Visualization of the Differences Between Many Genome Sequences

Michael DiBernardo*

Department of Computer Science
University of British Columbia

ABSTRACT

As high-thoroughput sequencing technology continues to advance, large-scale sequence comparisons are becoming more commonplace. There do not yet exist any visualization tools that are capable of compactly displaying the differences between thousands of genome sequences.

In this paper, we take the first steps towards a solution to this problem by attempting to design a static overview to provide a detailed summary of the sequence differences between HIV viral genomes extracted from two different populations. The key challenges that are inherent to the problem of large-scale sequence comparison are identified and discussed to provide a framework for further research.

Keywords: information visualization, biological sequence analysis, genome browsers, catastrophic failures

1 INTRODUCTION

The publication of the first draft of the human genome sequence was rightfully considered to be a milestone event in the history of molecular biology. In the five years that have passed since then, much work has been done to interpret this data so that it can be leveraged to develop new diagnostics and therapies.

However, as high-thoroughput techniques continue to improve, there is an increasing focus on retrieving and analyzing *multiple* copies of a genome sequence from specific population or individuals from a species, instead of sequencing a single 'consensus' genome to represent the entire species. For example, the field of pharmacogenomics is concerned with examining the particular genetic inheritance details of a patient in order to develop a custom treatment protocol for that patient. Another application is the comparison of differences between different gene versions or *alleles* across human populations: the recently completed HapMap project sequenced the genomes of 270 individuals across four populations with the purpose of catalouging as many of these differences as possible [2].

The space of useful large-scale analyses of this sort is not restricted to the comparison of human genomes. Amidst the large population in Africa that is exposed to AIDS, there exists a small subpopulation that appears to be completely immune to the disease. Researchers at the British Columbia Genome Sciences Centre have isolated thousands of HIV samples from both immune and susceptible populations and have sequenced the genomes of all of these viral samples. The question that they seek to answer is if the viruses in the immune population differ genetically from the viruses isolated from the susceptible population. This analysis is being done by comparing each sample against a single canonical HIV sequence and determining what the differences are relative to this canonical sequence.

*e-mail: mddibern@cs.ubc.ca

Interpreting volumes of sequence data of this magnitude is a difficult task, and one in which visualization tools would undoubtedly be of use: One need only skim quickly through any introductory textbook in molecular biology to see that biologists quite frequently reason about sequence properties using sketches and other visual abstractions. While there are many extant sequence data visualization tools, most have been designed with the intention of comparing a small number of sequences in great detail, and very few allow a comprehensive simultaneous overview of thousands of sequences.

In this paper, I investigate a specific instance of the problem of large-scale sequence difference visualization. The problem being addressed is the comparison of HIV viruses isolated from the immune and susceptible populations described above. The researchers performing this analysis expressed a desire for a single, static overview that could be used to succinctly describe the important aspects of the data in a figure in a journal publication; however, such an overview would also be extremely useful as a view in an exploration tool. The major contribution of this work is not an effective solution to this problem, but it has the benefits of identifying the primary challenges in designing such an overview and of suggesting possible directions for further research.

Before tackling this problem, however, it is beneficial to have a general awareness of the work that has already been done in the field of biological sequence data visualization so that we might identify approaches that are particularly effective. This related work is discussed in the following section. I then provide a more detailed description of the problem at hand in section 3. Section 4 describes the proposed solution, and sections 5 and 6 discuss the implementation and the evaluation of this solution. The concluding section summarizes the strengths and weaknesses of my approach and identifies the central difficulties that researchers will need to address in order to construct an effective visualization tool for high-volume sequence comparison data.

2 RELATED WORK

Visualization tools for biological sequence analysis are typically used to browse sequence *alignments* or sequence *features*. Alignment viewers are used to identify features that are shared among sequences for the purposes of generating an annotation, while a feature viewer is used to explore and modify a mature sequence annotation.

A sequence alignment can be viewed as a relation among two or more biological sequences that maps each nucleotide or amino acid in a sequence to a nucleotide or amino acid in all other sequences in the set. If one sequence contains a contiguous segment of nucleotides or amino acids that another does not, this difference can be interpreted as an *insertion* in the first sequence, or a *deletion* from the second sequence. The basic representation of a multiple sequence alignment across two or more genomes is a textual description of the alignment, as in Figure 1. Insertions and deletions, or *indels*, are identified by the use of a *gap character*, usually a '-'. Alignment viewers essentially take a textual alignment representation like the one in Figure 1 as input, and produce a visual representation or summary of the alignment.

A sequence feature is merely a region or attribute of interest in

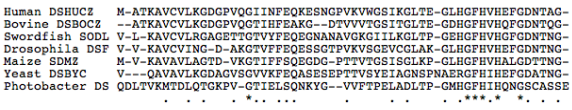

Figure 1: An example of the raw output of a multiple sequence alignment program (taken from the clustalw documentation)

a particular sequence. Classes of sequence features include *genes* and *intergenic regions*, as well as *GC-rich* regions (blocks of sequence that have a relative abundance of guanine and cytosine nucleotides) and *single-nucleotide polymorphisms* (single nucleotide substitutions that are known to occur regularly in different instances of a particular species). Sequence features are often inferred by sequence alignment, and are the building blocks that biologists and bioinformaticians use to make inferences about the function of a sequence region.

2.1 Alignment Viewers

There are a variety of tools available to visualize sequence alignments, although many of them share common features. The NCBI MapViewer [15] serves as a browser for genome assemblies, which are essentially collections of alignments of sequence fragments used to infer the entire genome sequence. Of particular interest to this project is the Evidence Viewer (Figure 2) that identifies sequence features that have been inferred from alignments of known messenger RNAs to a genomic region. Sequences are represented as lines, and raised 'blips' in the lines are used to identify mismatches. The interface is rather spartan, and no overview is provided.

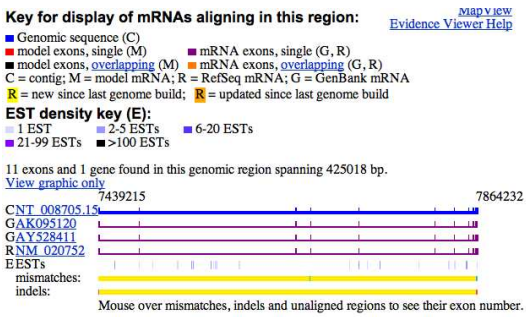

Figure 2: The NCBI Evidence Viewer

The VISTA alignment viewer [9] is a more full-featured tool that is used specifically for processing raw alignments. A sliding window of a preset number of nucleotides is moved a nucleotide at a time over the alignment, and the percent identity of each sequence with respect to some 'base' sequence is computed in that window. The percent identity is then plotted as a line graph (Figure 3) so that one may quickly identify highly conserved and highly divergent regions. Colour is used to distinguish coding and intergenic regions. Although elegant, this approach does not scale to more than a handful of sequences. Also, sequences at the top and bottom of the display are more difficult to compare than those that are directly adjacent to one another.

Synplot [5] is another basic alignment viewer that relies on percent-identity plots (PIPs) to express sequence differences. However, Synplot also explicity represents insertions and deletions as gaps in the lines representing the first or second sequence, respectively. Colour is again used to highlight sequence features, although some spatial encoding is also used to emphasize coding regions (Figure 4).

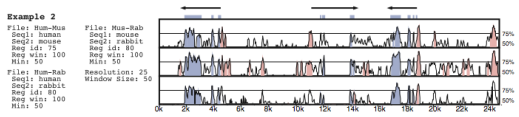

Figure 3: The VISTA alignment viewer

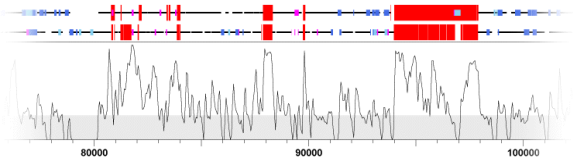

Figure 4: The Synplot alignment viewer

KBrowser [17] is essentially a feature viewer that compares the features of multiple genomes simultaneously; however, alignment data (in the form of percent identity plots) is also displayed. KBrowser unfortunately lists all of the data tracks for a single genome in a continguous block, instead of interleaving the tracks for each genome (Figure 5). This makes it very difficult to compare specific sequence features to one another. Displaying indels as grey bands that stretch across all tracks is effective, but because substitutions are only shown across one track it appears as if they are less important than indels, which may or may not be the case depending on the problem being addressed.

