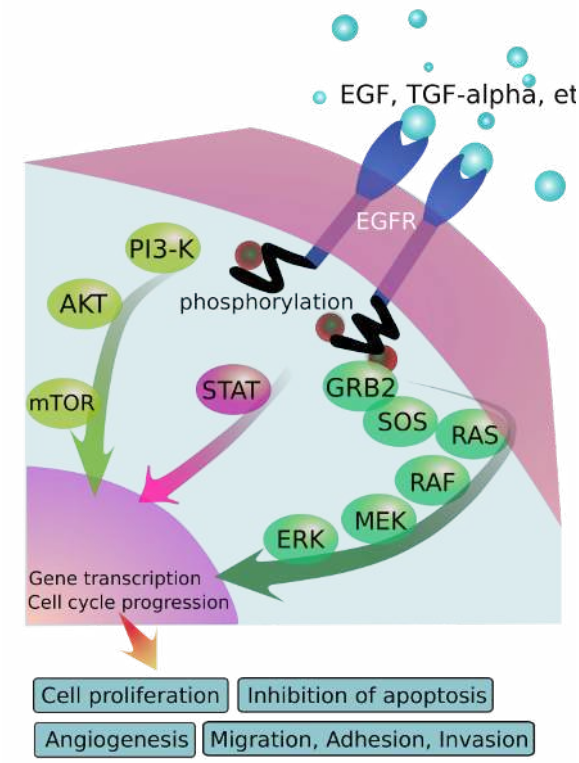
Introduction to Bioinformatics
2nd practice - Linux II.

During the session we will get more familiar with 1D plot and some standard unix tools that are useful for manipulating data in large scale. You are going to investigate the biophysical properties of the Epidermal Growth Factor Receptor (EGFR) protein for human, with a view to discovering membrane spanning region(s). You will also practice how to build simple pipelines, filter large body of data. More specifically, you will understand the concept of regular expressions and you will apply the concept in the *sed*, *grep* and *awk* commands. The required softwares are a web browser (Using the site: http://web.expasy.org/protscale/) and a unix terminal, since all the above mentioned programs are normally available in a unix environment.

Files you need:

- 5j8v.pdb - protein structure information

1. Locating the membrane spanning region in the EGFR protein for human



For the sake of simplicity, we will investigate a protein with just one, very clear membrane spanning region in its middle. “Real Life” is often not so kind … but “Real Life” is not the problem of the day. As we will hopefully have discussed, when a protein is traveling through a membrane it is typically: hydrophobic, and highly hydrophobic regions are membrane spanning. Also, there is a statistical property of amino acids to be found in membrane spanning regions. This is correlated with hydrophobicity, but it is determined by independent methods. Where both properties coincide, maybe we can reasonably suspect a membrane spanning region?

```
> EGFR_HUMAN
MRPSGTAGAALLALLAALCPASRALEEKKVCQGTSNKLTQLGTFEDHFLSLQRMFNNCEVVLGNLEITYVQRNYDLSFLKTIQEVAGYVLI
ALNTVERIPLENLQIIRGNMYYENSYALAVLSNYDANKTGLKELPMRNLQEILHGAVRFSNNPALCNVESIQWRDIVSSDFLSNMSMDFQN
HLGSCQKCDPSCPNGSCWGAGEENCQKLTKIICAQQCSGRCRGKSPSDCCHNQCAAGCTGPRESDCLVCRKFRDEATCKDTCPPLMLYNPT
TYQMDVNPEGKYSFGATCVKKCPRNYVVTDHGSCVRACGADSYEMEEDGVRKCKKCEGPCRKVCNGIGIGEFKDSLSINATNIKHFKNCTS
ISGDLHILPVAFRGDSFTHTPPLDPQELDILKTVKEITGFLLIQAWPENRTDLHAFENLEIIRGRTKQHGQFSLAVVSLNITSLGLRSLKE
ISDGDVIISGNKNLCYANTINWKKLFGTSGQKTKIISNRGENSCKATGQVCHALCSPEGCWGPEPRDCVSCRNVSRGRECVDKCNLLEGEP
REFVENSECIQCHPECLPQAMNITCTGRGPDNCIQCAHYIDGPHCVKTCPAGVMGENNTLVWKYADAGHVCHLCHPNCTYGCTGPGLEGCP
TNGPKIPSIATGMVGALLLLLVVALGIGLFMRRRHIVRKRTLRRLLQERELVEPLTPSGEAPNQALLRILKETEFKKIKVLGSGAFGTVYK
GLWIPEGEKVKIPVAIKELREATSPKANKEILDEAYVMASVDNPHVCRLLGICLTSTVQLITQLMPFGCLLDYVREHKDNIGSQYLLNWCV
QIAKGMNYLEDRRLVHRDLAARNVLVKTPQHVKITDFGLAKLLGAEEKEYHAEGGKVPIKWMALESILHRIYTHQSDVWSYGVTVWELMTF
GSKPYDGIPASEISSILEKGERLPQPPICTIDVYMIMVKCWMIDADSRPKFRELIIEFSKMARDPQRYLVIQGDERMHLPSPTDSNFYRAL
MDEEDMDDVVDADEYLIPQQGFFSSPSTSRTPLLSSLSATSNNSTVACIDRNGLQSCPIKEDSFLQRYSSDPTGALTEDSIDDTFLPVPEY
INQSVPKRPAGSVQNPVYHNQPLNPAPSRDPHYQDPHSTAVGNPEYLNTVQPTCVNSTFDSPAHWAQKGSHQISLDNPDYQQDFFPKEAKP
NGIFKGSTAENAEYLRVAPQSSEFIGA
```

