Rgb : a native genome browser for R

Sylvain Mareschal [1, 2], Philippe Ruminy [2], Fabrice Jardin [2], Herve Tilly [2]

[1] University of Rouen, France
[2] Centre Henri Becquerel, INSERM 918, France

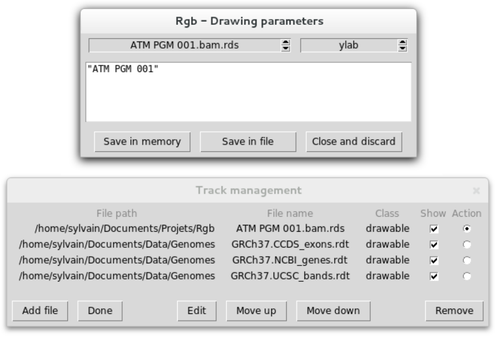
The growing demand from the biology community for statistically robust approaches have made the R statistics-oriented scripting language an essential part of the bioinformatics toolbox. Its graphical capabilities make it a valuable tool to produce publication-grade complex figures, while its computational efficiency allows it to handle huge datasets, as currently required in fields like transcriptomics or next generation sequencing (NGS). These qualities come with an open-source licensing and various operating system ports that make it available virtually everywhere. Thanks to the Bioconductor initiative, a large amount of software is freely available as R packages for tasks like microarray processing, feature annotation, parallel computing or sequence analysis. While most of the available software would largely benefit from "genome browser" representations, there is still a lack of a native and efficient R solution.

For biologists

Available

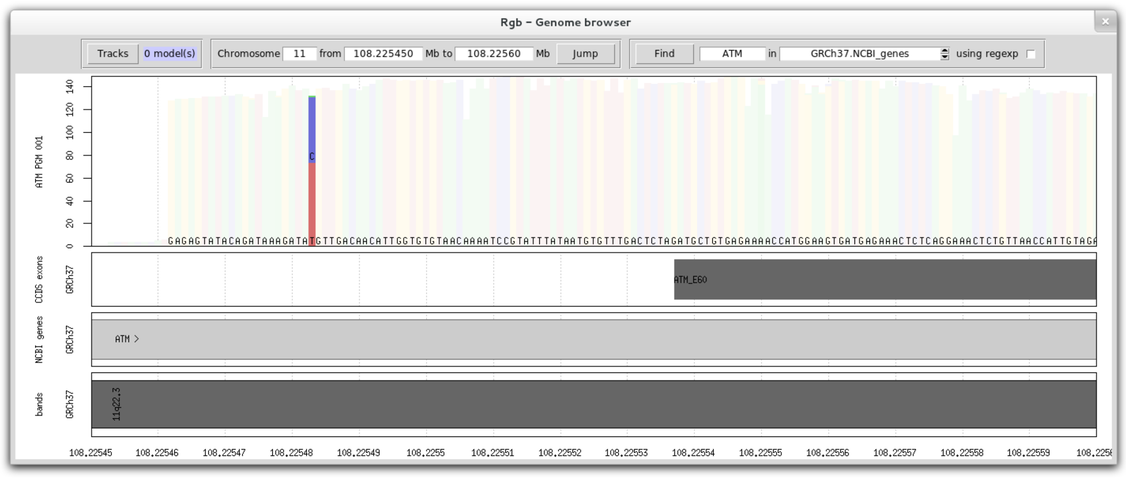
Freely distributed under the **GNU Public License**, Rgb can be deployed on all systems supporting the R statistical language. R itself is free and open-source software, ported to various operating systems (Windows, MacOS, Debian, Redhat, Solaris ...) and currently included in most package repositories. Rgb source code, R package and stand-alone builds can currently be found at http://bioinformatics.ovsa.fr/Rgb, and may be parts of Bioconductor or the Comprehensive R Archive Network in a near future.

