2.1.2.4.3

Command-Line Data Analysis and Reporting – Session III

- Reza's challenge
- prompt tools

Extracting non-overlapping n-mers

- last time we saw how to extract non-overlapping 7-mers from first 1 Mb of a sequence file

```
grep –v “>” seq.fa | tr –d “\n” | fold –w 1000 | head -1000 | tr –d “\n” | fold –w 7
```

GATCACAGGTCTATCACCCTATTAACCACTCACGGGAGCTCTCCATGCAT

- how about overlapping 7-mers

GATCACAGGTCTATCACCCTATTAACCACTCACGGGAGCTCTCCATGCAT

Extracting non-overlapping n-mers

- the problem can be rephrased, instead

```
extract overlapping 7-mers from a string, s
```

- cast it as 6 smaller problems which we can solve

```
extract non-overlapping 7-mers from a substring s(1:n)
extract non-overlapping 7-mers from a substring s(2:n)
extract non-overlapping 7-mers from a substring s(3:n)
extract non-overlapping 7-mers from a substring s(4:n)
extract non-overlapping 7-mers from a substring s(5:n)
extract non-overlapping 7-mers from a substring s(6:n)
extract non-overlapping 7-mers from a substring s(2:n)
```

Extracting non-overlapping n-mers

- creating substring s(m:n) done with cut

```
cut –c m- seq.fa
```

- extracting 7-mers from this string

```
grep –v “>” seq.fa | tr –d “\n” | cut –c m- | fold –w 1000 | head -1000 | tr –d “\n” | fold –w 7
```

- we need to let m run over 1...6
  - need a loop
  - xargs is the command-line loop maker

xargs in one slide

- xargs is arcane but very useful
  - reads a list of arguments from stdin
  - arguments separated by white space
  - executes a command, once for each input argument
    - by default the command is **/bin/echo**
  - flexible
    - –t tells you what it's doing
    - -I tells xargs to use newline as argument separator
    - -iSTRING replaces STRING with argument
- you can construct complex commands by making xargs run bash –c COMMAND
  - STRING in COMMAND will be replaced by arguments read by xargs

28.17

```
# args.txt
1
2
3
4 5 6

cat args.txt | xargs -t -l
/bin/echo 1
1
/bin/echo 2
2
/bin/echo 3
3
/bin/echo 4 5 6
4 5 6

cat args.txt| xargs -t -l -i ls {}.txt
ls 1.txt
ls: 1.txt: No such file or directory
ls 2.txt
ls: 2.txt: No such file or directory
ls 3.txt
ls: 3.txt: No such file or directory
ls 4 5 6.txt
ls: 4 5 6.txt: No such file or directory

cat args.txt | xargs -t -l -i bash -c "echo 'Hello from {}.'"
bash -c echo "Hello from 1."
Hello from 1.
bash -c echo "Hello from 2."
Hello from 2.
bash -c echo "Hello from 3."
Hello from 3.
bash -c echo "Hello from 4 5 6."
Hello from 4 5 6.
```

Extracting non-overlapping n-mers

```
>alphabet
abcdefghijklmnopqrstuvwxyz0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ)!@#$%^&*(
```

- we'll extract all 7-mers from a mock alphabet fasta file

```
grep –v “>” alphabet.fa |
tr –d “\n” | cut –c 1- |
fold –w 7
abcdefg
hijklmn
opqrstu
vwxyz01
2345678
9ABCDEF
GHIJKLM
NOPQRST
UVWXYZ)
!@#$%^&
*(
```

```
grep -v ">" alphabet.fa |
tr -d "\n" | cut -c 2- |
fold -w 7
bcdefgh
ijklmno
pqrstuv
wxyz012
3456789
ABCDEFG
HIJKLMN
OPQRSTU
VWXYZ)!
@#$%^&*
(
```

. . .

```
grep -v ">" alphabet.fa |
tr -d "\n" | cut -c 6- |
fold -w 7
fghijkl
mnopqrs
tuvwxyz
0123456
789ABCD
EFGHIJK
LMNOPQR
STUVWXY
Z)!@#$%
^&*(
```

- create a loop with xargs

```
echo -e "1\n2\n3\n4\n5\n6"
  | xargs -l -i bash -c 'grep -v ">" alphabet.fa | tr -d "\n" | cut -c {}- | fold -w 7'
  | sort
```

