5

Visualization of Next-Generation Sequencing Data

Phillip Ross Smith, Kranti Konganti, and Stuart M. Brown

Large, complex data sets such as next-generation sequencing (NGS) are very difficult to understand even when relatively straightforward analyses provide ready access to the information they contain. Techniques of data visualization often provide a powerful way to pick out important features in the data, such as correlations and skew, that are not readily apparent and allow an investigator to devise new data analytic approaches that permit quantification. Although it is not generally possible to visually inspect results for read alignments across every interval of the **genome** or all of the genes in genome-wide assays, it is very helpful for investigators to view results for a few favorite genes, high scoring regions, or highly significant differentials.

This chapter will explore data visualization in the context of **ChIP-seq**, **RNA-seq**, and other deep-sequencing experiments. The data that are obtained are voluminous, from millions to many hundreds of millions of data records that are delivered to the investigator in huge text files or spreadsheets. However, visualization in the context of appropriate analytics can be very revealing of important biology because the experimental information can be viewed in an appropriate context.

We begin by exploring the data formats used for data representation in NGS, both for experimental data and **reference genomes**, as well as various types of annotation. We will then look at the visualization challenges posed by each of the main types of NGS experimental applications including ChIP-seq, RNA-seq, variant discovery, deep sequencing of **amplicons**, and **de novo assembly**. The development of data indexing methods has been very important in allowing software on ordinary workstations or laptops to access only the relevant parts of data stored in multigigabyte files. To conclude we will try to outline the thinking that an investigator might want to follow when considering a decision to choose among existing visualization software or to create a new visualization tool required to fill local needs.

DATA FORMATS

Data Representation

The raw data output from sequencing machines contains all the information that the machine can provide. These data formats are proprietary and depend on the manufacturer of the sequencing equipment. Among these is SFF (standard flowspace format), which is the output of choice of the Roche 454 sequencing machines. There are also qseq.txt and .bcl files created by Illumina and many variants of **FASTQ formats** (sequence and quality) adopted by different vendors and modified across various versions of their software.

These output formats are very voluminous because they contain the sequence data; quality measures for the entire read and for individual bases that are called; identifiers for read, plate location, direction, and pairing; and other data that may be needed to resolve **sequence alignment** issues as the data are assembled. Raw data are generally very voluminous and most analyses only require a subset of the fields provided in each record or a quality filtered subset of all reads collected by the machine. Filtering methods are used to remove the unneeded fields and to produce a more compact representation of the data.

In many deep-sequencing projects, short **sequence reads** are aligned to a **reference genome**. In this case the actual sequence can be fully defined by the assembly ID of the reference genome (i.e., hg19), the chromosome, the location of the start position on that chromosome, the length of the sequence string, and any differences between the reference genome and the read. These differences are recorded as mismatches by the **alignment algorithm**, which may later be identified as **sequence variants** by a **variant detection** algorithm. This set of alignments and mismatches is also the basis for reference-based data compression, which is discussed in more detail in Chapter 13. Different data formats are needed for optimal use of different tools. Fortunately, there are several easy to implement toolboxes, such as SAMtools (http://samtools.sourceforge.net/), with software to interconvert between data formats, so that data can be made available for packages that do not read them natively.

FASTA and FASTQ

Historically, the **FASTA** and **FASTQ** sequence data formats have played pivotal roles in data storage and exchange. These are plain text files with a single header line for each sequence element (which may be a read or a database entry), beginning with an identifying start character, ">" in the case of FASTA and "@" in the case of FASTQ.

FASTA files store simple sequence data (either nucleic acid or protein) of usually fewer than 80 characters in a line. In the case of a nucleic acid sequence there are 13 additional "ambiguity" characters aside from the usual A,C,G,T,U that code for

uncertainties in the sequence identifications (see http://en.wikipedia.org/wiki/FASTA_format). FASTQ files store sequence data along with a corresponding quality score (see http://en.wikipedia.org/wiki/FASTQ_format). FASTQ format was not widely used for Sanger sequence data, but has become the standard data storage and exchange format for NGS data. Data stored in FASTA and FASTQ formats can be processed by a broad range of software packages; they are the most common data exchange formats for bioinformatics software and the default download format for all public sequence databases.

There are many additional data formats that are used for NGS data. Depending on the use to which the data are put, these vary significantly in complexity and in the data elements stored in the file. Table 1 shows a compilation of most of the common file formats at the time of final revision of this chapter. The table contains links to the sources of the definitive descriptions of these formats. You should be aware that the definitions of the file formats can change and, with a few exceptions, are not necessarily subject to any formal review and standard setting. Also, software authors may modify these formats in inconsistent ways to contain data they need for their own program packages. It is often the case that a file produced by one software package or database will be rejected or misread by another. The nomenclature for the chromosomes should be consistent among the various data sources that you want to use; however, **GenBank** and UCSC name chromosomes with the "chr" prefix (e.g., "chr19"), whereas EMBL/ENSEMBL just use a plain number (e.g., "19"). Strand can be denoted by +/− or F/R. There are some formal file format validator programs, but they may not catch these inconsistencies—so a given file may be valid by the definitions of the file format, yet still fail to be read correctly by your software of choice.

TABLE 1. NGS data file types

Name	Web description	Main package	Type
ALN	http://www.biostat.jhsph.edu/~hji/cisgenome/index_files/tutorial_seqpeak.htm	CisGenome	TSV
BED	http://genome.ucsc.edu/FAQ/FAQformat.html#format1	UCSC	TSV
FASTA	http://en.wikipedia.org/wiki/FASTA_format	FASTA	TSV
FASTQ	http://en.wikipedia.org/wiki/FASTQ_format	SAMTools	TSV
GFF/GTF	http://www.sanger.ac.uk/resources/software/gff/	ENSEMBL	TSV
GFF3	http://gmod.org/wiki/GFF3	GMOD	TSV
VCF	https://github.com/samtools/hts-specs		
SAM/BAM	http://samtools.sourceforge.net/SAM-1.3.pdf	SAMTools	TSV/binary
USEQ	http://useq.sourceforge.net/useqArchiveFormat.html	IGB	Binary
WIG	http://genome.ucsc.edu/goldenPath/help/wiggle.html	UCSC	TSV

ALN

ALN is a simple, straightforward genomic position format that is used as the principal data format for the "CisGenome" package, among others. It is a flat file, one record for each tag location with three tab-delimited fields as follows:

```
chromosome[tab] location[tab] strand
```

An example might be

```
chr10   1021346   +
chr10   1021456   –
   etc. ...
```

In an ALN file the location is measured on the "+" strand irrespective of the read direction ("+" or "–"). Because there is a single field for location, the ALN format assumes that all sequence reads are the same length, and this length must be defined external to the file.

