2.1.2.4.4

Command-Line Data Analysis and Reporting – Session IV

- prompt tools

Perl Prompt Tools

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

we saw these last time

- extract/delete columns with column
  - col –delete –c 1,2,5 file.txt
- return lines based on complex booleans
  - extract –t "_1 > 5 && _2 < 10" file.txt
  - extract –fail –t "abs(_3 – 10) < 2" file.txt
- enumerate field numbers in a file
  - fields file.txt
- randomly sample lines from a file
  - sample –r 0.01 file.txt
- remove tabs and collapse spaces
  - shrinkwrap file.txt
- obtain descriptive statistics on a column
  - col –c 5 file.txt | stats
- obtain sum of columns
  - col –c 5 file.txt | sums
- swap/rotate column order
  - swapcol –r -1 file.txt
  - swapcol –c 2,5 file.txt

addband – annotate coordinates with cytogenetic bands

- cytogeneticists frequently use bands to identify regions, not coordinates
  - if you send a cytogeneticist coordinates, he'll probably want bands to go with them
  - by default the band associated with chromosome in –chrcol and position –startcol (or chrcol+1) is shown
  - if you specify the –endcol, you'll get all bands that overlap with the coordinate

```
#file.txt
object1 1 119993574 120022777
object8 3 115004140 118096960
object12 4 107475177 127547875
object16 5 119495561 159600067
object18 6 117866946 127941155

> addband –karyo ~martink/work/ucsc/hg17/karyotype.txt –chrcol 1
object1 1 119993574 120022777 p12
object8 3 115004140 118096960 q13.31
object12 4 107475177 127547875 q24
object16 5 119495561 159600067 q23.1
object18 6 117866946 127941155 q22.1

> addband –karyo ~martink/work/ucsc/hg17/karyotype.txt –chrcol 1 –endcol 3
object1 1 119993574 120022777 p12
object8 3 115004140 118096960 q13.31
object12 4 107475177 127547875 q24,q25,q26,q27,q28.1
object16 5 119495561 159600067
q23.1,q23.2,q23.3,q31.1,q31.2,q31.3,q32,q33.1,q33.2,q33.3
object18 6 117866946 127941155 q22.1,q22.2,q22.31,q22.32,q22.33
```

You provide the karyotype file (UCSC, Ensembl) format for the appropriate organism. By default, HG17 karyotype definition is used.

addwell – quickly create rearray lists

- you have a list of clones that you would like the lab to rearray
  - they require the source well and target well for each clone
  - **addwell** adds 96 or 384 well position to each line
  - format output using any of –format 384/96, –col or –row, –space, –nopad, –noplate

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

```
> addwell nums.txt
1 0001A01
2 0001A02
3 0001A03
4 0001A04
5 0001A05
6 0001A06
7 0001A07
8 0001A08
9 0001A09
10 0001A10
11 0001A11
12 0001A12
13 0001B01
14 0001B02
15 0001B03
16 0001B04
17 0001B05
18 0001B06
19 0001B07
20 0001B08
```

```
addwell -format 384 –col nums.txt
1 0001A01
2 0001B01
3 0001C01
4 0001D01
```

```
addwell -space –nopad nums.txt
1 1 A 1
2 1 A 2
3 1 A 3
4 1 A 4
```

```
> addwell -noplate
1 A01
2 A02
3 A03
4 A04
```

well – convert bewteen 96- and 384-well format

- quick, what's the 96 well mapping for P23?
  - umm, ehhh

```
> well P23
P23 H12c
```

- how about D12b converted to 384 well format?
  - if you provide the quadrant, **well** will assume that the input is 96-well format

```
> well D12b
D12b G24
```

well – conversion templates

- if you specify –t and do not supply a well position, **well** returns handy templates

```
      1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23   24
  A A01a A01b A02a A02b A03a A03b A04a A04b A05a A05b A06a A06b A07a A07b A08a A08b A09a A09b A10a A10b A11a A11b A12a A12b
  B A01c A01d A02c A02d A03c A03d A04c A04d A05c A05d A06c A06d A07c A07d A08c A08d A09c A09d A10c A10d A11c A11d A12c A12d
  C B01a B01b B02a B02b B03a B03b B04a B04b B05a B05b B06a B06b B07a B07b B08a B08b B09a B09b B10a B10b B11a B11b B12a B12b
  D B01c B01d B02c B02d B03c B03d B04c B04d B05c B05d B06c B06d B07c B07d B08c B08d B09c B09d B10c B10d B11c B11d B12c B12d
  E C01a C01b C02a C02b C03a C03b C04a C04b C05a C05b C06a C06b C07a C07b C08a C08b C09a C09b C10a C10b C11a C11b C12a C12b
  F C01c C01d C02c C02d C03c C03d C04c C04d C05c C05d C06c C06d C07c C07d C08c C08d C09c C09d C10c C10d C11c C11d C12c C12d
  G D01a D01b D02a D02b D03a D03b D04a D04b D05a D05b D06a D06b D07a D07b D08a D08b D09a D09b D10a D10b D11a D11b D12a D12b
  H D01c D01d D02c D02d D03c D03d D04c D04d D05c D05d D06c D06d D07c D07d D08c D08d D09c D09d D10c D10d D11c D11d D12c D12d
  I E01a E01b E02a E02b E03a E03b E04a E04b E05a E05b E06a E06b E07a E07b E08a E08b E09a E09b E10a E10b E11a E11b E12a E12b
  J E01c E01d E02c E02d E03c E03d E04c E04d E05c E05d E06c E06d E07c E07d E08c E08d E09c E09d E10c E10d E11c E11d E12c E12d
  K F01a F01b F02a F02b F03a F03b F04a F04b F05a F05b F06a F06b F07a F07b F08a F08b F09a F09b F10a F10b F11a F11b F12a F12b
  L F01c F01d F02c F02d F03c F03d F04c F04d F05c F05d F06c F06d F07c F07d F08c F08d F09c F09d F10c F10d F11c F11d F12c F12d
  M G01a G01b G02a G02b G03a G03b G04a G04b G05a G05b G06a G06b G07a G07b G08a G08b G09a G09b G10a G10b G11a G11b G12a G12b
  N G01c G01d G02c G02d G03c G03d G04c G04d G05c G05d G06c G06d G07c G07d G08c G08d G09c G09d G10c G10d G11c G11d G12c G12d
  O H01a H01b H02a H02b H03a H03b H04a H04b H05a H05b H06a H06b H07a H07b H08a H08b H09a H09b H10a H10b H11a H11b H12a H12b
  P H01c H01d H02c H02d H03c H03d H04c H04d H05c H05d H06c H06d H07c H07d H08c H08d H09c H09d H10c H10d H11c H11d H12c H12d

96a -> 384

      1a   2a   3a   4a   5a   6a   7a   8a   9a  10a  11a  12a
 Aa  A01  A03  A05  A07  A09  A11  A13  A15  A17  A19  A21  A23
 Ba  C01  C03  C05  C07  C09  C11  C13  C15  C17  C19  C21  C23
 Ca  E01  E03  E05  E07  E09  E11  E13  E15  E17  E19  E21  E23
 Da  G01  G03  G05  G07  G09  G11  G13  G15  G17  G19  G21  G23
 Ea  I01  I03  I05  I07  I09  I11  I13  I15  I17  I19  I21  I23
 Fa  K01  K03  K05  K07  K09  K11  K13  K15  K17  K19  K21  K23
 Ga  M01  M03  M05  M07  M09  M11  M13  M15  M17  M19  M21  M23
 Ha  O01  O03  O05  O07  O09  O11  O13  O15  O17  O19  O21  O23

. . .

96d -> 384

      1d   2d   3d   4d   5d   6d   7d   8d   9d  10d  11d  12d
 Ad  B02  B04  B06  B08  B10  B12  B14  B16  B18  B20  B22  B24
 Bd  D02  D04  D06  D08  D10  D12  D14  D16  D18  D20  D22  D24
 Cd  F02  F04  F06  F08  F10  F12  F14  F16  F18  F20  F22  F24
 Dd  H02  H04  H06  H08  H10  H12  H14  H16  H18  H20  H22  H24
 Ed  J02  J04  J06  J08  J10  J12  J14  J16  J18  J20  J22  J24
 Fd  L02  L04  L06  L08  L10  L12  L14  L16  L18  L20  L22  L24
 Gd  N02  N04  N06  N08  N10  N12  N14  N16  N18  N20  N22  N24
 Hd  P02  P04  P06  P08  P10  P12  P14  P16  P18  P20  P22  P24
```

unsplit - join lines

- recall that **fold** was used to break up a line into multiple lines
- **unsplit** does the opposite – joins multiple lines together
  - specify the number of lines to glue with –l
  - specify the line separator with –delim (; is default)

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

```
> unsplit -l 5 nums.txt
1;2;3;4;5
6;7;8;9;10
11;12;13;14;15
16;17;18;19;20

> unsplit –l 10 –delim “ “ nums.txt
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
```