You now must load the protein you wish to investigate from the sequence databases. To make this simple, we copied the sequence into the text above. This is in FASTA format, has a name after the “>” sign in the first line, and is followed by a long sequence in the second line. This is a continuous line now, but it can also be split into separate lines. Copy it into the form.

Now look at the hydrophobicity of your protein using the simple method of Kyte and Doolittle. Below the upload box you see a number of options. These are all physicochemical properties of the amino acids that you can plot along the sequence. Hyphob Kyte Doolittle is the default, so you do not need to do much just hit the submit button, but before that, select the window length of 21. This will give you a smooth curve. The server will generate a long output. First it will show your sequence. Then it will show the numerical values of the hydrophobicities used, this is called a “scale”

Using the scale **Hphob. / Kyte & Doolittle**, the individual values for the 20 amino acids are:

```
Ala:  1.800  Arg: -4.500  Asn: -3.500  Asp: -3.500  Cys:  2.500  Gln: -3.500
Glu: -3.500  Gly: -0.400  His: -3.200  Ile:  4.500  Leu:  3.800  Lys: -3.900
Met:  1.900  Phe:  2.800  Pro: -1.600  Ser: -0.800  Thr: -0.700  Trp: -0.900
Tyr: -1.300  Val:  4.200  : -3.500   : -3.500   : -0.490
```

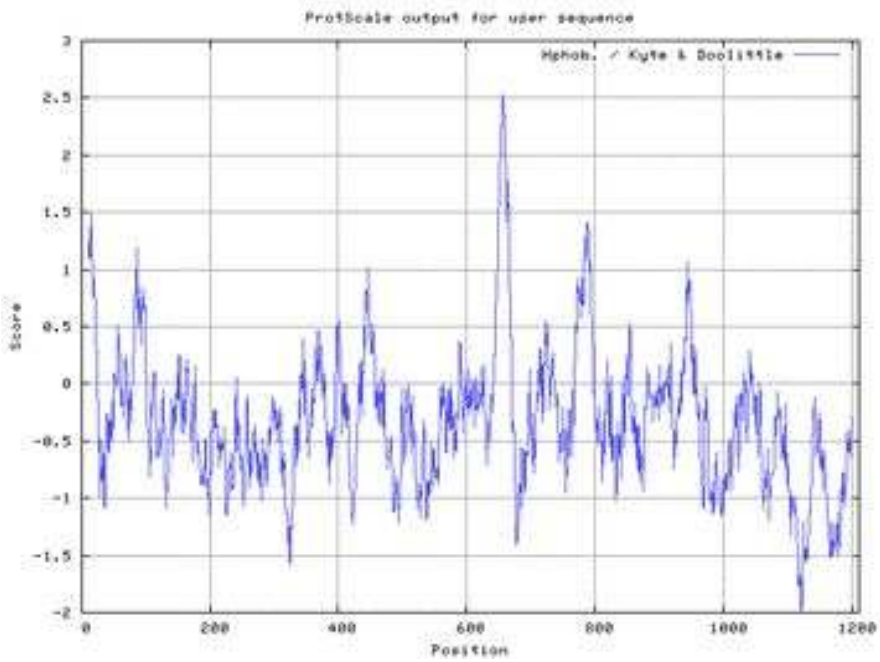
The program will replace these numbers instead of the amino acids of your sequence.
It will also show a weighting scheme that is used for the window of plotting.

