

Bioinformatics
Prof. M. Michael Gromiha
Department of Biotechnology
Indian Institute of Technology, Madras

Lecture – 28a
“awk” Programming I

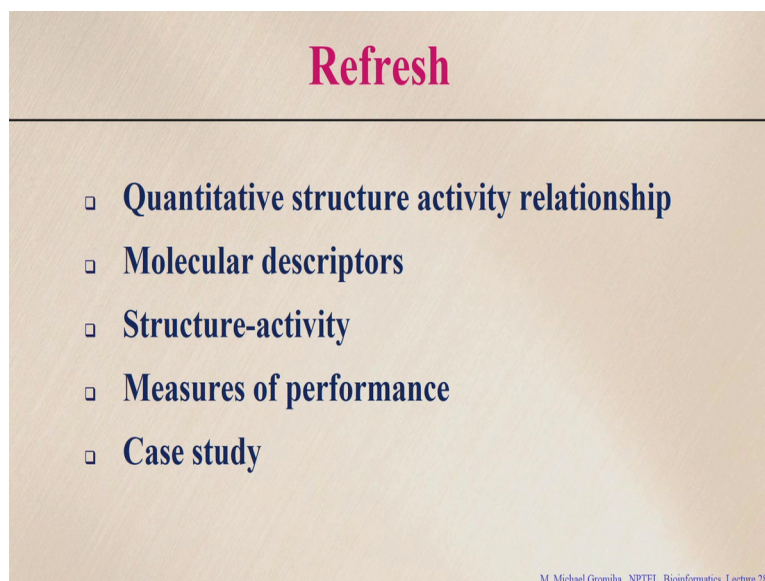
In this lecture, we will mainly discuss about awk programming, it is a script language; how to manipulate the data files or the output obtained from the unique programmings and so on.

So, in the past few classes we focused on different aspects of bioinformatics and the application of bioinformatics including the development of databases and algorithms, and how to deal with the different specific type of algorithms we developed for protein sequences or DNA sequences and so on.

Also we discussed about various properties or features derived from known amino acid sequences or from this 3D structures, and then we discussed about the applications of the different bioinformatics database and tools for predicting the 3D structure of a protein or folding rates or stability as well as protein interactions.

In the last class we discussed about the computer aided drug design and different aspects; such as docking or the pharmacophore modelling or we discussed about the virtual screening and the quantitative structure activity relationship.

(Refer Slide Time: 01:23)



Refresh

- ❑ Quantitative structure activity relationship
- ❑ Molecular descriptors
- ❑ Structure-activity
- ❑ Measures of performance
- ❑ Case study

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

In the quantitative structure activity relationship say QSAR. what are the various aspects we need to consider?

Student: (Refer Time: 01:34) biological activity.

Right mainly the main aspect is the biological activity can be described as a function of the molecular descriptors, like the structural parameters right. So, you can use various types of descriptors right for example, the 1d QSAR like molecular weight or hydrogen bond donor and so on. And they can use the connectivity as well as you can give the information regarding the surrounding residues in the environment.

Then we can relate these parameters with the activity and to measure the performance, whether this we can get the good confidence or not right by means of the correlation. So, depending upon the correlation as well as the cross validations right then we discussed few examples mainly on the Bcl-2 and the EGFR, mutants for the cancer.

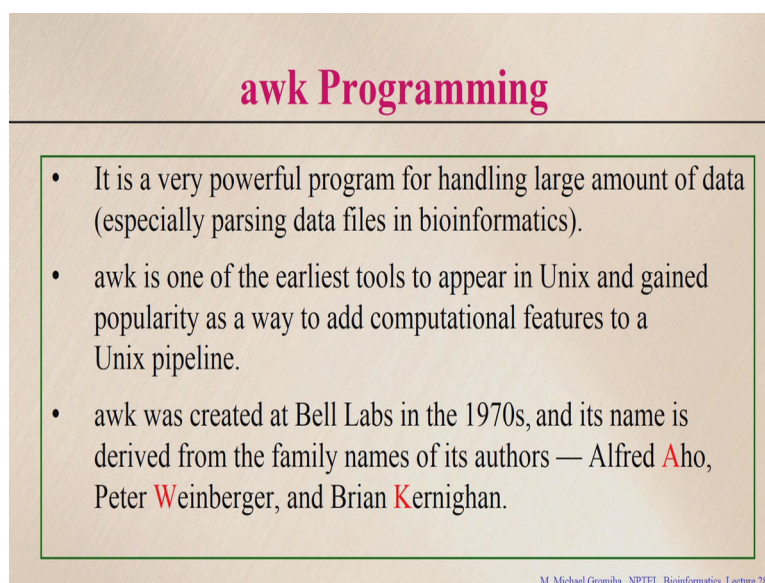
So, we try to see how we can identify new compound by fitting the QSAR models. In this class and the subsequent classes we discuss mainly about the algorithms as well as how we approach a problem, may it for a project or any type of analysis. So awk programming it is a scripting language, it is a kind of pattern matching program right for processing files because when you deal with the several databases or we write program

for analysing any structures or sequences, you will end up with getting a bulk amount of data.

So, doing the manipulating the data it takes time. So, in this case you can use this awk programming to get the desired output right in a very simple and easy way. See it is kind of pattern matching program for processing the files, especially if your databases or any output obtained from this programs. So, it is a data extraction reporting tool that uses a data driven scripting language right, which consists of set of actions to be taken against any data right for the purpose of producing output right.

You have any specific input data, you want to process the data with respect to any action you can give and; what do you want see the output you can format the output as per your desire. So, this is the main applications of this awk programming.

(Refer Slide Time: 03:53)



awk Programming

- It is a very powerful program for handling large amount of data (especially parsing data files in bioinformatics).
- awk is one of the earliest tools to appear in Unix and gained popularity as a way to add computational features to a Unix pipeline.
- awk was created at Bell Labs in the 1970s, and its name is derived from the family names of its authors — Alfred Aho, Peter Weinberger, and Brian Kernighan.

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

It is a simple programming, but it can it is very powerful to handle large amount of data especially parsing several data files in bioinformatics for example, we discussed about various databases right can you remember any of these databases?

Student: UniProt.

Protein sequence database: uniprot, protein structure database

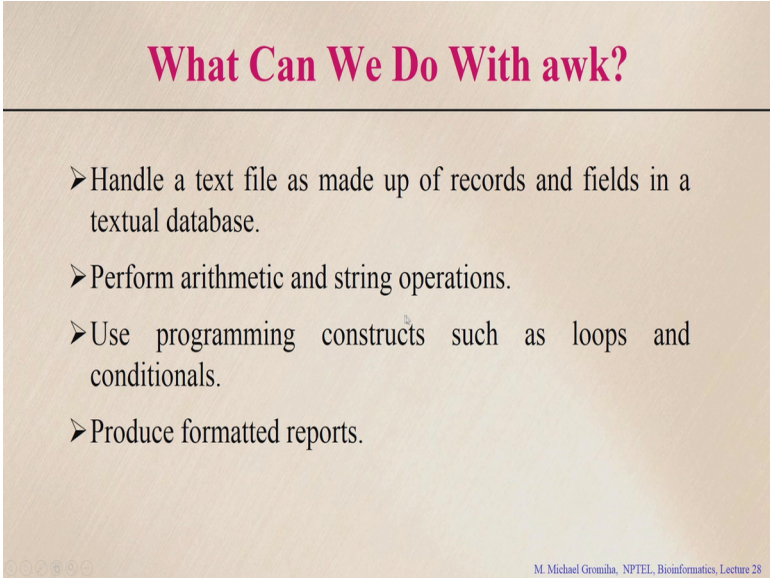
Student: PDB.

