

HowTo: Build and use chromosomal information

Jeff Gentry

June 25, 2008

1 Overview

The `annotate` package provides a class that can be used to model chromosomal information about a species, using one of the metadata packages provided by Bioconductor. This class contains information about the organism and its chromosomes and provides a standardized interface to the information in the metadata packages for other software to quickly extract necessary chromosomal information. An example of using *chromLocation* objects in other software can be found with the `alongChrom` function of the *geneplotter* package in Bioconductor.

2 The `chromLocation` class

The *chromLocation* class is used to provide a structure for chromosomal data of a particular organism. In this section, we will discuss the various slots of the class and the methods for interacting with them. Before this though, we will create an object of class *chromLocation* for demonstration purposes later. The helper function `buildChromLocation` is used, and it takes as an argument the name of a Bioconductor metadata package, which is itself used to extract the data. For this vignette, we will be using the *hgu95av2.db* package.

```
> library("annotate")
> z <- buildChromLocation("hgu95av2")
> z
```

Instance of a `chromLocation` class with the following fields:

```
Organism: Homo sapiens
Data source: hgu95av2
Number of chromosomes for this organism: 25
Chromosomes of this organism and their lengths in base pairs:
 1 : 246127941
 2 : 243615958
 3 : 199344050
 4 : 191731959
```

```

5 : 181034922
6 : 170914576
7 : 158545518
8 : 146308819
9 : 136372045
10 : 135037215
11 : 134482954
12 : 132078379
13 : 113042980
14 : 105311216
15 : 100256656
16 : 90041932
17 : 81860266
18 : 76115139
19 : 63811651
20 : 63741868
21 : 46976097
22 : 49396972
X : 153692391
Y : 50286555
M : 16571

```

Once we have an object of the *chromLocation* class, we can now access its various slots to get the information contained within it. There are six slots in this class:

```

organism:      This lists the organism that this object is describing.
dataSource:    Where this data was acquired from.
chromLocs:     A list with an element for every unique chromosome
               name, where each element contains a named vector where
               the names are probe IDs and the values describe the
               location of that probe on the chromosome. Negative
               values indicate that the location is on the antisense
               strand.
probesToChrom: A hash table which will translate a probe ID to the
               chromosome it belongs to.
chromInfo:     A numerical vector representing each chromosome, where
               the names are the names of the chromosomes and the
               values are the lengths of those chromosomes.
geneSymbols:   An environment that maps a probe ID to the appropriate
               gene symbol.

```

There is a basic 'get' type method for each of these slots, all with the same name as the respective slot. In the following example, we will demonstrate these basic methods. For the `probesToChrom` and `geneSymbols` methods, the return value is an environment which maps a probe ID to other values, we will be using the probe ID '32972_at', which was selected at random for these examples. We

are showing only part of the `chromLocs` method's output as it is quite long in its entirety.

```
> organism(z)

[1] "Homo sapiens"

> dataSource(z)

[1] "hgu95av2"

> names(chromLocs(z))

[1] "1"      "10"     "11"     "12"     "13"
[6] "14"     "15"     "16"     "16_random" "17"
[11] "17_random" "18"     "19"     "2"      "20"
[16] "21"     "22"     "3"      "4"      "4_random"
[21] "5"      "6"      "6_cox_hap1" "6_qbl_hap2" "7"
[26] "8"      "9"      "X"      "Y"      "2_random"
[31] "3_random" "5_h2_hap1" "8_random" "6_random" "19_random"
[36] "22_random" "X_random" "1_random"

> chromLocs(z)[["Y"]]

    266_s_at    31534_at    31911_at    32864_at    32930_f_at    32991_f_at    35885_at
-19611913    2863545    14324840    -2714896    15145847    -6793959    13322553
35929_s_at    35930_at    37583_at    38182_at    38355_at    40030_at    40097_at
   9914563    9914563    -20326690    20213723    13526170    7202013    21146998
  41214_at    1185_at    31412_at    31412_at    31415_at    31415_at    32677_at
   2769622    1415508    -22627290    23045931    -18390255    18756722    -14607046
 32677_at    34172_s_at    34215_at    34753_at    35073_at    35447_s_at    36553_at
 14677491    1670485    1670485    57623412    505078     1674347    -1482031
 36554_at    39168_at    40342_at    40342_at    40435_at    40436_g_at    41138_at
 -1482031    -2414454    -23684896    25389451    -1465044    -1465044    2619227
   629_at    31411_at    31411_at    31411_at    33593_at    33593_at    33593_at
 57739639    23539797    25173538    -25586437    -24600763    26177651    -24601329
 34477_at    34477_at    34477_at    41108_at    33665_s_at    33665_s_at    31601_s_at
-13944308    -13918783    -13869656    -161425     1361570     1347700     22082636
31601_s_at    31601_s_at    31601_s_at    31601_s_at    31601_s_at
 22106177    -22435611    -22459154    22082645     22106186

> get("32972_at", probesToChrom(z))

[1] "X"

> chromInfo(z)
```

1	2	3	4	5	6	7	8
246127941	243615958	199344050	191731959	181034922	170914576	158545518	146308819
9	10	11	12	13	14	15	16
136372045	135037215	134482954	132078379	113042980	105311216	100256656	90041932
17	18	19	20	21	22	X	Y
81860266	76115139	63811651	63741868	46976097	49396972	153692391	50286555
M							
16571							

```
> get("32972_at", geneSymbols(z))
```

```
[1] "NOX1"
```

Another method which can be used to access information about the particular *chromLocation* object is the `nChrom` method, which will list how many chromosomes this organism has:

```
> nChrom(z)
```

```
[1] 25
```

3 Summary

The *chromLocation* class has a simple design, but can be powerful if one wants to store the chromosomal data contained in a Bioconductor package into a single object. These objects can be created once and then passed around to multiple functions, which can cut down on computation time to access the desired information from the package. These objects allow access to basic but also important information, and provide a standard interface for writers of other software to access this information.