- construct a complex command line from individual commands
  - great for making cluster job files

```
# too many small jobs
dostuff –param 1,2
dostuff –param 2,3
. . .
```

```
# 50 calls to dostuff per command – easier on the scheduler
dostuff –param 1,2; dostuff –param 2,3; . . .
dostuff –param 51,52; dostuff –param 52,53; . . .
. . .
```

tagfield – create numberical identifiers for alpha fields

- suppose you have mixed numerical/text data and you want to associate with each distinct field value a unique numerical value
  - cat=>0, cow=>1, horse=>2, etc.

```
#data.txt
5 sheep White house tasty
4 cow White farm tasty
12 horse brown field not_tasty
5 cow white farm tasty
11 sheep white house tasty
3 pig pink farm tasty
2 dog brown house not_tasty
4 sheep white house tasty
8 pig pink farm tasty
2 cat brown house not_tasty
1 horse brown field not_tasty

# alpha->numerical ascending order
> tagfield -f 1 data.txt
5 5 sheep White house tasty
1 4 cow White farm tasty
3 12 horse brown field not_tasty
1 5 cow white farm tasty
5 11 sheep white house tasty
4 3 pig pink farm tasty
2 2 dog brown house not_tasty
5 4 sheep white house tasty
4 8 pig pink farm tasty
0 2 cat brown house not_tasty
3 1 horse brown field not_tasty
```

```
# alpha->numerical descending order
> tagfield -f 1:r data.txt
0 5 sheep White house tasty
4 4 cow White farm tasty
2 12 horse brown field not_tasty
4 5 cow white farm tasty
0 11 sheep white house tasty
1 3 pig pink farm tasty
3 2 dog brown house not_tasty
0 4 sheep white house tasty
1 8 pig pink farm tasty
5 2 cat brown house not_tasty
2 1 horse brown field not_tasty
```

tagfield – cont'd

- you can create tags for multiple fields
  - - f m,n,…
  - use :r to ask that the alpha->num mapping be done in descending order

```
> tagfield -f 1,2,3,4 data.txt
5 0 2 1 5 sheep White house tasty
1 0 0 1 4 cow White farm tasty
3 1 1 0 12 horse brown field not_tasty
1 3 0 1 5 cow white farm tasty
5 3 2 1 11 sheep white house tasty
4 2 0 1 3 pig pink farm tasty
2 1 2 0 2 dog brown house not_tasty
5 3 2 1 4 sheep white house tasty
4 2 0 1 8 pig pink farm tasty
0 1 2 0 2 cat brown house not_tasty
3 1 1 0 1 horse brown field not_tasty
```

```
> tagfield -f 1,2:r,3,4:r data.txt
5 3 2 0 5 sheep White house tasty
1 3 0 0 4 cow White farm tasty
3 2 1 1 12 horse brown field not_tasty
1 0 0 0 5 cow white farm tasty
5 0 2 0 11 sheep white house tasty
4 1 0 0 3 pig pink farm tasty
2 2 2 1 2 dog brown house not_tasty
5 0 2 0 4 sheep white house tasty
4 1 0 0 8 pig pink farm tasty
0 2 2 1 2 cat brown house not_tasty
3 2 1 1 1 horse brown field not_tasty
```

- you can sort entirely numerically on existing numerical fields and mapped text fields
  - don't have to mess around with alpha/num sort combinations

tagfield – cont'd

- mapping case insensitive with –lc

```
> tagfield -f 1,2:r,3,4:r data.txt
5 3 2 0 5 sheep White house tasty
1 3 0 0 4 cow White farm tasty
3 2 1 1 12 horse brown field not_tasty
1 0 0 0 5 cow white farm tasty
5 0 2 0 11 sheep white house tasty
4 1 0 0 3 pig pink farm tasty
2 2 2 1 2 dog brown house not_tasty
5 0 2 0 4 sheep white house tasty
4 1 0 0 8 pig pink farm tasty
0 2 2 1 2 cat brown house not_tasty
3 2 1 1 1 horse brown field not_tasty
```

```
> tagfield -f 1,2:r,3,4:r -lc data.txt
5 0 2 0 5 sheep White house tasty
1 0 0 0 4 cow White farm tasty
3 2 1 1 12 horse brown field not_tasty
1 0 0 0 5 cow white farm tasty
5 0 2 0 11 sheep white house tasty
4 1 0 0 3 pig pink farm tasty
2 2 2 1 2 dog brown house not_tasty
5 0 2 0 4 sheep white house tasty
4 1 0 0 8 pig pink farm tasty
0 2 2 1 2 cat brown house not_tasty
3 2 1 1 1 horse brown field not_tasty
```

- encode lines into numbers
  - number can be interpreted as base N (e.g. base 12, since our biggest number is 12)

```
» tagfield -f 1,2:r,3,4:r -lc data.txt | column -c 0-3 | tr -d " "
5020 # 8664 = 5*12^3+2*12 in base 12
1000
3211
1000
. . .
```

collapse – hashed statistics

- frequently you come across data keyed by another value (numerical or text)

```
#data.txt associates random number 0-999 with random letter (10,000 lines)
b 741
c 53
s 511
a 238
i 9
e 903
j 99
. . .
```

- for each letter (distinct value of a specific field), it would be nice to apply **stats** to all associated numbers
  - this is what **collapsedata** does

```
> collapse data.txt
a col 1 n 350 avg 478.502857142857 med 455 mode 479 sd 283.006138102776 p10 105 p90 887 sum 167476 range 985 min 0 max 985
b col 1 n 413 avg 510.38014527845 med 524 mode 479 sd 276.300303231825 p10 131 p90 878 sum 210787 range 991 min 2 max 993
c col 1 n 398 avg 499.203517587939 med 488.5 mode 473 sd 295.287319053224 p10 103 p90 928 sum 198683 range 992 min 3 max 995
d col 1 n 355 avg 480.935211267606 med 451 mode 123 sd 296.497460997785 p10 105 p90 898 sum 170732 range 991 min 5 max 996
e col 1 n 332 avg 489.686746987952 med 471.5 mode 720 sd 282.05697514091 p10 101 p90 888 sum 162576 range 994 min 1 max 995
f col 1 n 365 avg 518.112328767124 med 521 mode 369 sd 290.112446911673 p10 107 p90 925 sum 189111 range 999 min 0 max 999
. . .
```

collapse – cont'd

- of course there is more!
  - if your data is keyed by a alpha field, then the field acts as a key to a list of values
  - if your data is keyed by a numerical field, then the field could be manipulated before used as a hash key
    - round off for windowed statistics
- as an example, let's use GC fraction computed over 5 kb windows
  - what if you want average GC over 100 kb windows?
  - use the start of the window position as the numerical key
  - round the key off to nearest 100,000 using –round option

```
# GC fraction in 5kb windows
> cat gc.txt
1 0 5120 58.4375
1 5120 10240 58.4961
1 10240 15360 53.8281
1 15360 20480 48.7305
1 20480 25600 46.4844
1 25600 30720 49.1406
1 30720 35840 32.168
1 35840 40960 35.4688
1 40960 46080 38.3984
1 46080 51200 35.1367
1 51200 56320 32.5195
1 56320 61440 33.4375
1 61440 66560 38.5156
1 66560 71680 39.4727
```

collapse – cont'd

- extract all lines associated with a given chromosome (here chr1)
- specify reference column (key) and data column

```
# 100kb windows
> grep -w ^1 gc.txt | collapse –ref 1 –data 3 -round 100000
0 col 3 n 10 avg 45.62891 med 47.60745 mode 0 sd 9.8111367675775 p10 32.168 p90 58.4375 sum 456 range 26 min 32 max 58
100000 col 3 n 20 avg 41.25683 med 40.0879 mode 0 sd 7.83653420689878 p10 32.2266 p90 49.7656 sum 825 range 31 min 30 max 62
200000 col 3 n 10 avg 40.16603 med 40.49805 mode 0 sd 5.78647372336747 p10 28.7891 p90 43.7891 sum 401 range 22 min 28 max 51
300000 col 3 n 1 avg 28.418 med 28.418 mode 0 sd 0 p10 28.418 p90 28.418 sum 28 range 0 min 28 max 28
400000 col 3 n 19 avg 40.0901157894737 med 36.5625 mode 0 sd 7.74716877175592 p10 32.6562 p90 56.4844 sum 761 range 26 min 31 max 58
500000 col 3 n 12 avg 43.0777958333333 med 46.9336 mode 0 sd 14.1148181248138 p10 38.7109 p90 53.125 sum 516 range 54 min 0 max 55
. . .

# 5 Mb windows
> grep -w ^1 gc.txt | collapse –ref 1 –data 3 -round 5000000
0 col 3 n 432 avg 53.2971771990741 med 55.2897 mode 59.5117 sd 10.3278722300674 p10 38.9648 p90 65.4883 sum 23024 range 70 min 0 max 70
5000000 col 3 n 945 avg 50.9516499259259 med 50.7812 mode 45.7812 sd 7.26967981313169 p10 41.9727 p90 60.7812 sum 48149 range 59 min 9 max 68
10000000 col 3 n 977 avg 47.9971567041965 med 47.3438 mode 47.2852 sd 5.56385055081108 p10 40.8594 p90 55.5664 sum 46893 range 29 min 34 max 64
15000000 col 3 n 949 avg 47.2776190727081 med 46.9531 mode 47.1875 sd 5.46523408101615 p10 40.9766 p90 54.4336 sum 44866 range 55 min 9 max 64
20000000 col 3 n 976 avg 48.2608398565574 med 48.6719 mode 51.2695 sd 5.42601522244971 p10 40.4102 p90 55.0391 sum 47102 range 28 min 33 max 62
25000000 col 3 n 968 avg 47.7213857747934 med 47.5781 mode 48.9844 sd 5.31660850708476 p10 41.3477 p90 54.4141 sum 46194 range 61 min 2 max 64
. . .
```

collapse – multiple hash keys

- data keyed by multiple values can be creatively handled by constructing compound keys

