Solutions to the Exercises

Commandline tools

I. TAR & GZIP

1. Use gzip to compress the file P12931.txt

```
# gzip P12931.txt
```

2. Decompress the resulting file P12931.txt.gz (revert previous command)

```
# gunzip P12931.txt.gz  # or gzip –d P12931.txt.gz
```

3. Use tar to create an archive containing all fasta files in the current directory into an archive called "fastafiles.tar"

```
# tar –c -f fastafiles.tar *.fasta
```

4. Use gzip to compress the archive "fastafiles.tar".

```
# gzip fastafiles.tar
```

5. How can you achieve the two previous steps "using tar to create archive" and "gzip the archive" in one command?

```
# tar –c –z -f fastafiles.tar.gz *.fasta  # note the -z
```

6. Test (list the contents of) the compressed archive "fastafiles.tar.gz"

```
# tar –tf fastafiles.tar.gz
```

7. Download the compressed PDB file for entry 1Y57 from rcsb.org (eg. "wget http://www.rcsb.org/pdb/files/1Y57.pdb.gz") and decompress it.

```
# wget "http://www.rcsb.org/pdb/files/1Y57.pdb.gz"
# gunzip 1Y57.pdb.gz
```

II. GREP

1. Which of the DNA files ENST* contains 'TATATCTAA' as part of the sequence?

```
# grep TATATCTAA ENST0*
ENST00000380152.fasta:ACGGAAGAATGTGAGAAAAATAAGCAGGACACAATTA
CAACTAAAAAATATATCTAA
ENST00000544455.fasta:ACGGAAGAATGTGAGAAAAATAAGCAGGACACAATTA
CAACTAAAAAATATATCTAA
```

2. List only the names of the DNA files ENST* that contain 'CAACAAA' as part of the sequence.

```
# grep –l CAACAAA ENST0*
ENST00000380152.fasta
ENST00000544455.fasta
```

3. Considering the previous example, would you consider grep a suitable tool to perform motif searches? Why not? Try to find the pattern 'CAACAAA' by manual inspection of the first two lines of each sequence.
   Answer: When using grep as a motif searching tool, you need to keep in mind that grep (like sed and awk) is line-oriented, meaning that by default it only searches for a

given motif in a single line. In the given example, upon manual inspection you will find the given motif also in the file ENST00000530893.fasta, which grep missed. You would need to think about how to do multi-line searches (eg. Removing line-breaks etc.)

4. Count the number of ATOMs (lines starting with 'ATOM') in the file 1Y57.pdb.
5. Does this number agree with the annotated number of atoms (Search the REMARKs for 'protein atoms')

```
# grep –c "ATOM" 1Y57.pdb
3632
# grep –i "protein atoms" 1Y57.pdb
REMARK   3   PROTEIN ATOMS                  : 3600
```

This means there are 3600 atoms annotated in this PDB file, however we counted 3632. This is because grep also counted any occurrence of "ATOM" within REMARKS. We can avoid this by either filtering out the remarks:

```
# grep –v "REMARK" 1Y57.pdb | grep –c "ATOM"
3600
```

…or by telling grep to only count those lines that start with "ATOM":

```
# grep –c "^ATOM" 1Y57.pdb
3600
```

III. SED

IV. Use sed to print only those lines that contain "version" in the files P05480.txt and P04062.txt

```
# sed –n '/version/p' P05480.txt P04062.txt
```

1. Use sed to change the text 'sequence version 3' to 'sequence version 4' in the files P05480.txt and P04062.txt (without actually changing the files, just printing)

```
# sed 's/sequence version 3/sequence version 4/'
P05480.txt P04062.txt
```

2. Use sed to update the text 'sequence version 3' to 'sequence version 4' in the files P05480.txt and P04062.txt (this time, make the changes directly in the files)

```
# sed –i.bak 's/sequence version 3/sequence version 4/'
P05480.txt P04062.txt
```

3. Replace (transliterate) all occurrences of 'r' by 'l' and 'l' by 'r' (at the same time) in the file PROTEINS.txt (so that 'structural' becomes 'stluctular')

```
# sed 'y/rRlL/lLrR/' PROTEINS.txt
```

V. AWK

1. Use awk to print only those lines that contain "version" in the files P12931.txt and P05480.txt and think about how this procedure is different to sed.

```
# awk '/version/ {print}' P12931.txt P05480.txt
```

This is very similar to sed, you also have to use the slashes '/' to define the search pattern. However the sed notation is a little more concise…

2. For all FASTA files that begin with "P" ("P*.fasta") print only the second item of the header (split on "|") eg. for ">sp|P12931|SRC_HUMAN Proto-oncogene", print only "P12931"

```
# awk –F"|" '/>/ {print $2}' P*.fasta
```

3. The file 'P12931.csv contains phosphorylation sites in the protein P12931. (If the file 'P12931.csv' does not exist, use 'wget' to download it from "http://phospho.elm.eu.org/byAccession/P12931.csv").
   a. Column three of this file lists the amino acid position of the phosphorylation site. You are only interested in position 17 of the protein. Try to use 'grep' to filter out all these lines containing '17'.

```
# grep 17 P12931.csv
```

   b. Now use awk to show all lines containing '17'.

```
# awk '/17/ {print}' P12931.csv
```

   c. Next try show only those lines where column three equals 17 (Hint: The file is semicolon-separated…).

```
# awk –F";" '$3==17 {print}' P12931.csv
```

   d. Finally print the PMIDs (column 6) of all lines that contain '17' in column 3.

```
# awk –F";" '$3==17 {print $6}' P12931.csv
```

Quoting and Escaping

1. Familiarize yourself with quoting and escaping.
   a. Run the following commands to see the difference between single and double quotes when expanding variables:

```
# echo "$HOSTNAME"
# echo '$HOSTNAME'
```

   b. Next, use ssh to login to a different machine to run the same command there, again using both quoting methods:

```
# ssh pc-atcteach01 'echo $HOSTNAME'
```

```
# ssh pc-atcteach01 "echo $HOSTNAME"
```

Closely inspect the results; is that what you were expecting? Discuss this with your neighbour.