BIOINFORMATICS ORIGINAL PAPER Vol. 27 no. 14 2011, pages 1889–1893
doi:10.1093/bioinformatics/btr309

Genome analysis Advance Access publication May 19, 2011

GenPlay, a multipurpose genome analyzer and browser

Julien Lajugie[1,2] and Eric E. Bouhassira[1,2,*]

[1]Department of Medicine and [2]Department of Cell Biology, Albert Einstein College of Medicine, Bronx, NY 10461, USA

Associate Editor: John Quackenbush

ABSTRACT

Motivation: Rapidly decreasing sequencing cost due to the emergence and improvement of massively parallel sequencing technologies has resulted in a dramatic increase in the quantity of data that needs to be analyzed. Therefore, software tools to process, visualize, analyze and integrate data produced on multiple platforms and using multiple methods are needed.

Results: GenPlay is a fast, easy to use and stable tool for rapid analysis and data processing. It is written in Java and runs on all major operating systems. GenPlay recognizes a wide variety of common genomic data formats from microarray- or sequencing-based platforms and offers a library of operations (normalization, binning, smoothing) to process raw data into visualizable tracks. GenPlay displays tracks adapted to summarize gene structure, gene expression, repeat families, CPG islands, etc. as well as custom tracks to show the results of RNA-Seq, ChIP-Seq, TimEX-Seq and single nucleotide polymorphism (SNP) analysis. GenPlay can generate statistics (minimum, maximum, SD, correlation, etc.). The tools provided include Gaussian filter, peak finders, signal saturation, island finders. The software also offers graphical features such as scatter plots and bar charts to depict signal repartition. The library of operations is continuously growing based on the emerging needs.

Availability: GenPlay is an open-source project available from http://www.genplay.net. The code source of the software is available at https://genplay.einstein.yu.edu/svn/GenPlay.

Contact: eric.bouhassira@einstein.yu.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on February 4, 2011; revised on April 15, 2011; accepted on May 11, 2011

1 INTRODUCTION

With the advent of high-throughput sequencing technologies, bioinformatics analysis has become a bottleneck despite the creation of biocomputational units at major research centers. Many useful tools are available to analyze and visualize genomic and epigenomic data, but most are geared toward informaticians and can be difficult to use for biologists. The problem is not that most applications are run through a command line since commands, flags and parameters can rapidly be learned. Rather, the main bottleneck is the absence of tools that biologists can use to analyze and evaluate their data by the usual trial and error process that is so important to gain a real understanding of any phenomenon. Biologists need tools that allow them to scrutinize genomic and epigenomic data as easily as they can scrutinize a microscope slide by moving the stage, varying the light, etc. to focus on interesting details. With most existing tools, this is not possible because data analysis involves many hours of computing time and many decision points without any visual feedback before the data are sufficiently processed to be loaded in a genome browser.

In practice, most bioinformatics analyses are performed by informaticians that understand the tools but not necessarily the experiments, rather than by the biologists who designed the experiments. GenPlay was developed with the aim of producing a graphically oriented, user-friendly multipurpose tool that would help biologists visualize, analyze and transform their raw data into biologically relevant tracks. It was also developed to facilitate comparison of datasets generated on multiple platforms.

Among the currently available genome browsers, we can distinguish two categories. The first category is web-based client/server application such as the UCSC (Kent *et al.*, 2002) or Ensembl (Fernandez-Suarez and Schuster, 2010) genome browsers. They include rich databases and users do not need to have the data on their personal computer. These browsers are mostly used for data visualization and they generally do not offer tools to process data. The second category of browsers are desktop oriented and rely on local databases and can be graphically much faster. This category is particularly adapted to processing as well as visualizing user-generated data. Examples of desktop-oriented browsers include the software Apollo (Lewis *et al.*, 2002), Artemis (Rutherford *et al.*, 2000), BamView (Carver *et al.*, 2010), Gambit-viewer (http://code.google.com/p/gambit-viewer/), GBrowse (Stein *et al.*, 2002), IGV (Robinson *et al.*, 2011), IGB (Nicol *et al.*, 2009), LookSeq (Manske and Kwiatkowski, 2009), Magic Viewer (Hou *et al.*, 2010), MochiView (Homann and Johnson, 2010), Savant (Fiume *et al.*, 2010), Tablet (Milne *et al.*, 2010).

