Th2 cells during
            helminth infection
  JOURNAL   J. Immunol. 188 (2), 615-623 (2012)
   PUBMED   22156341
  REMARK    GeneRIF: Th1 and Th17 cells can be repolarized to acquire
            expression of IL-4 and lose the expression of their characteristic
            cytokines IFN-gamma and IL-17A, respectively; however, regulatory T
            cells are largely resistant to repolarization.
REFERENCE   2  (bases 1 to 921)
  AUTHORS   Liu,X., Wang,G., Hong,X., Wang,D., Tsai,H.J., Zhang,S.,
            Arguelles,L., Kumar,R., Wang,H., Liu,R., Zhou,Y., Pearson,C.,
            Ortiz,K., Schleimer,R., Holt,P.G., Pongracic,J., Price,H.E.,
            Langman,C. and Wang,X.
  TITLE     Gene-vitamin D interactions on food sensitization: a prospective
            birth cohort study
```

Grepping genbank

- Let's see how many sequences there are in the files contained in the examples folder:
  - grep 'LOCUS' *

How many PLoS articles are in the files?

- grep 'PLoS' *

Grepping multiple patterns

- Use the -e option:
  - grep -e 'PLoS' -e 'Cancer Lett'

Exercises

- Let's check together:
  - How many articles by “Powers” are cited?
  - How many CDS sequences are in the files?

Other commands: cut and sort

Another command: “cut”

- cut allows to extract columns from a tab/comma separated file
- We will play with it on some vcf files

Go to the vcf folder

- You should be in the vcf folder. To go to the genbank folder:
  - cd ..
  - ls
  - cd vcf

The VCF format

- VCF is a tab-separated format used to store SNP genotypes

```
##fileformat=VCFv4.0
##source=BCM:SNPTools:hapfuse
##reference=1000Genomes-NCBI37
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=AP,Number=2,Type=Float,Description="Allelic Probability, P(Allele=1|Haplotype)">
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  HG00096 HG00097 HG00099 HG00100 
2       39967768        rs11124691      T       A       100     PASS    .       GT:AP   0|1:0.000,1.000 
2       39967778        .       G       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.00
2       39967824        rs114023135     C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 
2       39967950        .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.00
2       39968061        .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.00
2       39968140        rs72936091      C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 
2       39968210        rs6716262       G       T       100     PASS    .       GT:AP   0|0:0.000,0.000 
2       39968231        rs78390685      T       A       100     PASS    .       GT:AP   0|0:0.000,0.000 
2       39968486        rs118064218     T       A       100     PASS    .       GT:AP   0|0:0.000,0.000 
2       39968596        rs76809080      C       T       100     PASS    .       GT:AP   0|0:0.000,0.000 
2       39968633        .       G       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.00
2       39968753        .       T       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.00
2       39968829        .       C       A       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.00
2       39968879        .       T       G       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.00
2       39969136        rs73924914      G       T       100     PASS    .       GT:AP   0|0:0.000,0.000 
2       39969173        .       T       C       100     PASS    .       GT:AP   0|0:0.000,0.000 0|0:0.00
```

“cut” on a VCF file

- Basic cut usage:
  - cut -f <column_number> <filename>
- The sample VCF file in the exercise folder has a lot of columns
- Let's use cut to extract only the second column of the file:
  - cut -f 2 sample_vcf.vcf

Specifying ranges in cut

- You can also specify ranges of columns
- For example, to print the first columns:
    - cut -f 1-5 sample_vcf.vcf

Let's get the genotype of HG00099

- Let's say that we want to extract the genotype of the individual HG00099
- HG00099 is on the 12th column of the vcf file
    - cut -f 12 sample_vcf.vcf

Let's look at other “cut” options

```
CUT(1)                          User Commands                          CUT(1)

NAME
       cut - remove sections from each line of files

SYNOPSIS
       cut OPTION... [FILE]...

DESCRIPTION
       Print selected parts of lines from each FILE to standard output.

       Mandatory  arguments  to long options are mandatory for short options
       too.