PDB right for example, if you are interested only on the coordinates in the PDB file right. It just if you are writing one line command right we can get only the coordinates in the PDB file without looking into these PDB file. If we do not have to open the PDB file just if you know the input name right in one line you can get all the information regarding the coordinates right. This is I will explain how to use this awk program to get the files.

So, it is very powerful program. So, it is one of the earliest tools to appear in Unix right, then we can get the popularity now to use the various other features and also you can use other combine other programs right to make a program using this awk. See if we talk about awk it was created in Bell Labs in 1970s right and the names came a w k that is from the family names of these authors. So, three persons that is Aho right and Weinberger and this Kernighan.

So, with the names they have put a w k. So, this how they made this initially they started the name awk right. So, what can we do with awk?

(Refer Slide Time: 05:19)



What Can We Do With awk?

- Handle a text file as made up of records and fields in a textual database.
- Perform arithmetic and string operations.
- Use programming constructs such as loops and conditionals.
- Produce formatted reports.

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

So, what are the various applications of awk? You can handle a text file right made up of any records and fields any text database, and you can perform the arithmetic and sorting of string operations you can also search with several strings, you can also do arithmetic operations right they have explained some examples how to do this operation.

And you can use a programming this by constructing loops and the conditions, you can also write a program using this awk right. So, let us also you can use to produce the formatted reports right in any specific format.

(Refer Slide Time: 05:52)

What Can We Do With awk?

- With **nawk**, you can also:
- Execute Unix commands from a script.
- Process the results of Unix commands.
- Process command-line arguments more gracefully.
- Work more easily with multiple input streams.
- Perform more powerful string substitutions (**gawk**)

AWK - the original from AT&T
NAWK - A newer, improved version from AT&T
GAWK - The Free Software foundation's version

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

Then there are different types of awk right. The first original form, this is the AWK, the original form right then they developed new versions right one is the NAWK right this is newer and improved versions right. Using this version you can execute unix commands from a script, and you can process the results of unix commands, and also you can do the work more easily with multiple input steps.

Then if you take the GWAK this is another version of this awk this is a free software which is available from this foundations version. So, they are also perform very more powerful string substitutions than the previous versions like a simple awk or this nawk right. So, what is syntax?

(Refer Slide Time: 06:35)

awk One Liners

Syntax

```
awk [options] 'script' var=value file(s)
```

Pattern-action statement

```
awk 'pattern {action}' file name
```

E.g. awk 'NF>1 {print \$1}' abc.dat

Handwritten notes:
 - Above the example: $\$1$ 2 3 4 5
 - A box around 'pattern' with 'input' written above it.
 - A box around '{action}' with 'column' written next to it.
 - A table with columns labeled A, B, C, D, E.
 - An arrow pointing from 'column' to the first column (A).

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

How the program awk works right. So, it is kind of pattern action statement, this is syntax awks you can use some options and this is script you can use, variable equal to value of these files.

So, this is the pattern how to write this syntax for the awk? First you write “awk” this is the programming name “awk” and you need the input file name. So, right here this is your input file name we can put the input; and here the action; action means what you have to do right for example, here we have to do, the program tells we need to print. So, which one we need to print?

First column, the dollar one this, means a column right for example, if this is the A B C D E this is column 1, this is 2, 3, 4, 5. So, you use this dollar symbol to specify the column. So, this is the action means what we need to do that. So, here we have to print only the first column. So, first column means this is the first column you have to print this, and this is a pattern. Pattern means if it satisfies which conditions we need to print we need to do the action right we can use some conditions right for example, if there is no empty line if any line contains some characters or any line contains any numbers.

Any number contains, any line contains the numbers more than 50 right. You can give, find any pattern that. If it satisfies the pattern then do the action right here, you can see just print right this is a input file. So, here input file is abc dot dat. So, it considers the condition NF greater than 1 I will explain what is this right, this will check this condition

in this file abc dot dat and if it satisfies the condition then we will print the first column ok.

This is the syntax or this awk pattern action statement that is the awk. So, I show some of the input files.

(Refer Slide Time: 08:43)

Sample Input Files	
test1.dat	test2.dat
<pre>% Summary reports on aggregation-prone regions % Three types of aggregation-prone regions: % Type1: TANGO score >= 10% % Type2: PAGE Zscore >= 1.96 % Type3: TANGO score >= 5% and PAGE Zscore >= 1 # 1dbuacMS1 3 APPs 41 45 SFK--TLIVA--ENG 2 53 61 QKK--LACFLVATA--NLN 2 89 93 KST--GTLVG--GIS 1 # 1prxacaMS2 5 APPs 29 34 GDS--UGILFS--HFR 3 63 68 NVK--LIALSI--DSV 2 105 112 NRE--LAILLOBL--DPA 4 128 133 TAB--VVFVFG--PRK 4 158 165 ILR--VVISLQLT--A 1 # 1n5sacaMS3 6 APPs 23 27 IAR--VLITA--ATK 2 32 38 TER--VALVAAT--EAT 3 43 48 ATG--FATSVI--MCP 1 72 79 RFG--VYVOICTF--KYE 4 121 127 GFK--LEFFADG--NES 2 164 171 IAG--GNFFIFGD--SGH 0 # 11q6acaMS4 2 APPs 16 22 AAE--VAFAAL--SED 3 49 59 VHG--MLLASLFSGLL--GOQ 0 # 1spvacaMS5 3 APPs 9 18 QGS--ITKLAVDVIV--NAA 1 61 66 TGH--AVITLA--GDL 1 149 155 LPE--QVVFVCY--DEE 5</pre>	<pre>HEADER STONALIND PROTEIN 17-MAR-00 1PC3 TITLE THE CRYSTAL STRUCTURE OF TRANS-ACTIVATION DOMAIN OF THE COMPOUND RESPONSE REGULATOR, SPQAL COMPND 1 CHAIN: A, B, C SOURCE 1 ORGANISM SCIENTIFIC: DEBACILLUS STEAROTHEROPHILUS; 2 ORGANISM TAXID: 1421; 3 EXPRESSION SYSTEM: ESCHERICHIA COLI BL21; 4 EXPRESSION SYSTEM TAGS: 616691; 5 EXPRESSION SYSTEM TAGS: 616691; 6 A 120 AMN LYS PRO LYS AMN LYS ASP ALA SER ILE THR SER ILE 7 A 120 ILE SER ILE ILE GLY VAL PRO ALA SER ILE LYS GLY THR 8 A 120 LYS THR LYS ARG GLY ALA ILE ALA MET VAL THR SER ASP 9 A 120 ILE GLY LYS LYS GLY SER ILE THR VAL VAL LYS THR PRO 10 A 120 ASP ILE ALA LYS LYS THR AMN THR THR ALA SER ARG VAL 11 A 120 GLY ARG ALA ILE ARG SER ALA ILE GLY VAL ALA THR SER 12 A 120 AMN GLY AMN LYS GLY SER ILE SER SER LYS PRO GLY THR 13 A 120 THR VAL SER VAL SER LYS ALA LYS PRO THR AMN SER GLY 14 A 120 PRO ILE ALA MET VAL ALA ASP LYS LYS ARG LYS GLY SER 15 A 120 LYS ALA SER 16 A 120 LYS A 180 GLY A 187 1 17 A 120 ILE A 180 ASP A 178 1 18 A 120 ILE A 178 ILE A 185 5 19 A 120 VAL A 188 AMN A 186 1 20 A 120 THR A 200 ARG A 218 1 21 A 120 ILE A 224 GLY A 229 1 22 A 120 THR A 229 VAL A 234 1 23 A 120 THR A 240 GLY A 255 1 24 A 120 THR B 141 GLY B 157 1 25 A 120 ILE B 162 ASP B 170 1 26 A 120 ILE B 170 ILE B 180 5 27 A 120 VAL B 188 THR B 197 1 28 A 120 THR B 200 GLY B 219 1 29 A 120 ALA B 240 ILE B 254 1 30 A 120 THR A 140 -14.345 40.311 15.200 1.00 41.86 W 31 A 120 LYS A 141 -16.052 40.480 13.250 1.00 39.17 W 32 A 120 CA LYS A 141 -8.497 42.864 12.475 1.00 38.28 C 33 A 120 C LYS A 141 -6.184 41.860 12.471 1.00 37.13 C 34 A 120 O LYS A 141 -7.793 40.866 11.439 1.00 36.08 O 35 A 120 CB LYS A 141 -7.165 41.539 12.000 1.00 38.69 C 36 A 120 CG LYS A 141 -6.899 44.766 11.997 1.00 39.74 C 37 A 120 CD LYS A 141 -6.400 46.860 12.004 1.00 41.23 C 38 A 120 CE LYS A 141 -6.600 46.873 12.073 1.00 42.17 C 39 A 120 HE LYS A 141 -5.900 46.238 12.500 1.00 40.43 W 40 A 120 HE LYS B 141 -12.713 31.145 12.352 1.00 24.71 W 41 A 120 CD LYS B 210 11.114 31.174 13.044 1.00 24.85 C 42 A 120 HE LYS B 210 11.532 33.027 11.744 1.00 23.59 W 43 A 120 HE LYS B 210 11.200 31.705 11.352 1.00 23.41 W 44 A 120 ALA B 211 14.746 32.050 16.235 1.00 56.10 W 45 A 120 CA ALA B 211 16.084 32.634 16.246 1.00 26.12 C 46 A 120 C ALA B 211 17.080 31.560 16.400 1.00 26.17 C 47 A 120 O ALA B 211 18.154 31.442 16.068 1.00 24.46 O</pre>