GenPlay is a desktop genome analyzer and visualizer that was written in Java because of the cross-platform capability of this language and the ease-of-development features from the JDK (Java Development Toolkit).

One of the challenges we faced during development was to obtain fast performance on common desktop computers. The challenge was met by extensive use of multithreading, by paying close attention to data structures and by preloading data to be visualized in the computer's main memory. One effective solution for some data types was to divide the genome into bins of fixed size. One of GenPlay advantages over most of the other browsers is that it places emphasis on letting biologists take control of their own data by providing constant visual feedback combined with extremely rapid browsing at every decision point during an analysis. The speed of the analyses performed in GenPlay is critical to let users experiment

*To whom correspondence should be addressed.

with parameters and test multiple methods to look at their data rather than using standardized methods that are often not optimal.

Second, the structure of GenPlay is highly innovative: loading the entire human genome in the main memory was a challenge. The result is a seamless browsing experience and rapid complex computations in random order that significantly improves genome-wide data analysis.

2 RESULTS

2.1 Web site

GenPlay can be launched from a dedicated web site at the www.genplay.net URL. The user starts the program using a Web Start procedure that also allocates a specific amount of memory to the software. The number of tracks and the level of resolution of the tracks that can be loaded concurrently are limited by the amount of memory selected at start up.

A Library link accessible from the main page of the web site contains annotation and configuration files for the last versions of the human and mouse genome assemblies. Configuration files must be downloaded and installed to work with an assembly and a genome different from the default, which is human hg19. Commonly used annotation files can also be found in the Library or can be accessed through the integrated DAS client.

Help files are available through the Documentation link. Detailed information on all operations and on all formats recognized by GenPlay is available. We strongly recommend that users read the documentation before starting working with GenPlay.

There is also a tutorial page, showing an example of ChIP-Seq analysis and an example of TimEX analysis. There are detailed examples of how to use the operations of the software.

A link allows users to report bugs, ask for new functionalities and submit ideas for future development.

The FAQ page compiles the frequent questions received from users followed by our answers.

Older versions of GenPlay are also available for download from the web site. This can be useful if there is a problem of compatibility between the current version and a project file saved with an older version.

Being open source, the source code is available from the link specified on the about page.

2.2 Graphical user interface

One of our goals was to create a clear, user-friendly interface to visualize complex datasets. The interface that we designed is similar to music recording or commonly used spreadsheet software and consists of a list of tracks composed of rows that contain independent datasets (Fig. 1). Each row can be expanded, collapsed or moved the same way rows move in a spreadsheet. Chromosomal location can be changed by simply dragging the mouse while the wheel allows the user to zoom in and out, seamlessly from a 1 base resolution to a whole chromosome view. On top of the track list, there is a ruler that shows the position of the displayed window on the selected chromosome or scaffold.

All operations available for loaded tracks are initiated by a right mouse click on the track handlers which reveals dynamic menus.

The Control Panel, located right under the track list, allows navigation in the genome. The zoom, the position and the chromosome or scaffold selection can be modified from there as well. In addition, genomic position can be specified in a text field.

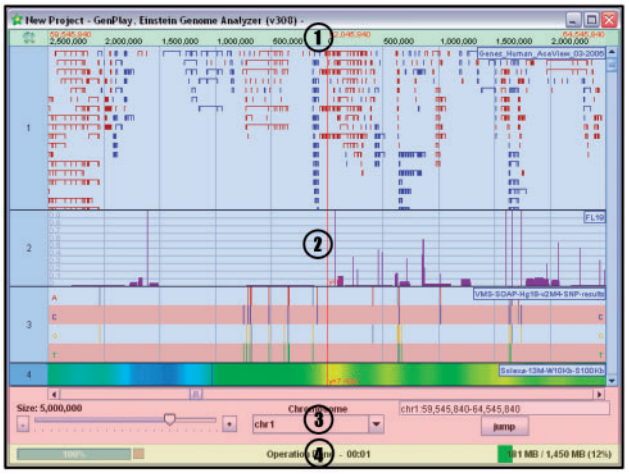

Fig. 1. GUI overview with (1) ruler; (2) track list; (3) control panel; (4) status bar.

