Case study: CTG repeat expansion modelling using a Java^{TM} applet and its PJama extension providing persistent storage for genetics data.

Ela Pustułka-Hunt, David A. Jack,
G. Fraser Hogg, Darren G. Monckton

{*ela,david*}*@dcs.gla.ac.uk*, {*ghogg,dmonck*}*@bio.gla.ac.uk*

Department of Computing Science
and Division of Molecular Genetics (IBLS),
University of Glasgow, Glasgow G12 8QQ

TR-1999-31

September 2, 1999

Abstract

We analyse the use of a Java graphical tool to model CTG repeat expansion in the DMPK gene which is the underlying cause of myotonic dystrophy. A modelling applet is described and the underlying mathematical and statistical model is discussed. Possible ways of using this applet to handle the expected large volumes of data are outlined and a PJama implementation of persistent data storage using servlets is described.

Contents

1 Introduction **4**

2 Problem definition **4**
2.1 Genetics 4
2.2 Computing 5

3 Methods **5**

4 Results **6**
4.1 The CTG repeat project description 6
4.2 Data gathering 6
4.3 Data preprocessing 8
4.4 The modelling 8
4.4.1 Human-computer interface 9
4.4.2 Applet structure and logic 12
4.4.3 The model 13
4.5 Statistics 14
4.6 Other data required for the full assessment of patient status . . 15

5 Discussion **16**
5.1 Applet potential use 16
5.2 Functionality extensions 17
5.3 Persistent data storage options 18
5.4 Persistent Java 20
5.5 Using PJama 20

6 Conclusions **21**

7 Acknowledgements **21**

1 Introduction

Myotonic dystrophy (DM) affects 1 in 8,000 people [8], is presently untreatable and poses problems in stating a prognosis for affected patients. DM's cause has been traced back to an expanding DNA triplet repeat (CTG) in the DMPK gene DNA structure, which changes the amount (expression) of the DMPK protein produced [1]. Patients diagnosed with DM are tested for CTG repeat expansions, but the exact relationship between the patient's age, disease onset, observed repeat lengths, and disease prognosis is unknown. Understanding the relationships in the clinical data and developing a mathematical model of the disease progression may help in stating the patients' prognosis. A modelling tool, developed in Java, explores those relationships and investigates how different models of repeat expansion influence the distribution of expanding repeats. This tool has a graphical user interface (a Java applet) incorporated in a web page, and is presently implemented as a stand-alone tool which does not use any data storage technology. As more patient data will be gathered, input data and simulation results will have to be stored in a database, and ways of running chains of simulations and comparing their results will have to be provided.

2 Problem definition

2.1 Genetics

The gene DMPK which has a number of CTG repeats in its DNA structure has been found to be responsible for myotonic dystrophy. The CTG repeat is located in the 3 primed (3') untranslated region of the gene, as shown in Figure 1. DM is a dominant disease. If a child inherits it from one of its parents, it will develop the disease at some point in life, and the probability of inheriting the disease is 50 per cent. The DMPK gene is inherited stably, i.e. in most cases 5 to 37 CTG repeats will be inherited from parent to child. However, very occasionally a normal allele will mutate into the disease range (over 50 CTG repeats), which happens at a very low rate. If the parent has an expanded CTG repeat, and the child inherits it, the number of repeats seen in child DNA is often much larger. This repeat size growth has been thought to happen only in intergenerational mutations during meiosis which produces ova and sperm, but has now been shown to happen throughout life. CTG repeat mutations start in fetal life and are hardly visible at 16 weeks, but can be clearly seen at birth. However, at birth minimal mosaicism (presence of different length alleles) is observed. In a healthy individual the number of repeats stays between 5 and 37. When that number goes over 50, disease

[1] A new form of DM is also being investigated. It involves another gene which has not been sequenced yet.