So, some of them are familiar to you already I showed several times as some of the new files, can you see this test two dot dat what is this?

Student: 3D.

It is kind of a it is a pdb file right. So, you can see the source, compound, sequence, right Helix, Atom and so on right this is a PDB file we can get the protein databank.

Now, in test one dat right here this is the output obtained from a program right we can see the various numbers this is the some scores and some ids right and some blank lines. So, these are the various types of information available in the test one dot, dat right we can use this data for manipulation.

(Refer Slide Time: 09:26)

Sample Input Files

[illegible]

Because this is test three dot dots are you familiar with this output.

Student: (Refer Time: 09:33).

This DSSP output this you obtain from DSSP what is DSSP.

Student: Dictionary of secondary structure of proteins.

Dictionary of secondary structure of proteins right. So, you can see this one. So, here we have the comment lines right, how they identify the hydrogen bonds right and so on right. So, here from this it starts a data from one, here we give the residue name and the chain information here, and here we give the residue right and this is secondary structure, and we can see solvent accessibility right, what is accessibility? So, you know the how far the residue exposed or it is buried so, from this you can see that.

So, from this only we can easily identify the residues, which are at the surface or which are buried. So, because you have to search with this column right for example, if you need the completely buried say less than 5 angstrom square. So, see this column and see where we have less than 5 for example, here here right. So, you can use write one line awk code right to get the residues; residues are here right see here right which has the ASA of less than 5, easily you can see that right.

(Refer Slide Time: 10:46)

Sample Input Files

test4.dat

hptxacaMS2	79	WSK	DINATN	SEE	32.1852	6
1m5sacaMS3	20	IKI	APVLIIT	AAT	11.4172	6
1m5sacaMS3	164	AGG	NFFIFQ	BDQ	96.2088	6
1m5sacaMS3	242	DVN	AVTIV	ING	19.8543	6
1iq6acaMS4	50	GHL	LASLFS	GLL	10.7893	6
1spvacaMS5	147	ALP	EQVFFV	CYD	96.1525	6
1x7dacaMS7	62	ESR	YAFKTV	MOB	58.0513	6
1x7dacaMS7	149	GIE	ETVATD	TDP	47.4865	6
1x7dacaMS7	236	NAR	VFFVTE	POT	74.9182	6
1l1zacaMS9	130	DCY	AALLYI	HAH	48.2344	6
1l1zacaMS9	228	EDP	DVSIYA	APS	25.9341	6
1u1lacaMS10	35	MSI	YQFLIA	VRQ	95.5041	6
1u1lacaMS11	61	HSE	AKGLFL	EPE	10.7389	6
1sxjacaMS12	22	DCV	QLVNFQ	CKE	81.4959	6
1sxjacaMS13	26	LSD	SINIIT	KET	42.2134	6
1sxjacaMS13	111	ESG	FLQFFL	APK	14.4927	6
1rpsacaMS14	15	GRN	FQVEYA	VKA	87.3043	6
1rpsacaMS15	45	EHV	DIIMFL	RET	10.2064	6
1l1jacaMS16	0	T	TIITFC	TGN	15.1024	6
1p1jacaMS17	61	IAS	NDILYN	DEL	93.6116	6
1p1jacaMS17	92	KVA	MDVYTS	ELM	44.0619	6
1m3sacaMS18	76	EEO	DLVIIG	SGS	54.9764	6

test6.dat

X Y Z

ATOM	9	N	LYS	A	141	-9.552	43.465	13.292	1.00	39.17	N
ATOM	10	CA	LYS	A	141	-8.497	42.864	12.775	1.00	38.38	C
ATOM	11	C	LYS	A	141	-8.194	41.362	12.471	1.00	37.33	C
ATOM	12	O	LYS	A	141	-7.793	40.846	11.439	1.00	36.98	O
ATOM	13	CB	LYS	A	141	-7.165	43.539	12.800	1.00	38.89	C
ATOM	14	CG	LYS	A	141	-6.899	44.766	11.997	1.00	39.74	C
ATOM	15	CD	LYS	A	141	-6.400	45.848	12.886	1.00	41.23	C
ATOM	16	CE	LYS	A	141	-5.650	46.873	12.073	1.00	42.17	C
ATOM	17	NZ	LYS	A	141	-5.910	48.238	12.590	1.00	42.63	N
ATOM	1497	ND1	HIS	B	210	12.713	31.145	12.352	1.00	24.71	N
ATOM	1498	CD2	HIS	B	210	13.114	33.174	13.046	1.00	24.85	C
ATOM	1499	CE1	HIS	B	210	13.280	31.795	11.352	1.00	23.41	C
ATOM	1500	NE2	HIS	B	210	13.532	33.027	11.746	1.00	23.59	N
ATOM	1501	N	ALA	B	211	14.746	32.053	16.235	1.00	26.10	N
ATOM	1502	CA	ALA	B	211	16.084	32.634	16.246	1.00	26.32	C
ATOM	1503	C	ALA	B	211	17.083	31.563	16.650	1.00	26.17	C
ATOM	1504	O	ALA	B	211	18.156	31.442	16.068	1.00	26.46	O

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

here I get few more data files, here I give one data, this is output for another program, test four and test 6 again it is a pdb file. So, what is this specific information in test 6?