The last part of the interface is the status bar at the bottom of the window that allows the user to monitor memory usage. The status bar displays text messages describing the current operation, as well as the time elapsed since the beginning of the operation and the percentage complete. The stop button offers the opportunity to cancel most operations while they are in progress.

On the top left of the software interface, there is an option button that controls the general configuration, toggles the full screen mode and saves the current project (all the tracks from the list except the sequence tracks). Saved project are compressed copies of the RAM memory. As a result, the output project files are considerably smaller than the size of the original loaded files. This allows for efficient reloading of a project and is a convenient means of collaboration between investigators. There is also a link to the Help page and the About page on the web site.

2.3 Track types

Another important goal was to allow users to visualize, summarize and integrate any data file that can be laid onto a genome, regardless of the acquisition method. The major data types that we wished to visualize were the results of gene expression, epigenetic (chromatin immunoprecipitation, DNA methylation, timing of replication, etc.) and single nucleotide polymorphism analyses. Because of the widely different structure of these data types, we developed several track types. The track types currently available in GenPlay are the Variable Window tracks, Fixed Window tracks, Gene tracks, Sequence tracks, SNP tracks and Repeat tracks (Fig. 2).

Variable Window tracks allow the visualization of the position and of the score of interval of variable sizes. These tracks are adapted to the visualization of data that can be summarized on genomic intervals of predefined size. The prototypic datasets are microarray experiments in which each interval represents a probe or a gene.

Fixed window tracks are most useful to visualize raw sequencing results that have not been projected to annotations. This track type is particularly adapted to the visualization of ChIP-Seq, RNA-Seq and Timex-Seq (Desprat *et al.*, 2009) experiments. To create a Fixed

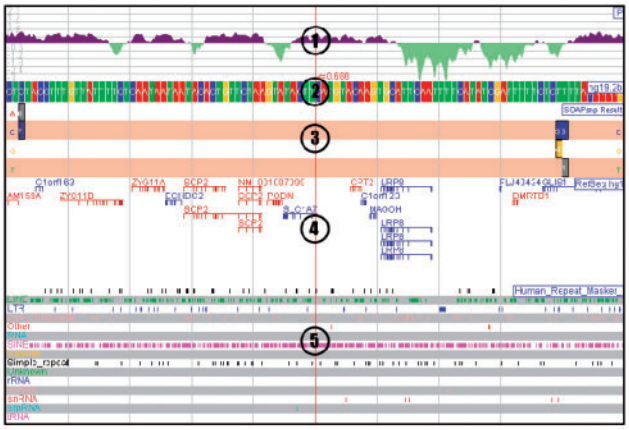

Fig. 2. Examples of different track types with (1) fixed window track; (2) DNA sequence track; (3) SNP track; (4) gene track; (5) repeat family track.

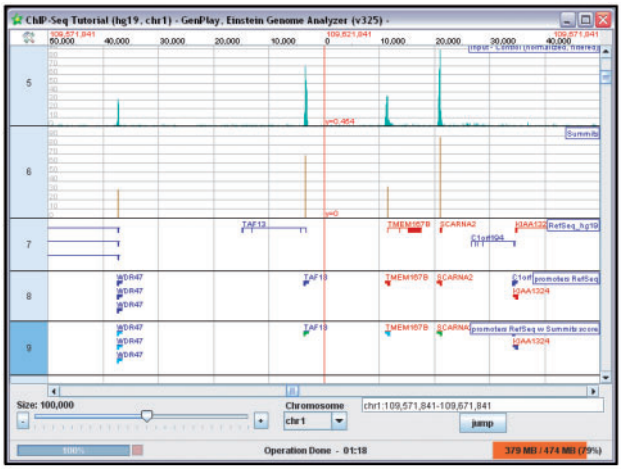

Fig. 3. Example of a ChIP-Seq experiment with track 5: IP; track 6: summits of IP; track 7: RefSeq annotation; track 8: promoters of RefSeq; track 9: promoters of RefSeq genes with score from summits.