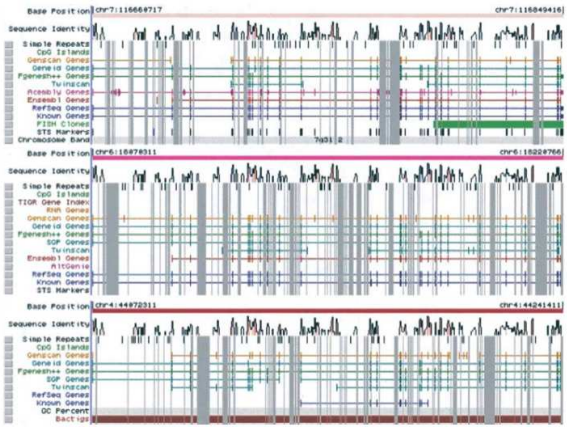

Figure 5: The KBROWSER alignment and feature viewer

ECRBrowser [11] is a multiple alignment viewer that seeks to emphasize evolutionarily conserved regions across vertebrate genomes, regardless of position or function. These regions are identified by shared peaks in PIP plots and are outlined with a box when detected by the software (Figure 6). Such an ECR can then be easily extracted and viewed in more detail. Making this particular operation a fast one is a good design decision, because the extraction of conserved sequence regions is almost always the purpose of a sequence analysis. A chromosomal overview is linked to the main view to maintain positional context, and features of sequence regions are again encoded with colour.

While the browsers that have been discussed thus far have their own individual strengths and weaknesses, they all share a common 'look and feel'. Sequences are displayed as lines, sequence features are displayed on surrounding lines or 'tracks', and percent-identity plots are used to display an overview of sequence similarity. While this homogenuity makes it moderately difficult to select one tool

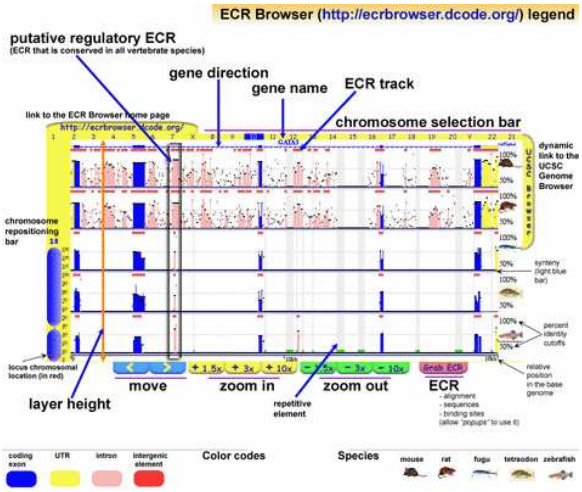

Figure 6: The ECR browser and its visual components (taken from http://ecrbrowser.dcode.org/)

over another, it does suggest that there are certain types of displays that biologists are very familiar with when it comes to sequence analysis.

However, some alignment viewers have sought to break the mold in an attempt to overcome the inherent weaknesses of this approach. SequenceJuxtaposer [16] uses an 'accordian drawing' rubber sheet metaphor to display a large number of sequences at one time while maintaining some overall context (Figure 7). Regions of interest can be identified across some number of sequences by drawing a box around them and 'stretching' them so that they are made larger with respect to their surroundings. Nucleotides are represented as colour-coded boxes, and when they are large enough in the display, their single-letter codes are rendered as text within the box. Indels are encoded as greyed-out boxes in the sequences that do not share the insertion. The rubber-sheet metaphor permits the display of a very large number of sequences at a very high resolution.

Figure 7: SequenceJuxtaposer

SockEye [10] is a 3D alignment viewer that differs from the 2D alignment viewers discussed thus far by using the third spatial dimension to encode the significance of the alignment at a given region (Figure 8). However, the perspective view makes it difficult to compare the thin vertical bar charts that represent the alignment quality, and the fact that they are rendered with transparency only amplifies this problem. No attempt is made to use the third dimension to alleviate the difficulty of having to lay sequences side-by-side for direct comparison; sequences at the top and bottom of the layout are still much more difficult to compare against each other than those that are directly adjacent.

The biological arc diagram [20], known as a BARD, was developed to reduce the difficulty of comparing sequences that are not laid out directly adjacent to one another. Arcs are drawn between aligned bases in line representations of the sequences that compose

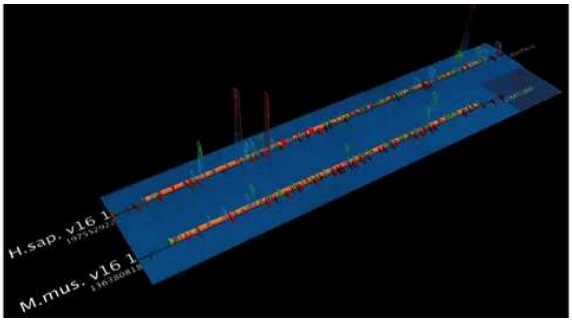

Figure 8: The SockEye alignment viewer

the alignment (Figure 9). However, this results in frequent line crossings that usually have little or no significance in the structure of the alignment, giving a general impression of visual clutter. Also, effective comparisons are still easier between adjacent sequences, because one does not have to follow the arcs as far.

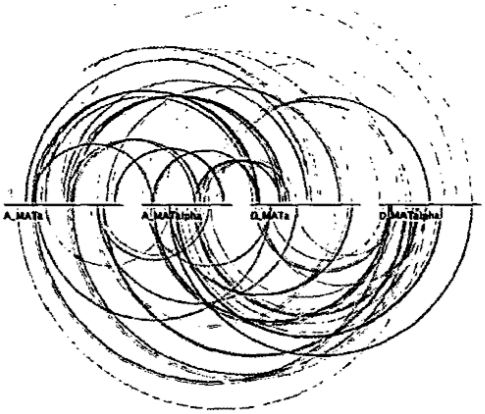

Figure 9: A biological arc diagram representing a multiple sequence alignment across four sequences

Some researchers have also applied dimensionality reduction techniques to alignment display. For example, Tsalenko et al. were interested in classifying certain sequence divergences with respect to their ability to be used as markers for predisposition to diabetes [14]. Thus, instead of using a standard display method to identify potential divergent regions of interest, they instead enumerated all combinations of computationally identified sequence differences and tested the efficacy of each such combination when it was used as a marker for disease. For instance, if a particular collection of SNPs was found only in the diabetic population, one would strongly expect that an undiagnosed individual that also carries these SNPs in their genome sequence would be highly predisposed to diabetes. This concept was implemented as an assessment of 'information gain' for each combination of sequence differences, and a visualization of the information gain provided by each combination was used to select the best markers.

2.2 Feature Viewers

Feature viewers differ from alignment viewers in that they are more concerned with displaying the unique features of a sequence or a set of sequences than they are with displaying differences between the sequences in the set.

Caryoscope [3] and the Progenetix.net browser [18] are simple feature viewers that allow a high-level overview of an entire

genome with respect to a single quantitative variate of interest. The genome is segmented into chromosomes, and coloured bar charts on each side of a chromosome indicate the value of the variate in the contained region (Figures 10 and 11). Caryoscope allows the depiction of any sequence feature that can be described in a standard format known as GFF (General Feature Format); Progenetix.net displays the relative sequence loss and gain in tumour sequences with respect to the 'healthy' chromosomal sequence. Neither browser allows a more detailed view, apart from zooming in on an individual chromosome. I evaluated the freely-available Caryoscope software myself, and although it is elegant, the inability to provide more detail is frustrating, and performance degrades heavily when using even reasonably-sized datasets (such as those provided on the author's website).

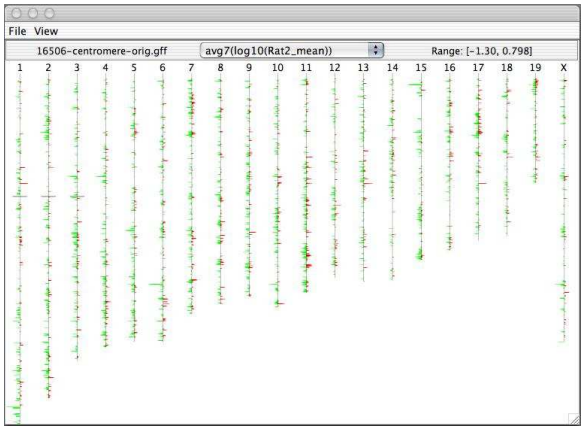

Figure 10: The Caryscope single feature viewer

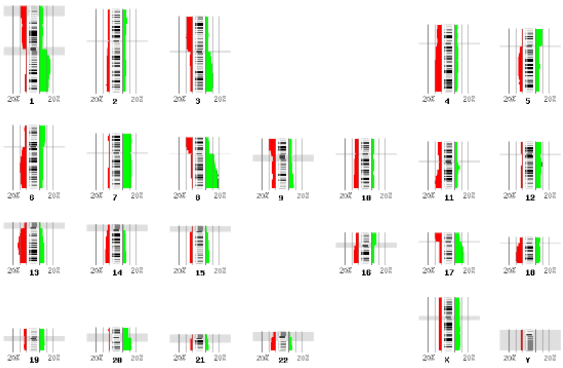

Figure 11: The Progenetix browser

The ENSEMBL map viewer [6], the UCSC Genome Browser [7], and the open-source GBROWSE [13] are track-based feature viewers that share many similarities. All provide a chromosome-level overview of the genome being examined, and allow multiple levels of zoom in the secondary view (Figures 12, 13, and 14). The user can add or remove feature tracks by manipulating a feature list. Glyphs on tracks are usually hyperlinked to specialized views: For instance, clicking on a glyph that represents a SNP will open a window that displays the list of known organisms that share the SNP, and any potential physiological differences that have been attributed with it. These viewers provide a great deal of detail that is easily interpreted for a small number of sequences (four or five on an average-sized display), largely because most of the data is spatially encoded. The disadvantage of these viewers is that they are delivered via dynamically generated HTML, and so any changes to a view require the page to be refreshed. Because most browsers will return focus to the top of the page after a refresh, adding or removing a feature track requires that users reorient their view before continuing with their exploration.