Student: Atom records.

Atom record this one is atom record we have the coordinates right X Y Z coordinates right this is test 6, this is another text file.

(Refer Slide Time: 11:10)

Sample Input Files

test5.dat

16	11	1	8	5	11	2	10	19	16	0	6	6	5	3	7	9	11	0	3
17	12	0	9	6	12	0	11	20	17	0	7	7	6	4	8	10	12	1	4
10.74	7.38	.67	5.37	3.36	7.38	1.34	6.71	12.75	10.74	.00	4.03	4.03	3.36	2.01	4.70	6.04	7.38	.00	2.01
11.49	8.11	.00	6.08	4.05	8.11	.00	7.43	13.51	11.49	.00	4.73	4.73	4.05	2.70	5.41	6.76	8.11	.68	2.70
14	13	0	11	10	9	3	11	10	21	2	7	11	1	10	8	10	11	2	2
15	14	1	12	11	10	4	12	11	22	0	0	12	2	11	9	11	12	3	3
8.43	7.83	.00	6.63	6.02	5.42	1.81	6.63	6.02	12.65	1.20	4.22	6.63	.60	6.02	4.82	6.02	6.63	1.20	1.20
9.55	8.92	.64	7.64	7.01	6.37	2.55	7.64	7.01	14.01	.00	.00	7.64	1.27	7.01	5.73	7.01	7.64	1.91	1.91
41	10	4	29	16	28	1	26	22	17	6	11	13	7	7	11	20	19	1	8
42	11	5	30	17	29	2	27	23	18	7	12	14	8	8	12	21	20	2	9
13.80	3.37	1.35	9.76	5.39	9.43	.34	8.75	7.41	5.72	2.02	3.70	4.38	2.36	2.36	3.70	6.73	6.40	.34	2.69
14.14	3.70	1.68	10.10	5.72	9.76	.67	9.09	7.74	6.06	2.36	4.04	4.71	2.69	2.69	4.04	7.07	6.73	.67	3.03
18	4	0	8	9	12	2	4	6	17	1	6	5	5	7	9	11	0	1	
19	5	0	9	10	13	3	5	7	18	0	2	7	6	6	8	10	12	1	2
14.29	3.17	.00	6.35	7.14	9.52	1.59	3.17	4.76	13.49	.79	.79	4.76	3.97	3.97	5.56	7.14	8.73	.00	.79
14.18	3.73	.00	6.72	7.46	9.70	2.24	3.73	5.22	13.43	.00	1.49	5.22	4.48	4.48	5.97	7.46	8.96	.75	1.49

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

This is the output obtained from the another program this one, you can also see how to use this test 5 also to do some kind of arithmetic calculations here.

(Refer Slide Time: 11:21)

Sample Input Files

test8.dat

```
139 1a1sa.dsspl
139 1a1sb.dsspl
219 1a2za.dsspl
219 1a2zb.dsspl
350 1a59a.dsspl
350 1a59b.dsspl
385 1a5aa.dsspl
385 1a5ab.dsspl
100 1a6fa.dsspl
100 1a6fb.dsspl
103 1a81a.dsspl
103 1a81b.dsspl
93 1adja.dsspl
93 1adjb.dsspl
153 1ahja.dsspl
153 1ahjb.dsspl
100 1aipa.dsspl
100 1aipb.dsspl
93 1aipc.dsspl
93 1aipd.dsspl
341 1aj8a.dsspl
341 1aj8b.dsspl
164 1amua.dsspl
164 1amub.dsspl
80 1amuc.dsspl
80 1amud.dsspl
83 1amue.dsspl
83 1amuf.dsspl
```

test9.dat

9	N	-9.552	1.00	39.17
10	CA	-8.497	1.00	38.38
11	C	-8.194	1.00	37.33
12	O	-7.793	1.00	36.98
13	CB	-7.165	1.00	38.89
14	CG	-6.899	1.00	39.74
15	CD	-6.400	1.00	41.23

test10.dat

9	N	-9.552	39.17
10	CA	-8.497	38.38
11	C	-8.194	37.33
12	O	-7.793	36.98
13	CB	-7.165	38.89
14	CG	-6.899	39.74
15	CD	-6.400	41.23

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

There is some list of input files this is input register input files. So, this is the number of lines and this is the file names. So, this is input file names.

Here this is the PDB coordinates we have only some information, not the complete information. Then this also this not in format right there are several misalignments in this particular test 9 and test 10. Now I will show you some examples. So, we have several input files and we desire to get the output files right based on some of our requirements right, in this case I will explain few commands using awk how to manipulate the data how to get the information what we need.

(Refer Slide Time: 12:06)

1. Print column 6

```
awk '{print $6}' test4.dat
```

Output

32.1852	lpxxacaNS2	79	USK	DINAYN	SEE	32.1852	6
11.4172	lmsacaNS3	20	IKI	ARVLIT	AAT	11.4172	6
96.2088	lmsacaNS3	164	AGG	NFFIFG	DSQ	96.2088	6
19.8543	lmsacaNS3	242	DVN	AVTEIV	ING	19.8543	6
10.7893	liq6acaNS4	50	GHL	LASLFS	GLL	10.7893	6
96.1525	lspvacaNS5	147	ALP	EQYFV	CYD	96.1525	6
10.7893	lx7dacaNS7	62	KSR	YAFYV	NGW	58.0513	6
96.1525	lx7dacaNS7	149	GIE	EIVAYD	TDP	47.4865	6
58.0513	lx7dacaNS7	236	NAR	VFVEYE	PQT	74.9182	6
47.4865	li1lacaNS8	130	DCY	AALLVI	HAN	48.2344	6
74.9182	li1lacaNS8	228	EDP	DVSIYA	APS	25.9341	6
48.2344	li1lacaNS10	35	NSI	YQFLA	VRQ	95.5041	6
25.9341	li1lacaNS11	61	HKE	ARQLFL	EPE	10.7389	6
95.5041	lsxjacaNS12	22	DCV	QLVNFQ	CKE	81.4939	6
10.7389	lsxjacaNS12	26	LSQ	SINIIT	KET	42.2134	6
81.4939	lsxjacaNS13	111	KSG	FLQFFL	APK	14.4927	6
42.2134	lrypacaNS14	15	GRN	FQVEYA	VKA	87.3043	6
	lpx3acaNS15	45	EHU	DIINFL	REV	10.2064	6
	li1lacaNS16	0	X	TIFVIC	TGN	15.1024	6
	lpljacaNS17	61	IAS	NDILYN	DRI	93.6116	6
	lpljacaNS17	92	KVA	MDEYYS	ELN	44.0619	6
	lmsacaNS18	76	EEG	DLVIIG	SGS	54.9764	6

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

So, first one is, start from the simplest one, we have a file, this is the input file right this is the input file. Now the first question is print column 6, and which one is column 6? 1, 2, 3, 4, 5, 6 this is column 6 we need to print right. So, in this case do we need any pattern? We don't need any pattern because we are not looking for anything, just we want to print the column 6 because it contains some numbers; I want to see the numbers.