- random number [0,1) for each (x,y) pair
  - (x,y) pair is the key
  - apply collapse to the random numbers associated with a given (x,y)

```
#data.txt
19 33 0.350931
66 79 0.476591
55 75 0.226481
1 41 0.567170
62 2 0.496846
90 63 0.682545
. . .

> sed 's/ /_/' data.txt
19_33 0.350931
66_79 0.476591
55_75 0.226481
1_41 0.567170
62_2 0.496846
. . .
```

collapse – cont'd

- recover original key by reversing the transformation

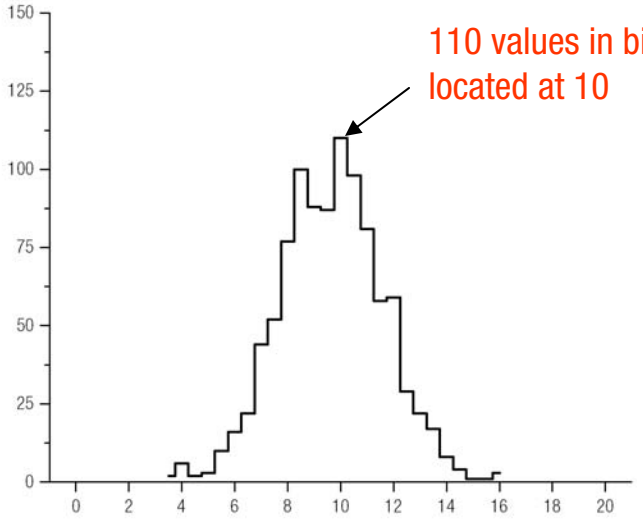
```
> sed 's/ /_/' data.txt | collapse
0_59 col 1 n 8 avg 0.25752 med 0.2129595 mode 0 sd 0.245312208624613 p10 0.010452 p90 0.793974 sum 2 range 0 min 0 max 0
0_4 col 1 n 17 avg 0.481874941176471 med 0.491576 mode 0 sd 0.296086204194503 p10 0.105238 p90 0.875281 sum 8 range 0 min 0 max 0
0_60 col 1 n 8 avg 0.562096875 med 0.615799 mode 0 sd 0.212182386251907 p10 0.149880 p90 0.746615 sum 4 range 0 min 0 max 0
0_79 col 1 n 11 avg 0.722206090909091 med 0.789877 mode 0 sd 0.257336590054914 p10 0.416782 p90 0.973580 sum 7 range 0 min 0 max 0
0_61 col 1 n 8 avg 0.39884125 med 0.3777855 mode 0 sd 0.289251165509814 p10 0.050638 p90 0.921020 sum 3 range 0 min 0 max 0

> sed 's/ /_/' data.txt | collapse | sed 's/_/ /' | sort -n
0 0 col 1 n 4 avg 0.6543155 med 0.79027 mode 0 sd 0.419419550856896 p10 0.042876 p90 0.993846 sum 2 range 0 min 0 max 0
0 10 col 1 n 14 avg 0.3922315 med 0.3027925 mode 0 sd 0.305107130974767 p10 0.068562 p90 0.862939 sum 5 range 0 min 0 max 0
0 11 col 1 n 12 avg 0.439400583333333 med 0.4288565 mode 0 sd 0.346069350978951 p10 0.003669 p90 0.837403 sum 5 range 0 min 0 max 0
0 12 col 1 n 20 avg 0.49009105 med 0.391537 mode 0 sd 0.307805604706415 p10 0.162336 p90 0.947040 sum 9 range 0 min 0 max 0
0 13 col 1 n 4 avg 0.5113005 med 0.504385 mode 0 sd 0.371854436377552 p10 0.075696 p90 0.960736 sum 2 range 0 min 0 max 0
0 14 col 1 n 15 avg 0.510980666666667 med 0.513594 mode 0 sd 0.276092816476289 p10 0.068124 p90 0.889914 sum 7 range 0 min 0 max 0
```

- e.g. average value for pair (0,14) is 0.51 (15 values seen)

histogram – obtain frequency and cumulative histograms

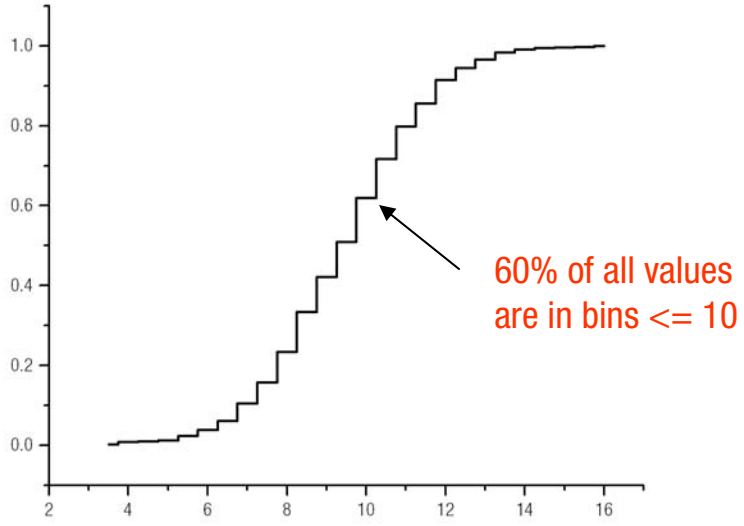
- closely on the heels of collapse is the histogram tool
- histograms are extremely common and useful in presenting data
- there are two types of histograms
  - they help answer very different questions

frequency histogram



how many values in bin X?

cumulative histogram



how many values smaller/bigger than bin X?

histogram – cont'd

- histogram is used on 1D data
  - as example, consider 1,000 normally distributed random values with mean 10, stdev 2

```
cat /dev/zero | fold -1 | head -1000 | perl -ne 'use Math::Random; printf("%f\n",random_normal(1,10,2))‘
9.185922
6.367009
8.223804
9.947639
8.430323
9.801682
10.775383
. . .
```

- let's check with stats

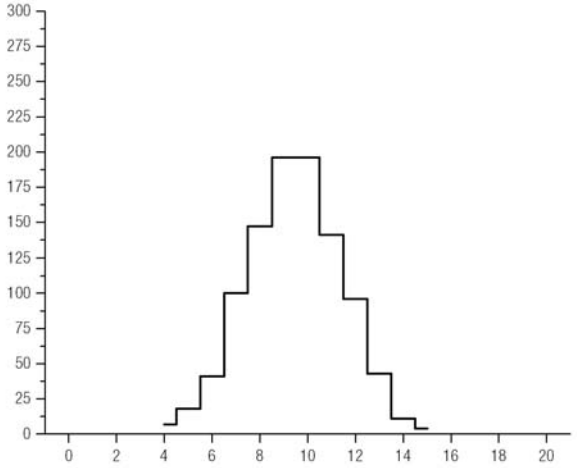
```
» stats r.txt
n 1000 mean 9.955 median 9.971 mode 0.000 stddev 1.9705 min 4.104 max 15.962
p01 5.126 p05 6.781 p10 7.397 p16 7.959441 p84 11.950 p90 12.508 p95 13.163 p99 14.366
```

histogram – cont'd

- now generate a histogram of the data
  - use binsize = 1

```
> histogram -bin 1 r.txt
4.00 4 count 7 7.00 0.00700 sum 30 30.63 0.00308
5.00 5 count 18 25.00 0.02500 sum 98 129.59 0.01302
6.00 6 count 41 66.00 0.06600 sum 273 403.00 0.04048
7.00 7 count 100 166.00 0.16600 sum 754 1157.34 0.11626
8.00 8 count 147 313.00 0.31300 sum 1258 2415.59 0.24266
9.00 9 count 196 509.00 0.50900 sum 1863 4279.58 0.42990
10.00 10 count 196 705.00 0.70500 sum 2058 6338.16 0.63669
11.00 11 count 141 846.00 0.84600 sum 1619 7957.99 0.79941
12.00 12 count 96 942.00 0.94200 sum 1195 9153.82 0.91954
13.00 13 count 43 985.00 0.98500 sum 580 9734.11 0.97783
14.00 14 count 11 996.00 0.99600 sum 158 9892.62 0.99376
15.00 15 count 4 1000.00 1.00000 sum 62 9954.79 1.00000
```

bin value **bin index** **frequency count** **cumulative count absolute and relative**