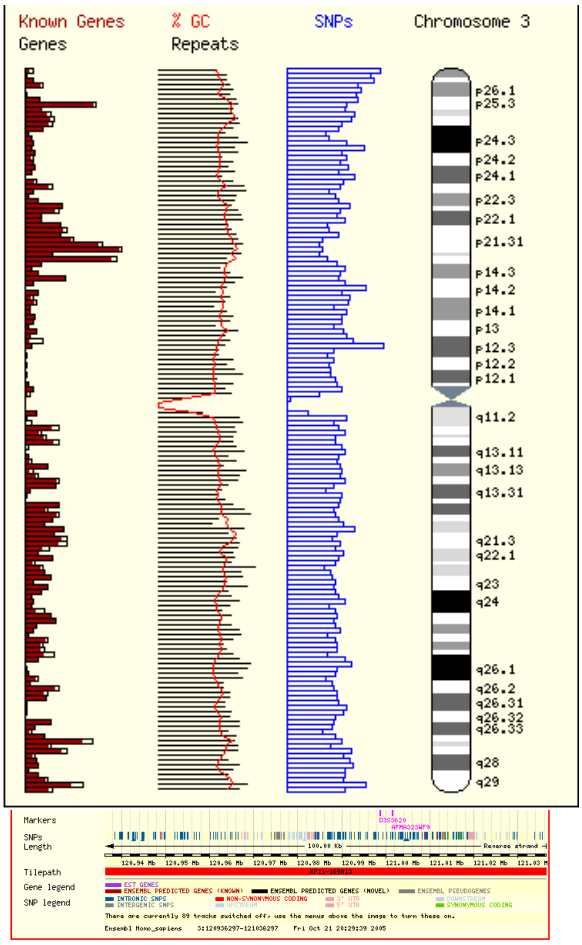

Figure 12: Overview (above) and SNP feature track view (below) of the ENSEMBL map viewer

The SNP Launcher [4] is a feature viewer that provides very detailed information about the quality of SNP assignments in a variety of genomes. All of the expressed sequence tag (EST) data that was used to infer the existence of a particular SNP is visualized spatially as a straight line beneath the canonical genome sequence. The relative amount of sequence evidence is depicted as a bar chart above the line that represents the genome. Clicking on a particular EST or genome region will open a window with a textual alignment view; unfortunately, there is no way to rearrange the order of the sequences presented in this detailed view. Further, SNPs in low- and high-covered regions are distinguished by the case of the character that represents the nucleotide, a difference that is difficult to pick out preattentively. SNPs in coding regions are coloured red, a common practice in many viewers, but the saturated red is hard to distinguish from the black background. There is no single overview that can be used as an entry point to the view depicted in Figure 15: One must specify a region of interest via a textual search query in an HTML form. Finally, the windows that are opened by the application are very difficult to close: A small square button in the lower-right of the window must be clicked, which is nonintuitive in almost any user interface.

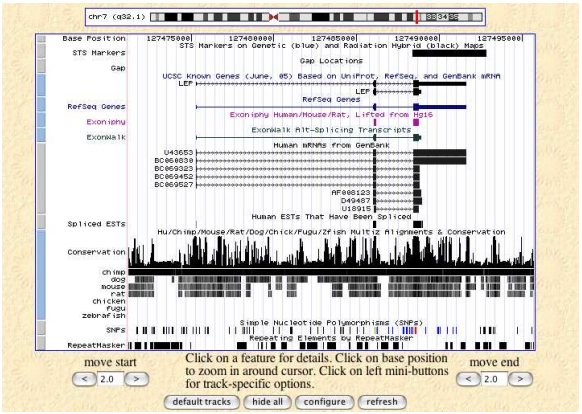

Figure 13: The UCSC genome browser

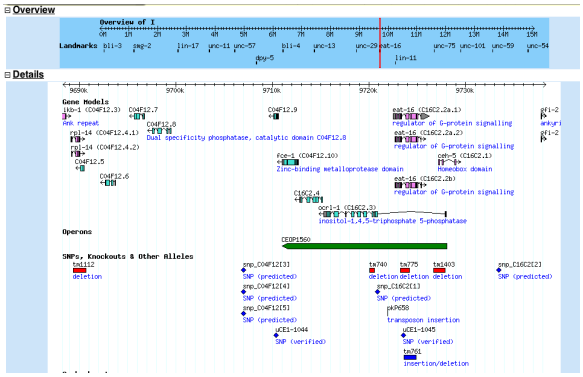

Figure 14: The GMOD GBROWSE feature browser

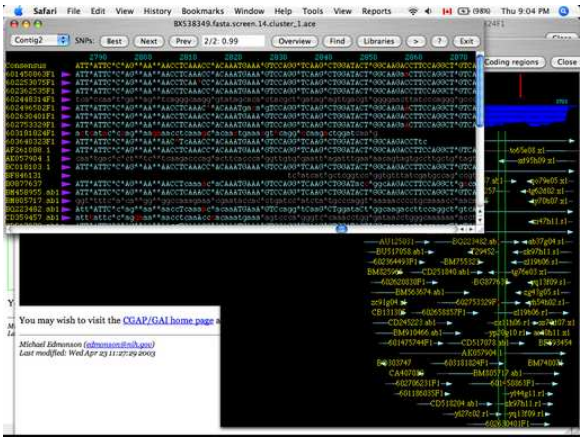

Figure 15: The SNP Launcher

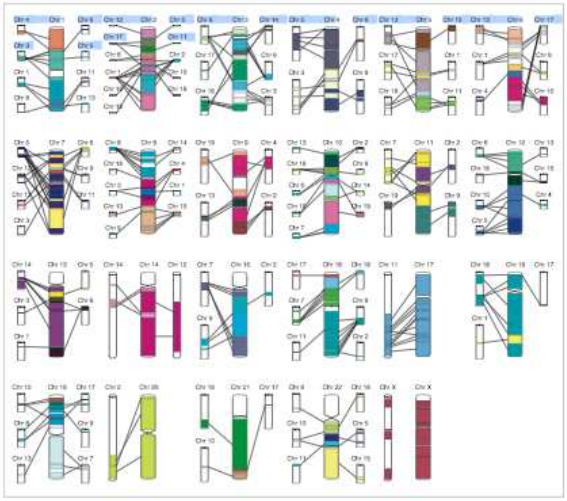

Figure 16: The Apollo chromosomal synteny view

Artemis [12] and Apollo [8] enable the user to access staggeringly detailed views of a single genome annotation. Apollo (Figure 16) provides more overviews than Artemis, such as depictions of high-level cross-chromosomal comparisons between vertebrate genomes (in this example, human and mouse), while Artemis provides a very detailed view that shows, among other things, all six open reading frames of the genome sequence at the region of interest, line-plots of GC content, hydrophobicity plots, and splicing and SNP patterns (Figure 17). In my (admittedly limited) experience working in genetics wet labs, this information is extremely useful when designing DNA sequence probes or primers that are to interact with the genomic region of interest, and so Artemis can get away with not providing any sort of overview; the intention is that it will only be invoked when a very specific region of interest has been identified.

It is evident from this survey that certain types of data have a common representation in many different tools, and that these representations are thus likely to be familiar to researchers in molecular biology. With these considerations in mind, we can move on to a discussion of the problem and a proposed solution.

3 Discussion of Problem

The description of the problem that we are trying to solve is not that complicated. The subjects of the experiment are individuals living in Africa. Each subject belongs to one of two populations, or *cohorts*. The first population is the *susceptible* cohort, and is composed of individuals who are infected with HIV and who demonstrate symptoms of the disease. The second population is the *immune* cohort, and is composed of individuals who are infected with HIV and who do not demonstrate symptomos of having the disease.

Multiple HIV viral samples were extracted from each individual in each population. These samples were processed to remove the genetic material, and each viral genome was sequenced. Each of these viral genomes was then compared to a canonical HIV viral genome sequence to determine the insertions, deletions, and substitutions in the sample relative to the canonical sequence.