Some software authors "load" the records with additional values. We will discuss this later in the chapter.

BED

A record in the BED format is similar to ALN. In its simplest form it looks like

```
chromosome[tab] StartLoc[tab] EndLoc ...
```

There are, however, nine additional fields that can be optionally included in a BED-formatted record. These are needed because **BED files** are used primarily to contain information needed for data display in the UCSC Genome Browser. Beginning with field 4, the names of these fields in a BED record are 4-name, 5-score, 6-strand, 7-thickStart, 8-thickEnd, 9-itemRgb, 10-blockCount, 11-blockSizes, and 12-blockStarts. The order of these optional fields is required: The lower-numbered fields must always be populated if a higher-numbered field is used. A record in BED format that contains the minimal information in an ALN file must have at least six fields if the "strand" information is required for display or analysis. More information as to the detailed meaning and use of these fields can be obtained from the reference in Table 1. Once again, it is important to note that "BED" files defined in the context of software other than the UCSC Genome Browser may not be identical or compatible (http://qed.princeton.edu/main/Wiggle_(BED)_files).

Wiggle (WIG)

The WIG format is used in the UCSC Genome Browser website as a method to store information for display in a compact way. WIG format defines display parameters for

intervals of the genome—not for individual **sequence reads**. It is often used to display integrated information or analysis results. It works best for the "display of dense, continuous data such as GC percent, probability scores, and transcriptome data." Individuals who are heavy users of the UCSC Genome Browser will find this format of value for the storage and exchange of data for display, but it is not particularly useful for storing data as part of an analysis pipeline. The UCSC web page provides more about this format (http://genome.ucsc.edu/goldenPath/help/wiggle.html).

GFF/GTF

The GFF (general feature format) consists of one line per feature, each containing nine columns of data, plus optional track definition lines. The GTF (general transfer format) is identical to GFF version 2. Features may be genes, **exons**, or other defined genomic regions. The columns are (1) chromosome name, (2) annotation source (program that generated this feature), (3) feature type, (4) start position, (5) end position, (6) score (a floating point value), (7) strand (+ or −), (8) frame ("0," "1," or "2"—indicating if the first base of this feature is first, second, or third base of a protein coding codon), and (9) attribute (a semicolon-separated list that may contain any type of information about this feature). A "." character is used in any column that lacks information.

GFF3

GFF3 is the Generic Feature Format, version 3 (http://gmod.org/wiki/GFF3), which is a newer version than the originally proposed GFF format. GFF3 is a format native to the GMOD (General Model Organism Database) package in which genomic sequence features are stored in a tab-delimited flat file. Complete gene structures with annotations are represented in a nested pattern, which stores information about the location of these features on the chromosome. Genomic Sequence and its annotation in GFF3 format is mainly used in displaying its features in graphical format in Generic Genome Browser (GBrowse). Readers are encouraged to learn about GFF3 format from The Sequence Ontology Project (http://www.sequenceontology.org/gff3.shtml).

SAM and BAM

SAM stands for "sequence alignment/map," the format defined formally by the "SAM Format Specification Working Group," which completed v1.3-r882 of the specification in December 2010 (the definition is in continual revision). The SAM file format is complex and its details are best read about on the website given in Table 1. SAM is used to store sequence reads aligned to a reference genome.

BAM files are binary files that consist of indexed, block-compressed, SAM data constructed so that the index can provide fast, direct access to any part of the file. This format is therefore ideal for many display tasks, particularly using the UCSC Genome Browser, because only the data needed for the display of the currently selected genomic interval need be uploaded, resulting in a significant increase in display speed. The format of BAM files is strictly defined and is machine independent (http://samtools.sourceforge.net/SAM-1.3.pdf).

VCF

Variant call format (VCF) is a text file that contains only information about sequence variants—those bases in the NGS data from a given sample that differ from the reference genome, one variant per line. This implies that the raw data (in BAM files) has been processed by a variant detection software and each variant "called" meets the various thresholds for quality, sequencing depth, **allele** frequency, etc. Many NGS scientists have suggested the use of VCF files as the permanent database and data exchange format for most NGS data (at least for humans and other model organisms) allowing a huge reduction in the size of data files. However, a majority of scientists have insisted that the primary read data is still required because the "calls" in VCF files cannot be trusted, and all of the actual sequence reads must be continually reprocessed by ever more sophisticated variant detection algorithms.

The VCF file contains headers with metadata lines (##fileformat, ##INFO, ##FORMAT, ##SAMPLE, ##FILTER, etc.) followed by data (one variant per line) described by eight "fixed" fields (tab-delimited columns) and a user-defined number of additional optional fields. The fixed fields are (1) chromosome, (2) positions (variants are generally defined as located at a single base position rather than an interval as in BED and GFF files), (3) identifier (such as a dbSNP rs number), (4) reference base, (5) alternate base (the variant found in the sample), (6) quality score for the ALT base, (7) filter ("PASS" or a list of software-specific filters that this base has failed), (8) additional info (a semicolon-separated list of key-value pairs, with many predefined keys including AF, allele frequency; SB, strand bias at this position; DB, dbSNP membership for this variant; DP, combined depth across samples), etc. INFO keys are also used to describe structural variants including SVTYPE (DEL, INS, DUP, INV, CNV, BND) and SVLEN. A full description of the VCF file specification is available at https://github.com/samtools/hts-specs.

Useq

Finally, we mention "Useq," which is a binary format allowing a directory of genomic data to be zip-compressed. Using this scheme, very large amounts of preindexed genomic information can be archived and transmitted. The web page describing it

indicates the packages that can access these data directly and provides details as to the organization of the data within the archive (http://useq.sourceforge.net/useqArchiveFormat.html).

ChIP-seq

The optimum choice of visualization tool will depend on the biological problem being studied. We begin by looking at a simple task, that of identifying binding sites of **transcription factors** to DNA using ChIP-seq. The purpose of a ChIP-seq experiment is to find parts of the genome where proteins bind preferentially. In a simple example, a transcription factor is bound to the genome and is cross-linked to stabilize the DNA–protein interaction. The DNA is fragmented and the DNA–protein complex is immunoprecipitated using an antibody specific to the transcription factor. The **DNA fragments** are released from cross-links to the bound protein and sequenced. The DNA sequences are then aligned to the reference genome. The aligned sequence reads are called "tags."