Window track, the software divides the genome into bins of fixed size and then computes how many reads fall into each bin. The bin size is selected by the user. Users also define the method of computation of the scores of each bin (sum, max or average number of the scores per bin). Bin sizes of 25–250 bases are generally used for Chip-Seq analysis. Larger bins (1–5 kb) are sufficient to visualize replication timing. RNA-Seq data is most demanding and require very small bin size.

A common issue for the analysis of genomic or epigenomic data is the treatment of overlapping intervals. Gene expression data contain overlapping intervals because genes in the genome overlap. Sequencing data also often contains overlapping intervals.

Both Fixed and Variable Window tracks can only display a single value per interval. When data containing overlapping intervals are loaded, GenPlay summarizes the interval according to user selected parameters. In the case of Variable Window tracks, GenPlay calculates the intersection between all overlapping intervals, creating new intervals whenever necessary. The score of each new interval can be the weighted average, the maximum or the weighted sum of each overlapping interval score. In the case of Fixed Window tracks, GenPlay computes the weighted intersection of all overlapping intervals with the fixed sized bins and assigns the scores as above. In the case of sequencing data, the problem of overlapping intervals can be avoided by only considering the start position of each read. This latter solution is the default mode in GenPlay for loading files that are the output of sequence aligners (Bowtie, Eland, etc.).

The summarization process described above is not optimal for all analysis. Gene expression results often must be visualized in tracks where a single genomic interval can have multiple values, for instance to visualize the expression of multiple isoforms or overlapping genes. To allow this type of analysis, we have developed Gene tracks. Gene tracks can be used to visualize data containing overlapping intervals without summarizing the intervals. Gene Tracks also allow the visualization of exon and intron boundaries and of a score, which can be associated either to the whole gene or to each exon. Scores are visualized by color coding the exons; numeric scores are displayed when the mouse cursor hovers on a gene or on a gene exon. Genes (or other objects) visualized on a Gene Track can be hyperlinked to various databases allowing the software to open a page in a browser showing details about a selected gene.

Sequence tracks display the nucleotide sequence of the selected genome assembly. This is the only track where the data are not completely loaded in memory. The software reads a file in random access only when there is enough room on the screen to print all the nucleotides.

Repeat tracks display repeat windows organized by family or class.

SNP tracks display the frequency at which each base has been found in a particular sample.

2.4 Operations

Another objective during GenPlay development was to create an interactive multipurpose tool for data mining that would allow biologists to rapidly transform and compare their raw data with data obtained from other sources. To achieve this goal, we created a number of operations that can be applied to one or more tracks. These operations are greatly sped-up and facilitated by the software architecture since all data loaded in GenPlay (except for sequence data) are kept in the main memory at all times. The fixed window track format is also well adapted to data transformation and to statistical analysis because of its regularity. The library of operations is still growing and the Report Bugs link of the web site can be used by any user to request new features.

Each track type has different operations available: in the case of Fixed Windows tracks, GenPlay offers normalization, standard score, indexation, Gaussian smoothing, moving window average, Loess regression, correlation calculation, various filters, peak finding, as well as many arithmetic, statistical and masking operations.

Figure 3 illustrates how ChIP-Seq sequencing data can be transformed by GenPlay to visualize enriched regions in gene promoters. In order to remove artifactual peaks, we compared our Immuno-Precipitated (IP) to a control sample after normalization of the data. We also developed and used an algorithm inspired by the work of Chongzhi Zang on the identification of enriched domains

(Zang *et al.*, 2009). This allowed us to isolate peak summits. After that, we extracted gene promoters of the RefSeq genes. A promoter was defined as a region that starts 500 bp before a gene start position and ends 500 bp after. Finally, if a summit fell within a promoter, we attributed the summit score to the promoter. This study is available in the Tutorial section of the web site.

It is also possible to transform tiling array or RNA-Seq data into Transfrag and to annotate the Transfrag using a gene annotation track.

Finally, GenPlay includes powerful export functions that allow users to create concatenated files containing analyses results and chromosome position for each interval genome wide for any number of the loaded tracks. These files can be used for downstream analyses in other software.