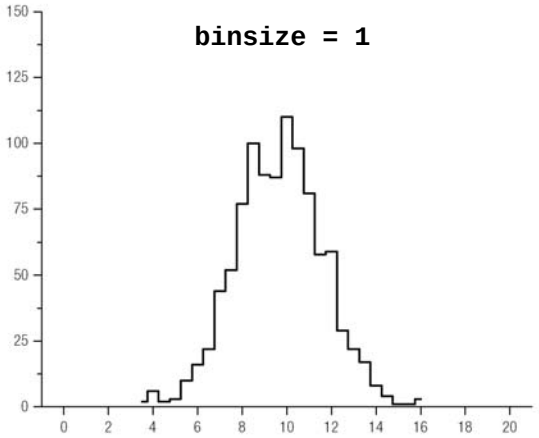


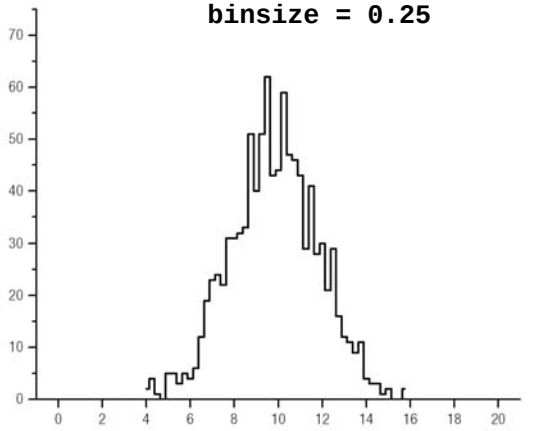

```
# plot of columns 0,3
> histogram -bin 1 r.txt |
  column –c 0,3
```

histogram – cont'd

- smaller bins stratify the data
  - use binsize = 0.25

```
> histogram -bin 0.25 r.txt
. . .
8.25 33 count 32 229.00 0.22900 sum 268 1678.10 0.16857
8.50 34 count 33 262.00 0.26200 sum 284 1962.69 0.19716
8.75 35 count 51 313.00 0.31300 sum 452 2415.59 0.24266
9.00 36 count 40 353.00 0.35300 sum 364 2779.98 0.27926
9.25 37 count 51 404.00 0.40400 sum 478 3258.38 0.32732
9.50 38 count 62 466.00 0.46600 sum 596 3854.81 0.38723
9.75 39 count 43 509.00 0.50900 sum 424 4279.58 0.42990
10.00 40 count 44 553.00 0.55300 sum 445 4725.14 0.47466
10.25 41 count 59 612.00 0.61200 sum 612 5337.46 0.53617
10.50 42 count 47 659.00 0.65900 sum 500 5837.51 0.58640
10.75 43 count 46 705.00 0.70500 sum 500 6338.16 0.63669
11.00 44 count 43 748.00 0.74800 sum 478 6816.93 0.68479
11.25 45 count 29 777.00 0.77700 sum 330 7147.55 0.71800
11.50 46 count 41 818.00 0.81800 sum 477 7624.71 0.76593
11.75 47 count 28 846.00 0.84600 sum 333 7957.99 0.79941
12.00 48 count 30 876.00 0.87600 sum 363 8321.57 0.83594
. . .
```



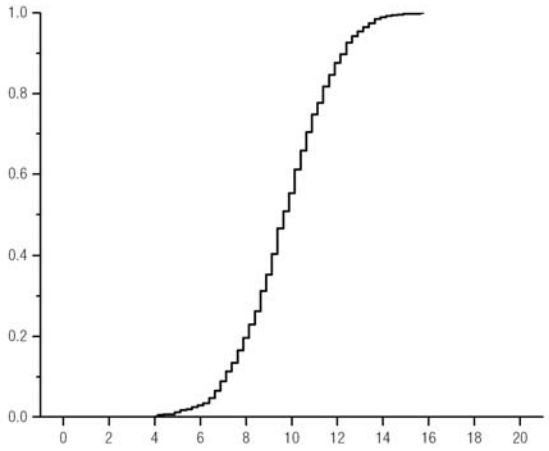


- do not make the bin too small for frequency histograms

histogram – cont'd

- the cumulative histogram is returned as both total cumulative count (0..counts) and relative count (0..1)

```
> histogram -bin 0.25 r.txt
. . .
8.25 33 count 32 229.00 0.22900 sum 268 1678.10 0.16857
8.50 34 count 33 262.00 0.26200 sum 284 1962.69 0.19716
8.75 35 count 51 313.00 0.31300 sum 452 2415.59 0.24266
9.00 36 count 40 353.00 0.35300 sum 364 2779.98 0.27926
9.25 37 count 51 404.00 0.40400 sum 478 3258.38 0.32732
9.50 38 count 62 466.00 0.46600 sum 596 3854.81 0.38723
9.75 39 count 43 509.00 0.50900 sum 424 4279.58 0.42990
10.00 40 count 44 553.00 0.55300 sum 445 4725.14 0.47466
10.25 41 count 59 612.00 0.61200 sum 612 5337.46 0.53617
10.50 42 count 47 659.00 0.65900 sum 500 5837.51 0.58640
10.75 43 count 46 705.00 0.70500 sum 500 6338.16 0.63669
11.00 44 count 43 748.00 0.74800 sum 478 6816.93 0.68479
11.25 45 count 29 777.00 0.77700 sum 330 7147.55 0.71800
11.50 46 count 41 818.00 0.81800 sum 477 7624.71 0.76593
11.75 47 count 28 846.00 0.84600 sum 333 7957.99 0.79941
12.00 48 count 30 876.00 0.87600 sum 363 8321.57 0.83594
. . .
```




```
# plot of columns 0,5
> histogram -bin 0.25 r.txt |
  column –c 0,5
```

- small bins are ok for cumulative histograms

histogram – cont'd

- the histogram tool helps answer the question
  - how many numbers in bin X?
  - how many numbers smaller/larger than bin X?
- sometimes you have a slightly different question
  - what is the sum of numbers in bin X?
  - what is the sum of numbers smaller/larger than bin X?
- this arises when the numbers represent genomic coverage, for example
  - consider a list of sequence contig sizes
  - non-overlapping assemblies of genomic regions

```
#ctgsizes.txt
324136
407986
219279
249268
203036
. . .
```

histogram – cont'd

- you want to add the contig sizes, not just count how many you have, because the sum (coverage) is more important than the number of contigs

```
sums ctgsizes.txt
2699545299
```

- let's histogram the contigs with binsize = 100,000

```
» cat clones.formap.contigs.txt | c2 | histogram -bin 50000 -max 1000000
0.00 0 count 2 2.00 0.00157 sum 73868 73868.00 0.00014
50000.00 1 count 116 118.00 0.09269 sum 9209875 9283743.00 0.01817
100000.00 2 count 147 265.00 0.20817 sum 18037872 27321615.00 0.05346
150000.00 3 count 118 383.00 0.30086 sum 20509729 47831344.00 0.09359
200000.00 4 count 102 485.00 0.38099 sum 22798404 70629748.00 0.13821
250000.00 5 count 86 571.00 0.44855 sum 23372337 94002085.00 0.18394
300000.00 6 count 71 642.00 0.50432 sum 23104770 117106855.00 0.22915
350000.00 7 count 66 708.00 0.55617 sum 24916624 142023479.00 0.27791
400000.00 8 count 83 791.00 0.62137 sum 35081975 177105454.00 0.34655
450000.00 9 count 67 858.00 0.67400 sum 31759456 208864910.00 0.40870
500000.00 10 count 48 906.00 0.71170 sum 25385781 234250691.00 0.45837
550000.00 11 count 41 947.00 0.74391 sum 23549448 257800139.00 0.50445
600000.00 12 count 57 1004.00 0.78869 sum 35533332 293333471.00 0.57398
650000.00 13 count 45 1049.00 0.82404 sum 30485619 323819090.00 0.63364
700000.00 14 count 51 1100.00 0.86410 sum 36833001 360652091.00 0.70571
750000.00 15 count 43 1143.00 0.89788 sum 33224141 393876232.00 0.77072
800000.00 16 count 28 1171.00 0.91987 sum 23138867 417015099.00 0.81600
850000.00 17 count 39 1210.00 0.95051 sum 34124871 451139970.00 0.88277
900000.00 18 count 30 1240.00 0.97408 sum 27782341 478922311.00 0.93714
950000.00 19 count 33 1273.00 1.00000 sum 32125211 511047522.00 1.00000
```