So, here you put the awk right and then what is the action want to do. So, there this bracket is important right. So, you have to give the action within quotes. So, you put the print what you want to do is you have print the 6. So, print this is the sixth column. So, you mention dollar 6 right then get the output. What is the output you will get if you work this on this file?

Student: sixth column

Right 6 column, like 32.8, 11.41 this is just you can see this output this you will get, right, that is fine for example, if you want to print two columns.

(Refer Slide Time: 13:29)

2. Print columns 1 and 6
`awk '{print $1,$6}' test4.dat`

3. Print in reverse order (columns 6 and 1)
`awk '{print $6, $1}' test4.dat`

4. Write the results in a file
`awk '{print $1 " " $6}' test4.dat > test1.result`

1prxacaMS2	32.1852
1m5sacaMS3	11.4172
1m5sacaMS3	96.2088
1m5sacaMS3	19.8543
1iq6acaMS4	10.7893
1spvacaMS5	96.1525
1x7dacaMS7	58.0513
1x7dacaMS7	47.4865
1x7dacaMS7	74.9182
1lzlacaMS8	48.2344
1lzlacaMS8	25.9341
1ul1acaMS10	95.5041
1ul1ccaMS11	10.7389
1sxjacaMS12	81.4939
1sxjccaMS13	42.2134

test 1. result

32.1852 1prxaca..

\$

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

For example dollar one and dollar 6 right we need the; because this is only the numbers. So, I want to get the codes plus the numbers right. So, this case if you want to, how to do? Start awk and in the quotes you have to give the action, you have to print dollar 1 as well as dollar 6 right then you give the input file name, this is the input file name.

Then you will see this is the first column and this is the sixth column right you get the answers. If you see the input file this is your input file this is the 1, this is 6 and you get one and 6. The question is it possible to write in reverse order, right I do not want to get this 1 and 6, I want to write the results in 6 and 1, because I want to deal with 6 first.

Is it possible to write the data in the reverse order it is possible how to do this in this case? You can see the order. So, here this will follow the same order what you give an action. If you ask to write a first column we will write first column, 6 column 6 column if you want to take several columns what are the order you give accordingly you can write. See if you want to first and 6 means you will get first and 6. If you write 6 and 1 what will be the result? Yeah this will be like this.

32.1852 then 1 p r x a c a like this right you can get the answer. Now here this is the screen. So, if you write just the input files right and if you give the awk command you can see the data on the screen right for example, if we want to write the answer in a file right, in this case what to do right we have to give the greater than symbol; that means,

the result obtained from this command will go to this file right this is the extra we give this is the same.

Awk print 1 comma 6 right and you need to specify where you want to write. So, you could put the greater than symbol, this is the output will go to the file what do you specify right. So, you can see test one dot result right. If you do it where can we see it on the screen now or only on the file it is on the file.

So, if you do like this then the next will come with this dollar symbol for the next command. Because if you open this file right then you can see if you open test one dot result there you can see this one comma 6 means this result right. This is stored in test one dot result right that is fine. So, for any operations you can either see your results on the screen or you can write in a file.

So, I have a question if you have several empty lines for example.

(Refer Slide Time: 16:33)

5. Delete empty lines

NF: number of fields

awk 'NF>1 {print}' test1.dat

```
% Summary reports on aggregation-prone regions
% Three types of aggregation-prone regions:
% Type1: TANGO score >= 10%
% Type2: PAGE Zscore >= 1.96
% Type3: TANGO score >= 5% and PAGE Zscore >= 1

# 1dbuacaMS1      3 APFs
41 45           SFK--TLLVA--ENG  2
53 61           QWK--LACFVLATA--NLN  2
89 93           KST--GVLVG--GIS  1

# 1prxacaMS2      5 APFs
29 34           GDS--WGILFS--HPR  3
63 68           NVK--LIALSI--DSV  2
105 112         NRE--LAILLOHL--DPA  4
128 133         TAR--VVFVFG--PDK  4
158 165         ILR--VVISLQLT--A  1

# 1mSeacaMS3      6 APFs
23 27           IAR--VLITA--ATK  2
32 38           TER--WALVAAT--EAT  3
43 48           ATG--FATSVI--MCP  1
72 79           RFG--VTVQICTF--KYE  4
121 127         GFK--LKFADG--MES  2
164 171         IAG--GNFFIFGD--SQM  0
```

If this is the data there are several empty lines right, in this case how to eliminate the empty lines. So, I need the file without empty lines right. So, here we use this NF, NF is number of fields. what is the meaning of number of fields?

Student: Number of columns.

Number of columns. How many fields we have. So, if you want to eliminate the free lines. So, at least this contain only one column right there is one field. So, in this case you put NF is greater than 1 right, you can see the codes it started from this your conditions till your action right. So, here the condition is NF is greater than 1. What is the meaning of NF greater than 1?

Student: number of fields is

The number of fields will be more than 1 that should be more if it is more than one then you print right, but here you will get the result in the screen or in a file.

Student: Screen.

On the screen right you can see it on the screen. So, here if you do if you work on this one right then you will get this result; these empty lines are eliminated right all others are fine. Type three these empty lines eliminated and this empty line eliminated right this is fine. So, now, next question is I have a file.

(Refer Slide Time: 17:52)

6. Number each line
FNR: file line number
`awk '{print FNR $0}' test4.dat`

entire line

\$1

1prxacaNS2	79	WSK	DINAYN	SEE	32.1852	6
1m5sacaNS3	20	IKI	ARVLIT	AAT	11.4172	6
1m5sacaNS3	164	AGG	NFFIFG	DSQ	96.2088	6
1m5sacaNS3	242	DVN	AVTEIV	ING	19.8543	6
1iq6acaNS4	50	GHL	LASLFS	GLL	10.7893	6
1spvacaNS5	147	ALP	EQUFTV	CYD	96.1525	6
1x7dacaNS7	62	ESR	YAFKTV	NGH	58.0513	6
1x7dacaNS7	149	GIE	EIVATD	TDP	47.4865	6
1x7dacaNS7	236	NAR	VFVEYE	PQT	74.9182	6
1lzlacaNS8	130	DCY	AALLYI	HAH	48.2344	6
1lzlacaNS8	228	EDP	DVSLYA	APS	25.9341	6
1ul1acaNS10	35	MSI	YQFLIA	VRQ	95.5041	6
1ul1acaNS11	61	HZE	AHQFLA	EPE	10.7389	6

1prxacaNS2	79	WSK	DINAYN	SEE	32.1852	6
21m5sacaNS3	20	IKI	ARVLIT	AAT	11.4172	6
31m5sacaNS3	164	AGG	NFFIFG	DSQ	96.2088	6
41m5sacaNS3	242	DVN	AVTEIV	ING	19.8543	6
511iq6acaNS4	50	GHL	LASLFS	GLL	10.7893	6
61spvacaNS5	147	ALP	EQUFTV	CYD	96.1525	6
71x7dacaNS7	62	ESR	YAFKTV	NGH	58.0513	6
81x7dacaNS7	149	GIE	EIVATD	TDP	47.4865	6
91x7dacaNS7	236	NAR	VFVEYE	PQT	74.9182	6
1011zlacaNS8	130	DCY	AALLYI	HAH	48.2344	6

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

I want to do number in each lines right. So, in this case, what we have to do. So, here we have this command FNR, FNR gives file line number right for each fields for each line. This will give a number.