The data, then, is a list of changes relative to the canonical sequence. The canonical sequence itself is about 9300 bases in length. Bases in the canonical sequence are numbered starting at 1. Each change is labelled as originating from the susceptible cohort or the immune cohort. Substitutions are described by the position of the base in the canonical sequence that was modified, and the letter code of the base that it was changed to. Deletions are described by an inclusive range of bases in the canonical sequence that were deleted or missing in the viral sample. Insertions are described by the number of the base in the canonical sequence at which the insertion begins, the length of the sequence that was inserted, and the actual sequence of bases that was inserted. For instance, an insertion of the sequence ACGTT at base 4 means that in a comparison of the canonical sequence and a viral sample, the extra sequence ACGTT was found in between the viral bases that correspond to bases 4 and 5 in the canonical sequence. See Figure 18 for a visual

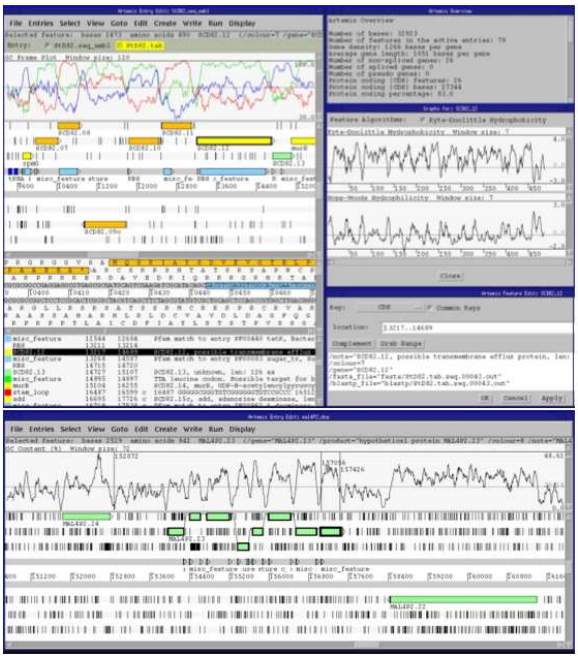

Figure 17: The Artemis feature browser

representation of this insertion; the concept is quite simple, but can be difficult to describe in words.

The problem is to find a method of visually representing all of these changes in a single overview, subject to some restrictions. This overview must be extremely intuitive to interpret, because the intention is that it will serve as a 'visual introduction' to the findings of the study. This suggests that an effective solution would use display paradigms that biologists are used to working with. We have seen in the previous section that sequence difference data is often laid out on horizontal, spatial 'tracks', and that line graphs are commonly used to display quantitative data. Abstraction techniques such as semantic zoom and glyphing would likely require too much cognitive 'overhead' to decipher for the average reader.

Data aggregation and other reduction techniques must also be used sparingly, because difference data quickly becomes useless when detail is lost. This is because small changes in biological sequences can produce profound effects; for instance, because the DNA bases that code for amino acids are read in triplets, the deletion of one or two bases is a catastrophic event because it changes the composition of the subsequent triplets. This is known as a 'frame shift' mutation. Single-nucleotide polymorphisms are perhaps the most innocuous change that one could imagine, because they only change the identity of a single base. However, many SNPs have been associated with disease or loss of function in a variety of organisms, and so even these minute changes are important.

A one sentence summary of the ideal solution, then, is a single overview that is easily interpreted by biologists and that allows for a detailed description of the changes that occured at each base in the canonical sequence.

```
1 2 3 4           5 6 7 8 9
T A C T - - - - - G G C G A
T A C T A C G T T G G C G A
```

Figure 18: Raw textual representation of an insertion in a viral sample relative to the canonical sequence. The top sequence is the canonical sequence, while the bottom sequence is the viral sample.

4 Proposed Solution

In the previous section, I described the problem and the nature of the data. Specifically, I identified the variates that are required to completely describe each of the possible sequence changes. Designing a solution to the problem required that I first select a subset of these variates to display, and then decide how to display them.

4.1 Selection of Variates

Substitutions are specified by a position in the canonical sequence, the letter of the base that was changed, and the letter that the base was changed to. For the purposes of the overview, I decided that the existence of the substitution was more important than the nature of the change itself; that is, the fact that a substitution occurs at base 24 is more important than the fact that base 24 changed from an adenine to a guanine. Thus, substitution data can be encoded as the probability that a substitution will occur at a given position in the canonical sequence.

Deletions are usually specified by a start position and a length. However, in the previous section I described a deletion as an inclusive range in a sample sequence. This is because for most applications, the fact that a base is missing from a sequence is more important than the fact that the deletion started at particular base. For instance, if in one sample, bases 10-20 are missing, and in another bases 15-25 are missing, we could make two types of observations. The first observation is that bases 10-14 and 21-25 are missing 50% of the time, whereas bases 15-20 are missing 100% of the time. The second observeration is that deletions originate 50% of the time after base 14, and 50% of the time after base 19. The latter sort of observation is useful when searching for specific 'hot spots' in the genome that are especially prone to being clipped; however, most of the time the knowledge that a base is frequently removed in a deletion of some length is more than sufficient. This is because regions that are frequently deleted can be assumed to have little effect on the continuance of the organism, and thus likely do not contribute to any particular function. Thus, I decided to represent deletions with a single variate: The probability that particular base in the canonical sequence will be missing in an extracted sequence. Deletion origins and lengths are completely ignored. The actual sequence that was deleted is also not considered to be required knowledge.

Insertions are a difficult case. Both the origin and the length of the insertion are important, while the actual sequence being inserted is perhaps of lesser concern. However, there is no simple single-variate reduction that can be used to describe an entire insertion, and so for every position in the canonical sequence, I chose to display both the probability that an insertion will occur after that particular base, and the average length of an insertion that begins at that position. Averaging lengths is a particularly dangerous thing to do, as it precludes the interpretation of length distributions.

In summary, the variates that were selected were probability of substitutions at each base; probability of deletion of each base, and probability and average length of an insertion occuring after each base.

4.2 Visual Encoding of Variates

As discussed earlier, one of the design goals was to encode the selected variates in a way that was immediately familiar to biologists. As such, the use of line and area charts and the like was preferred to the use of glyphs and other more abstract encodings. An initial mockup of the proposed solution can be seen in Figure 19.

Because the majority of sequence viewers represent the genome sequence as a horizontal line, I chose to adhere to this convention; the position of bases on the canonical sequence is encoded spatially across the x-axis. Substitution probabilities for each cohort are encoded spatially in the positive y-axis, using a line graph with two

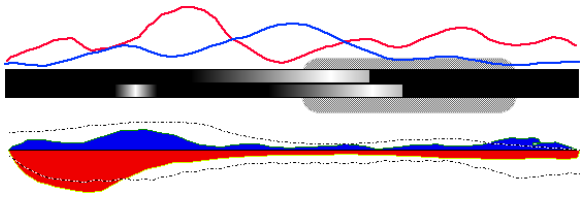

Figure 19: Mockup of proposed solution. Substitution data is represented with line graphs at top of display; insertion data is represented with area and line plots at bottom of display; deletion data is represented by gradient bars that double as the *x*-axis. Genes are represented by rounded rectangular regions behind the gradient bars.

series to maintain continuity across the sequence. Colour is used to distinguish the two populations; the susceptible population data is drawn in red, while the immune population data is drawn in blue. The human perceptual system is particularly sensitive to line crossings [1], which in this context is useful because such crossings indicate regions where the substitution probability in one population overtakes that of the other population.

Insertions are encoded spatially in the negative *y*-axis. Because insertions are represented by two variates, two data points must be plotted per column for each population. In order to reduce clutter, the susceptible population data is displayed separately from the immune population data. Insertion probabilities are plotted as an area chart, which can be viewed as a very high-resolution histogram. Area charts are used to distinguish this display from the line graphs that describe the substitution data. Again, colour is used to distinguish populations. Average insertion lengths are plotted as a line chart overtop of the area chart. A textured line is used to 'soften' the effects of the line/area crossings, as they are not informative in this display and should not draw attention to themselves. The inversion of one of the plots allows the display of 4 data points per column without a great deal of clutter, but admittedly makes it difficult to compare insertion probabilities and lengths between the two populations.

Deletions are represented with a pair of gradient maps that are positioned along the *x*-axis. The top bar represents the susceptible population, while the bottom bar represents the immune population. The lightness of the bar at a given position is used to represent the probability that the base at that position will be deleted in a viral sample. The gradient bar allows for a compact representation of the deletion data, and also serves as a visual anchor for the *x*-axis in the display.

Note that this overview could concievably be used both as a figure in a publication, or as a overview in a multi-view visualization tool; a rectangular 'window' could be placed in the overview to denote the region being examined in a more detailed view, and this window could be moved by clicking and dragging the mouse.