histogram – cont'd

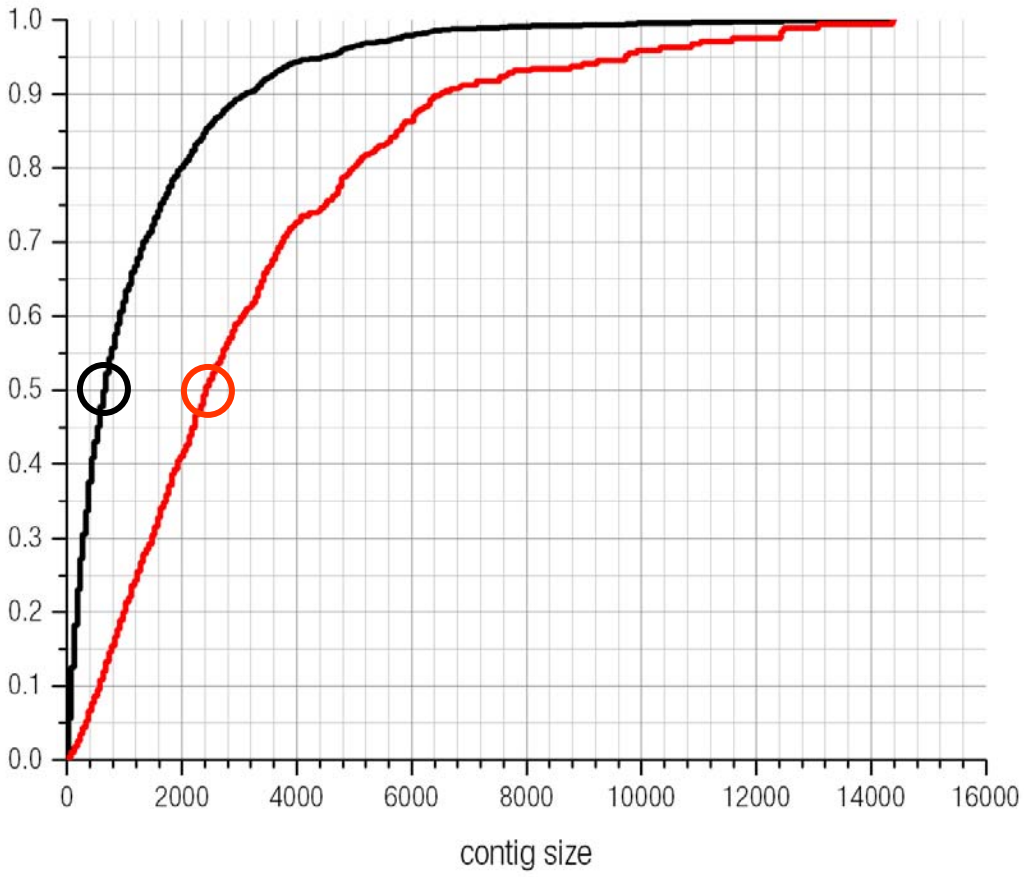
- the second half of the output of histogram reports the **sum**, not the count, of values in bins

```
» cat clones.formap.contigs.txt | c2 | histogram -bin 50000 -max 1000000
0.00 0 count 2 2.00 0.00157 sum 73868 73868.00 0.00014
50000.00 1 count 116 118.00 0.09269 sum 9209875 9283743.00 0.01817
100000.00 2 count 147 265.00 0.20817 sum 18037872 27321615.00 0.05346
150000.00 3 count 118 383.00 0.30086 sum 20509729 47831344.00 0.09359
200000.00 4 count 102 485.00 0.38099 sum 22798404 70629748.00 0.13821
250000.00 5 count 86 571.00 0.44855 sum 23372337 94002085.00 0.18394
300000.00 6 count 71 642.00 0.50432 sum 23104770 117106855.00 0.22915
350000.00 7 count 66 708.00 0.55617 sum 24916624 142023479.00 0.27791
400000.00 8 count 83 791.00 0.62137 sum 35081975 177105454.00 0.34655
450000.00 9 count 67 858.00 0.67400 sum 31759456 208864910.00 0.40870
500000.00 10 count 48 906.00 0.71170 sum 25385781 234250691.00 0.45837
550000.00 11 count 41 947.00 0.74391 sum 23549448 257800139.00 0.50445
600000.00 12 count 57 1004.00 0.78869 sum 35533332 293333471.00 0.57398
650000.00 13 count 45 1049.00 0.82404 sum 30485619 323819090.00 0.63364
700000.00 14 count 51 1100.00 0.86410 sum 36833001 360652091.00 0.70571
750000.00 15 count 43 1143.00 0.89788 sum 33224141 393876232.00 0.77072
800000.00 16 count 28 1171.00 0.91987 sum 23138867 417015099.00 0.81600
850000.00 17 count 39 1210.00 0.95051 sum 34124871 451139970.00 0.88277
900000.00 18 count 30 1240.00 0.97408 sum 27782341 478922311.00 0.93714
950000.00 19 count 33 1273.00 1.00000 sum 32125211 511047522.00 1.00000
```

- 550kb bin
  - 41 contigs in this bin (74% of contigs in this and smaller bins) (26% contigs are larger)
  - total coverage of contigs in this bin **23.5Mb (50% of coverage in contigs in this bin and smaller)**

histogram – cont'd

- cumulative histograms of contigs
  - black trace gives cumulative count
    - 0.5 on y-axis corresponds to median contig number on x-axis
    - median contig size is ~650kb
  - red trace gives cumulative coverage
    - 0.5 on y-axis corresponds to N50
    - size cutoff s.t. all larger contigs provide 50% coverage
    - 50% coverage in contigs larger than 2.4 Mb
- cumulative coverage (sum) is shallower because less of smaller contribution by smaller contigs



enzyme – restriction enzyme information

- this is absolutely useless to you unless you work with restriction enzymes
  - get the cut site for an enzyme, size of site, uniqueness, GC content

```
# list all enzymes (Bio::Tools::RestrictionEnzyme)
> enzyme
AatII gacgt^c 6 5 gacgtc 0.67 unique
AccI gt^mkac 6 2 gtmkac 0.33 flex
AclI aa^cgtt 6 2 aacgtt 0.33 unique
AcyI gr^cgyc 6 2 grcgyc 0.67 flex
AflII c^ttaag 6 1 cttaag 0.33 unique
. . .

# data for HindIII
» enzyme -enzyme HindIII
HindIII a^agctt 6 1 aagctt 0.33 unique

# data for all 4-cutters with unique restriction sites
» enzyme | grep unique | extract -t "_2 == 4"
AluI ag^ct 4 2 agct 0.50 unique
CviRI tg^ca 4 2 tgca 0.50 unique
DpnI ga^tc 4 2 gatc 0.50 unique
FnuDII cg^cg 4 2 cgcg 1.00 unique
HaeIII gg^cc 4 2 ggcc 1.00 unique
HhaI gcg^c 4 3 gcgc 1.00 unique
HpaII c^cgg 4 1 ccgg 1.00 unique
MaeI c^tag 4 1 ctag 0.50 unique
MaeII a^cgt 4 1 acgt 0.50 unique
MseI t^taa 4 1 ttaa 0.00 unique
RsaI gt^ac 4 2 gtac 0.50 unique
TaqI t^cga 4 1 tcga 0.50 unique
```

At the risk of putting you to sleep, I will not cover the

digestvector

prompt tool. If you want restriction maps of vector, or other sequence, read the man page for this tool.

matrix – construct matrix representation of 3D data

- 3D data is most flexibly stored as lines of x,y,z triplets

```
1 a 1a
1 b 1b
2 a 2a
2 d 2d
3 b 3b
4 c 4c
5 a 5a
10 b 10b
15 d 15d
30 a 30a
30 c 30c
30 d 30d
```

- what if you want this represented a-la spreadsheet?
  - matrix treats first column as row label, second column as column label and third column as (row,col) contents

```
» ./matrix -width 4 data.txt
   \    a    b    c    d
   1   1a   1b    -    -
  10    -  10b    -    -
  15    -    -    -  15d
   2   2a    -    -   2d
   3    -   3b    -    -
  30  30a    -  30c  30d
   4    -    -   4c    -
   5   5a    -    -    -
```

matrix – cont'd

- missing data can be represented by any text with –missing
- data can be delimited arbitrarily with –outdelim

```
» ./matrix -missing 0 -outdelim , data.txt
\,a,b,c,d
1,1a,1b,0,0
10,0,10b,0,0
15,0,0,0,15d
2,2a,0,0,2d
3,0,3b,0,0
30,30a,0,30c,30d
4,0,0,4c,0
5,5a,0,0,0
```

- to obtain transpose, swap columns before calling matrix

```
» swapcol data.txt | ./matrix -width 4 -missing "xxx"
   \    1   10   15    2    3   30    4    5
   a   1a  xxx  xxx   2a  xxx  30a  xxx   5a
   b   1b  10b  xxx  xxx   3b  xxx  xxx  xxx
   c  xxx  xxx  xxx  xxx  xxx  30c   4c  xxx
   d  xxx  xxx  15d   2d  xxx  30d  xxx  xxx
```

matrix – cont'd

- recall the (x,y,z) random triplets used to illustrate **collapse**
  - (x,y) were random numbers [0,99], z was random number [0,1)