Extracting non-overlapping n-mers

- the 7-mer from the last line in the folded file isn't always a 7-mer
  - can be shorter if cut –c m- produced a file whose length isn't a multiple of 7
- filter through egrep ".{7}"
  - selects lines with 7 characters
- final command is

```
echo -e "1\n2\n3\n4\n5\n6"
  | xargs -l -i bash -c
    'grep -v ">" alphabet.fa | tr -d "\n" | cut -c {}- | fold -w 7'
  | sort
  | egrep ".{7}"
```

```
!@#$%^&
#$%^&*(
$%^&*(
%^&*(
(
*(
0123456
2345678
3456789
456789A
56789AB
6789ABC
789ABCD
9ABCDEF
@#$%^&*
ABCDEFG
BCDEFGH
CDEFGHI
DEFGHIJ
EFGHIJK
GHIJKLM
HIJKLMN
IJKLMNO
JKLMNOP
KLMNOPQ
LMNOPQR
NOPQRST
OPQRSTU
PQRSTUV
QRSTUVW
RSTUVWX
STUVWXY
UVWXYZ)
VWXYZ)!
WXYZ)!@
XYZ)!@#
YZ)!@#$
Z)!@#$%
^&*(
```

```
abcdefg
bcdefgh
cdefghi
defghij
efghijk
fghijkl
hijklmn
ijklmno
jklmnop
klmnopq
lmnopqr
mnopqrs
opqrstu
pqrstuv
qrstuvw
rstuvwx
stuvwxy
tuvwxyz
vwxyz01
wxyz012
xyz0123
yz01234
z012345
```

Perl Prompt Tools

- collection of Perl scripts that extend/add to existing command line tools
  - http://gin.bcgsc.ca/Members/martink/Documents/System Utilities/prompttools/view

- addband
- addwell
- collapsedata
- column
- cumul
- cumulcoverage
- digestvector
- enzyme
- extract
- fields
- histogram
- matrix
- mergecoordinates
- sample
- shrinkwrap
- stats
- sums
- swapcol
- tagfield
- unsplit
- well
- window

Perl Prompt Tools

- installed in /usr/local/ubin
- SCRIPT –h
  - usage information
- SCRIPT –man
  - read man page
- field numbering is 0-indexed

```
> ./column -h

Usage:
      # extract a single column
      cat data.txt | column -c 1 [-1]
      column -c 1 data.txt

      # extract multiple columns
      column -c 1,2,3 data.txt
      column -c 3,1,2 data.txt
      column -c 1-3,4,5 data.txt
      column -c 4,1-3,5 data.txt
      column -c "5-)" data.txt
      column -c "(-8,10" data.txt

      # delete columns
      column -delete -c 4,1-3,5 data.txt

      # cX is equivalent to column -c X
      ln -s column c1
      c1 data.txt

      # c0-c10 may be preinstalled on your system
      c0 data.txt
      c1 data.txt
      ...
      c10 data.txt
```

Cleaning House with **shrinkwrap**

- files come as space delimited, tab delimited, whatever delimited
- shrinkwrap replaces/collapses all whitespace with DELIM
  - by default, DELIM is a space

```
#data.txt
chr1        1           616         1           F           AP006221.1  36116       36731       -
chr1        617         167280      2           F           AL627309.15 241         166904      +
chr1        167281      217280      3           N           50000       clone       no          #

shrinkwrap data.txt
chr1 1 616 1 F AP006221.1 36116 36731 –
chr1 617 167280 2 F AL627309.15 241 166904 +
chr1 167281 217280 3 N 50000 clone no # Unfinished_sequence

shrinkwrap –delim : data.txt
chr1:1:616:1:F:AP006221.1:36116:36731:-
chr1:617:167280:2:F:AL627309.15:241:166904:+
chr1:167281:217280:3:N:50000:clone:no:#:Unfinished_sequence
```

Manipulating Columns with **column**

- less verbose than cut to extract a single column
  - if symlinked to cN, column will extract Nth column

```
c0 file.txt    vs   cut –d” “ –f 1 file.txt
```

- supports closed and open ranges
  - 1, 1-5, 5-), (-3

```
column –c “1,2,6-)” file.txt
```

