Basic Data processing in UNIX

grep, sort, counting

More resources for these topics

The topics in today's lecture are thorougly covered in Chapter 7 of the *Bioinformatics Data Skills* book.

I would suggest you sign up for a free trial and read this chapter online if you aren't going to buy the book at this time (though buy the book if you are going to be doing bioinformatics in any serious capacity).

Chapter 7: Insecting and Manipulating Text Data with Unix Tools

The data files in this lecture are in data and also in the https://github.com/biodataprog/2018_programming-intro/tree/master/data.

Sorting with sort

```
$ sort file.txt > file.sorted.txt
```

Type of sorting:

1. -d/--dictionary_order : consider only blanks & alphanumeric characters
2. -n/--numeric-sort : compare according to string numerical value
3. -f/--ignore-case : upper/lower doesn't matter
4. -r/--reverse : reverse the order
5. -k : specify the key positions to sort by

Let's try some examples : sort alphabetical

```
$ sort data/numbers_only.dat | head -n 10
10
10
12
25
30
34
39
42
49
49
```

Let's try some examples : sort numeric

```
$ sort -n data/numbers_only.dat | head -n 10
7
7
7
10
10
12
25
30
34
39
```

More sort

```
$ sort -r -n numbers_floating.dat  | head -n 10
49.6859213710444
49.6454233452118
49.5141651980655
49.2878027550901
48.5007601226085
45.15231125553
43.0392927946809
41.8950131857132
41.7844270115886
39.63172550297467
```

Unique with uniq

`uniq` collapses adjacent lines in a file which are identical

```
$ sort -n data/numbers_only.dat | uniq | head -n 10
7
10
12
25
30
34
39
42
49
51
```

uniq - count the number of occurrences

```
$ sort -n data/numbers_only.dat | uniq -c | head -n 8
3 7
2 10
1 12
1 25
1 30
1 34
1 39
1 42
```

Hey let's sort this list so we know the numbers that show up most frequently

```
$ sort -n data/numbers_only.dat | uniq -c | sort -r -n | head -n 8
3 7
2 86
2 83
2 66
2 64
2 58
2 49
2 10
```

Sort multicolumn data

```
$ head -n 10 data/rice_random_exons.bed
Chr7	21408673	21408826
Chr9	16031526	16031938
Chr11	4762531	4762595
Chr8	54040	54193
Chr10	19815475	19815747
Chr3	16171331	16172869
Chr10	2077882	2077938
Chr3	20517604	20517936
Chr10	9777446	9777527
Chr2	4967096	4967246
$ sort -k1,1 -k2,2n data/rice_random_exons.bed | head -n 5
Chr1	12152	12435
Chr1	98088	98558
Chr1	216884	217664
Chr1	291398	291534
Chr1	338180	338310
$ sort -k1,1 -k2,2n data/rice_random_exons.bed | tail -n 5
Chr9	22369724	22369776
Chr9	22508926	22509014
Chr9	22753347	22753458
Chr9	22924316	22924424
ChrSy	136034	136323
```

Column processing with cut

cut - subselect and print certain columns from a file

```
YAR060C Chr_I 100.00 336 0    0    336    1   217148    217483    8.6e-83    298.8
YAR060C Chr_I 64.00  325 95  22    330   14   198385    198695    4.1e-18    84.0
YAR060C Chr_I 74.07  108 25   3    110    6   211012    211119    2.1e-10    58.4
YAR060C Chr_I 97.02  336 8    2    1    336    14799    15132    1.3e-77    281.6
YAR060C Chr_I 72.48  109 25   5    6    110    20974    21081    2.3e-10    58.2
YAR061W Chr_I 100.00 204 0    0    1    204   218131    218334    3.4e-54    203.1
YAR061W Chr_I 70.62  194 57   0    1    194   203400    203593    6.5e-23    99.2
YAR061W Chr_I 94.61  204 7    4    204    1    13951    14150    5e-48    182.6
YAR061W Chr_I 67.88  193 62   0    194    2    27770    27962    3.9e-20    90.0
YAL030W Chr_I 100.00 252 0    0    103  354    87502    87753    2.5e-55    207.7
```

Query sequence names

```
$ cut -f1 data/yeast_orfs-to-chr1.FASTA.tab | head -n 10
YAR060C
YAR060C
YAR060C
YAR060C
YAR060C
YAR061W
YAR061W
YAR061W
```

Cut examples

Get the Hit sequence names

```
$ cut -f2 data/yeast_orfs-to-chr1.FASTA.tab | head -n 5
Chr_I
Chr_I
Chr_I
Chr_I
Chr_I
```

Get the Query name and Percent Identity

```
$ cut -f1,3 data/yeast_orfs-to-chr1.FASTA.tab | head -n 5
YAR060C    100.00
YAR060C    64.00
YAR060C    74.07
YAR060C    97.02
YAR060C    72.48
YAR061W    100.00
YAR061W    70.62
YAR061W    94.61
YAR061W    67.88
YAL030W    100.00
YAL030W    98.15
```