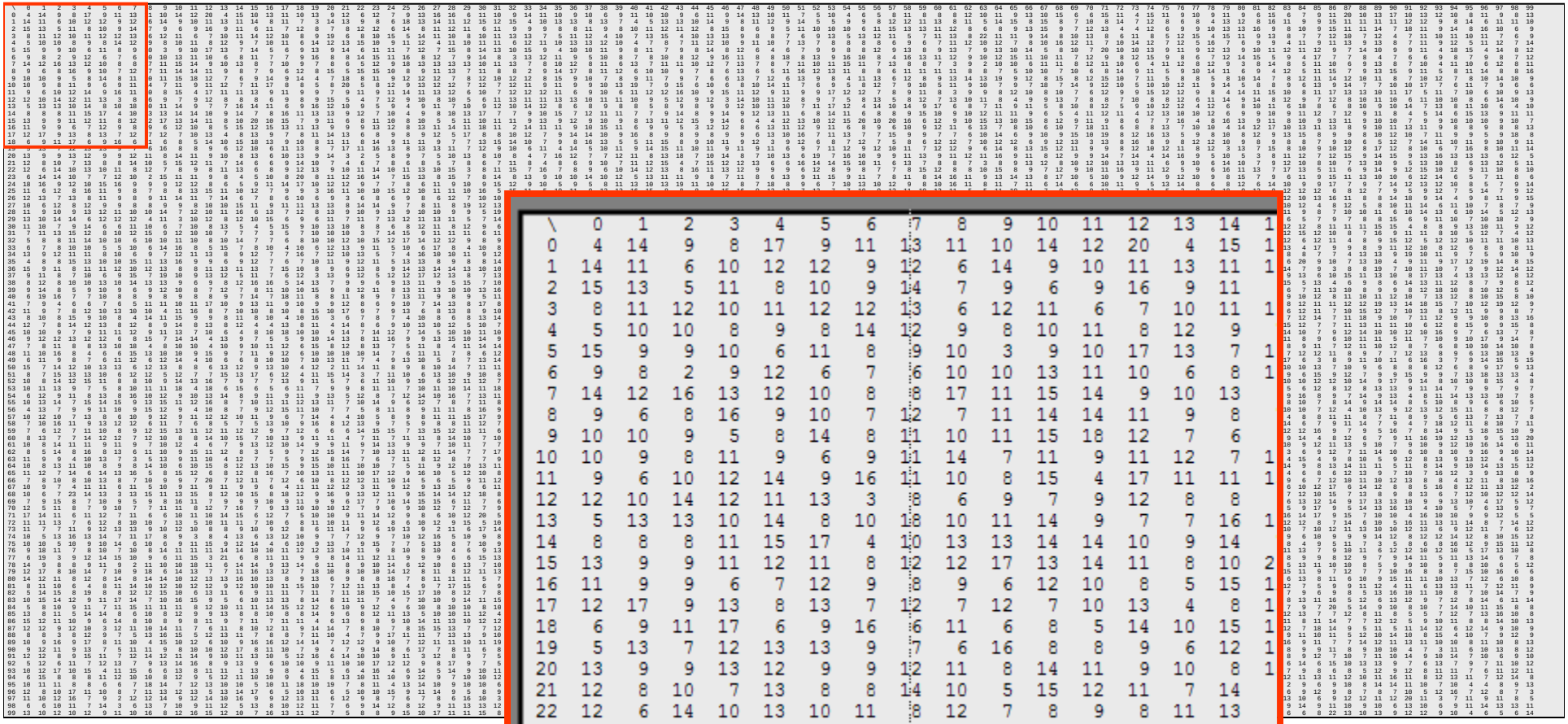
```
#data.txt
19 33 0.350931
66 79 0.476591
55 75 0.226481
1 41 0.567170

>cat data.txt | sed 's/ /_/' | collapse | cut -d " " -f 1,5 | sed 's/_/ /'| ../matrix/matrix -width 3
```

- don't send anyone the output of matrix unless they really really want it
  - you can ruin someone's day
  - consider sparse data and doing this to an enemy

```
matrix –missing “ “ data.txt | shrinkwrap
```

2.1.2.4 – Command-Line Data Analysis and Reporting

matrix – cont'd



\	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	14	9	8	17	9	11	13	11	10	14	12	20	4	15	1
1	14	11	6	10	12	12	9	12	6	14	9	10	11	13	11	1
2	15	13	5	11	8	10	9	14	7	9	6	9	16	9	11	1
3	8	11	12	10	11	12	12	13	6	12	11	6	7	10	11	1
4	5	10	10	8	9	8	14	12	9	8	10	11	8	12	9	1
5	15	9	9	10	6	11	8	9	10	3	9	10	17	13	7	1
6	9	8	2	9	12	6	7	6	10	10	13	11	10	6	8	1
7	14	12	16	13	12	10	8	8	17	11	15	14	9	10	13	1
8	9	6	8	16	9	10	7	12	7	11	14	14	11	9	8	1
9	10	10	9	5	8	14	8	11	10	11	15	18	12	7	6	1
10	10	9	8	11	9	6	9	11	14	7	11	9	11	12	7	1
11	9	6	10	12	14	9	16	11	10	8	15	4	17	11	11	1
12	12	10	14	12	11	13	3	8	6	9	7	9	12	8	8	1
13	5	13	13	10	14	8	10	18	10	11	14	9	7	7	16	1
14	8	8	8	11	15	17	4	10	13	13	14	14	10	9	14	1
15	13	9	9	11	12	11	8	12	12	17	13	14	11	8	10	2
16	11	9	9	6	7	12	9	8	9	6	12	10	8	5	15	1
17	12	17	9	13	8	13	7	12	7	12	7	10	13	4	8	1
18	6	9	11	17	6	9	16	6	11	6	8	5	14	10	15	1
19	5	13	7	12	13	13	9	7	6	16	8	8	9	6	12	1
20	13	9	9	13	12	9	9	12	11	8	14	11	9	10	8	1
21	12	8	10	7	13	8	8	14	10	5	15	12	11	7	14	
22	12	6	14	10	13	10	11	8	12	7	8	9	8	11	13	
23	6	14	14	10	7	7	12	10	2	15	11	11	9	8	4	
24	18	16	9	12	10	15	16	9	9	9	12	12	8	6	5	
25	11	6	12	8	16	11	9	8	7	8	8	13	15	11	10	1

mergecoordinates – create lists of coordinate unions

- remember that list of sequence contig sizes that we used to show how histogram works?
  - such a list can be prepared with mergecoordinates
- start with a list of features with chr/start/end positions and an optional identifier
  - I will use BAC clone positions on chr22 obtained by end sequence alignments

```
#data.txt
22 23703501 23923465 CTD
22 32977027 33141332 CTD
22 20220054 20400887 CTD
22 25860238 26112542 CTD
22 21490995 21657228 CTD
22 46508803 46689801 CTD
. . .
```

- mergecoordinates can help answer
  - what is the coverage of these clones?
  - what is the coverage for a given depth of these clones?
  - what are the disjoint sets of overlapping clones (contigs)

mergecoordinates – cont'd

- mergecoordinates constructs the union of all coordinates and reports disjoint spans
  - chr/start/end/size of each span
  - number of coordinate elements contributing to the span
  - CSV list of element IDs, if provided

```
mergecoordinates data.txt
22 14440103 15064781 624679 44 CTD,CTD,RP11,RP11,CTD,CTD,CTA,CTD,. . .
22 15300713 18578058 3277346 202 CTD,RP11,RP11,RP11,RP11,RP11,CTD,RP11,RP11,CTB,CTB,CTD, . . .
22 18691760 19022333 330574 5 CTD,CTD,RP11,CTD,CTD
. . .

> sed 's/-.*//' bes.txt | ./mergecoordinates2 | column  -col 3,4
624679 44
3277346 202
330574 5
3990407 210
4378840 309
10751130 711
3922161 227
736576 39
698120 22
829546 51
2551832 135
1309638 60
257801 3
195538 6
```

mergecoordinates – cont'd

- let's look at all clones with end sequence alignments
  - 200,000 clones mostly from RP11 and CTA/D libraries

```
> sed 's/-.*//' bes.all.txt |
   ./mergecoordinates2 |
   column  -col 3,4
10821584 47
18063366 388
20395693 150
25487533 266
28638188 108
. . .
```

- profile of contig sizes can be obtained with histogram

```
» column –c 3 ctgs.txt | awk '{print $1/1e6}' | histogram -bin 5
0.00 0 count 292 292.00 0.70702 sum 437 437.48 0.15444
5.00 1 count 36 328.00 0.79419 sum 256 693.71 0.24489
10.00 2 count 24 352.00 0.85230 sum 290 984.65 0.34760
15.00 3 count 17 369.00 0.89346 sum 298 1283.29 0.45303
20.00 4 count 11 380.00 0.92010 sum 246 1530.04 0.54013
25.00 5 count 10 390.00 0.94431 sum 269 1799.67 0.63532
30.00 6 count 10 400.00 0.96852 sum 323 2122.82 0.74940
35.00 7 count 3 403.00 0.97579 sum 116 2239.80 0.79069
40.00 8 count 4 407.00 0.98547 sum 167 2407.74 0.84998
45.00 9 count 0 407.00 0.98547 sum 0 2407.74 0.84998
50.00 10 count 1 408.00 0.98789 sum 50 2457.83 0.86766
55.00 11 count 1 409.00 0.99031 sum 57 2514.94 0.88782
60.00 12 count 1 410.00 0.99274 sum 64 2579.84 0.91074
65.00 13 count 1 411.00 0.99516 sum 69 2649.19 0.93521
70.00 14 count 0 411.00 0.99516 sum 0 2649.19 0.93521
75.00 15 count 0 411.00 0.99516 sum 0 2649.19 0.93521
80.00 16 count 1 412.00 0.99758 sum 82 2731.52 0.96428
85.00 17 count 0 412.00 0.99758 sum 0 2731.52 0.96428
90.00 18 count 0 412.00 0.99758 sum 0 2731.52 0.96428
95.00 19 count 0 412.00 0.99758 sum 0 2731.52 0.96428
100.00 20 count 1 413.00 1.00000 sum 101 2832.70 1.00000
```