Intuitively, it is clear that the tags obtained should cluster in the regions of the genome where the transcription factor binds. When the experiment is visualized, the investigator needs to see "peaks" consisting of clusters of tags on the DNA strands that indicate preferred binding. The "problem" is not so much the visualization of the peaks, but defining how and where to look for them. It is very clear that even if there are thousands of peaks, it is unlikely that looking at a single stretch of DNA/RNA will reveal much.

Although deciding whether a given peak is significant is a task requiring sophisticated analysis software (see Chapter 7), it is easy and very quick to generate a **coverage** depth "score" from the binding locations that can be used to pinpoint the locations on the DNA strands where tags are clustered. The investigator can then step through the locations of local maxima in the score, confident that the large majority of peaks will have been inspected. Alternately, the visualization tool can incorporate a set of gene names and locations from an authoritative reference source (i.e., a GFF file from GenBank or EMBL) and use those names to jump to the genomic locations of genes of interest (http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE13047).

In Figure 1 we show a tag peak located using a score computed similarly to the MACS algorithm and visualized using a simple display tool developed in our laboratory here at New York University Langone Medical Center (NYULMC). Other packages can provide similar results. The tool used most commonly in the community is probably the UCSC Genome Browser.

How to Choose Visualization Tools

Visualization tools need to be convenient to use, which means, practically, that they need to run on the computer typically used for work at the desktop.

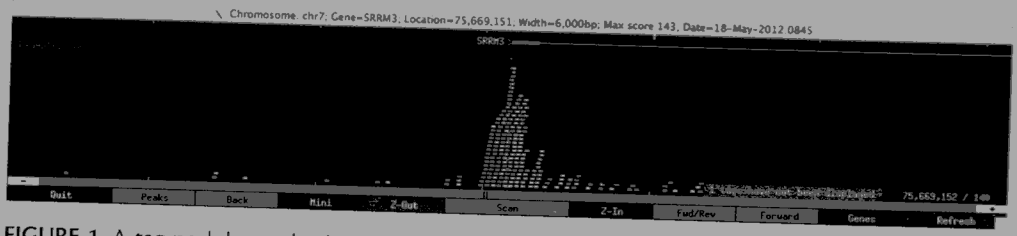

FIGURE 1. A tag peak located using a score computed similarly to the MACS algorithm and visualized using a simple display tool developed in our laboratory at NYULMC.

Right now, investigators will almost certainly have a PC running a flavor of MS Windows, a Macintosh, or a Linux workstation. Because most laboratories are likely to have investigators using all of these platforms, there is a benefit to have software serving a common task that is capable of running on all three platforms, so that everyone can participate equally in data analysis and the inspection of the results.

The software investigators need may run either locally, on the desktop, or remotely on some other platform. There are many options for the latter, the obvious example being software that is accessed via the Web, such as the UCSC Genome Browser.

There are three ways, in general, to create cross-platform software that is to run locally. The obvious method is to create functionally identical packages built for each of the platforms individually. This usually produces the fastest versions of the software, but it can significantly slow development if each platform has different requirements, and it will add dramatically to the cost and time for maintenance as the developer is forced to keep up with system software upgrades for each platform.

The second method is to use a scripting language or a cross-platform software toolkit (such as Perl or the R language) to build the software. In this case the software code is identical across platforms, which simplifies development and deployment. However, the scripted software requires platform-specific libraries to execute. Because these libraries are, usually, highly standardized and optimized, the software author does not need to worry about low-level implementation issues. The price paid is that the library tools are generic and may not be tailored to the specific needs of the package, thus overall performance may be mediocre. Performance is a critical issue for NGS analysis packages because the data are usually very large.

The final method is to write the software in Java. The big advantage of Java is that it is a true cross-platform system. As with the scripting method, a high-performance Java virtual machine needs to be installed on the platform where the software is to run, but Java is now very well established and it is hard to find a desktop where Java is not available or easily installed. The drawback of software built in Java is

that the Virtual Machine often requires more available RAM than executing similar operations in code that is optimized to run natively on the desktop computer's own operating system. There may also be some challenges in configuring a local Java virtual machine to take advantage of the large amount of physical RAM installed on a user's computer.

The bottom line is that software built native to the platform to be used has the potential to run faster and to be able to take advantage of particular features of the specific machine being used, such as high-performance storage, multiple CPUs, very large memory, and software optimizations at the CPU level. These potentials are no guarantee that they will be exploited, and so in many cases scripted software and software implemented in Java will have very competitive performance. Some investigators who write their own software do so initially in a scripting language, such as Perl, and then port it to native code for large-scale production only if the Perl version is unacceptably slow. In this fashion the relative simplicity of the scripting language speeds algorithm development and natively built software is created only when and if it is needed.

To decide how best to visualize data, you need to know some basic facts. First, where is the data stored? Considerable processing is needed to prepare data for analysis and display, which often will be performed remotely on the analysis computing cluster associated with the sequencing laboratory. The decision to be made is whether to move the data files to the local workstation or to access them over a network connection.

What are the capabilities of the local workstation? The key factors are local disk data storage capacity, RAM, CPU power (including the number of processor cores), the workstation operating system, and the design of the visualization software. Is the workstation able to display data stored remotely?

In general, nearly all data analysis occurs as part of a pipeline, so the "best" data analysis choices are those that optimize the efficiency of the process overall, minimizing the minor "waits" for data to transfer or display and facilitating the move to the next steps in the analysis. From the standpoint of data visualization, "best" choices allow an image to appear on the screen quickly and are most responsive to local commands. Usability and the learning curve are also important factors. Software that is slow and cumbersome to use consumes time constantly throughout all analyses. Software that requires substantial learning to operate requires an initial investment of time. Software that is sophisticated and complex to operate (or that provides a very open-ended toolkit) creates a premium on expert users, so that the analysis of a single data set may produce different results in the hands of different data analysts.

Another key issue is the costs associated with software licenses, the time taken to install packages on the target machines (a local workstation or a more central server), and the costs of software maintenance. Various open source versus commercially

distributed software may have different levels of documentation, user training, technical support, response to error reports, and long-term commitment to software maintenance. Locally developed software needs to be maintained and the maintainer is usually the owner. Time spent installing and updating software is time not spent working with the data and moving a project forward. Senior investigators need to be aware of these types of trade-offs and encourage efficient software choices for members of a research team.

SOFTWARE COMPARISONS

A reader of this chapter will probably be on the point of making choices for the best software to meet a project need. It is almost inevitable that the options available at the time you read this will be different from those that seemed most attractive at the time we prepared this material, and the user's research needs may be different than the use cases presented here. The reader's goal should be to understand the strengths of each package, to decide whether these are key strengths for the project, and then to look for them in the latest software offerings. These include a subset of the file formats that can be read natively, whether the software needs to be run remotely or locally and therefore installed on the workstation, the display mode, the amount of data that can be displayed on the screen at one time, and the time to display an image on the screen from a suitable input file.