Cut two columns out, and run sort to sort on

```
$ sort -k3,1nr data/yeast_orfs-to-chr1.FASTA.tab | cut -f1,3 |  head -n 5
HRA1	100.00
YAL001C	100.00
YAL002W	100.00
YAL003W	100.00
YAL003W	100.00

$ sort -k9,1n data/yeast_orfs-to-chr1.FASTA.tab |  head -n 5
YAL069W	100.00	335	649
YAL068W-A	100.00	538	792
YAL068C	100.00	1807	2169
YAR020C	79.76	2008	2169
YAL067W-A	100.00	2480	2707
```

Column combining with paste

(note this is a useful tool but this example is kind of made up!)

```
$ cut -f1,3,4 data/yeast_orfs-to-chr1.FASTA.tab > first_cols.tab
$ cut -f1,7 data/yeast_orfs-to-chr1.FASTA.tab > second_cols.tab
$ paste first_cols.tab second_cols.tab | head -n 5
YAL027W 100.00  786     YAL027W 1
tL(CAA)A        100.00  44      tL(CAA)A        39
tL(CAA)A        100.00  38      tL(CAA)A        1
YAL028W 100.00  1587    YAL028W 1
YAL029C 100.00  4416    YAL029C 4416
```

Simple column filtering with awk

awk is another programming language. It has a very simple syntax.

It is really useful for column delimited data as well

```
$ awk '{print $1}' blast.out.tab   # print out the first column of a file

$ awk -F, '{print $1,$2}' data/random_exons.csv | head -n 3
Chr5 27781790
Chr11 14656670
Chr3 14560358
```

Awk is a programming language too

Can do math or other operations with awk like build up conditional filtering

```
$ awk -F, '{print $1,$2,$3, $3 - $2}' data/random_exons.csv |
head -n 3
Chr5 27781790 27781888 98
Chr11 14656670 14656778 108
Chr3 14560358 14560608 250

# restrict to features larger than 100 bp
$  awk -F, '$3 - $2 > 100 {print $1,$2,$3,$3 - $2}' data/random_exons.csv |
sort -k4,1nr | head -n 3
Chr9 399276 402077 2801
Chr11 3528895 3530426 1531
Chr11 16238576 16239304 728
```

Powerful searching with grep

Tools for finding pattern matches in text data. Operates line-by-line and reports the lines that match a particular pattern. Some cmdline arguments

1. -i -- case insensitive
2. -n -- report line the match is on
3. -c -- simple count the number of matches
4. -v -- report lines which DON'T match the pattern

```
$ grep YAL data/yeast_orfs-to-chr1.FASTA.tab
YAL027W    Chr_I    100.00    786     0     0     1     786     94688     95473
YAL028W    Chr_I    100.00    1587     0     0     1     1587     92901     94487
YAL029C    Chr_I    100.00    4416     0     0     4416     1     87856     92271
YAL064W-B    Chr_I    100.00    381     0     0     1     381     12047     12427
$ grep -c YAL data/yeast_orfs-to-chr1.FASTA.tab
 114
$ grep -v -c YAL data/yeast_orfs-to-chr1.FASTA.tab
138
$ grep -n tL data/yeast_orfs-to-chr1.FASTA.tab
2:tL(CAA)A    Chr_I    100.00    44     0     0     39     82     181205     1812
3:tL(CAA)A    Chr_I    100.00    38     0     0     1     38     181135     18117
```

Regular expressions

Grep can also match patterns not just exact strings

- . - matches everything
- \. - matches literally a period (remember this if you want to match a period!)
- ? - one character match
- [] - can match anything in the brackets

There is a more extended grep pattern match using Perl regular expressions with the -E options

```
$ perl -E '/[0-9]+[A-Z]+/' datafile.txt
```

Putting things together

```
$ more celegans_gene_names.txt
aagr-1
aagr-2
aagr-3
aak-1
aak-2
aakb-1
aakb-2
aakg-1
aakg-2
aakg-4
aakg-5
aap-1
aars-2
```

Putting things together - what are sizes of C.elegans named

genes families

```
$ cut -d\- -f1 data/celegans_gene_names.txt | sort | uniq -c | \
  sort -nr | head -n 10
196 srh
174 clec
168 nhr
141 fbxa
133 str
123 col
68 srw
66 unc
66 fbxb
64 ugt
```

Putting things together - How many hits are 100

```
$ cut -d\- -f1 data/celegans_gene_names.txt | sort | uniq -c | \
  sort -nr | head -n 10
196 srh
174 clec
168 nhr
141 fbxa
133 str
123 col
68 srw
66 unc
66 fbxb
64 ugt
```

Comparing what is different with diff

diff is a tool used to compare the content of two files and report the differences. This is a line-by-line comparison (not within a line).

Can deal with insertions/deletions.

```
$ grep -v "\." celegans_gene_names.txt > celegans_gene_names.no_dot.txt
$ diff  celegans_gene_names.txt celegans_gene_names.no_dot.txt
185,188d184
< ant-1.1
< ant-1.2
< ant-1.3
< ant-1.4
213,214d208
< arf-1.1
< arf-1.2
287,288d280
< atg-4.1
< atg-4.2
476,478d467
< cdc-25.1
< cdc-25.3
< cdc-25.4
```