ctg size (mb) **cumulative count** **cumulative coverage**

mergecoordinates – cont'd

- what about the average size of contig as function of the number of clones in the contig?
  - collapse useful here

```
> sed 's/-.*//' bes.all.txt | ./mergecoordinates2 | column  -col 3,4 | swapcol
4 251165
3 123153
63 2006581
37 1104592
. . .

> collapse –round 50 ctgsize.txt
0 n 148 avg 497595.189189189
100 n 91 avg 1872787.54945055
200 n 31 avg 3266716.19354839
300 n 22 avg 4591344.77272727
400 n 16 avg 5763975.5
500 n 10 avg 7445101.1
600 n 8 avg 9511708
700 n 9 avg 10384564.8888889
800 n 4 avg 10570086.25
900 n 5 avg 12873527
1000 n 4 avg 12617995.75
1100 n 8 avg 14450961.875
1200 n 3 avg 18434004.6666667
1300 n 6 avg 17721645.1666667
. . .
```

mergecoordinates – cont'd

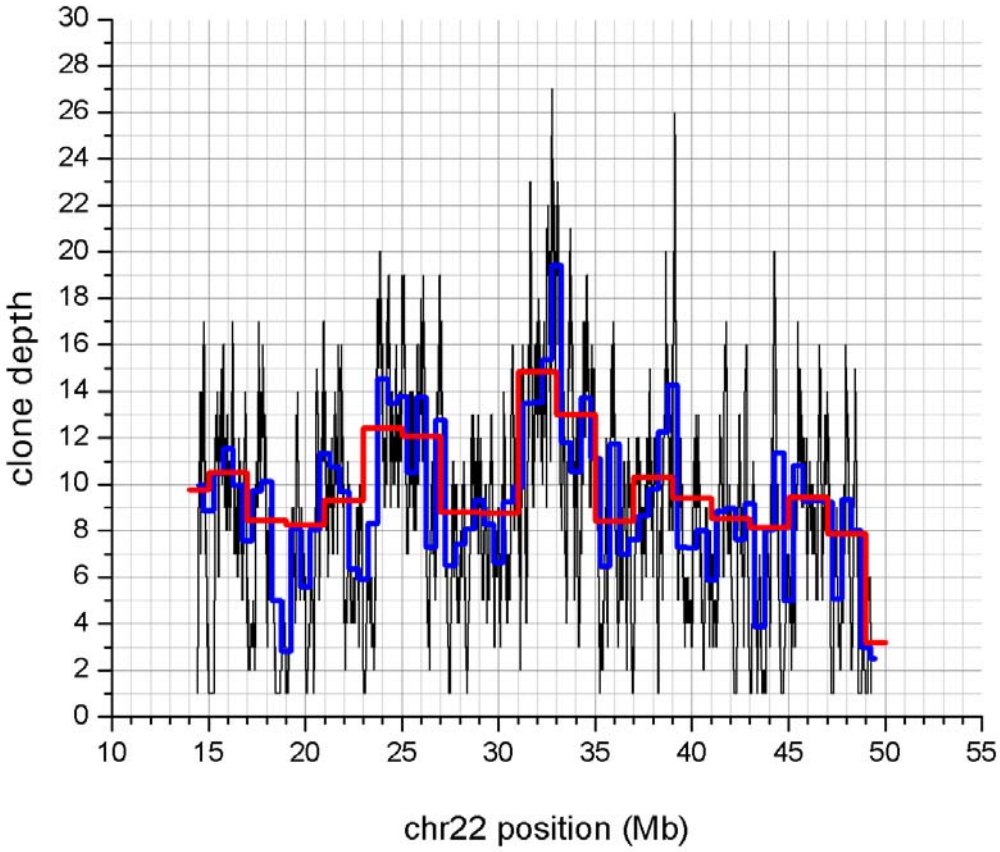
- with the optional –depth flag, mergecoordinates reports not just the contigs, but all depth covers within a contig

```
> mergecoordinates –depth data.txt
22 14440103 14458076 17974 1 CTD
22 14458077 14466337 8261 2 CTD,CTD
22 14466338 14466340 3 3 CTD,CTD,RP11
22 14466341 14473300 6960 4 CTD,CTD,RP11,RP11
22 14473301 14485638 12338 5 CTD,CTD,RP11,RP11,CTD
22 14485639 14486146 508 6 CTD,CTD,RP11,RP11,CTD,CTD
22 14486147 14491290 5144 5 CTD,RP11,RP11,CTD,CTD
22 14491291 14491343 53 6 CTD,RP11,RP11,CTD,CTD,CTA
22 14491344 14493186 1843 7 CTD,RP11,RP11,CTD,CTD,CTA,CTD
22 14493187 14509740 16554 8 CTD,RP11,RP11,CTD,CTD,CTA,CTD,CTD
22 14509741 14512284 2544 9 CTD,RP11,RP11,CTD,CTD,CTA,CTD,CTD,CTD
22 14512285 14512285 1 10 CTD,RP11,RP11,CTD,CTD,CTA,CTD,CTD,CTD,RP11
22 14512286 14528211 15926 9 CTD,RP11,RP11,CTD,CTA,CTD,CTD,CTD,RP11
22 14528212 14540324 12113 10 CTD,RP11,RP11,CTD,CTA,CTD,CTD,CTD,RP11,CTD
22 14540325 14554197 13873 11 CTD,RP11,RP11,CTD,CTA,CTD,CTD,CTD,RP11,CTD,CTD
22 14554198 14560524 6327 10 RP11,RP11,CTD,CTA,CTD,CTD,CTD,RP11,CTD,CTD
22 14560525 14560642 118 11 RP11,RP11,CTD,CTA,CTD,CTD,CTD,RP11,CTD,CTD,CTD
22 14560643 14562414 1772 12 RP11,RP11,CTD,CTA,CTD,CTD,CTD,RP11,CTD,CTD,CTD,RP11
22 14562415 14567954 5540 13 RP11,RP11,CTD,CTA,CTD,CTD,CTD,RP11,CTD,CTD,CTD,RP11,RP11
22 14567955 14573360 5406 14 RP11,RP11,CTD,CTA,CTD,CTD,CTD,RP11,CTD,CTD,CTD,RP11,RP11,CTC
22 14573361 14604998 31638 13 RP11,RP11,CTD,CTA,CTD,CTD,RP11,CTD,CTD,CTD,RP11,RP11,CTC
22 14604999 14610770 5772 12 RP11,RP11,CTD,CTA,CTD,CTD,RP11,CTD,CTD,CTD,RP11,RP11
22 14610771 14614794 4024 11 RP11,RP11,CTA,CTD,CTD,RP11,CTD,CTD,CTD,RP11,RP11
22 14614795 14614853 59 10 RP11,RP11,CTA,CTD,RP11,CTD,CTD,CTD,RP11,RP11
22 14614854 14615060 207 9 RP11,CTA,CTD,RP11,CTD,CTD,CTD,RP11,RP11
22 14615061 14615427 367 10 RP11,CTA,CTD,RP11,CTD,CTD,CTD,RP11,RP11,CTD
22 14615428 14620168 4741 9 RP11,CTA,CTD,RP11,CTD,CTD,RP11,RP11,CTD
22 14620169 14622789 2621 8 RP11,CTA,CTD,RP11,CTD,RP11,RP11,CTD
22 14622790 14623419 630 9 RP11,CTA,CTD,RP11,CTD,RP11,RP11,CTD,CTD
22 14623420 14624555 1136 8 RP11,CTD,RP11,CTD,RP11,RP11,CTD,CTD
22 14624556 14625187 632 7 RP11,RP11,CTD,RP11,RP11,CTD,CTD
22 14625188 14625195 8 8 RP11,RP11,CTD,RP11,RP11,CTD,CTD,CTC
22 14625196 14626375 1180 7 RP11,RP11,RP11,RP11,CTD,CTD,CTC
22 14626376 14628965 2590 8 RP11,RP11,RP11,RP11,CTD,CTD,CTC,CTC
22 14628966 14628972 7 9 RP11,RP11,RP11,RP11,CTD,CTD,CTC,CTC,CTD
22 14628973 14639369 10397 10 RP11,RP11,RP11,RP11,CTD,CTD,CTC,CTC,CTD,CTD
22 14639370 14647707 8338 11 RP11,RP11,RP11,RP11,CTD,CTD,CTC,CTC,CTD,CTD,CTD
. . .
```

mergecoordinates – cont'd

```
> mergecoordinates –depth data.txt |
  column –c 1,4
14440103 1
14458077 2
14466338 3
14466341 4
. . .
```

- black trace shows (x,d), depth for each cover start position
- blue trace shows average d calculated over 500kb windows
  - collapsedata –round 5e5 data.txt
- red trace uses 2Mb windows
  - collapsedata –round 2e6 data.txt

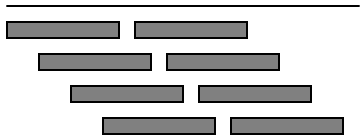


mergecoordinates – cont'd

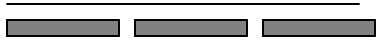
```
> mergecoordinates –depth data.txt
22 14440103 14458076 17974 1 CTD
22 14458077 14466337 8261 2 CTD,CTD
22 14466338 14466340 3 3 CTD,CTD,RP11
22 14466341 14473300 6960 4 CTD,CTD,RP11,RP11
. . .

# total coverage by library
> grep RP11 depth.txt | c3 | sums
31491641
> grep CTA depth.txt | c3 | sums
17076253
> grep CTD depth.txt | c3 | sums
31869061

# coverage unique to RP11 library
> cat bes.depth.txt | grep RP11 | grep -v CT | c3 | sums
990500
```

window – statistics across sliding windows

- window is similar to collapse
  - offers statistics across a sliding window
  - you select window size and step size