Staden

One of the first software tools developed to assist investigators in analyzing sequencing data was the Staden package, the first elements of which were authored in 1977, and which was finally published in 1998 (Staden et al. 1998). Its particular value is that it was first to indicate the central role of the computer in managing the tasks of nucleic acid **sequence assembly** and visualization (see Fig. 2). The package has achieved very high usage and is actively maintained as a project on the SourceForge site (http://staden.sourceforge.net/).

UCSC Genome Browser

The UCSC Genome Browser is a web-based genome browser that is very widely used for visualization of all types of location-specific annotation on human and model organism genomes. UCSC was commonly used to visualize early versions of NGS data (after alignment to a reference genome) as "custom tracks" uploaded by the

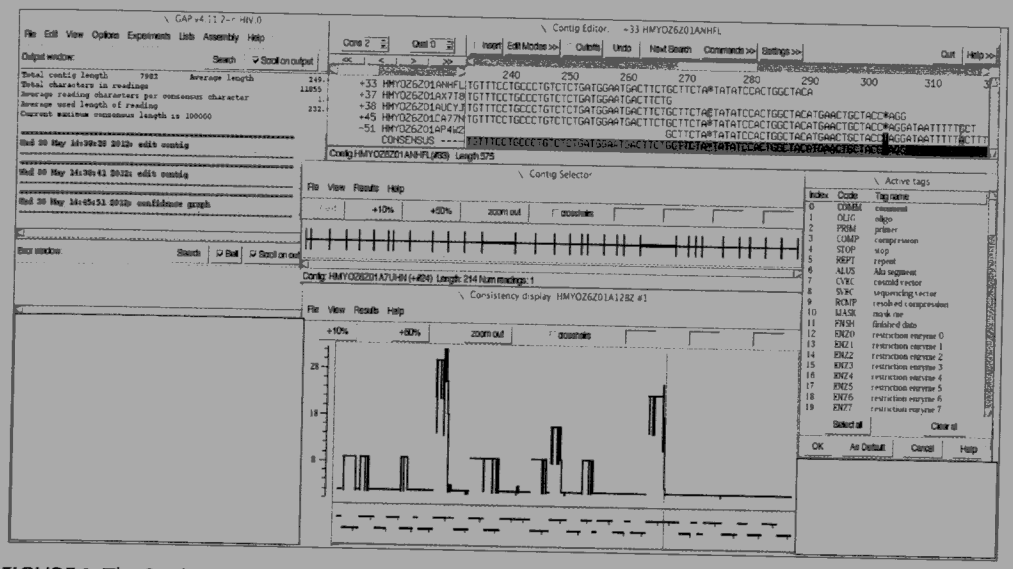

FIGURE 2. The Staden GAP4 package provides a visual interface for assembly and editing of DNA sequence contigs.

user. However, the UCSC Genome Browser limits the amount of data that can be uploaded as custom tracks to their website. Because NGS data is on the order of tens of gigabase pairs (Gb), this has become impractical as a routine method to visualize entire data sets, although it can still be very valuable for subsets of sequence data at loci of interest.

The UCSC browser also supports an alternative method known as URL upload for large data files (such as BAM files). However, to take advantage of this method, the investigator must place the data files on a computer that allows free access from the internet by http protocol (i.e., an open web server). This level of open access to data and computers generally contradicts university and corporate security policy and conceptually allows access to the data to anyone who knows the URL.

The key advantages of the UCSC browser include the huge number of different annotation tracks optionally available to the user for display (GenBank sequences, predicted gene models, gene expression experiments, gene function, sequence variants, comparative genomics, **chromatin** structure, regulatory motifs, etc.) and the instant availability of the resource on the web. UCSC only supports .BED and .WIG data formats so the user must convert their data from BAM format into bedGraph. The BED format is a simple set of genome intervals that appear as blocks (i.e., exons) in the browser. The WIG format allows each base to be represented with a bar at a different height, which can be used to summarize all of the sequence reads in a data file, such as the coverage graph shown in Figure 3.

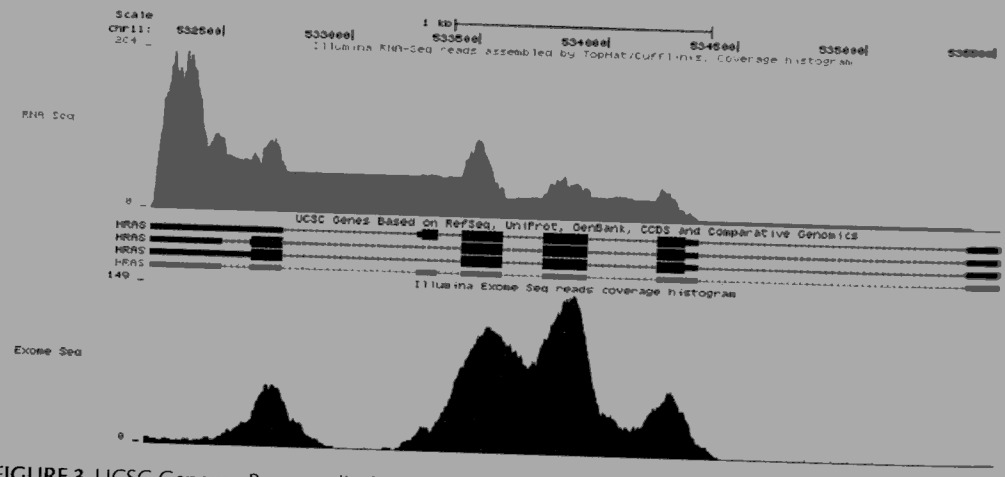

FIGURE 3. UCSC Genome Browser display of coverage graphs of Illumina RNA-seq reads assembled into transcripts by TopHat/Cufflinks package (green) and Illumina Exome sequencing reads (blue) at the *HRAS* gene.

Illumina Genome Studio

Genome Studio (http://www.illumina.com/software/genomestudio_software.ilmn) is commercial software from Illumina that is sold to facilitate the analysis of data collected on their sequencing platforms. The package is built around modules that can allow the investigator to tackle all the major challenges in nucleic acid sequencing research (the modules include the DNA Sequencing Module, the RNA Sequencing Module, the ChIP Sequencing Module, the Genotyping Module, the Profile miRNA expression, the Methylation Module, and the Protein Analysis Module). Genome Studio runs on the Windows platform. Because of the very large data sets that it integrates, at least 8 GB of RAM and a multicore CPU are required for good performance. Genome Studio can read and export data from all of the major data formats and can therefore be integrated into any workflow that an investigator chooses (see Fig. 4).