- delete columns

```
column –delete “1,2,6-)” file.txt
```

Manipulating Columns with **swapcol**

- manipulates order of columns in a file
  - swap columns
  - roll columns

```
#data.txt
0 1 2 3 4 5 6 7 8 9

swapcol –c 0,2 data.txt
2 1 0 3 4 5 6 7 8 9

swapcol –c 5 data.txt
5 1 2 3 4 0 6 7 8 9

swapcol –r 1 data.txt
9 0 1 2 3 4 5 6 7 8

swapcol –r 2 data.txt
8 9 0 1 2 3 4 5 6 7

swapcol -r -2 data.txt
2 3 4 5 6 7 8 9 0 1
```

Application

- extract lines with 8th column starting with “13”
  - make 8th column first with swapcol
  - grep with ^13
  - swapcol back to original order

```
#data.txt
chr1        1           616         1           F           AP006221.1  36116       36731       -
chr1        617         167280      2           F           AL627309.15 241         166904      +
chr1        167281      217280      3           N           50000       clone       no          #

cat data.txt | shrinkwrap | swapcol -c 7 | grep ^13 | swapcol -c 7
chr1 1038213 1167191 12 F AL390719.47 2001 130979 +
chr1 1925617 2056500 22 F AL391845.49 2001 132884 +
chr1 3443871 3572010 41 F AL513320.30 2002 130141 +
chr1 3572011 3708951 42 F AL136528.11 1 136941 -
chr1 4487300 4618626 52 F Z98747.1 1 131327 - 6
```

- swap last two columns in a file without knowing number of columns
  - roll +2, swap 0/1, then roll -2

```
cat data.txt | shrinkwrap | swapcol -r 2 | swapcol | swapcol -r -2
```

Application

- reverse all abutting 7-mers in a sequence file

```
>alphabet
abcdefghijklmnopqrstuvwxyz0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ)!@#$%^&*(
```

- what's going on
  - make 7mers
  - add space after each character
  - swap columns 0/6, 1/5 and 2/4 (reverses 7 mer)
  - remove newlines and spaces

```
grep -v ">" alphabet.fa |
  tr -d "\n" | fold -w 7 | egrep ".{7}" |
  sed 's/\(.\)/\1 /g' | shrinkwrap |
  swapcol -c 0,6 | swapcol -c 1,5  swapcol -c 2,4 |
  tr -d "\n" | sed 's/ //g‘

gfedcbanmlkjihutsrqpo10zyxwv8765432FEDCBA9MLKJIHGTSRQPON)ZYXWVU&^%$#@!
```

```
a b c d e f g
h i j k l m n
o p q r s t u
v w x y z 0 1
2 3 4 5 6 7 8
9 A B C D E F
G H I J K L M
N O P Q R S T
U V W X Y Z )
! @ # $ % ^ &


g f e d c b a
n m l k j i h
u t s r q p o
1 0 z y x w v
8 7 6 5 4 3 2
F E D C B A 9
M L K J I H G
T S R Q P O N
) Z Y X W V U
& ^ % $ # @ !
```

Extracting lines with **extract**

- grep tests an entire line with a regular expression
  - burdensome to apply a regex to a specific field
  - v difficult to apply a simple numerical test to a field (e.g. field3 > 5)

- extract applies a test to each line
  - _N replaced by value of column N
  - returns lines that pass or fail (-f) the text

```
#data.txt
chr1        1          616        1          F          AP006221.1     36116       36731       -
chr1        617        167280     2          F          AL627309.15    241         166904      +
chr1        167281     217280     3          N          50000          clone       no          #
```

```
> cat data.txt | extract -t "_7 > 180000"
chr1 852348 1038212 11 F AL645608.29 2001 187865 +
chr1 6770942 6975335 80 F AL590128.11 1 204394 +

> cat data.txt | extract -t "abs(_1 - 5e6) < 1e6 && _7 > 1e5"
chr1 4093705 4232977 49 F AL611933.30 2001 141273 +
chr1 4232978 4390136 50 F AL591916.8 2001 159159 +
chr1 4487300 4618626 52 F Z98747.1 1 131327 -
. . .
```

Identifying file contents with **fields**

- for files with a large number of fields, it hurts the eyes to figure out which column your data is in
  - quick - which column is the accession number in?

```
#data.txt
chr1        1          616        1          F          AP006221.1     36116       36731       -
chr1        617        167280     2          F          AL627309.15    241         166904      +
chr1        167281     217280     3          N          50000          clone       no          #
```