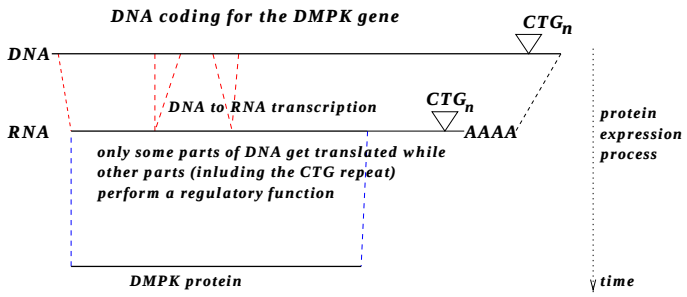

Figure 1: DMPK DNA produces DMPK protein. Untranslated parts of DNA regulate the expression process, in particular long CTG repeat sections present in DM patient DNA possibly affect the amount of the DMPK protein expressed, and have been shown to influence the expression of other genes [6].

symptoms appear. Research done so far suggests that the severity of the disease increases with repeat length expansion. Similarly, longer inherited CTG repeat sequences lead to earlier disease manifestation. To understand the complex relationships between the age of disease onset, patient age, clinical symptoms, and repeat lengths observed in different tissues, more patient data has to be gathered and analysed, and the existing mathematical model has to be improved and extended, to include more variables and their relationships.

2.2 Computing

Data gathered by clinicians and produced in the lab is presently stored in flat files. A Java applet designed by a biologist performs the mathematical modelling. Patient data is pasted in, five simulation parameters are adjusted on the screen, and the results of the simulation are displayed. Additionally, graphs of input and simulation data distributions are shown, and some statistics for the output data are calculated. At the moment there is no provision for automatic data transfer from the file system to the applet. Similarly, there is no provision for printing, other than by getting a screen dump. And, finally, the simulation parameters and results cannot be permanently stored, or compared. There is no facility to perform a sequence of simulations either.

3 Methods

This study started with computer scientists being informed of the need for a database to store patient CTG repeat data. Computer scientists gained an initial understanding of the problem area by attending a seminar where the software was presented. An interview with the biologist who designed the software followed, and the research issues involved in CTG repeats analysis were discussed. After reflection and code analysis, this report was drafted,

and it was discussed with the biologists. Comments and corrections were incorporated, suggestions for further software development were discussed and plans for further cooperation were drawn.

4 Results

4.1 The CTG repeat project description

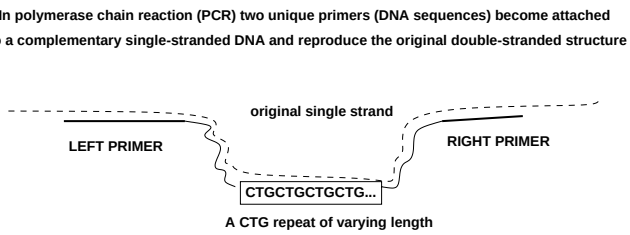

Figure 2: PCR amplification of the CTG repeat.

The severity of DM and the age of disease onset are related to the initial number of CTG repeats inherited from a patient's parents. As the disease progresses, CTG repeat expansions observed in patients increase in size. DM research aims to understand and quantify those relationships so that clinicians working with patients can use this knowledge in stating a patient's prognosis and designing a suitable treatment. In this study some 120 patients' blood samples will be used, of which 70 are available now. DNA from those blood samples is isolated, then the PCR technique is used to amplify the DNA containing the CTG repeat. Using PCR (polymerase chain reaction), shown in Figure 2, it is possible to produce a large volume of DNA sequences with the repeat, which can then be put in an agarose gel in the electric field (electrophoresis). As the speed of molecule movement in the gel depends on size, it is then possible to measure the relative lengths of CTG repeats. An example gel is shown in Figure 3. The sensitivity of length measurement in a gel depends on several factors, one of them being the concentration of DNA in the sample. This concentration can be directly related to the number of cells for which different repeat lengths are being measured. For instance, 6pg DNA is equivalent to a single cell's worth of DNA. To find out the optimum experimental conditions for each sample, a series of gels is made, and once the results are satisfactory, and the readings clear, gel pictures will be processed and the repeat lengths gathered.