5 Implementation

The view described in the previous section was implemented using Java 1.4.2. A model/controller/view architecture was adhered to even though the current state of the application does not permit interaction: This was done to make future extensions to the software as easy as possible.

After evaluating many visualization toolkits, I decided to use the JFreeChart SDK to implement the line and area charts used in the substitution and insertion views. I could not find any existing software to create the gradient bars for the deletion plots, and so I implemented those myself using the Java2D library.

The data itself is stored in the General Feature Format (GFF), at the request of my collaborators at the BC Genome Sciences Centre. The GFF is a rigid, plain-text, human-readable file format that is used specifically for describing sequence features.

5.1 Challenges Encountered

Several challenges were encountered in the process of translating the high-level design into software. These challenges and my attempted solutions are described here. The efficacy of the solutions is evaluated in the following section.

5.1.1 Lack of Critical Feedback

The one lesson that I have learned from working in biology wet-labs is that science can and often does go completely and horrifically awry. It is not uncommon for Ph.D projects that have an expected duration of 3 years to require more than double that time to complete. Thus, I was not especially surprised when similar problems cropped up throughout the course of this project. Very early on in the design phase, my collaborators at the Genome Sciences Centre became extremely involved in their work and were essentially unavailable throughout the duration of the project. Thus, I was forced to abandon my original plan, which was to engage in an iterative design process that would involve constant feedback from the researchers that would eventually be using the software.

I used two techniques to attempt to produce a useful solution in the absence of informed criticism. First, I performed cognitive walkthroughs of how I would use the software as a user of the system; however, these were somewhat limited in utility given my lack of domain knowledge and of the specific details of the project. Second, I requested input from friends and colleagues that I have worked with before in the fields of bioinformatics and molecular biology. Ultimately, though, there is little that I was able to do to ensure that this tool would be useful to the knowledge workers that requested it. The lack of feedback also drastically increased the time of development, because I found myself constantly evaluating and re-evaluating design decisions that I could not adequately reason about with my limited exposure to the actual research problem being addressed.

5.1.2 Lack of Real Data

Another unfortunate consequence of the project delay was that I was not able to gain access to the data that my collaborators wished to analyze. This was not a large problem in the early stages of development, since I was able to do basic testing with hand-constructed datasets. However, as the project matured, I required larger and more representative datasets to allow for evaluation of the system.

I accomodated for this difficulty in two different ways. First, I wrote software to produce datasets from probabalistic models of sequence evolution. When modelling sequence evolution, it is common to assume that individual mutation events are Poisson distributed, and that the lengths of insertions and deletions are geometrically distributed [19]. This data was useful in that it represented a 'worst case' in terms of noisiness and density: for example, Figure 20 demonstrates a scenario where the mutation rates are the same in both populations, but the indel lengths are longer in the immune population than in the susceptible population.

However, this did not allow me to evaluate the effectiveness of the system when used with real data. To this end, I designed a toy experiment based on an existing dataset that could be used as a testbed for the system. HIV genome sequences were downloaded from the HIV Sequence Compendium website[21]. A total of 75 sequences were downloaded, with 37 isolated from individuals in Kenya, 37 isolated from individuals in Uganda, and 1 isolated from an individual in Ethiopia. Ethiopia and Kenya share a border, but Ethiopia and Uganda do not; the experiment was to see if the samples isolated from Kenyan individuals were more divergent from

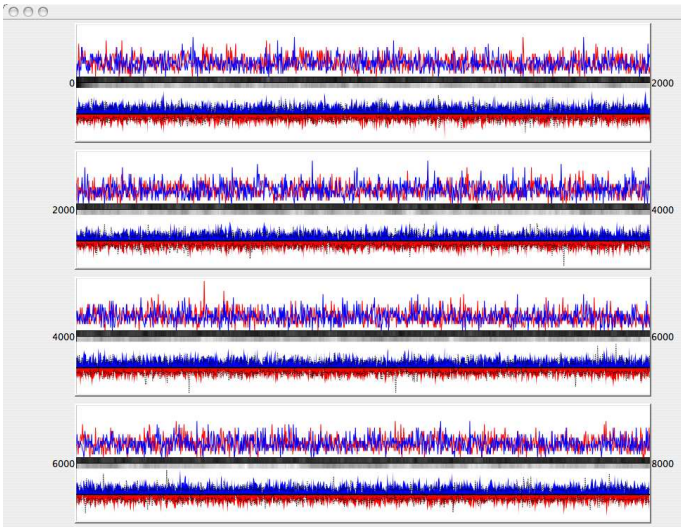

Figure 20: Display of random data. Each population consists of 5000 samples. Mutation frequencies are Poisson-distributed with a mean of 20 mutations per sample. Indel lengths are geometrically distributed with length 10 in the susceptible population (blue, top gradient) and length 20 in the immune population (red, bottom gradient)

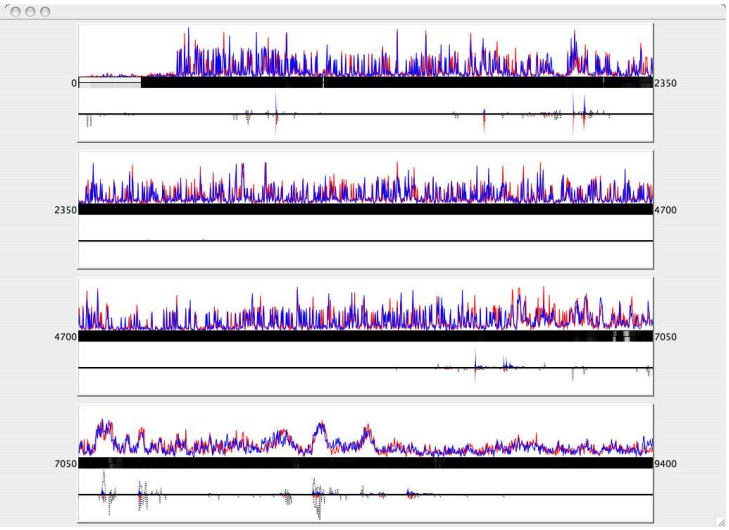

Figure 21: Display of substitute dataset. Canonical sequence is an HIV viral genome sequence extracted from an Ethiopian patient. Blue/top population is composed of 37 Kenyan HIV sequence isolates, and red/bottom population is composed of 37 Ethiopian HIV sequence isolates.

the Ethiopian sequence than the samples isolated from Ugandan individuals. In this scenario, the Ethiopian sequence serves as the canonical sequence, and the Kenyan and Ugandan sequences served as the two sample populations.

Unfortunately, I was unable to acquire a genetic map of the Ethiopian sequence, and thus I could not identify what regions of the sequence coded for genes; the original intention was display gene regions as rounded rectangles as seen in Figure 19.

The sequence data were downloaded as a multiple alignment of all 75 sequences. The sequence alignments provided by the HIV sequence compendium have been manually curated and are thus considered to be of very high quality. I wrote a Perl script to translate this multiple sequence alignment into GFF-formatted data. The results of this experiment are discussed in the following section. A view of the data can be seen in Figure 21.

5.1.3 Density of Data

As discussed earlier, the HIV genome is approximately 9300 bases in length. Because the width in pixels of the average monitor ranges from 800 to roughly 1300 pixels, it became evident early in the project that each of the positions in the canonical sequence could not be assigned a display column of even a single pixel in width. However, this sort of resolution is necessary with the proposed solution in order to display detailed features such as single-nucleotide polymorphisms.

My first attempt at a solution for this problem was to split the canonical sequence into n pieces and to create a separate display component for each segment. These components were stacked vertically and labelled with base pair ranges. However, there is a limit to how many components can be stacked vertically before the length-to-width ratios of the individual graphs become low enough to make interpretation of the data almost impossible. Figure 20 shows a display with n=4, which I found to be a happy medium when the application window is maximized to occupy the entire screen. However, there is still the issue that one column of pixels of any particular component must represent 3 to 4 base pairs of the canonical sequence, and this is in 'best-case' scenario where the application window is maximized: If this tool was to be used as a linked overview, it would only occupy a small portion of the screen space.

This problem is especially dangerous because of the reliance of the solution on the JFreeChart library. The behaviour of the JFreeChart plots is essentially undefined when there more data points to plot along a single axis than there are available pixels along that dimension. Thus, some reduction of the data was required to produced a one-to-one mapping of data points to pixels along the x-axis.

The approach that was taken was simple, and less than ideal. For all quantities that are displayed in each component, a windowed average was computed and plotted instead of the raw data. For instance, in the substutition plot, if the data density is 4 bases to 1 pixel, an average of the substitution probabilities across those 4 bases is computed and displayed. It is disconcerting that when this technique is applied to the insertion length component, this results in the computation of an average of *average* lengths, which is certainly not an intuitive quantity to reason about. A comparison of an averaraged and an unaveraged display can be seen in Figure 22. The extensive drawbacks of windowed averaging are discussed in the next section.

