The **rtracklayer** package

Michael Lawrence

April 30, 2008

1 Introduction

The **rtracklayer** package is an interface (or *layer*) between **R** and several genome browsers. Its main purpose is the visualization of genomic annotation *tracks*, whether generated through experimental data analysis performed in R or loaded from a separate data source. The features of **rtracklayer** may be divided into two categories: 1) the representation and import/export of track data and 2) the control and querying of external genome browser sessions and views.

For working with track data, the package defines a data structure named *trackSet*, which extends the Bioconductor *eSet* class. A *trackSet* contains information on the features of a track, including their chromosome, start and end positions in the genome, as well as any associated experimental measurements. Track data may be read from or written to connections (e.g. files on the disk) in the following formats: Browser Extended Display (BED), versions 1, 2 and 3 of the General Feature Format (GFF), and Wiggle (WIG). Support for additional formats may be provided by other packages through a plugin system.

The **rtracklayer** package currently interfaces with the **UCSC** web-based genome browser as well as the locally installed Java-based **Argo** browser. Other packages may provide drivers for other genome browsers through a plugin system. With **rtracklayer**, the user may start a genome browser session, create and manipulate genomic views, and import/export tracks and sequences to and from a browser. Please note that not all features are necessarily supported by every browser interface.

The rest of this vignette will introduce these features through a demonstration.

2 Demonstration

In order to demonstrate the features of **rtracklayer**, we will import some track data from a file and visualize it in the **UCSC** genome browser.

The first step is to attach the **rtracklayer** package, as in:

```
> require(rtracklayer)
```

2.1 Working with track data

Before we can demonstrate the features in **rtracklayer** for manipulating, analyzing and visualizing annotation track data, we must first load a track into the R session. The track information is stored in a *trackSet* object. It is possible to construct a *trackSet* directly from information within the R session, but for the purpose of this demonstration, we will import the track data from a file.

The *import* function imports track data encoded in one of the supported standard formats (BED, GFF and WIG are built-in). The source of the data may be given as a connection, a filename or a character vector containing the data. In this case, we will import the data from a GFF-formatted file included with the **rtracklayer** package, as in the following code:

```
> track <- import(system.file("tests",
+     "v1.gff", package = "rtracklayer"))
```

The track information is now stored in the R session as a *trackSet* object. As the *trackSet* class inherits from *eSet* defined in the **Biobase** package, it has slots for storing feature information (*featureData*), experimental design information (*phenoData*) and experimental measurements (*assayData*).

Most of the information for each feature in a track is stored in the *featureData*. This includes the chromosome name, numeric start and end positions, the DNA strand (+/-/NA), and any other information available. There are methods for accessing the most commonly used fields. For example, the following code retrieves the chromosome names and then start positions for each feature in the track:

```
> featChrom(track)

[1] chr22 chr22 chr22
Levels: chr22

> featStart(track)

[1] 1000000 1010000 1020000
```

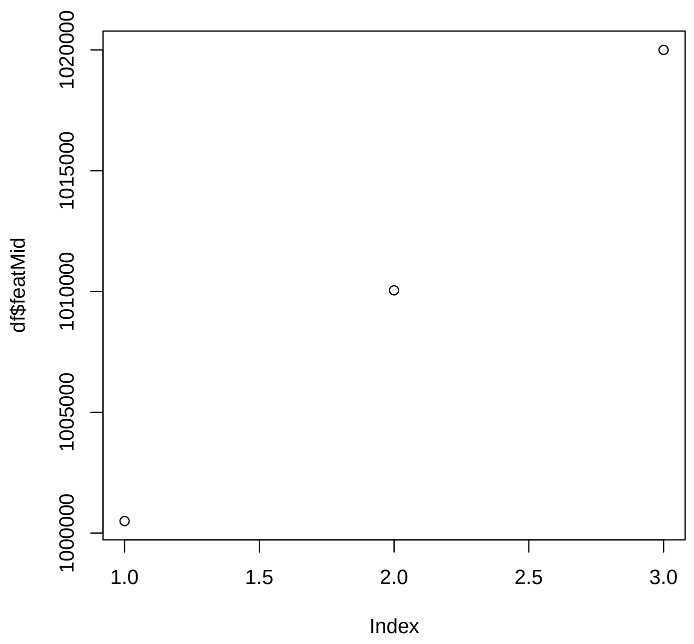
The data values for a track are stored as a numeric matrix under the name *dataVals* within the *assayData*. The values may be retrieved with the following code:

```
> dataVals(track)

    1
1 500
2 900
3 800
```

Sometimes, it may be convenient to extract the track information as a *data.frame*. The *trackData* function does this by combining the *featureData* matrix with the *dataVals*. It also adds a column named *featMid*, which gives the mid-points (the mean of the start and end positions) of each feature in the track. Here is an example of using *trackData* to plot the data values in the track vs. their mid-points.

```
> df <- trackData(track)
> plot(df$featMid, df$dataVals)
```



2.2 Interacting with genome browsers

For the next step in our example, we will load the track into a genome browser for visualization with other genomic annotations. The **rtracklayer** package is capable of interfacing with any genome browser for which a driver exists. In this case, we will interact with the web-based **UCSC** browser, but the same code should work for any browser.