MAUVE

MAUVE (http://asap.ahabs.wisc.edu/software/mauve/) is a package designed to help construct multiple genome alignments (Darling et al. 2010). It permits the project to be followed from its earliest stages through to the final characterization of the alignment. The package is written in C and can be downloaded and compiled from source if desired for Windows, UNIX, and Mac OSX platforms (see Chapter 7 for a more detailed discussion of MAUVE).

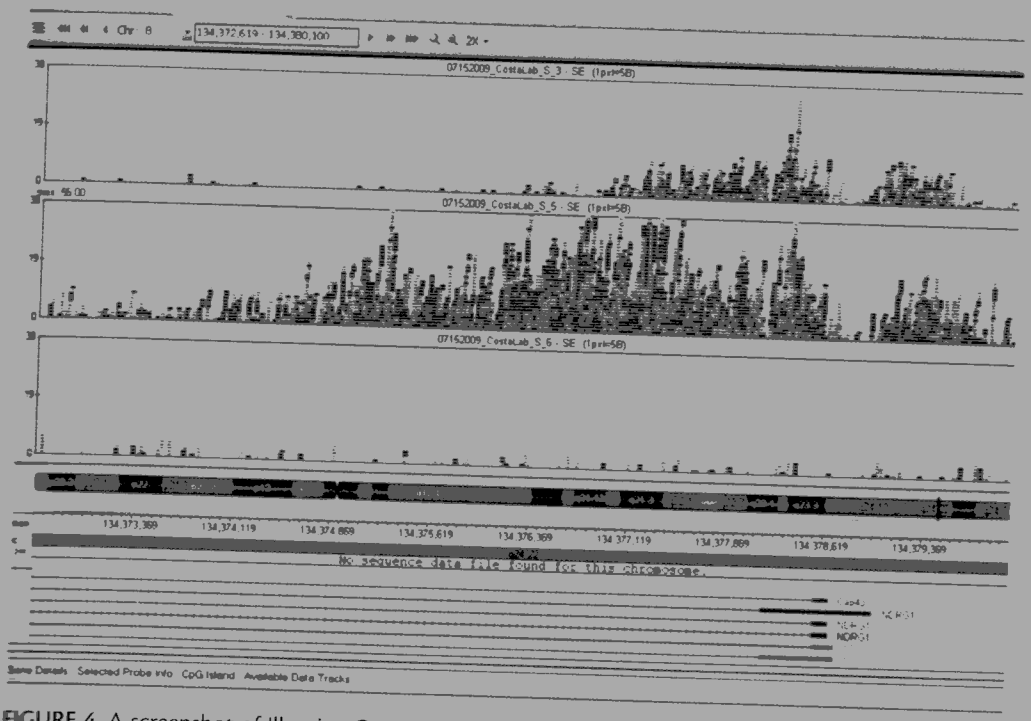

FIGURE 4. A screenshot of Illumina Genome Studio visualizing reads from two ChIP-seq samples and a genomic DNA control, aligned to the reference human genome near the 5′ end of gene *NRDG1*.

Newbler, Amplicon Variant Analyzer

Newbler (formally known as GS de novo **Assembler**; http://contig.wordpress.com/2010/02/09/how-newbler-works/) is a package developed for de novo assembly from data obtained from the **Roche 454 Genome Sequencer**. Amplicon Variant Analyzer is a software package developed by 454 for **multiple alignment** and visualization of very deep sequencing of defined **sequence fragments** (polymerase chain reaction [PCR] amplicons). It has been used to identify rare mutations in in vitro populations of HIV. Roche 454 software runs on Linux and other Unix platforms. It is freely available to Roche customers (see Fig. 5).

Integrative Genomics Viewer (IGV)

The IGV is free Java-based stand-alone desktop software developed at the Broad Institute (Robinson et al. 2011). It can visualize NGS data in a variety of native file formats (FASTA, FASTQ, SAM and BAM, VCF) as well as **microarray** gene expression and genotyping data. Annotation tracks in BED and GFF format can also be loaded, so the IGV display can be configured very similarly to the UCSC Genome Browser or

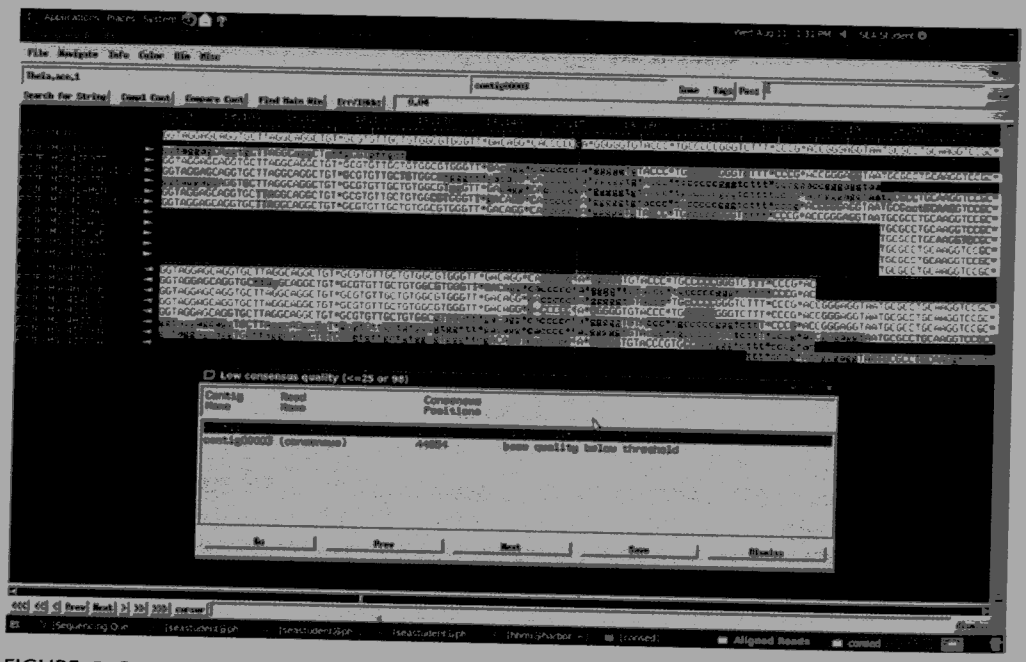

FIGURE 5. Base-pair view of a genome assembly in the 454 Newbler application. Note the strong resemblance to the Staden Gap package; see also Anders et al. (2014).

a GBrowse server. As Java software, it is easily installed by a quick download on any computing platform and it accesses data stored on a local workstation or remotely mounted network drive. Common reference genomes and genome annotations can be accessed from a server maintained by the Broad Institute, or can be manually installed. Even a draft genome composed of many thousands of **contigs** and scaffolds can be installed as a reference genome. Visualization of large data sets requires substantial amounts of RAM on the local workstation (see Fig. 6). IGV has become the most commonly used workhorse of NGS data visualization.