5.1.4 Data Distribution

A comparison between the initial conception of the display in Figure 19 and an example of the display of real data in Figure 21 makes it very clear that the mockup made some unrealistic assumptions as to the nature of the data. In the mockup, the data is very 'well-behaved' in that it forms smooth, normally distributed curves that are easy to follow and compare. However, the tendency of mutation frequencies to follow a Poisson distribution and indel lengths to follow a geometric distribution means that any practical depiction of real data will not be nearly as pleasing to look at. Especially difficult is the tendency for indels to occur infrequently in very narrow regions of sequence: This results in large, simultaneous spikes in both the insertion frequency and insertion length in very narrow regions of the chart. These spikes result in the obfuscation of the insertion probability plot by the insertion length plot, which makes

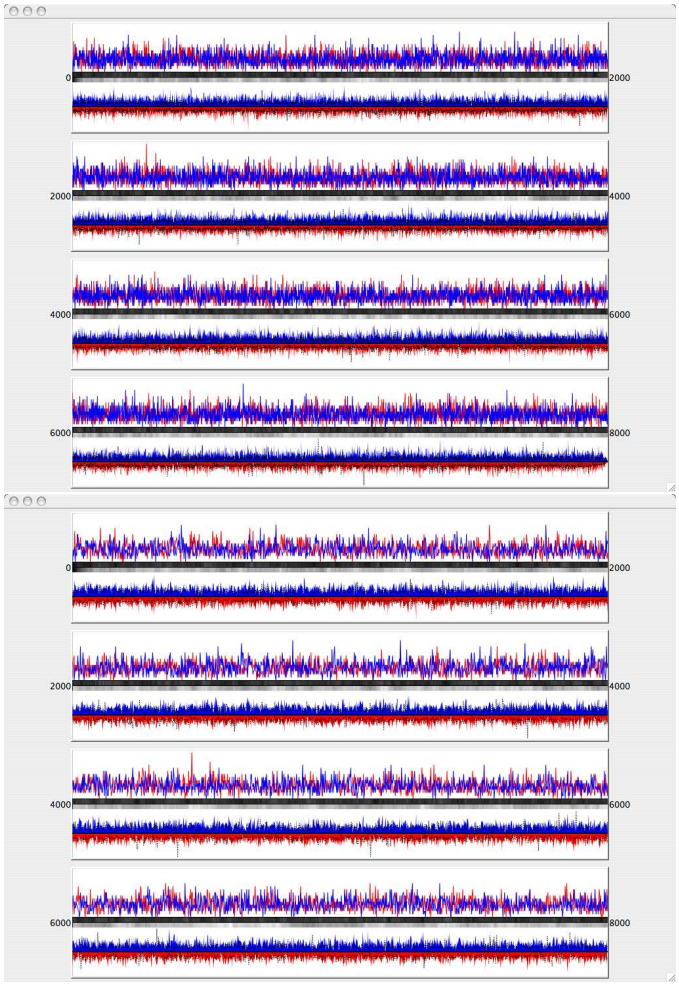

Figure 22: Comparison of raw data (top) and window-averaged data (bottom) for the random data display from Figure 20

the interpretation of these quantities extremely difficult, and additionally make it hard to compare indel lengths and probabilities that are relatively small compared to the largest data point in the plot. There is also a tendency for every third residue in a coding region to experience a heavy substiution rate because the redundancy of the genetic code means that such mutations will generally not result in a change to the translated amino-acid sequence, and so the substitution chart appears very noisy, with frequent spikes that make it difficult to pick out more salient substitutions (such as those that would actually change the encoded amino acid).

An attempt was made to reduce the difficulty of comparing insertion data by plotting them on a log scale; however, reactions to this change from my informal correspondents was overwhelmingly negative. The general consensus was that the log plot 'homogenized' the length quantities to such an extent that it made it impossible to accurately gauge what those lengths actually were, and so the decision was made to leave the linear axes in place.

5.1.5 Performance

There are four computationally intensive tasks that are performed by the system: Parsing the raw difference data, aggregating the data, storing the aggregated data in datasets that can be used by the JFreeChart renderers, and actually rendering the data. The first three tasks are performed at launch, whereas the rendering process must be repeated each time the display is resized or otherwise modified.

Table 1: Speed of system launch before and after optimization. Measurements are of average walltime, in seconds. 5 trials were run on a 1.2GHz iBook with 768MB of RAM. All numbers are with respect to random dataset displayed in Figure 20

Task	Before (s)	After (s)
Parsing	15.4	15.4
Aggregation	0.3	0.3
Dataset Population	96.2	0.02

The initial prototype of the system was implemented without any sort of optimization; emphasis was placed instead on keeping the design clean and easy to extend or modify in the event that optimization of certain components would be needed after profiling. After the first prototype was complete, a performance evaluation on the random datasets described above (composed of approximately 200 000 individidual change events) showed that the system was painfully slow in both the initialization and rendering phases.

The computational bottleneck in the initalization phase was identified as being the transfer of the aggregated data into the JFreeChart dataset objects. The JFreeChart data storage library as provided was not specifically designed to handle extremely large datasets, and so I extended it to add this functionality. A preliminary analysis of the east African HIV data described previously revealed that the data was sparse in many regions; for instance the insertion probability for most of the genome is 0. Also, data is always accessed in ascending order when rendering the plot. A linked list was thus used to store the nonzero data points, and a pointer to the previously accessed nonzero list element is maintained as part of the state of dataset. This makes the initial population of the dataset and the retrieval of the data during the rendering process much more efficient: See Table 1 for details.

The issue of rendering speed was significantly improved when the windowed averaging technique described in the previous section was implemented, because of the reduction in the number of actual data points that are rendered. However, the rendering process is still somewhat slow, requiring 4-8 seconds to redisplay the data when the display window is changed.

6 Results

In order to evaluate the strengths and weaknesses of the system described in the previous two sections, a total of four scenarios of use were conceived, using four different datasets. In each of these scenarios, the question to be answered is very simple: What is the difference between the two populations with respect to the canonical sequence?

An assessment of the accessibility of the software to individuals with colour-blindness was also performed.

6.1 Scenarios of Use

6.1.1 Finding a simple trend in the first random dataset

A random dataset was generated where both populations had similar rates of mutation, but where indels tended to be larger in one population than the other. I randomized the selection of the population with the longer mean indels so that I would not know *a priori* which population I should expect to exhibit the trend. I visualized this data with and without the windowed averaging enabled. The results can be seen in Figure 22.

In both displays, the trend that is most easily seen is that the red/bottom population more frequently experiences base deletion than the other population. While the dataset was designed to have equal mutation rates for both populations, because the deletions in red population tended to be longer, more bases were deleted on average than in the blue population, even though the *number* of deletion events was roughly equal. This demonstrates one of the problems with choosing not to display the probability of a deletion event; it is impossible to distinguish whether the high deletion rate in a region was caused by frequent, short deletions, or infrequent long deletions.

Despite the fact that both insertion and deletion lengths were larger in the red population, it is not at all apparent from either display that insertion lengths in either population differ by much. The reason for this is that each population sustained a single insertion that was extremely long compared to the others in the dataset, but relatively similar in length to one another. The scale of the insertion length plot is thus adjusted to accomodate for these very long insertions, and this makes it difficult to distinguish any sort of global difference in average insertion length between the two populations. Also, the insertion length plot is distracting in both displays because it appears to be an airbrush pattern instead of an actual line as a result of the irregular distribution of the data.

The substitution plot does not confer much information other than that the substitution rates seem to be uniform and roughly equal in magnitude across the whole canonical sequence for both populations, which is certainly an accurate description of the underlying model.

Finally, notice that the averaged plot smooths away some of the lines that were present in unaveraged plot. It was not clear to me if the extra lines in the average plot were actually informative, or if they were produced as a side-effect of the antialiasing that the JFreeChart line renderers use.

6.1.2 Finding a simple trend in the second random dataset

A second random dataset was generated where both populations had the exact same mutation rates and mean indel lengths (Figure 23). These mutation rates and indel lengths were smaller in magnitude than those in the previous dataset.

It is fairly obvious from the both displays that the probability of any mutation in both populations is lower than in the previous scenario: The lines in the substitution and insertion plots are closer to the *x*-axis, and the deletion bars are relatively dark in comparison. However, because of the density of the data in the lower regions of the substitution plot, the red population data in the unaveraged substitution display is obscured by the equally noisy blue population data that is rendered last.

6.1.3 Finding a simple trend in the third random dataset

A third random dataset was generated where one population had both a higher mutation rate and mean indel length than the other. Again, a population was randomly selected to receive the high mutation rate. The visualization can be seen in Figure 24.