Interactive

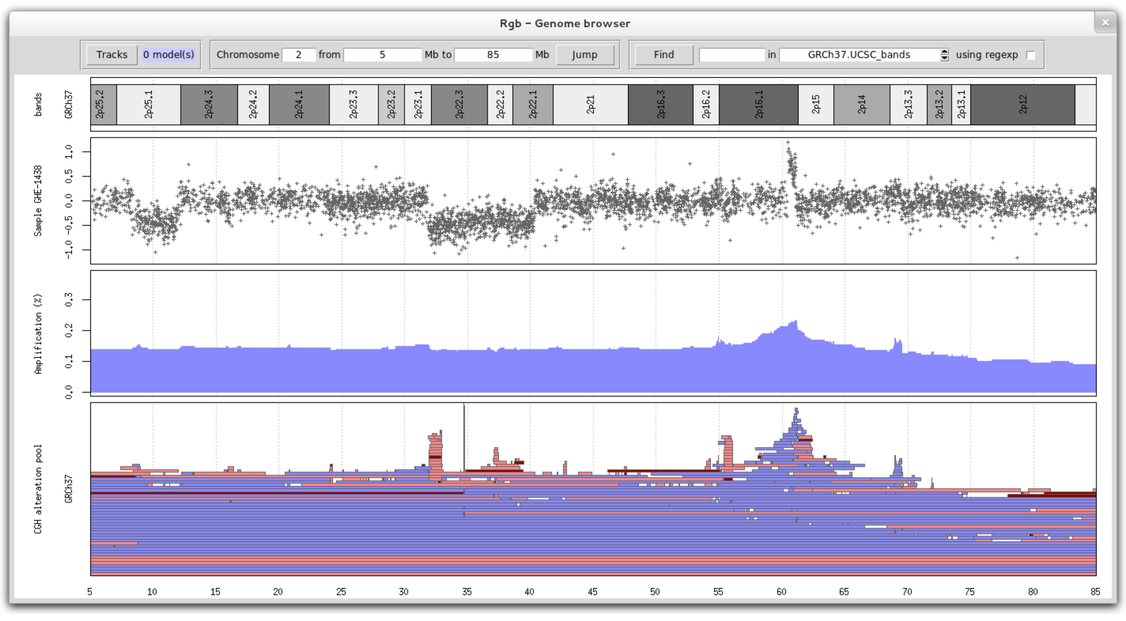


"Genome browser" representations are basically stacks of tracks, each one representing a collection of biological features (genes, Giemsa banding, experiment results ...). First convert your tabular data (CSV or GFF3) using the **file conversion utility**, and import them in Rgb for visualization. If the default representation does not suit your data, **edit the drawing parameters** of your tracks using the provided utility (height, colors, labels, titles ...). Utilities are provided to build **common tracks** such as genes, exons, Giemsa banding and somatic CNV from public databases.

Visual

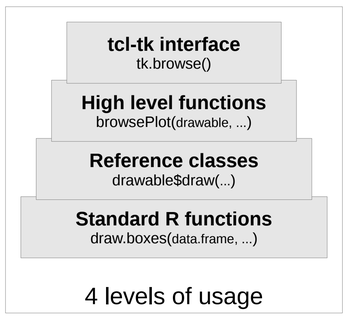


Whatever the genomic experiments you run, benefit from a visual representation of your data that makes it easier to share and understand. Above, a pileup representation of **Next-Generation Sequencing** that clearly shows a polymorphism in an ATM intron. Below, various levels of **CGH-array** results, from single sample probes to whole series alterations, showing the well-known amplification of REL in lymphomas.