GenomeView

An important new addition to the visualization packages for a next-generation stand-alone genome browser is GenomeView. The package provides numerous important features including the interactive visualization of sequences, annotation, multiple alignments, syntenic mappings, and short read alignments. Support has been provided for many standard file formats and a plug-in system allows new functionality to be added as user needs evolve.

Popularity of this package has increased significantly as its enhanced visualization features have become well known. Important among these are the fact that BAM

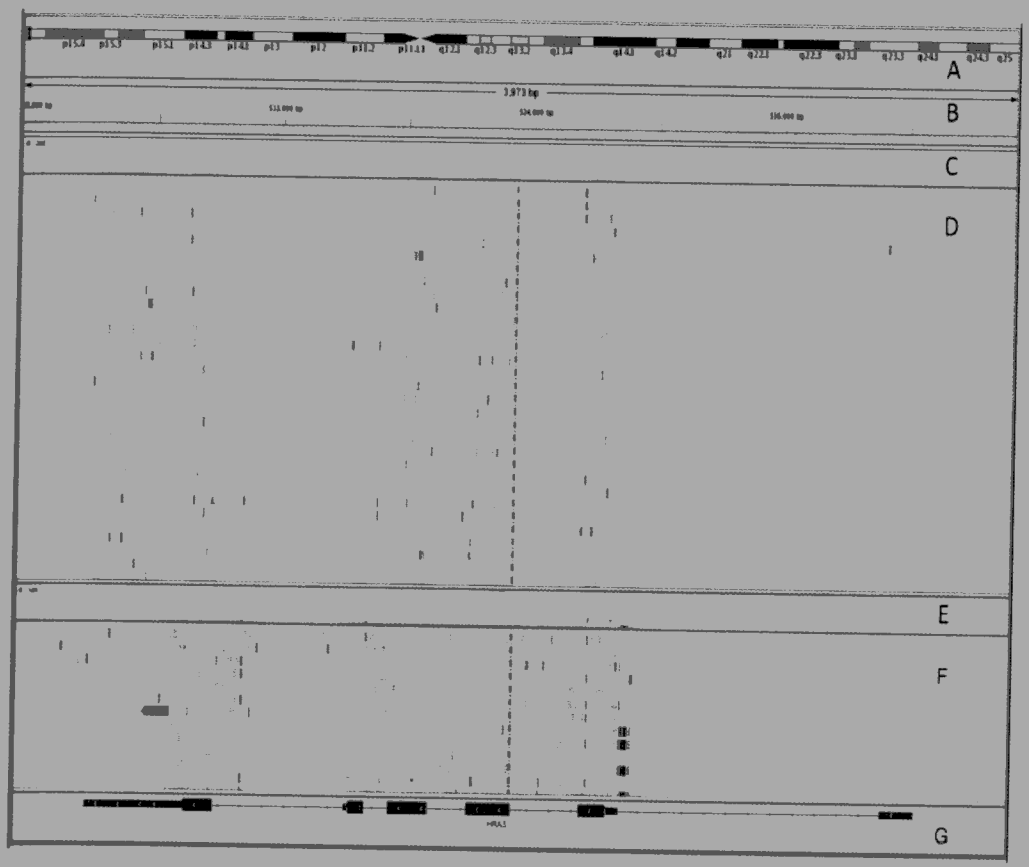

FIGURE 6. This is the main window of the Integrative Genome Viewer (IGV) from Broad Institute. (*A*) Chromosome overview of human genome hg19, Chr11 is shown here. (*B*) Zoomed-in view of a ~4 kb segment on Chr 11. (*C*) The coverage of TopHat-aligned transcripts on the genome. (*D*) RNA-seq reads aligned to the reference genome with thin lines indicating reads split across introns. (*E*) The coverage of Exome capture data aligned to the genome. (*F*) Exome reads aligned to the genome. (*G*) Compact view of *HRAS* RefSeq gene model.

files can be read and written, and the user's ability to zoom rapidly and smoothly from the level of a chromosome down to individual bases in a multiple alignment analysis. The improvement in usability plus the fact that a large-memory version of the package is available largely remove any limitations to the projects that can be tackled using state-of-the-art methods.

Development of the package was initiated by Thomas Abeel in 2008 at the Bioinformatics & Systems Biology (BSB) group at VIB (Viaams Instituut voor Biotechnologie) and now continues at the Broad Institute, his current institution. The package is written in Java and can be initiated from a local copy of the software or using a "webstart" from http://www.genomeview.org, the homepage of the package (see Fig. 7).

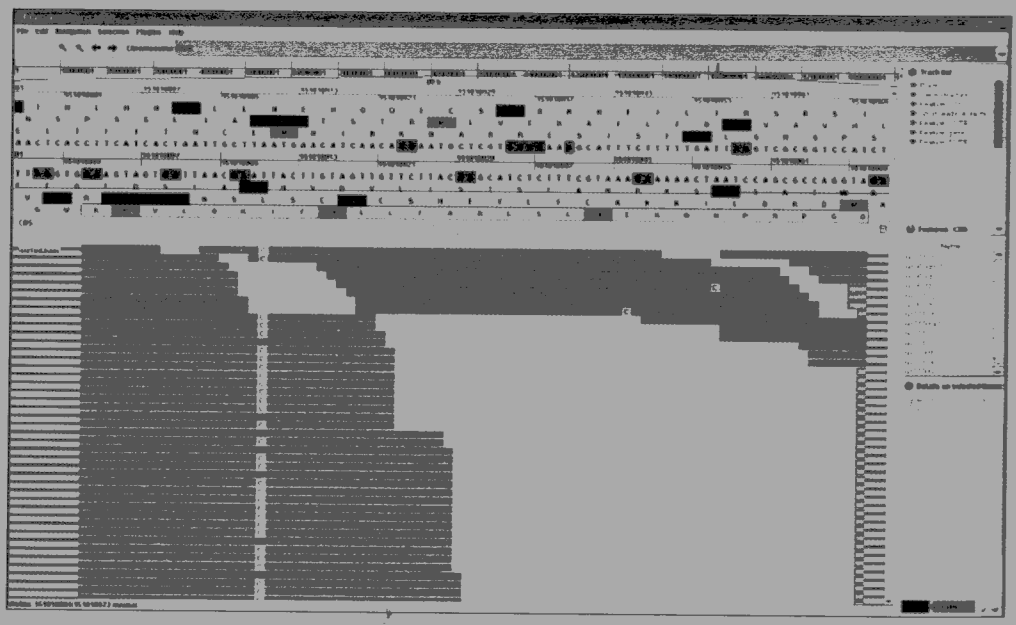

FIGURE 7. GenomeView visualization of **Illumina sequencing** reads to inspect a sequence variant.