In this case, it is obvious that the blue population has the higher substitution rate, deletion rate, and insertion frequency. However, there is still the issue that it is impossible to tell if the higher deletion rate is a result of longer deletions or more frequent deletions, and again the insertion lengths in both populations are not easy to distinguish from one another because of several very long insertions that expand the scale.

6.1.4 Exploring a real dataset

The dataset described in section 5.1.2 was visualized with and without the application of windowed averaging. The Kenyan sequences were assigned to the blue/top population and the Ugandan sequences were assigned to the red/bottom population. A comparison of the unaveraged and averaged views of this data can be seen in Figure 25.

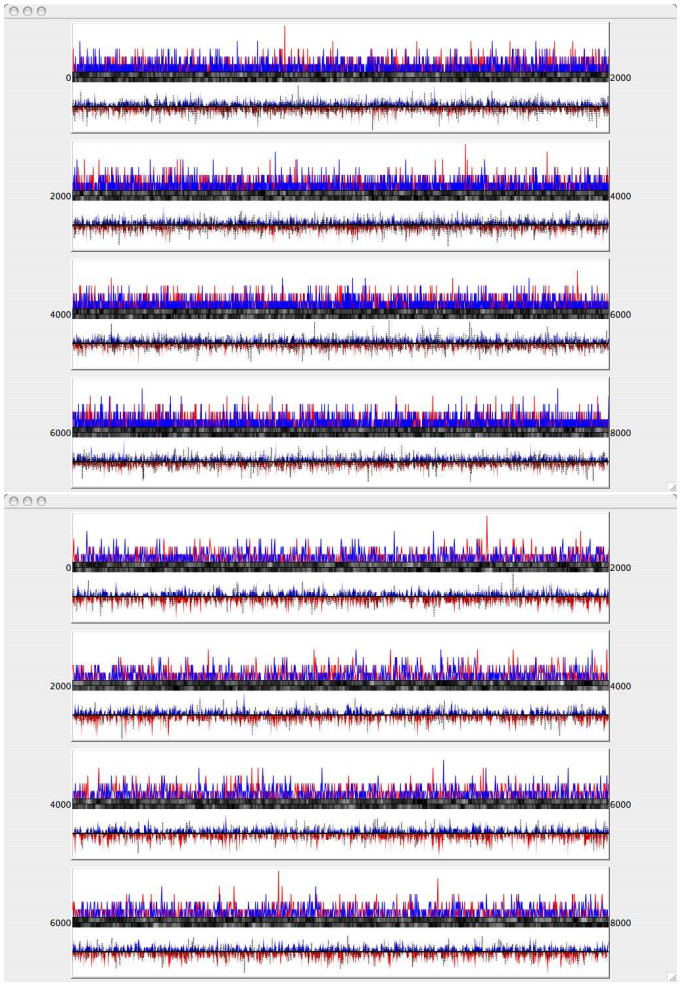

Figure 23: Comparison of raw data (top) and window-averaged data (bottom) for random data with equally low mutation rates in both populations

Several things are noticeable in both views almost immediately. Both populations frequently have a large deletion with respect to the canonical sequence at the beginning; this is not uncommon, and is usally a result of sequencing and assembly techniques and not indicative of an evolutionary difference. Deletions are rare but usually occur concurrently, except at around position 2000 where a deletion is much more common in the Kenyan samples than in the Ugandan samples. Substitution rates also occur similarly across both populations, with substitions being particularly rare at the sequence ends. That substitutions are rare near the terminal end is not as obvious in the averaged view because the overall scale is smaller and the averaging tends to 'flatten' this distribution; this is evidence that the averaging does in some way 'lie' about the true distribution to the user. Also, the substitution plots in both displays are quite divergent at times, which again implies that the windowed averaging may be producing an unrepresentative oversimplification of the data.

The insertion display holds a few surprises. Insertions happen infrequently on the whole, but there are several 'hotspots' at very narrow regions of the sequence. The unaveraged display actually hides some of these because they have a span of less than a few

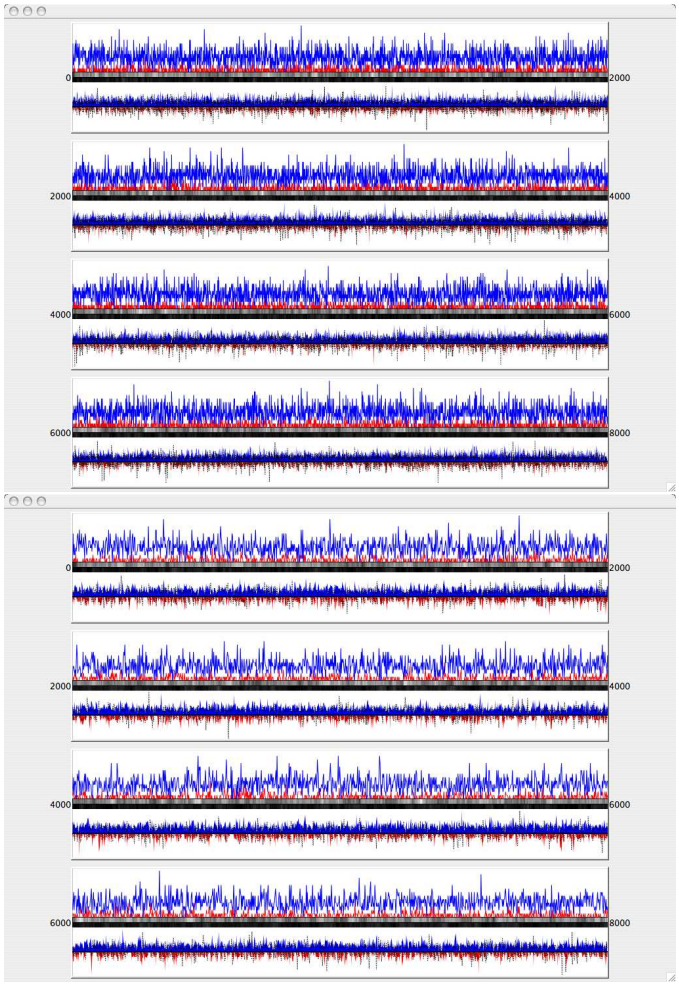

Figure 24: Comparison of raw data (top) and window-averaged data (bottom) for random data, where one population has a higher mutation rate and indel length than the other population

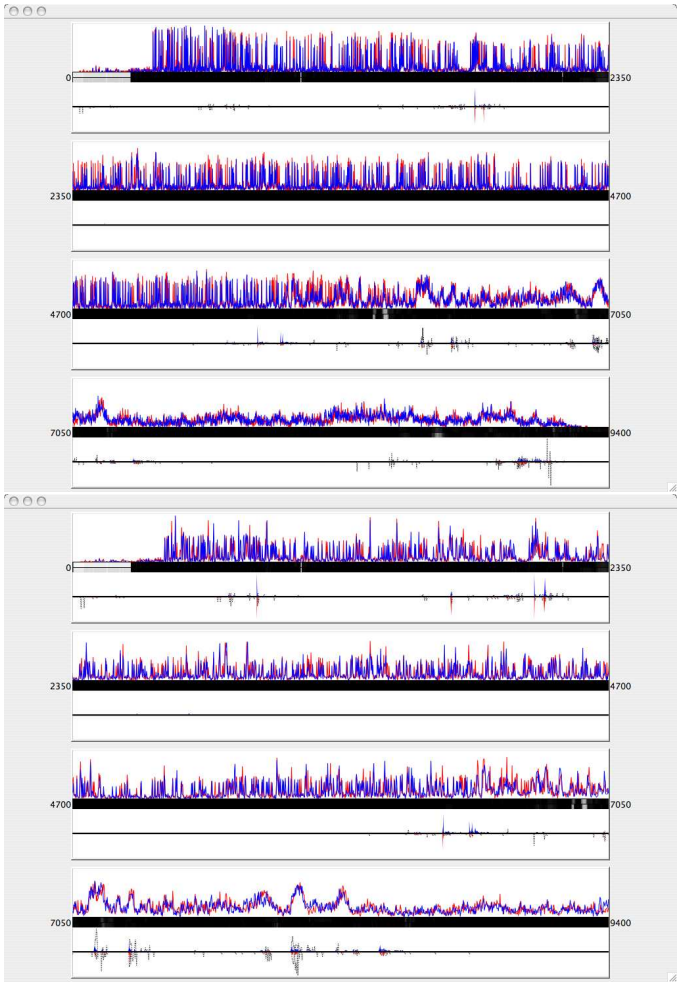

Figure 25: Comparison of 37 HIV viral genome sequences extracted from Kenyan patients (top, blue) and 37 sequences extracted from Ugandan patients (bottom, red) with respect to a single 'canonical' HIV sequence isolated in Ethiopia. Both unaveraged (top figure) and averaged views (bottom) are shown here.

bases and thus will be rendered only at the whim of the JFreeChart plot renderer. The averaged view exposes more of these insertions – particularly obvious examples can be seen around position 700, and in the last quarter of the sequence – but appears to have obfuscation problems of its own, because very small insertions that are surrounded by regions with no insertions at all are de-emphasized. Thus, some insertions that appear in the unaveraged view are not visible in the averaged view. This indicates that both displays are blatantly misleading the user in some way, which is a severe inadequacy of the system. Also, the sparseness of the insertions indicates that they could conceivably be represented with some sort of compact glyph without producing excessive clutter.