Weights for window positions 1,..,21, using **linear weight variation model**:

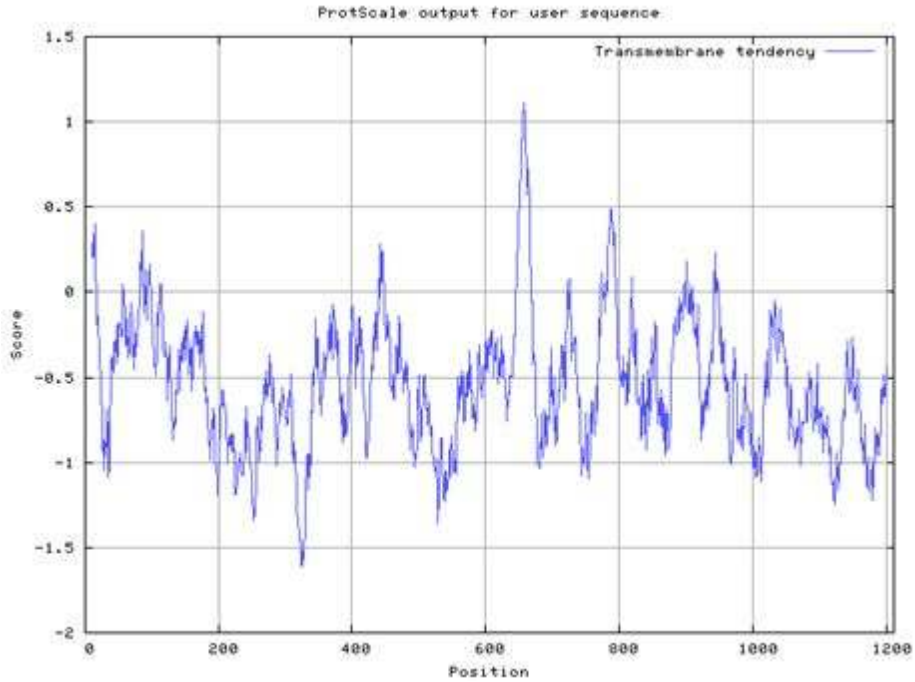
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
edge										center										edge

As you see, all weights are equal, i.e. we will simply plot the average of the values in the middle of the window.

Now comes plot showing the way the hydropathy of the protein varies along its length. It should be clear to you already where the most likely position of the membrane spanning region might be. We put a red arrow there, for the sake of security .