4.2 Data gathering

Gels used in this project have 35 lanes each, and for each patient several gels will be made. In each lane several different lengths will be observed. This leads

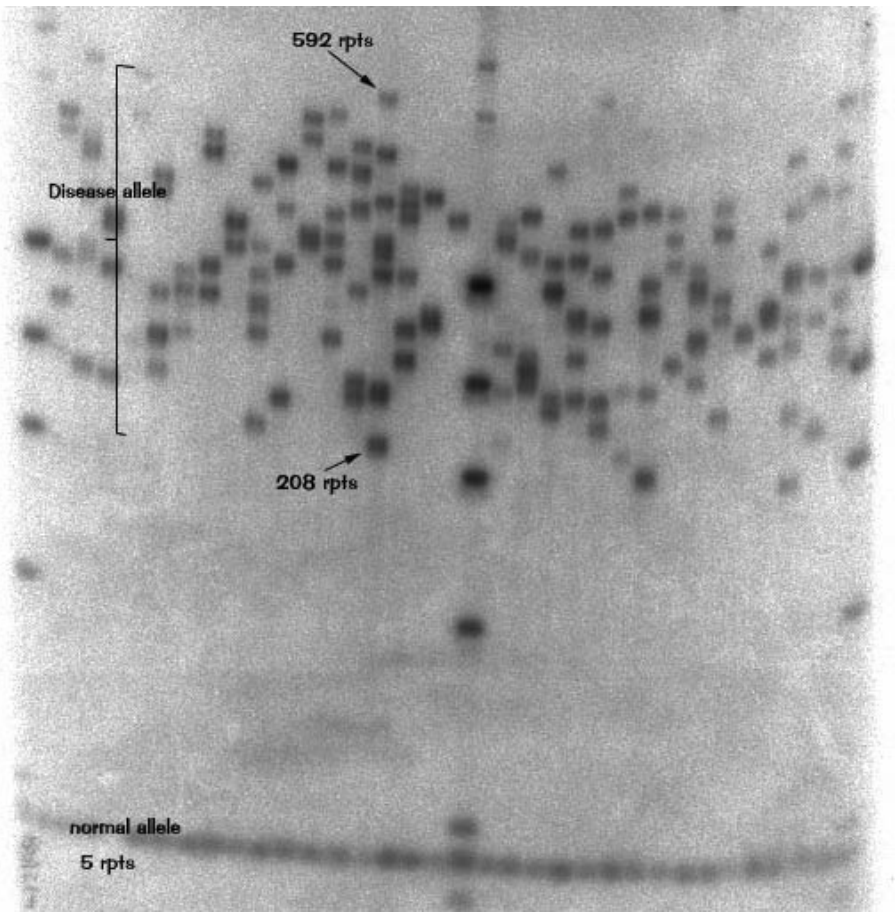

Figure 3: A gel picture showing a set of different CTG repeat lengths observed in a DM patient blood sample. The gel shows a single band at the bottom of the picture which is the normal copy of the gene (containing 5 repeats), and further up many bands (ranging from 200 - 600 repeats) produced by the diseased gene copy. An average repeat number observed is 390 repeats. This individual showed onset of the symptoms as an adult.

to a product space of 120 patients x 35 lanes x 3 gels x 3-8 lengths i.e. some 40,000 to 100,000 data points. DNA in the gel can be made visible using certain chemicals and photographed under ultraviolet light. To exclude the possibility of contaminants creating spurious bands on the gel, gel is hybridised to a radioactive probe containing CTG repeats using southern blot hybridisation technique. Bands containing CTG are then visible in the gel picture. Gel pictures are scanned in and processed using Kodak Digital Science software. Observed DNA fragment lengths are delivered by software, but some manual adjusting may be required. The resulting gel data is a list of numbers, and for each patient several such lists will be merged. Further data processing is performed in Excel, as described in the next section.

