

HowTo: Build and use chromosomal information

Jeff Gentry

November 8, 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 : 247249719
 10 : 135374737
 11 : 134452384
 12 : 132349534
```

```
13 : 114142980
14 : 106368585
15 : 100338915
16 : 88827254
17 : 78774742
18 : 76117153
19 : 63811651
2  : 242951149
20 : 62435964
21 : 46944323
22 : 49691432
3  : 199501827
4  : 191273063
5  : 180857866
6  : 170899992
7  : 158821424
8  : 146274826
9  : 140273252
M  : 16571
X  : 154913754
Y  : 57772954
```

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" "7"          "8"
[26] "9"          "X"          "Y"          "2_random"   "3_random"
[31] "5_h2_hap1" "22_h2_hap1" "8_random"   "6_qbl_hap2" "6_random"
[36] "19_random" "22_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 -2714895  15145847  -6793958  13322553
35929_s_at  35930_at  37583_at  38182_at  40030_at  40097_at  40342_at
 9914563   9914563  -20326688  20213723  7202013  21146998  -23684889
 41214_at  1185_at  31412_at  31412_at  31415_at  31415_at  32677_at
 2769622   1415508 -22627290  23045931  18756722  -18390253  14677491
 32677_at  34172_s_at  34215_at  34753_at  35447_s_at  36553_at  36554_at
-14607045  1670485   1670485  57623412  1674347  -1482031  -1482031
 38355_at  38355_at  39168_at  40435_at  40436_g_at  41108_at  41138_at
 13526092  13525412  -2414454  -1465044  -1465044  -161425  2619227
 629_at  31411_at  31411_at  31411_at  31601_s_at  31601_s_at  31601_s_at
 57739639  25173538  23539797  -25586437  22082645  22106186  -22435610
 34477_at  34477_at  34477_at  33593_at  33593_at  33593_at  33593_at
-13869656  -13944307  -13918782  -24601327  26177651  26177651  -24600763
33665_s_at 33665_s_at  35073_at  35073_at
 1347700   1361570   505078   505078

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

[1] "X"

> chromInfo(z)
```

```

      1      10      11      12      13      14      15      16
247249719 135374737 134452384 132349534 114142980 106368585 100338915 88827254
      17      18      19      2      20      21      22      3
78774742 76117153 63811651 242951149 62435964 46944323 49691432 199501827
      4      5      6      7      8      9      M      X
191273063 180857866 170899992 158821424 146274826 140273252 16571 154913754
      Y
57772954

```

```
> 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.