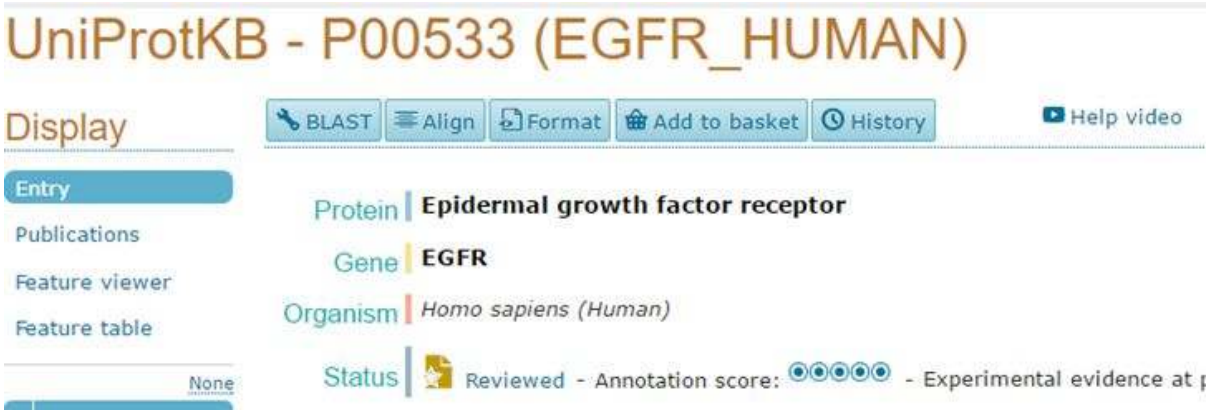


Next, take a look at a statistical property called Transmembrane tendency. This is a scale that represents the frequency of an amino acid found in transmembrane regions, so we expect a similar curve. Go back with the back arrow, your sequence should still be in the loading window. Select Transmembrane tendency, do not forget to set the window size to 21. Hit Submit.



Hopefully you can see that the charge around the most probable membrane spanning region is as expected. From the two plots we can conclude that there is a membrane spanning region around say 610 and 680. We can now check if we are right. In the central protein sequence database, UNIPROT, the transmembrane regions are annotated. Go to www.uniprot.org

In the search window, key the identifier of this protein: *egfr_human*
You will see the only protein corresponding to this name to come up:



Below this you can read the entire CV of this protein. However, we want to see if we were right with our prediction.
Then go to the “Topology” section. Since his is a large record, type control F and key in topology. This is what you will hopefully see:

Topology

Feature key	Position(s)	Length	Description	Graphical view
Topological domain	25 – 645	621	Extracellular Sequence analysis	
Transmembrane	646 – 668	23	Helical Sequence analysis	
Topological domain	669 – 1210	542	Cytoplasmic Sequence analysis	

So, in fact we see a transmembrane segment in the region we located, and one which was confirmed experimentally. This tells us an important lesson: looking at 1D curves is simple but not very accurate.

2. Regular expressions

Many times various patterns exist in text files. In many cases the data is stored as simple, unstructured text file, rather than in XML. Usually, each row represents a single database record, where the attributes are separated by tabulators, semicolons, etc.
These are often non normalized data, with various dependency between the attributes.
In an other typical unstructured data representation the data belonging to one object is scattered through many lines. Usually these are descriptional information (annotations, structural information, etc.) An example for this format is genbank (.gb), pdb (.pdb), etc.
The common in these is that there are “rules” or patterns in the text file, therefore they are machine readable. The patterns are described by so called regular expressions. Some examples are shown in table 1 (for more details see i.e. https://www.math.utah.edu/docs/info/gawk_5.html). The regular expressions are almost the same through various programming languages (i.e. one may be able to use the same regular expression in python and awk).

Table 1. Examples for regular expression

PATTERN	DEFINITION	EXAMPLE
OPERATORS		
\	escape character	\$ - match to $sign
^	matches the beginning of a string	^aa - matches any line starting with two aa
$	matches the ending of string	^aa - matches any line ending with two aa
.	matches any single character	aa.aaa - mathes to aabaaa or aacaaa, but not aabbaaa
*	match to any number of occurrences	a*bb - matches to acbb or abbbb
+	match to at least one occurrences	a+bb - matches to abb or aaabbbb
[....]	set of characters	a[abc]d - matches to aad, abd, acd;
[:alnum:]	matches to alphanumeric characters (i.e. words)	matches to words like cat434
[:digit:]	matches to numbers	i.e. 51464545

(.....)	grouping	It makes the partitioning of the matches possible.
^	not operator	[^abc]+ it matches to every string that does not contain a or b or c.
|	logical OR	A|b - matches to strings beginning with A or b
SPECIAL CHARACTERS		
\\	backslash	Matches to the '\' (it is important, since '\' is an escape character)
\n	newline	matches to the end of line
\t	tab	matches to the tabulator
\r	carriage return	matches to the carriage return
SPECIAL CHARACTERS (more language specific, i.e. available perl, python, c++, etc.)		
\d	matches to numbers	
\D	matches to NOT numbers	
\s	matches to whitespaces (equivalent to [\t\n\r\f\v])	
\S	matches to NOT whitespaces	