       -b, --bytes=LIST
              select only these bytes

       -c, --characters=LIST
              select only these characters

       -d, --delimiter=DELIM
              use DELIM instead of TAB for field delimiter
 Manual page cut(1) line 1 (press h for help or q to quit)
```

“cut -c”

- Use cut -c to define a range of characters instead of a range of columns
- This is useful to deal with files that are not comma-separated

Exercise

- Which parameter can be used to define a different field separator? (for example: a comma-separated file instead of a tab-separated file)

The “sort” command

- Let's look at another command: sort
- The sort command can be used to sort a csv file by a column

Let's sort

- Basic sort usage:
  - sort -k <column_number> filename
- Let's sort the sample vcf file by chromosome position:
  - sort -k 2 sample_vcf.vcf

“sort -k 2 sample_vcf.vcf”

- Note: when sorting numerical columns, the “-n” option should be used

Let's sort by genotype state of HG00099

- sort -k 12 sample_vcf.vcf

How to sort by multiple columns

- Use the -k option multiple times:
  - sort -k 2,2 -k 12,12 sample_vcf.vcf
- Add the -n option to sort numbers, and -v to reverse the sorting order:
  - sort -k 2,2n -k12,12v sample_vcf.vcf

Unix philosophy & Piping

The history of Unix systems

- Let's go back in time to when the original Unix system was developed:
  - Graphical interfaces didn't exist
  - No facebook or gmail
  - Computers were used mainly by programmers, and for data analysis / document formatting

The history of Unix systems (2)

- When Unix was designed, it was used mostly for data analysis and document formatting
- Unix was important for the diffusion of the first best practices on data analysis and document formatting
- By studying Unix's history and philosophy, we can learn how the first programmers approached these problems

The Unix Philosophy

- The Unix Philosophy can be summarized as:
  - Store everything on text files
  - Write small programs that carry out a single task
  - Combine the small programs together to perform more complex tasks

The Unix Philosophy

- “Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.”
    - (Doug McIlroy, inventor of the Unix pipes system)

Each Unix tool is specialized on a task

- Did you notice that each of the tools we have seen today is specialized on one single task?
  - grep searches for a word in a file
  - cut extracts columns
  - sort orders a file

Each Unix tool is specialized on a task

- Did you notice that each of the tools we have seen today is specialized on one single task?
  - grep searches for a word in a file
  - cut extracts columns
  - sort orders a file
- Almost all Unix command line tools are specialized on a task:
  - diff, find, echo, wait...

Chaining commands together

- So, the Unix approach to programming is to write small programs specialized on single tasks, and organize them together
- To chain commands together, we will use the "pipe" command ("|")

The “pipe” system

- The “|” symbol can be used to “chain” commands together
- To type the “|” symbol, use AltGr+> on your keyboard:

Basic Pipe usage

- The pipe command redirects the output of a command into another one
- Let's try it:
    - man ls | head → prints the first lines of the “ls” manual

Piping to “less”

- The command “less” allows to scroll the contents of a file (we have seen it this morning)