For developers

Scriptable



As a R package, Rgb allows users to take direct control of its working parts (classes and functions), at the level that suits their needs :

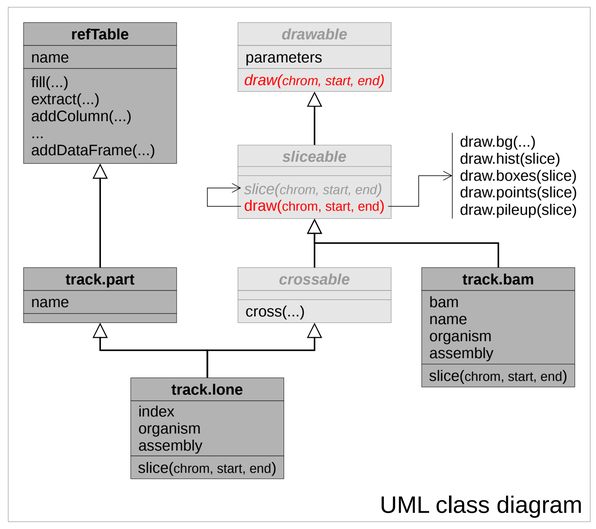
- **R console users** may find it more comfortable to summon the tcl-tk interface during their R interactive sessions (*tk.browse*).
- **script writers** may directly call the underlying rendering function (*browsePlot*) in their scripts, passing track files and coordinates by arguments. More advanced users may also use *drawable* objects directly, for plotting and computation.
- **package developers** manipulating their own classes may find it more convenient to directly call the drawing functions, using standard R classes (vectors, data.frames ...) instead of Rgb reference classes.

In Rgb, scripting is not an additional feature but the **natural** consequence of its development scheme. Users have access to the same classes and functions that the software naturally uses, with a rich and fast scripting language already mastered by most of the bioinformatics community (R).

Extendable

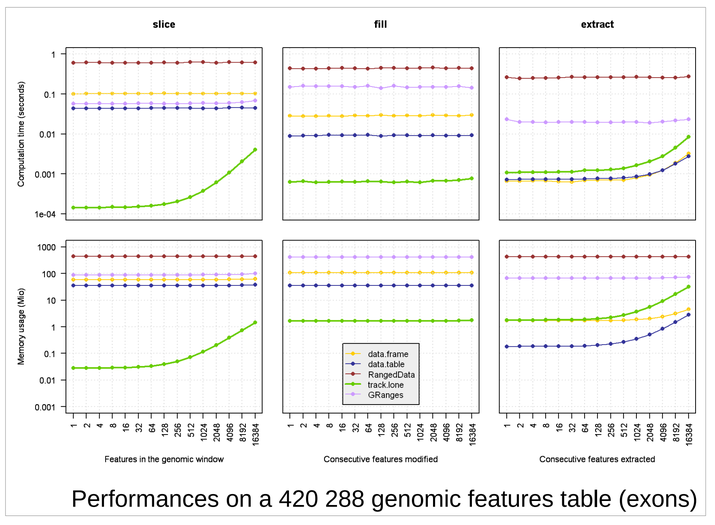
Rgb is developed in the **Object-Oriented paradigm**, which provides great potential for extension. As an example, BAM tracks were implemented later, simply providing a 3 line *slice()* method based on Rsamtools functions to a *sliceable*-inheriting class, and a suitable drawing function.

As most genomic data is currently stored as feature tables, extension can also be achieved by developing **drawing functions** for the *track.lone* class that handles such data representation, a process that requires no knowledge of the Rgb reference classes but only relies on R *base* classes and plotting system.



UML class diagram

Efficient



Performances on a 420 288 genomic features table (exons)

Genomic data are usually queried by "**slicing**" : all features in a genomic window are extracted, for plotting or computation (annotation ...).

Bioconductor provides classes for this purpose (*RangedData*, *GRanges*) that are far too slow for our needs (each rendering needs this operation to be applied to each track), so a new storage solution (*refTable*) was developed from scratch. This reference class relies on R environments, which minimize memory and time wasting due to the "copy-on-write" paradigm of R. The slicing operation itself was implemented in C for data.frames and extended to the *track.lone* class, **dividing the computation time by up to 1000** through row ordering and indexing (see Figure).

21st Annual International Conference on Intelligent Systems for Molecular Biology (ISMB)
12th European Conference on Computational Biology (ECCB)
2013 July 21-23, Berlin