Use of GBrowse at a Sequencing Core Facility

The Generic Genome Browser (GBrowse) is one of the software components from the GMOD community. It was developed by Lincoln Stein and coworkers (Stein et al. 2002) as a web-based genome visualization tool, and therefore is platform-independent software. It is widely used by most of the model organism genome sequencing and annotation projects such as WormBase (http://wormbase.org), FlyBase (http://flybase.org), The *Arabidopsis* Information Resource (www.arabidopsis.org), etc. GBrowse is a scalable web application to efficiently view large amounts of sequence information in a user-friendly graphical format. Standard reference genomes (human genome hg19, mouse genome mm9, etc.) can be stored on the server and used to annotate any uploaded data file. Gene structures (**introns**, exons, UTRs, etc.), microarray expression data, RNA-seq data, etc., are represented in uniquely identifiable images called glyphs, and users can search for the landmark of their interest and browse by scrolling and zooming around the sequence tracks. Additional tracks that provide summary annotation for experimental samples can be computed automatically by standard GBrowse tools (GC content, six-frame translation) or uploaded as additional GFF files.

GBrowse is written in Perl and is highly customizable according to the user's specific needs. The latest version of GBrowse enables users to upload custom tracks and

annotations without the involvement of the maintainer/system administrator. The option of uploading files through remote URLs allows the user to visualize data stored on large-scale devices located in data centers at their institution, or elsewhere on the internet, without the need to download the data files to their desktop computer.

A local installation of GBrowse can facilitate the distribution and analysis of sequence information at a research institution. Scientists can access sequence data files located on a network storage server via a web browser in the GBrowse visualization (see Figs. 8 and 9). The following workflow has been implemented at the sequencing core facility at NYULMC.

1. Customer's sample is sequenced using any NGS technology at the core laboratory.
2. Automated base-calling and demultiplexing using manufacturer-supplied software are performed.
3. After necessary quality-control steps, the reads are either aligned to the reference genome (ELAND or BWA) or a de novo assembly is performed.
4. The data are then converted to one of the formats compatible with GBrowse, preferably GFF3.
5. The data are stored on the high-performance computing (HPC) storage cluster and the location of the GBrowse formatted file (e.g., GFF3) is uploaded as a custom track remotely.

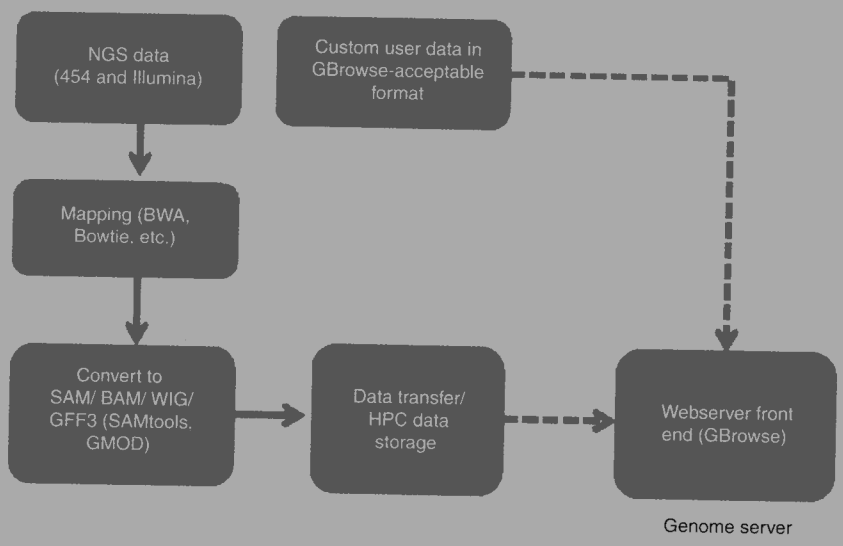

FIGURE 8. Data workflow for a locally installed instance of GBrowse visualizing data stored on a network device located in the HPC Center.