  - collapse bins data into disjoint groups, then does the stats



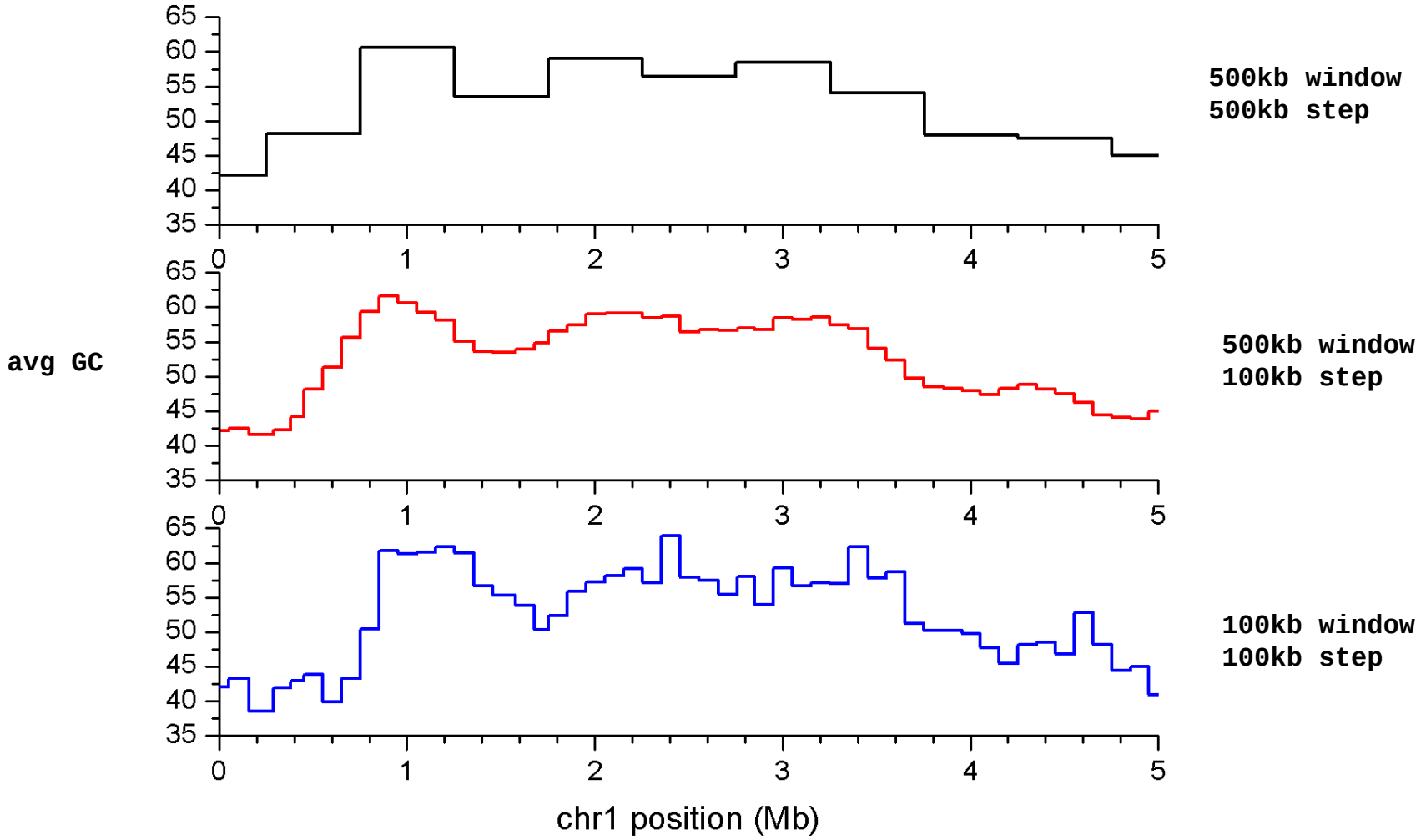
- let's go back to the GC content example

window – cont'd

- when step is the same size as window, the output is equivalent to what is produced by collapse

```
> window –window 500000 –step 500000 –statistic average data.txt
0 0 495820 42.1231507246377
1 500940 996430 48.1479297752809
2 1001550 1496835 60.6502977272727
3 1501955 1999975 53.5362876404494
4 2005095 2496615 59.0800164948454
5 2501735 2997980 56.4708053932584
6 3003100 3499740 58.4767602040816
7 3504860 3998900 54.093388372093
8 4004020 4495540 47.9353154639175
9 4500660 4997300 47.4406081632653
10 5002420 5499295 44.9424160919541
11 5504415 5565855 44.4320846153846

> window –window 500000 –step 100000 –statistic average data.txt
0 0 495820 42.1231507246377
1 102400 597070 42.4648991666666
2 217280 699470 41.5535111940299
3 357580 796750 42.3039724358974
4 403660 899150 44.1576275280899
5 500940 996430 48.1479297752809
6 602190 1098830 51.3173591836735
7 704590 1196110 55.656406185567
8 801870 1298510 59.4080816326531
9 904270 1354830 61.6862696629213
10 1001550 1496835 60.6502977272727
11 1103950 1599235 59.2943170454545
. . .
```

window – cont'd



window – cont'd

- window also supports window sizes in units of **lines**
  - for cases when your data doesn't lie on a distance scale
  - for 1D data (time series)

```
# data.txt
372
395
499
443
424
476
496
539
. . .

> cat –n data.txt | shrinkwrap | window –line 10 –step 5 –strict data.txt
# window_id, window_start, window_end, window_statistic
0 1 10 413.6
1 6 15 513.1
2 11 20 270.4
3 16 25 156.5
4 21 30 416.3
5 26 35 533.8
6 31 40 570.6
7 36 45 589.3
. . .
```

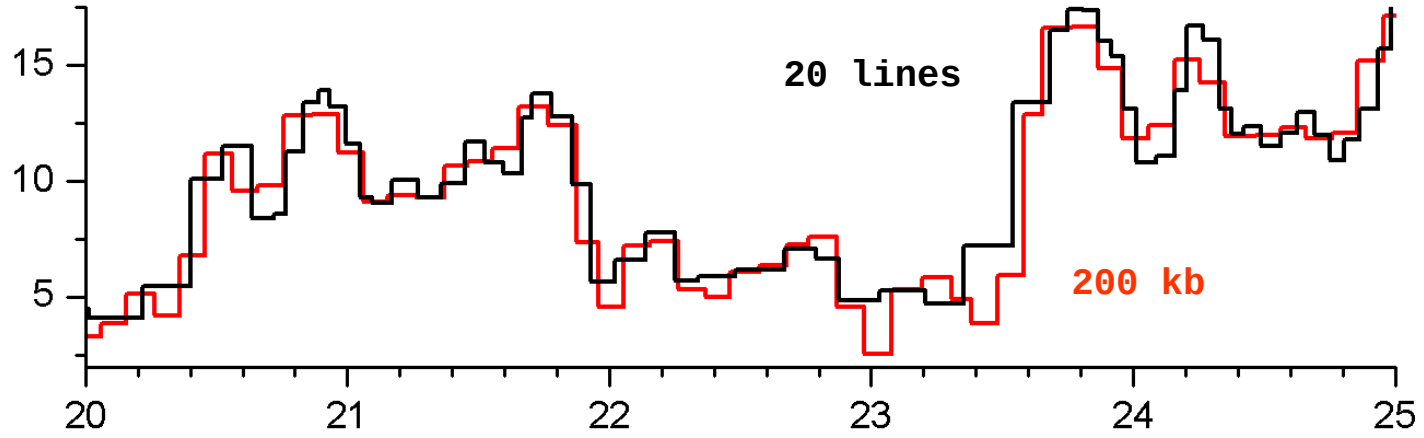
window – cont'd

- sizing your windows by the number of lines is also useful when your data is not uniformly distributed
  - recall the contig depth spans from a previous example

```
# data.txt
# span_start, span_depth
14440103 1
14458077 2
14466338 3
14466341 4
14473301 5
14485639 6
14486147 5
14491291 6
14491344 7
. . .

> window -line 10 -step 2 -statistic average data.txt
0 14440103 14493187 4.7
1 14466338 14512285 6.3
2 14473301 14528212 7.5
3 14486147 14554198 8.5
4 14491344 14560643 9.7
5 14509741 14567955 10.9
6 14512286 14604999 11.5
7 14540325 14614795 11.7
8 14560525 14615061 11.5
9 14562415 14620169 10.9
. . .
```

window – cont'd

```
> window -line 20 -step 10 -statistic average data.txt > w1.txt
> window -window 200000 -step 100000 -statistic average data.txt > w2.txt
```



Prompt tools - recap

- http://gin.bcgsc.ca/Members/martink/Documents/System Utilities/prompttools/view

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

2.1.2.4.4

Command-Line Data Analysis and Reporting – Session iv

- next time you think data analysis, think command line
- don't write a script, investigate UNIX tools and prompt tools
- share your tricks with others