- It is often useful to redirect the output of a command to less
- Example:
    - grep -v '#' sample_vcf.vcf | less

Piping exercise on vcf files

- You should be in the exercises/vcf folder
- Let's get the first 5 columns of the first 10 lines of the vcf file:
    - cut -f 1-5 sample_vcf.vcs | head -n 10
- Explanation:
    - cut -f 1-5 sample_vcf.vcf → extracts columns 1 to 5
    - head -n 10 → prints first 10 lines

Piping exercise on vcf files

- Let's repeat the previous example, but removing all the lines starting with “#” first:
  - grep -v '#' sample_vcf.vcf | cut -f 1-5 | head -n 10
- Explanation:
  - grep -v '#' sample_vcf.vcf → extract all the lines that do <not> contain a “#” (note the -v flag)
  - cut -f 1-5 → extracts columns 1 to 5
  - head -n 10 → prints first 10 lines

“uniq”

- uniq removes all duplicated lines from a file
- Useful to elaborate the results of cut or grep
- Example:
    - cut -f 5 sample_vcf.vcf | uniq

“uniq”

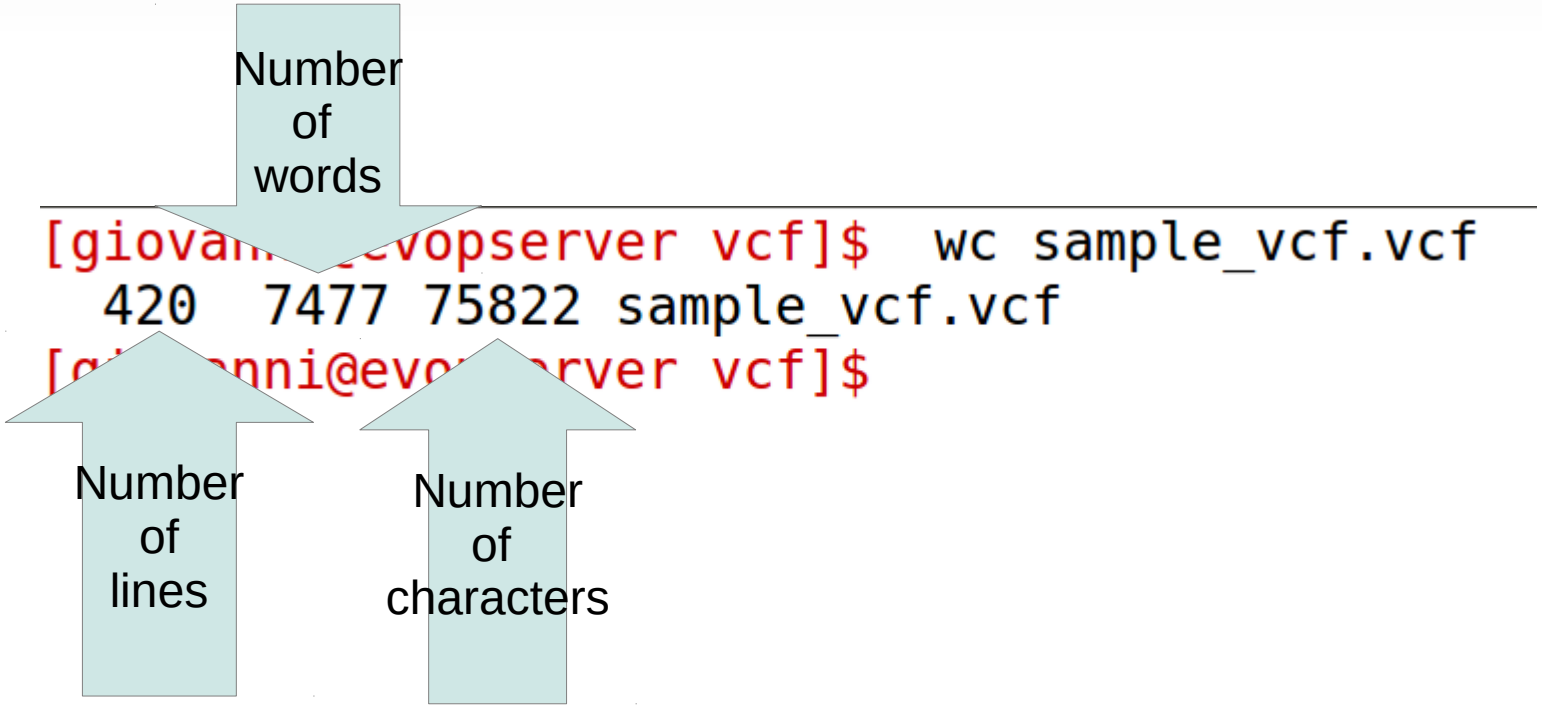
- uniq removes all duplicated lines from a file
- Useful to elaborate the results of cut or grep
- Example:
    - cut -f 5 sample_vcf.vcf | uniq
- Important: uniq expects the input to be already sorted
    - cut -f 5 sample_vcf.vcf | sort | uniq

concatenating grep calls

- Multiple grep calls are useful to apply multiple conditions
- Example:
  - grep -v '#' sample_vcf.vcf | cut -f 10 | grep '1:1' | sort | uniq

Another command: wc

- wc stands for “word count”
- It counts the number of words, lines and characters in a file



“wc” is useful to count the number of results

- Let's add it to the previous example:
  - grep -v '#' sample_vcf.vcf | cut -f 10 | grep '0:1' | sort | uniq | wc -l
  - (counts the number of SNPs for which the individual in the 10th column, HG0096, has the 0:1 genotype)

Unix philosophy applied to Programming

- The Unix philosophy is a general approach to data analysis
- When you will have learned programming, you can apply it to any data analysis task
    - Split your analysis in small segments
    - Write a different script for each segment
    - Use piping to put everything together

Example of a bioinfo pipeline

- Imagine that we need to plot the CG content of a genome:
  - Approach 1 (not Unix): write a script that downloads the sequence of the genome, calculate the CG content, and draw a plot
  - Approach 2 (Unix): write three separate scripts: one to download the sequence, one to calculate CG content, and another to draw the plot. Then, pipe them together.

Questions

- Which of the commands seen today (grep, cut, sort) would be useful for your work?
- What do you find more difficult about Unix systems?
- Which kind of analysis workflows do you need to do for your research?

Unix Approach to Bioinformatics Conclusions

- The Unix approach is more flexible and adaptable to other situations
- You don't need to use the Unix approach always, but you should be aware of it

Advanced Commands

- tr → replace a character in a text file
- comm → compare two files
- join → join files by a common column
- paste → merge lines of files

More advanced commands

- sed → stream editor
- awk → spreadsheet-like programming language
- make → define simple pipelines

Resume of the session:

- grep: search files for a text
- cut: extract columns from a csv file
- sort: sort a csv line
- The pipe symbol (|) can be used to combine Unix tools together

Time for a break!

Next session starts at 17:00