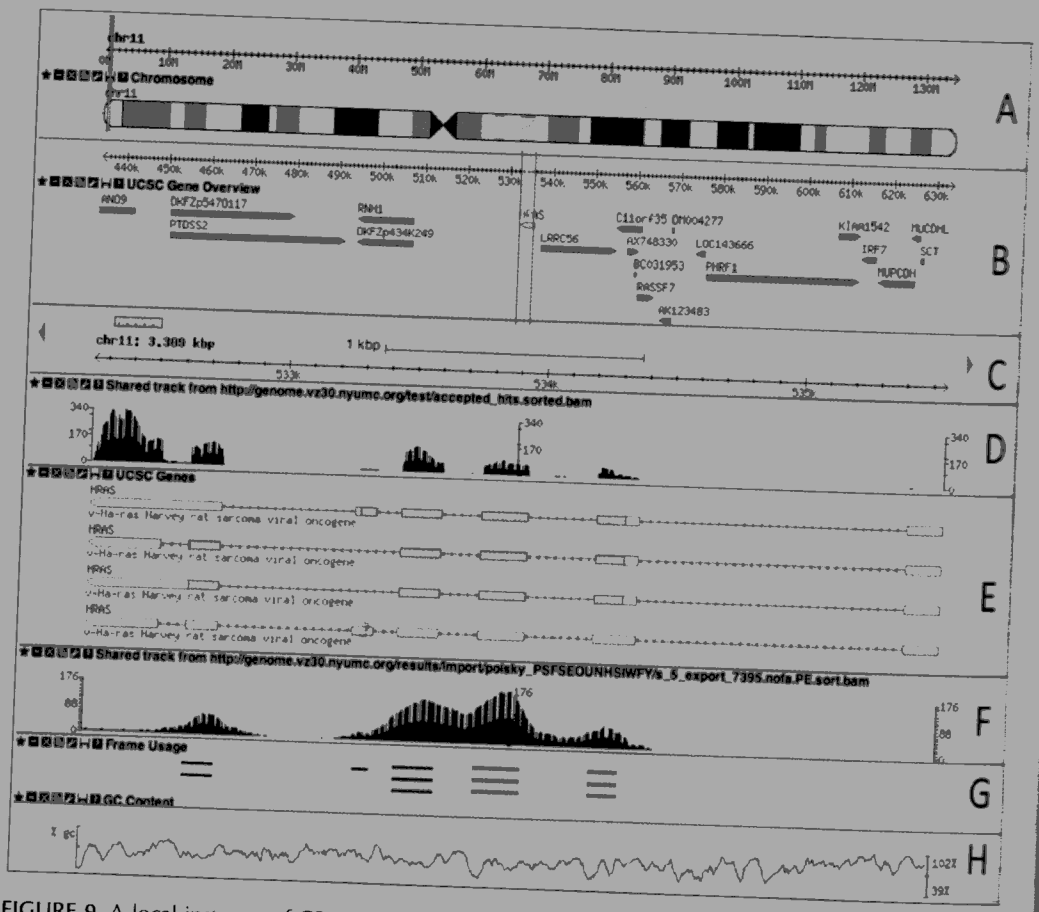

FIGURE 9. A local instance of GBrowse showing lanes of experimental RNA-seq and genomic data. (*A*) Bird's-eye view of chromosome 11 with the region of interest indicated by two vertical red lines. (*B*) A zoomed-in view of the gene *HRAS* (highlighted in yellow) between coordinates 530,000 and 540,000 on chromosome 11, with neighboring genes from UCSC. (*C*) The finer zoomed-in view of the *HRAS* gene region with 1-kbp scale and options to scroll left and right. (*D*) A shared track (loaded by URL from a network storage device) displaying Illumina RNA-seq reads as TopHat/Cufflinks assembled transcripts in SAM/BAM format. Peaks are clearly visible aligning with the exon regions of the transcripts below. (*E*) The gene *HRAS* and its different transcript isoforms with defined gene structures (exons are in box shape, CDS are blue, and UTRs are white.) Small arrows in the intron regions and the skewed box to the *left* indicate the direction of transcription. (*F*) Shared Illumina Exome sequencing data in SAM/BAM format, showing the depth of coverage across annotated exons. (*G*) CDS reading frame in "musical staff" notation. (*H*) %GC content of the current view region.

6. The link is then shared with the investigator for further examination and analysis.
7. The custom tracks can be made permanent as part of the reference GBrowse (hg19, hg18, mm9, mm8) or hosted as a password-protected browser.

REFERENCES

Abeel T, Van Parys T, Yvan S, Galagan J, Van de Peer Y. 2012. GenomeView: A next generation genome browser. *Nucleic Acids Res* **40:** e12.

Anders S, Pyl PT, Huber W. 2014. HTSeq—A Python framework to work with high-throughput sequencing. *Bioinformatics* doi: 10.1093/bioinformatics/btu638.

Darling AE, Mau B, Perna NT. 2010. ProgressiveMauve: Multiple genome alignment with gene gain, loss, and rearrangement. *PLoS ONE* **5:** e11147.

Robinson JT, Thorvaldsdóttir H, Winckler W, Guttman M, Lander ES, Getz G, Mesirov JP. 2011. Integrative genomics viewer. *Nat Biotechnol* **29:** 24–26.

Staden R, Beal K, Bonfield JK. 1998. The Staden package. Computer methods in molecular biology. In *Bioinformatics methods and protocols* (ed. Misener S, Krawetz SA), Vol. 132, pp. 115–130. Humana, Totowa, NJ.

Stein LD, Mungall C, Shu S, Caudy M, Mangone M, Day A, Nickerson E, Stajich JE, Harris TW, Arva A, Lewis S. 2002. The generic genome browser: A building block for a model organism system database. *Genome Res* **12:** 1599–1610.

WWW RESOURCES

http://asap.ahabs.wisc.edu/software/mauve/ Mauve, a system for efficiently constructing multiple genome alignments in the presence of large-scale evolutionary events such as rearrangement and inversion. Genome Evaluation Laboratory, University of Wisconsin, Madison.

http://contig.wordpress.com/2010/02/09/how-newbler-works/ Newbler, how it works. An assembly of reads, contigs, and scaffolds. A blog on all things newbler and beyond.

http://flybase.org FlyBase, a database of *Drosophila* genes and genomes.

http://genome.ucsc.edu/goldenPath/help/wiggle.html Wiggle Track Format (WIG), UCSC Genome Bioinformatics.

https://github.com/samtools/hts-specs GitHub samtools/hts-specs. Specification of SAM/BAM and related high-throughput sequencing file formats.

http://gmod.org/wiki/GFF3 GFF (generic feature format) is a standard file format for storing genomic features in a text file.

http://qed.princeton.edu/main/Wiggle_(BED)_files Wiggle (BED) files page, Princeton University.

http://samtools.sourceforge.net/ SamTools, SourceForge. SAM tools provide efficient utilities on manipulating alignments in the SAM (Sequence Alignment/Map) format.

http://staden.sourceforge.net Staden package on the SourceForge site.

http://useq.sourceforge.net/useqArchiveFormat.html USeq Compressed Binary Data Format 1.0, a zip-compressed "directory" containing genomic data split by chromosome, strand, and slices of observations.

http://www.genomeview.org GenomeView is a stand-alone sequence browser (see Abeel et al. 2012).

http://www.illumina.com/software/genomestudio_software.ilmn GenomeStudio software package (Illumina) is used to visualize and analyze sequencing and microarray data.

http://www.ncbi.nlm.nih.gov/projects/geo/query/acc.cgi?acc=GSE13047 Genome-wide mapping of in vivo protein–DNA interactions, GSE13047.

http://www.sequenceontology.org/gff3.shtml Generic Feature Format, version 3 (GFF3), Sequence Ontology Project, version 1.31, February 26, 2013, L. Stein.

http://en.wikipedia.org/wiki/FASTA_format FASTA format, from Wikipedia, is a text-based format for representing either nucleotide sequences or peptide sequences, in which nucleotides or amino acids are represented using single-letter codes.

http://en.wikipedia.org/wiki/FASTQ_format FASTQ format, from Wikipedia, is a text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores.

http://wormbase.org WormBase, insights into nematode biology, version WS243.

www.arabidopsis.org The *Arabidopsis* Information Resource Page (TAIR) maintains a database of genetic and molecular biology data for the model higher plant, *Arabidopsis thaliana.*