3. Simple UNIX utilities

Grep is typically used for filtering lines according to certain criteria (i.e. list the atoms that are in a certain chain, described by an id). The **awk** is a very simply but powerful tool for basic string manipulation. The **sed** is also useful tool for simple text transformation (i.e. replacing a string described by a pattern, to another).
The PDB file contains the 3D structure of a protein. In the PDB file format, each line – so called record – starts with maximum 6-character long record name that indicates the type of information stored in the record. For example: HEADER – the header section, CMPND – the compound name and properties, ATOM – the names and 3d coordinates of the atoms, etc. Most records have strict formats.
For more information on PDB file format, see
http://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html

GREP

Syntax:

grep [OPTIONS] PATTERN [FILE...]

Some options:

-i	ignore case sensitivity
-E	use extended regular expression

Examples: finding all users having nag in its name:

```
ls /home/ | grep -i -E 'nag'
```

SED (for string replacement)

Syntax:

sed ‘s/regexp/replacement/flags’ INPUTFILE

Examples:

Replacing the the usernames starting with ‘nag’ with ‘XX’:

```
ls /home/ | grep -E '^nag' | sed s/^nag/XX/
```

AWK

Syntax:

awk ['program'] [file ...]

where ‘prog’ is the *awk* program; file is the input *file.* The ‘program’ part has a match part and an action part. The columns (or fields) can be accessed directly by their index (i.e. $0 - all column, $1, first, ... $5 fifth column, etc). You can also create variables, loops, condition, etc. Some examples are described here. For detailed description about the progs please see http://www.computerhope.com/unix/uawk.htm.

Examples:

I.e. simple processing of a tabular file */home/olaba/bioinfo/table.tsv*:

Selecting the first 2 columns with awk:

```
awk '{print $1 "\t" $2}' /home/olaba/bioinfo/table.tsv
```

Selecting the records, where the row contains the character ‘D’:

```
awk '/D/ {print $0}' /home/olaba/bioinfo/table.tsv
```

Adding the values in the 4th and 5th columns:

```
awk ' {x=$4; y=$5; print x+y}' /home/olaba/bioinfo/table.tsv
```

Printing the records having an id between 10 and 50.

```
awk ' {if ($1 > 10 && $1<50) print $0}'
/home/olaba/bioinfo/table.tsv
```

Connect to a server (tempus.itk.ppke.hu) via terminal or putty.
Download the 5j8v.pdb from **users.itk.ppke.hu/~olaba/**5j8v.pdb using the **wget** command:
`wget users.itk.ppke.hu/~olaba/5j8v.pdb`

For answering all the following questions, use the downloaded PDB file called 5j8v.pdb.

1. How many lines, words and bytes does the 5j8v.pdb PDB file consists of?
2. What is the title of the PDB file? Print it on the console. (TITLE records)
3. How many authors does the PDB file have? Print the line(s) to the console that start(s) with “AUTHOR”.
4. Lines starting with “REMARK 2” provide information about the highest resolution. Print those lines to the console, without the record name.
5. How many CA atoms are there? (ATOM records, atom name is located in the 3rd column.)
6. Print the atoms with serial number between 500 and 550 to the console. (ATOM records, the serial number is located in the 2nd column.)
7. Print all the atoms to a file called 5j8v_atoms.pdb, but rename each C atom to CO.
8. Print the serial number of each atom, and their distance from the origin. Take care that the coordinates cannot be considered according to column number, because in certain lines there is no whitespace between the columns. Use instead the subsrt function, based on the character positions of the x, y and z coordinates in the ATOM records. (Example: subsrt($0,31,8) extracts 8 characters from the whole line starting from 31.) For reference: http://www.wwpdb.org/documentation/file-format-content/format33/sect9.html#ATOM
9. In certain CONECT lines, the “CONECT” string is immediately followed by any number of digits. Print those lines again in a way that there is one whitespace character immediately after the CONECT string.
10. How many atoms belong to the chains A, B, C and D, respectively? (ATOM records; chain ID is in position 22.)