See if you want to number all the, consider all the columns and print everything right. So, print FNR, FNR means give the file number and dollar 0. What is the meaning of dollar 0?

Student: Entire line.

Entire line.

So, if you write dollar one what will happen.

Student: First columns.

This will give only the first column if it is dollar 0 so; that means, the entire. So, you write the entire line right. So, in this case if you see this one whether in the output, this is the input right. So, get the output. So, did you see it added the line the line numbers?

Student: Yes.

Yes right we can see the line numbers, but this is merged with this first column right. So, in this case it is very difficult to see where the numbers are. So, in this case it is better if you add a tab or add some space between these two, because there are no space. So, it is merged right. So, in this case what can we do is it possible to add a tab in between these two? Yes it is we can add it you can have this command.

(Refer Slide Time: 19:24)

7. Number each line with tab

`awk '{print FNR "\t" $0}'`
test4.dat
\$0 denotes all fields

1	1pxvacaMS2	79	WSK	DINATN	SEE	32.1852	6
2	1m5sacaMS3	20	IKI	ARVLIT	AAT	11.4172	6
3	1m5sacaMS3	164	AGG	NFFIFG	DSQ	96.2088	6
4	1m5sacaMS3	242	DVN	AVTEIV	ING	19.8543	6
5	1iq6acaMS4	50	GML	LASLFS	GLL	10.7893	6
6	1spvacaMS5	147	ALP	EQVTFV	CTD	96.1525	6
7	1x7dacaMS7	62	ESR	YAFKTV	NGH	58.0513	6
8	1x7dacaMS7	149	GIE	EIVATD	TDP	47.4865	6
9	1x7dacaMS7	236	NAR	VFVEYE	PQT	74.9182	6
10	1lz1acaMS8	130	DCY	AALLYI	HAH	48.2344	6

8. Count lines (similar to `wc -l`)

`awk 'END {print NR}' test4.dat`

NR: line number

22

The same print FNR dollar 0 test four dot dat right, but here we added another information that is tab, that is backslash t right you put in the quotes right. If you do this then it will add lines right with a tab. Now can you see this this result and the previous result. Here there is no space, but here you can see this is a tab right. So, we can also give space right when we are adding numbers ok.

Now, the question is you can also count lines, if you take many documents in a file. So, how many lines ? In the Unix we give the command what is the command we use to count the line number of lines?

Student: wc -l.

wc minus l just we give wc minus l this will give you the lines right. So, here also you can give awk, you go from to the end. What is NR?

Student: line number

Number of lines. So, they are line numbers right.

Student: Number of records.

So, you can.

Student: Number of records.

Yeah number of records. So, you can see NR right. So, how many lines say if you go to this particular file test 4 dot dat the input file. So, you give the line number. So, count the all the lines. So, it will give total 22 records say 22 lines in this particular file. So, it is similar to wc minus l in the Unix right.

(Refer Slide Time: 20:55)

9. Print the last field of each line
`awk '{print $NF}' test6.dat`

ATOM	9	H	LYS	A	141	-9.552	43.465	13.292	1.00	39.17	N
ATOM	10	CA	LYS	A	141	-8.497	42.864	12.475	1.00	38.38	C
ATOM	11	C	LYS	A	141	-8.194	41.362	12.471	1.00	37.33	C
ATOM	12	O	LYS	A	141	-7.793	40.846	11.439	1.00	36.98	O
ATOM	13	CB	LYS	A	141	-7.165	43.539	12.800	1.00	36.89	C
ATOM	14	CG	LYS	A	141	-6.899	44.765	11.997	1.00	39.74	C
ATOM	15	CD	LYS	A	141	-6.400	45.848	12.886	1.00	41.23	C
ATOM	16	CE	LYS	A	141	-5.650	46.873	12.073	1.00	42.17	C
ATOM	17	NE	LYS	A	141	-5.910	48.238	12.590	1.00	42.63	N
ATOM	1497	ND1	HIS	D	210	12.713	31.143	12.332	1.00	14.71	N
ATOM	1498	CD2	HIS	D	210	13.114	33.174	13.046	1.00	24.85	C
ATOM	1499	CE1	HIS	D	210	13.280	31.795	11.352	1.00	23.41	C
ATOM	1500	NE2	HIS	D	210	13.532	33.027	11.746	1.00	23.59	N
ATOM	1501	H	ALA	D	211	14.746	32.053	16.235	1.00	26.10	N
ATOM	1502	CA	ALA	D	211	16.084	32.634	16.246	1.00	26.32	C
ATOM	1503	C	ALA	D	211	17.083	31.563	16.450	1.00	26.17	C
ATOM	1504	O	ALA	D	211	18.156	31.442	16.048	1.00	26.46	O

10. Print the last field of last line
`awk 'END {print $NF}' test6.dat`

O

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

So, now here the next question is print the last field of each line. So, we have several fields. If you know this is you require 5 field or 6 field, just you write print dollar number of fields 5 or field 6 right. If it is a last field then you put dollar NF right, In this file, so, what is the last field? This is the last field right then if you do this you will get the result. So, just it is simple awk command, awk print dollar NF this input file test 6 dot dat. So, we get the last field of each line.

Then also it is also possible the last field of last line right. Just now we saw about the total number of lines. So, you put the end right when you give this line number. So, same you combine. So, this is the last field this is the end and they print dollar NF right this is the last field or last line that is this is O. So, you can see the result over here right.

So, likewise you can manipulate right any data file right as we require, then I will show some of the important information where we can really manipulate the output files.

(Refer Slide Time: 22:06)

11. Print every line, where the value of the 6th field is more than 50
 awk '\$6 > 50 {print}' test4.dat

1 2 3 4 5 6

hprvacaNS2	79	USK	DINAYN	SEE	32.1852	6
1m5sacaNS3	20	IKI	ARVLIT	AAT	11.4172	6
1m5sacaNS3	164	AGG	NFFIFG	DSQ	96.2088	6
1m5sacaNS3	242	DVN	AVTEIV	ING	19.8543	6
1iq6acaNS4	50	GWL	LASLFS	GLL	10.7893	6
1spvacaNS5	147	ALP	EQVFFV	CYD	96.1525	6
1x7dacaNS7	62	KSR	YAFKTV	NGH	58.0513	6
1x7dacaNS7	149	GIE	EIVAYD	TDP	47.4865	6
1x7dacaNS7	236	NAR	VFVEYE	PQT	74.9182	6
1l1tacaNS8	130	DCV	AALLVI	HAH	48.2344	6
1l1tacaNS8	228	EDP	DVSIYA	APS	25.9341	6
1ul1acaNS10	35	HSI	YQFLIA	VRQ	95.5041	6
1ul1acaNS10	61	HKE	AHQFL	EPE	10.7389	6
1sxjacaNS12	22	DCV	QLVNFQ	CKE	81.4939	6
1sxjacaNS12	26	LSD	SINIIT	KET	42.2134	6
1sxjacaNS13	111	ESG	FLQFFL	APK	14.4927	6
1rypacaNS14	15	GRN	FQVEYA	VKA	87.3043	6
1yx3acaNS15	45	EHV	DINFL	REV	10.2064	6
1lj1acaNS16	0	X	TIFFIC	TGN	15.1024	6
1p1jacaNS17	61	IAS	NDILYN	DKL	93.6116	6
1p1jacaNS17	92	KVA	NDEYTS	ELN	44.0619	6
1m3sacaNS18	76	EEG	DLVIIG	SGS	54.9764	6