2.5 Files supported

GenPlay can recognize the following formats: SAM (Li *et al.*, 2009), Bed, BedGraph, GFF, GTF, pair, PSL, SOAPsnp, WIG, .2bit, Eland Extended. Care was taken to create a flexible file import architecture in order to facilitate the creation of additional file import functions in the future.

2.6 Memory requirement

High memory usage is the major trade-off that we had to make in order to obtain higher performance. GenPlay can be started with very little memory, whereas most genome-wide analysis for mammalian datasets requires a minimum of 4 GB and preferably 8 GB of RAM. Working on multiple tracks at the base pair level can be done in GenPlay if a workstation is available. We routinely perform such analysis using an 8 processor work station with 24 GB of RAM.

Details on the data structure used for the various tracks and memory usage calculations are provided as Supplementary Materials. Keeping annotation tracks (such as gene track) in memory requires little resources. For instance, <10 MB are required to keep all the refSeq genes in memory. The most memory intensive tracks are the Fixed Window tracks, especially when small bin sizes are specified by the user. A typical ChiP-Seq track loaded as 1000 bases bin require 15 MB of RAM, at 100 bases bins 150 MB, at 10 bases bins 1.5 GB and at 1 base bins 15 GB.

To maximize memory usage, GenPlay provides two additional functions. Fixed Windows track can be compressed after loading in memory and users can specify data precision (either at loading time or on the fly). Data stored with 64 bit precision data are stored in double arrays, 32 bit data in float arrays, 16 bit data in short arrays, 8 bit data in byte arrays and 1 bit in byte arrays where each element of the array contains eight windows. Supplementary Table S1 describes the memory requirements for tracks at various bin sizes and various data precisions. Supplementary Table S2 shows the ranges and precisions associated to each data type.

2.7 Limitations and future development

In the future, we will add a function to simultaneously display multiple reference genomes and to display annotations or tracks based on one genome on the coordinates of any other genome. We believe this will prove useful to exploit the large number of human genomes currently being sequenced. We will also add features to perform and record groups of operations on multiple tracks at the same time. Finally, we will improve the data structure to minimize memory usage. We expect that after these modifications, GenPlay users will be able to enjoy the seamless visualization of their data while working genome wide at 1 bp resolution with less powerful machines.

3 METHODS

A big challenge was to create a browser that allows users to seamlessly navigate an entire genome. Several optimizations were introduced in order to achieve this goal.

The first optimization was to dispatch all calculations to different threads. This allows GenPlay to fully exploit the power of modern multicore computers. The software determines the number of cores available using the JDK static method 'availableProcessors() of the RunTime class and starts new threads from a fixed thread pool on each available core.

The second optimization relates to memory usage. As discussed above, in order to accelerate the display and the operations, all data are kept in RAM memory at all times. This results in relatively slow track loading time but provides seamless browsing and tremendously accelerates the operation.

To minimize memory usage, we developed several data structures implementing the java.util.List interface that store data in arrays of primitives. The main advantage of this technique is that arrays of primitive occupy less memory than array of objects. In the last Sun implementations of the JVM (Java Virtual Machine) for Windows XP, the memory allocation works as follows:

(1) Non-array objects requires 8 bytes of 'housekeeping' space and arrays require 12 bytes (four additional bytes for the array length).

(2) Each Boolean or byte field takes up 1 byte, each char and short field takes up 2 bytes, each int and float field takes up 4 bytes and each long and double take up 8 byte. Each reference field takes up 4 bytes.

(3) The amount of memory used is increased in 8 byte blocks.

Hence, an array of 1000 int primitives (int[1000]) will require 4×1000 bytes (for the elements of the array) $+12$ bytes (for the array housekeeping) $= 4012$ $+ 4$ (because the memory is always increased in 8 byte blocks) $= 4016$ bytes.