The uncertainty that I was confronted with when using the system to perform this task implies that the solution is inadequate for any sort of real-world task. The only component of the display that appears to be useful and reliable is the gradient bar that is used to encode the deletion data.

6.2 Accessibility

A colour-blindness simulator was used to assess the useability of the tool for users who have various forms of colour blindness. Figures 26, 27, and 28 demonstrate what Figure 20 would look like to a deuteranope, a protanope, and a tritanope, respectively. The simulations were generated by the Vischeck online image generator at http://www.vischeck.com.

While the red curve in the substitution plots in Figures 26 and 27 are admittedly difficult to see, the displays are by and large interpretable in all of the simulations. The colour and lightness scheme should therefore be readily accessible to all users.

7 Lessons Learned, and Future Work

The objective of this project was to generate an effective overview to compare two large populations of genome sequences with respect to there differences relative to some canonical genome sequence. The overview was required to be easily interpretable by biologists and to provide sufficient detail to allow the recognition of fine sequence details such as SNPs.

It should be readily apparent from the previous section that this objective was not fulfilled. An attempt was made to use 'traditional' components such as line and area plots in the overview to allow for ease of interpretation; however, in order to accommodate the density of the data being displayed in these plots, some abstraction and averaging of the data had to be done, and these transformations

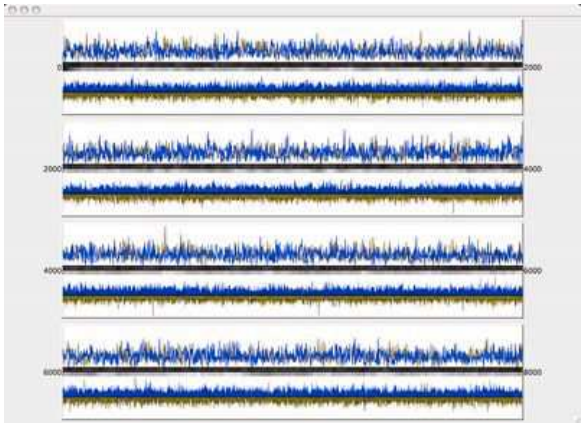

Figure 26: Figure 20, as seen by a deuteranope.

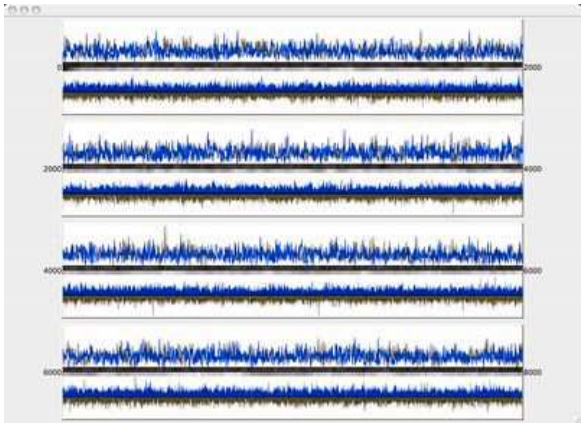

Figure 27: Figure 20, as seen by a protanope.

tended to eliminate detail and to introduce misleading artifacts into the display.

In my opinion, the fundamental challenge that makes this a difficult problem is that the display must on the one hand be easily interpretable, but on the other hand must accomodate the display of very dense data with minimal loss of detail. Techniques for dealing with density often involve some sort of semantic encoding that can abstractly represent a very detailed dimension of the data while occupying minimal display area, at the expense of some extra required effort on the part of the viewer to learn how to interpret these encodings. I believe that in order to realize the goal of creating an effective overview for large-scale sequence difference data, the requirement of 'instant familiarity' will have to be relaxed so that a more abstract representation can be used. Such an approach would be especially effective for displaying insertion data, because insertions are both the most difficult element of the data to display correctly, and because they tend to be very sparsely distributed and thus are good candidates for representation by some sort of glyph or symbol.

The most important thing that I learned throughout the course of this project is that a good design cannot be produced by a person working in isolation. If I could go back and change one of my decisions regarding this project, it would be the decision I made to work alone. The large amount of time I spent evaluating and re-evaluating my design decisions would have been greatly reduced had I been working with someone that I could discuss ideas with.

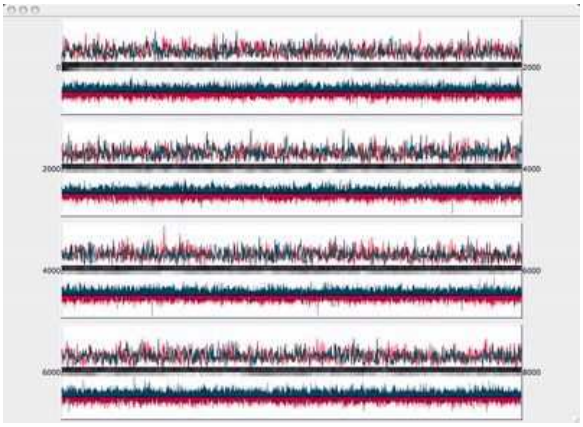

Figure 28: Figure 20, as seen by a tritanope.

REFERENCES

[1] Ware C. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, second edition, 2004.

[2] The International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437:1299–1320, 2005.

[3] Awad IA et al. Caryoscope: an open source java application for viewing microarray data in a genomic context. *BMC Bioinformatics*, 5:151, October 2004.

[4] Clifford RJ et al. Bioinformatics tools for single nucleotide polymorphism discovery and analysis. *Annals of the New York Academy of Science*, 1020:101–109, 2004.

[5] Gottgens B et al. Long-range comparison of human and mouse scl loci: Localized regions of sensitivity to restriction endonucleases correspond precisely with peaks of conserved noncoding sequences. *Genome Research*, 11(1):87–97, 2001.

[6] Hubbard T et al. Ensembl 2005. *Nucleic Acids Research*, 33:D447–D453, 2005.

[7] Karolchik D et al. The ucsc genome browser database. *Nucleic Acids Research*, 31(1):51–54, 2003.

[8] Lewis SE et al. Apollo: a sequence annotation editor. *Genome Biology*, 3(12):RESEARCH0082 (Epub), 2002.

[9] Mayor C et al. Vista: Visualizing global dna sequence alignments of arbitrary length. *Bioinformatics*, 16(11):1046–1047, 2000.

[10] Montgomery SB et al. Sockeye: A 3d environment for comparative genomics. *Genome Research*, 14:956–962, 2004.

[11] Ovcharenko I et al. Ecrbrowser: a tool for visualizing and accessing data from comparisons of multiple vertebrate genomes. *Nucleic Acids Research*, 32:W280–W287, 2004.

[12] Rutherford K et al. Artemis: Sequence visualization and annotation. *Bioinformatics*, 16(10):944–945, 2000.

[13] Stein LD et al. The generic genome browser: a building block for a model organism system database. *Genome Research*, 12(10):1599–1610, 2002.

[14] Tsalenko A et al. Methods for analysis and visualization of snp genotype data for complex diseases. In *Pacific Symposium on Biocomputing*, pages 548–561, 2003.

[15] Wheeler DL et al. Database resources of the national center for biotechnology information: 2002 update. *Nucleic Acids Research*, 30(1):13–16, 2002.

[16] Slack J, Hildebrand, Munzner T, and St. John K. Sequencejuxtaposer: Fluid navigation for large-scale sequence comparison in context. In *Proceedings of the German Conference on Bioinformatics*, pages 37–42, 2004.

[17] Chakrabarti K and Pachter L. Visualization of multiple genome annotations and alignments with the k-browser. *Genome Research*, 14:716–720, 2004.

[18] Baudis M and Cleary ML. Progenetix.net: an online repository for molecular cytogenetic aberration data. *Bioinformatics*, 17(12):1228–

1229, 2001.

[19] Durbin R, Eddy SR, Krogh A, and Mitchison G. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[20] Spell R, Brady R, and Dietrich F. Bard: A visualization tool for biological sequence analysis. In *2003 IEEE Symposium on Information Visualization*, page 28, 2003.

[21] Leitner T, Foley B, Hahn B, Marx P, McCutchan F, Mellors J, Wolinsky S, and Korber B. Hiv sequence compendium 2003, 2003.