1m5sacaNS3	164	AGG	NFFIFG	DSQ	96.2088	6
1spvacaNS5	147	ALP	EQVFFV	CYD	96.1525	6
1x7dacaNS7	62	KSR	YAFKTV	NGH	58.0513	6
1x7dacaNS7	236	NAR	VFVEYE	PQT	74.9182	6
1ul1acaNS10	35	HSI	YQFLIA	VRQ	95.5041	6
1sxjacaNS12	22	DCV	QLVNFQ	CKE	81.4939	6
1rypacaNS14	15	GRN	FQVEYA	VKA	87.3043	6
1p1jacaNS17	61	IAS	NDILYN	DKL	93.6116	6
1m3sacaNS18	76	EEG	DLVIIG	SGS	54.9764	6

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

So, here I put one condition print every line where the value of the 6 field is more than 50 right. So, how to do this? So, you can see the 6 field right, I have 6 field, more than 50, it is a more than 50 then you print if I just print.

What it will do? It will print, the entire, either print dollar 0 or you just give print. So, you can give the entire things if it is more than 50. So, if it is more than 50 no right this is more than 50 dollar this is 1, 2, 3, 4, 5, 6 right this is more than second one more than 50 no third one yes then again this no no this 6 right.

So, you can see the same way. So, here you can see this is 96 is present here and again the next stand is present here, and 58 is present here. So, you can see the values right which is more than 50. Now if I have this example I have the, I give the example. So, I want to see how many amino acids are in the buried region. So, what you will do?

If you have any pdbid you can if want to see the number of residues which are in the buried. So, you can run the DSSP. So, you will get this output from this is it possible to see get the number of residues, which are completely buried for example, 0 without touching any of these file is it possible to do we can do, what we have to do first?

Student: Select only records.

Yeah select only the records which number 1, 2, 3, 4, 5, 6, 7, 8, 9 right take this field and then see you have to write the records if this record is more than 50 and second what we

have to do? We can do the number of fields likewise if we make two commands and you can easily get the number of residues, which are at the buried because if you take the condition of 0 right. So, these are the two commands we have to do.

First we need to right check the file right if the column 9 right which is more than 50 right, and then this will give you the residues which are having the value of 0. Now if it is buried means you have to put that less than 1 or equal to 0 right and then you can use these command right this is the command to see the NF minus 1 right. So, then we will see how many records.

Now is it possible to identify how many residues which are in the exposed region with a condition of 75 angstrom square.

Student: Yes.

Right. So, first what we have to do? We have to put this column number which is greater than 75 then you print right and you print this two output, right this output will be the input of this the next command. So, here you can give the command, this is the output file you give here and this will give you a total number of records right. So, in this it will tell you without looking into the PDB file or DSSP file right we can see how many residues right, which are buried how many residues are which are in the exposed right.

Just first take the PDB file, run DSSP right then give the awk command. So, with field which contains the value of more than 75, then you find the line number, totally how many lines right. Without manipulating directly anything, we can easily get all the information. In this case also you can see how many alanines which are exposed, which are in the buried right likewise you can have different files we can manipulate the files, easily you can get all the informations right without editing any of these files.

(Refer Slide Time: 26:30)

12. print the lines starting from 15

```
awk 'NR > 14 {print}' test3.dat
```

[illegible]

So, the next question print the lines starting from 15 right. This is your DSSP file right. So, here these are your command the information right. So, actual result starts from here right. If you want only these data as the input to the another program because this they do not need all these unnecessary data. So, in this case if you want to have the lines right which is starting from 15.

So, here you can, what is NR.

Student: Records.

Records right. So, in this case line number this is more than 14, because we need from 15 right then you print from this input file. So, then you will eliminate all these things 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14.

So, then you can start this is 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 right. So, then we can get from this line right this line.

So, you can give any number according to the number you give. So, you will get the output so; that means, you delete the line numbers until 14 lines right this is you can check or you can get these a files. So, you can do with any numbers right 15 or 16 whatever you like. So, you can do that.

(Refer Slide Time: 28:00)

13. Separate with field separator
`awk -F "-" '{print $2}' test1.dat`

Summary reports on aggregation-prone regions
 Three types of aggregation-prone regions:
 Type1: TANGO score >= 10s
 Type2: PAGE Zscore >= 1.96
 Type3: TANGO score >= 5s and PAGE Zscore >= 1

idonacaM1 3 APs
 41 45 SFF--TLLVA--ENG 2
 53 61 QEK--LACFVLATA--MLN 2
 89 93 EST--GYLVG--GIS 1

ipcvacaM2 5 APs
 29 34 QDS--WILFS--BPP 3
 43 68 HVE--LIALSI--DVP 2
 105 112 HVE--LAILLOHL--PPA 4
 128 133 TAB--VVPVFG--PKR 4
 158 165 ILR--VVISLQLT--A 1

in5acaM3 6 APs
 23 27 TAB--VLITA--ATX 2
 32 38 TER--NALVAAT--EAT 3
 43 48 AVS--FATVVI--RCP 1
 72 79 RPK--VTVQICTF--KYE 4
 121 127 GFK--LAFVADG--RES 2
 164 171 LAG--GNFFIFGD--QDE 0

liq5acaM4 2 APs
 16 22 ALE--VAFAAL--SED 3
 49 59 VBO--MLLASLFSGLL--QQQ 0

ipcvacaM5 3 APs
 9 18 QDS--LITLAVVTV--HIA 1
 61 66 TDR--AVITLA--GDL 1
 149 155 LPE--QVTFVCT--DEE 5

TLLVA
 LACFVLATA
 GYLVG
 WILFS
 LIALSI
 LAILLOHL
 VVPVFG
 VVISLQLT
 VLITA
 NALVAAT
 FATSVI
 VTVQICTF
 LAFVADG
 GNFFIFGD
 VAFAAL
 MLLASLFSGLL

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

So, then the next question is you can separate the field separator for example, if there is any field separator this is the field separator, here you can see this one whether we can eliminate you can do this separate these things. So, if this is the case right, dollar 2. So, you can print the dollar two like test one dot dat. So, this is the test one right now here you can see the field separator, this is the 'F' for field separator and we can get this to here right the middle ones, separate these two files right.