In contrast, a java.util.ArrayList of 1000 Integer objects will require 8 bytes (for the class) $+4$ bytes (for the pointer to elementData) $+4$ bytes (for elementCount). The elementData array will take 16 bytes (for the elementData class and the length) plus $4\times$ elementData.length bytes. The list also contains the variable int modCount from the superclass java.util.AbstractList, which will take up the minimum 8 bytes. An ArrayList of 1000 elements will therefore require: $8+4+4+16+4\times1000+8=4040$ bytes. Each Integer object of the list will require 8 (for the class) $+4$ (for the int) $+4$ (because of the 8 byte blocks) $= 16$ bytes. Therefore, the list of objects takes up 20 040 bytes as compared with 4016 bytes for the array of int primitives.

We also created different structures for the Fixed Windows tracks and Variable Windows tracks. At the memory level, Fixed Windows tracks are optimized for data spread within the genome such as replication data and ChIP-Seq data. They are represented by arrays containing the scores of each window and the chromosomal position is deducted from the index position in the array. On the other hand, the Variable Windows tracks are optimized for data concentrated in specific regions such as RNA-Seq data. They are represented by list of windows with non-null scores.

The third optimization was the systematic creation of pre-computed displays for different levels of zooming on the chromosome currently displayed. Resolution of the pre-computed display is adapted to each zoom level by averaging the scores of the genomic intervals to fit the number of pixels available.

As a result of these optimizations, changing the position and the zoom level on a chromosome is almost instantaneous on most desktop computers even with many tracks loaded at the same time.

Track loading is relatively fast and range from a few seconds to a few minutes depending on the size of the files.

All operations were also optimized. In the case of a Fixed Window track, the length of the operation depends on bin size. In practice on a standard desktop computer, most operations are computed in <1 min.

ACKNOWLEDGEMENTS

We thank Nathalie Lailler and Erin Tomine for manuscript revisions and Nicolas Fourel for his contributions in some software operations.

Funding: J.L. and E.E.B. are funded by NYSTEM (grant C024172).

Conflict of Interest: none declared.

REFERENCES

Carver,T. *et al.* (2010) BamView: viewing mapped read alignment data in the context of the reference sequence. *Bioinformatics*, **26**, 676–677.

Desprat,R. *et al.* (2009) Predictable dynamic program of timing of DNA replication in human cells. *Genome Res.*, **19**, 2288–2299.

Fernandez-Suarez,X.M. and Schuster,M.K. (2010) Using the ensembl genome server to browse genomic sequence data. *Curr. Protoc. Bioinformatics*, Chapter 1, Unit1.15.

Fiume,M. *et al.* (2010) Savant: genome browser for high-throughput sequencing data. *Bioinformatics*, **26**, 1938–1944.

Homann,O.R. and Johnson,A.D. (2010) MochiView: versatile software for genome browsing and DNA motif analysis. *BMC Biol.*, **8**, 49.

Hou,H. *et al.* (2010) MagicViewer: integrated solution for next-generation sequencing data visualization and genetic variation detection and annotation. *Nucleic Acids Res.*, **38**, W732– W736.

Kent,W.J. *et al.* (2002) The human genome browser at UCSC. *Genome Res.*, **12**, 996–1006.

Lewis,S.E. *et al.* (2002) Apollo: a sequence annotation editor. *Genome Biol.*, **3**, RESEARCH0082.

Li,H. *et al.* (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Manske,H.M. and Kwiatkowski,D.P. (2009) LookSeq: a browser-based viewer for deep sequencing data. *Genome Res.*, **19**, 2125–2132.

Milne,I. *et al.* (2010) Tablet–next generation sequence assembly visualization. *Bioinformatics*, **26**, 401–402.

Nicol,J.W. *et al.* (2009) The Integrated Genome Browser: free software for distribution and exploration of genome-scale datasets. *Bioinformatics*, **25**, 2730–2731.

Robinson,J.T. *et al.* (2011) Integrative genomics viewer. *Nat. Biotechnol.*, **29**, 24–26.

Rutherford,K. *et al.* (2000) Artemis: sequence visualization and annotation. *Bioinformatics*, **16**, 944–945.

Stein,L.D. *et al.* (2002) The generic genome browser: a building block for a model organism system database. *Genome Res.*, **12**, 1599–1610.

Zang,C. *et al.* (2009) A clustering approach for identification of enriched domains from histone modification ChIP-Seq data. *Bioinformatics*, **25**, 1952–1958.