2.2.1 Starting a session

The first step towards interfacing with a browser is to start a browser session, represented in R as a *browserSession* object. A *browserSession* is primarily a container of tracks and genomic views. The following code creates a *browserSession* for the **UCSC** browser:

```
> session <- browserSession("ucsc")
```

Note that the name of any other supported browser could have been given here instead of "ucsc". To see the names of supported browsers, enter:

```
> genomeBrowsers()
```

```
[1] "ucsc" "argo"
```

2.2.2 Laying tracks

Before a track can be viewed on the genome, it must be loaded into the session using the *layTrack* function, as demonstrated below:

```
> session <- layTrack(session, track,
+     name = "Demo Track")
```

The *name* argument should be a character vector that will identify the track within *session*.

2.2.3 Viewing tracks

By default, the *layTrack* function creates a *browserView*, an object that represents a view of a particular set of tracks along a particular region of the genome. For **UCSC**, this roughly corresponds to one tab or window in the web browser. To override the automatic creation of a view, pass *view = FALSE* to the *layTrack* function.

The default view of the track attempts to show the entire track. The view region is determined by a call to *genomeSegment* on the track object, as in:

```
> genomeSegment(track)
```

```
An object of class "genomeSegment"
Slot "genome":
[1] "hg18"

Slot "chrom":
[1] "chr22"

Slot "start":
[1] 1e+06

Slot "end":
[1] 1020000
```

The returned value from *genomeSegment* is an instance of the *genomeSegment* class, which specifies a segment of a genome by its genome name, chromosome name and start and end positions.

In order to zoom in on the first two features of the track, one may subset the track and then create a browser view of the new region using the *browserView* function, as demonstrated below:

```
> subtrack <- track[1:2, ]
```

```
> view <- browserView(session, segment = genomeSegment(subtrack))
```

2.2.4 A shortcut

There is also a shortcut to the above steps. The *browseGenome* function creates a session for a specified browser, loads one or more tracks into the session and creates a view of a given genome segment. In the following code, we create a new **UCSC** session, load the track and view the first two features, all in one call:

```
> session <- browseGenome(tracks = track,
+     browser = "ucsc", segment = genomeSegment(subtrack))
```

It is even simpler to view the entire track in **UCSC** by relying on parameter defaults:

```
> session <- browseGenome(track)
```

2.2.5 Downloading tracks

It is possible to query the browser to obtain the names of the loaded tracks and to download the tracks into R. To list the tracks loaded in the browser, enter the following:

```
> loaded_tracks <- tracks(session)
```

One may download any of the tracks, such as the "Demo Track" that was loaded previously in this example, with the following code:

```
> demo_track <- trackSet(session,
+     name = "Demo Track")
```

By default, the segment of the track downloaded is the current default genome segment associated with the session. One may download track data for any genome segment, such as that displayed by a particular view, as in this code:

```
> view_track <- trackSet(session,
+     genomeSegment(view), "Demo Track")
```

2.2.6 Querying view state

The *view* variable is an instance of *browserView*. To programmatically query the segment displayed by a view, use the *genomeSegment* method for a *browserView*, as in:

```
> segment <- genomeSegment(view)
```

Similarly, one may extract the names of the visible tracks in the view by entering:

```
> visible_tracks <- tracks(view)
```

2.3 Conclusion

This short demonstration has explained a few of the most important features of **rtracklayer**, but many have been left unexplained. Please see the package documentation for more details.

The following is the session info that generated this vignette:

```
> sessionInfo()

R version 2.7.0 (2008-04-22)
x86_64-unknown-linux-gnu

locale:
LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=C;LC_MESSAGES=en_US;L

attached base packages:
[1] tools     stats     graphics
[4] grDevices utils     datasets
[7] methods   base

other attached packages:
[1] rtracklayer_1.0.0 RCurl_0.8-3
[3] Biobase_2.0.0

loaded via a namespace (and not attached):
[1] rJava_0.5-1 XML_1.93-2
```