(Refer Slide Time: 28:29)

14. Print alternate lines
`awk 'NR%2' test8.dat`

15. Omit the third line
`awk 'NR%3' test8.dat`

139 1a1sa.dssp1
 219 1a2za.dssp1
 350 1a59a.dssp1
 385 1a5aa.dssp1
 100 1a6fa.dssp1
 103 1a8la.dssp1
 93 1adja.dssp1
 153 1ahja.dssp1

139 1a1sa.dssp1
 139 1a1sb.dssp1
 219 1a2za.dssp1
 219 1a2zb.dssp1
 350 1a59a.dssp1
 350 1a59b.dssp1
 385 1a5aa.dssp1
 385 1a5ab.dssp1
 100 1a6fa.dssp1
 100 1a6fb.dssp1
 103 1a8la.dssp1
 103 1a8lb.dssp1
 93 1adja.dssp1
 93 1adjb.dssp1
 153 1ahja.dssp1
 153 1ahjb.dssp1
 100 1aipa.dssp1
 100 1aipb.dssp1
 93 1aipc.dssp1
 93 1aipd.dssp1
 341 1aj8a.dssp1
 341 1aj8b.dssp1
 164 1amua.dssp1
 164 1amub.dssp1
 80 1amuc.dssp1
 80 1amud.dssp1
 83 1amue.dssp1
 83 1amuf.dssp1

139 1a1sa.dssp1
 139 1a1sb.dssp1
 219 1a2zb.dssp1
 350 1a59a.dssp1
 385 1a5aa.dssp1
 385 1a5ab.dssp1
 100 1a6fb.dssp1
 103 1a8la.dssp1
 93 1adja.dssp1
 93 1adjb.dssp1
 153 1ahjb.dssp1

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

In this case you can also do other lot of other manipulations for example; you can print the alternate lines.

In this case if you see this input file. So, this is the you can see the same for example, one a one s, one a one s, one a two z, one a two z. So, each these two are the same right, only difference is a and b. So, I want to see the unique wants, in this case I like to delete the alternate lines because that input is necessary for the another program. So, if you want to print alternate lines, you can omit the second one using this equation like the NR dollar this is NR percent two; that means, it will cut the second one and finally, you get this result right

So, this is here this is omitted, this is here, this is omitted this here right you can get that one. So, if you want to go to the third line right then you put the 3. So, first two is present here and the third one this 2 one 9 is omitted and the again you start from this two right and the third one is omitted here you can see the omitted ones right you can do for any lines if you want to omit.

Now, if you want to do any substitutions for example, if you want any mutations. So, you can it is possible to substitute any text for example, using this command this sub. So, what do you want to substitute?

(Refer Slide Time: 29:57)

16. Substitute LYS by ARG

awk '{sub(/LYS/,"ARG"); print}' test6.dat

ATOM	9	N	LYS	A	141	-9.552	43.465	13.292	1.00	39.17	N
ATOM	10	CA	LYS	A	141	-8.497	42.864	12.475	1.00	38.38	C
ATOM	11	C	LYS	A	141	-8.194	41.362	12.471	1.00	37.33	C
ATOM	12	O	LYS	A	141	-7.793	40.846	11.439	1.00	36.98	O
ATOM	13	CB	LYS	A	141	-7.165	43.539	12.800	1.00	38.89	C
ATOM	14	CG	LYS	A	141	-6.899	44.766	11.997	1.00	39.74	C
ATOM	15	CD	LYS	A	141	-6.400	45.848	12.886	1.00	41.23	C
ATOM	16	CE	LYS	A	141	-5.650	46.873	12.073	1.00	42.17	C
ATOM	17	NZ	LYS	A	141	-5.910	48.238	12.590	1.00	42.63	N
ATOM	1497	ND1	HIS	B	210	12.713	31.145	12.352	1.00	24.71	N
ATOM	1498	CD2	HIS	B	210	13.114	33.174	13.046	1.00	24.85	C
ATOM	1499	CE1	HIS	B	210	13.280	31.795	11.352	1.00	23.41	C

ATOM	9	N	ARG	A	141	-9.552	43.465	13.292	1.00	39.17	N
ATOM	10	CA	ARG	A	141	-8.497	42.864	12.475	1.00	38.38	C
ATOM	11	C	ARG	A	141	-8.194	41.362	12.471	1.00	37.33	C
ATOM	12	O	ARG	A	141	-7.793	40.846	11.439	1.00	36.98	O
ATOM	13	CB	ARG	A	141	-7.165	43.539	12.800	1.00	38.89	C
ATOM	14	CG	ARG	A	141	-6.899	44.766	11.997	1.00	39.74	C
ATOM	15	CD	ARG	A	141	-6.400	45.848	12.886	1.00	41.23	C
ATOM	16	CE	ARG	A	141	-5.650	46.873	12.073	1.00	42.17	C
ATOM	17	NZ	ARG	A	141	-5.910	48.238	12.590	1.00	42.63	N
ATOM	1497	ND1	HIS	B	210	12.713	31.145	12.352	1.00	24.71	N
ATOM	1498	CD2	HIS	B	210	13.114	33.174	13.046	1.00	24.85	C
ATOM	1499	CE1	HIS	B	210	13.280	31.795	11.352	1.00	23.41	C
ATOM	1500	NE2	HIS	B	210	13.532	33.027	11.746	1.00	23.59	N

Here you want to substitute LYS to ARG right how to substitute this lysine to arginine? So, here is a record for the lysine right. So, the change it to arginine by giving this command that is awk you get this is the pattern action statement sub is your substitution from LYS. So, in this case you have to put the slash, wherever the exactly you get the LYS you will change to arginine.

So, here if you see this one LYS into arginine this one example, but you can do the manipulation for the different files at the different places or if you want to make any changes, like kind of this replace command ok.

(Refer Slide Time: 30:40)

11.4 Built-in Variables		
Version	Variable	Description
awk	FILENAME	Current filename
	FS	Field separator (a space)
	NF	Number of fields in current record
	NR ✓	Number of the current record
	OFMT	Output format for numbers ("% . 6g") and for conversion to string
	OFS	Output field separator (a space)
	ORS	Output record separator (a newline)
	RS	Record separator (a newline)
	\$0 →	Entire input record
	\$n	nth field in current record; fields are separated by FS
	ARGC	Number of arguments on command line
	ARGV	An array containing the command-line arguments, indexed from 0 to ARGC - 1
	CONVFMT	String conversion format for numbers ("% . 6g") (POSIX)
	ENVIRON	An associative array of environment variables
nawk	FNR	Like NR, but relative to the current file
	LENGTH	Length of the string matched by match() function
	RSTART	First position in the string matched by match() function
	SUBSEP	Separator character for array subscripts ("\"034")

Unix in a nutshell, Stever et al. O'REILLY

M. Michael Gromiha, NPTEL, Bioinformatics, Lecture 28

Now, in this awk there is several built in function available. So, I explained few of them. So, there are various built in functions available in awk. So, for example, FS is the field separator, I discussed earlier what is NF.

Student: Number of fields.

Number of fields in the current record, NR number of current record likewise there are various other options like record separator, this is dollar 0 is for the entire input record. Dollar n is the nth field in the current record right. So, likewise you can see this FNR like the NR, but related to the current file you see any of these variables and get the description right and accordingly you can manipulate these files using these awk command.

So, again this some of this available in awk and you can see some more information right we can also use nawk right for a more information. So, now, we go to next step now first one is we discussed about the substitution and you are printing the some lines some columns right and doing some sort of a file manipulations.