- fields takes the first line, splits up the fields by whitespace and reports each field on a numbered line
  - use -1 for 1-indexing

```
> fields data.txt
0 chr1
1 1
2 616
3 1
4 F
5 AP006221.1
6 36116
7 36731
8 -
```

```
> fields -1 data.txt
1 chr1
2 1
3 616
4 1
5 F
6 AP006221.1
7 36116
8 36731
9 -
```

Descriptive statistics with **stats**

- what is the average contribution to the sequence from an accession in the region 5-10 Mb?

```
#data.txt
chr1        1           616         1           F           AP006221.1    36116       36731       -
chr1        617         167280      2           F           AL627309.15   241         166904      +
chr1        167281      217280      3           N           50000         clone       no          #
```

```
extract -t "_1 > 5e6 && _2 < 6e6" data.txt | extract -fail -t "_4 eq 'N'" | c7
38618
118994
19766
74045
22519
86535
107587
88598
38529
72610
100302

extract -t "_1 > 5e6 && _2 < 6e6" data.txt | extract -fail -t "_4 eq 'N'" | c7 | stats
n 11 mean 69827.545 median 74045.000 mode 0.000 stddev 34823.4095 min 19766.000 max 118994.000
p01 0.000 p05 0.000 p10 22519.000 p16 22519.000000 p84 107587.000 p90 107587.000 p95 118994.000
p99 118994.000
```

- returns avg/median/mode, stdev/min/max, and percentile values at 1, 5, 10, 16, 84, 90, 95, 99%

Application

- what is the average size of clones aligned by end sequence?

```
#bes.txt
D2512K09 CTD-2512K9 9 78570317 78728221
D2512K10 CTD-2512K10 10 63853366 63952303
D2512K11 CTD-2512K11 3 56788975 57000624
D2512K12 CTD-2512K12 7 77009069 77131301
D2512K13 CTD-2512K13 20 30389236 30590735
...

> cat bes.txt | awk '{print $5-$4+1}' | stats
n 201063 mean 148064.263 median 157776.000 mode 210895.000 stddev 43148.5652 min 25001.000 max
349332.000 p01 30238.000 p05 61505.000 p10 89306.000 p16 102939.000000 p84 185958.000 p90
194082.000 p95 205555.000 p99 228901.000
```

Adding with **sums**

- **sums** computes sum of columns
- what is the total size of clones aligned by end sequence?

```
#bes.txt
D2512K09 CTD-2512K9 9 78570317 78728221
D2512K10 CTD-2512K10 10 63853366 63952303
D2512K11 CTD-2512K11 3 56788975 57000624
D2512K12 CTD-2512K12 7 77009069 77131301
D2512K13 CTD-2512K13 20 30389236 30590735
...

> cat bes.txt | awk '{print $5-$4+1}' | sums
29770244964
# 29.8 Gb = 10.5X
```

```
#nums.txt
1
2
3
4 5 6

sums nums.txt
10 5 6
```

Random data sets with **sample**

- you can create a random subset of a file with sample
  - sample each line, reporting it with a certain probability
  - -r PROB will print, on average, 1 line out of 1/0.00005 (1 in 2,000)

```
> sample -r 0.00005 ./bacend.parsed.txt
D2502M11 CTD-2502M11 X 85862968 86028861
D3247C07 CTD-3247C7 21 16791275 17009235
M2012I08 CTD-2012I8 2 19540783 19702289
M2012I15 CTD-2012I15 6 23056788 23175152
M2163J06 CTD-2163J6 13 82085160 82175765
N0016D10 RP11-16D10 2 58882086 59036111
N0142H21 RP11-142H21 1 146449864 146642906
N0153C15 RP11-153C15 5 22145127 22317162
N0349I02 RP11-349I2 11 106677214 106846264
N0521E07 RP11-521E7 13 23756421 23920317

» cat /dev/zero | fold -w 1 | head -5 |
    xargs -l bash -c
    "sample -r 0.00005 ./bacend.parsed.txt | awk '{print \$5-\$4+1}' | stats | column -col 1,3"
16 130561.188
8 117181.625
6 168703.833
4 160159.250
10 162199.600
```

2.1.2.4.3

Command-Line Data Analysis and Reporting – Session III

- many more Perl prompt tools next time