4.3 Data preprocessing

Number sequences representing different repeat lengths are then imported into Excel where basic statistical analysis of data is performed. For every individual the mean, minimum, median and standard error are calculated, and a distribution graph is drawn. At this point data is evaluated, taking into account the patient's age, age of disease onset, and symptoms. The minimum expanded repeat size is taken to represent the size of repeat at conception. Patient data can then be fed into the modelling applet where a simulation of the disease development can be attempted.

4.4 The modelling

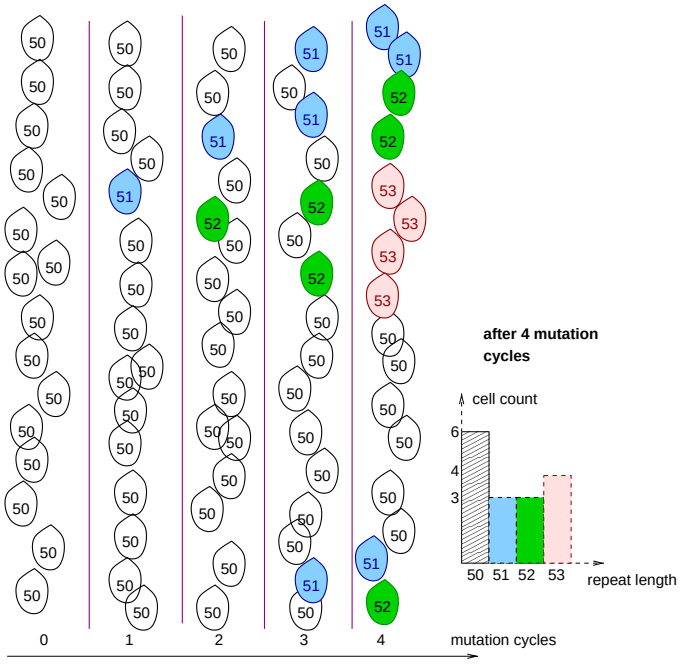

Figure 4: The principle of random mutations. A starting population of 16 randomly mutating cells with initial 50 CTG repeats is shown. After 4 mutation cycles, 10 out of 16 cells have an expanded CTG repeat.

The mathematical model used in this investigation tries to incorporate many variables which may influence the distribution of expanded CTG repeats observed in an individual. It is based on the principle of random mutations illustrated in Figure 4. The simulation starts with some cells mutating and this mutation is then inherited or expanded in the next mutation cycle, based on a supplied mutation frequency. Following parameters can be varied:

- mutation size (how many CTG repeats are added in one mutation)

- mutation rate (probability of a single cell mutation)
- starting number of repeats (taken as the minimum expanded length found in patient CTG repeat profile)
- number of mutation cycles for simulation
- no of cells used in the simulation.

Present model is a preliminary version and the simplest proposal for repeat expansion. It does not accurately describe the actual biological process yet. A better model would take into account the following factors:

- possible influence of the repeat length on the mutation rate (the longer the repeat, the higher the mutation rate)
- a spectrum of mutation sizes might be more appropriate than one fixed mutation size
- possibility of contraction mutations
- other mechanisms of repeat expansion
- different rules for mutations depending on which DNA strand they happen on.

4.4.1 Human-computer interface

The modelling tool is currently implemented as a Java applet embedded in an HTML page and shown in Figure 5. The applet is divided into three parts: left, right and bottom panel. Left panel accommodates two graphs: the lower one will show proportional frequencies of the input data points (count of alleles for each recorded repeat length, ordered by size and shown as a percentage of the input sample), the upper one will display the proportional frequencies of simulated repeat lengths. Right panel is subdivided into two parts which are separated by a **Continue** button. Its upper part provides input fields to set the input parameters for simulation, as described in the previous section. The lower part shows three statistical measures calculated during the simulation. These can be seen as simulation outputs and complement the graph shown in the upper part of the left panel. Following statistics are displayed, and they are fully discussed in a further section:

- best D value
- no of cycles
- significance threshold.

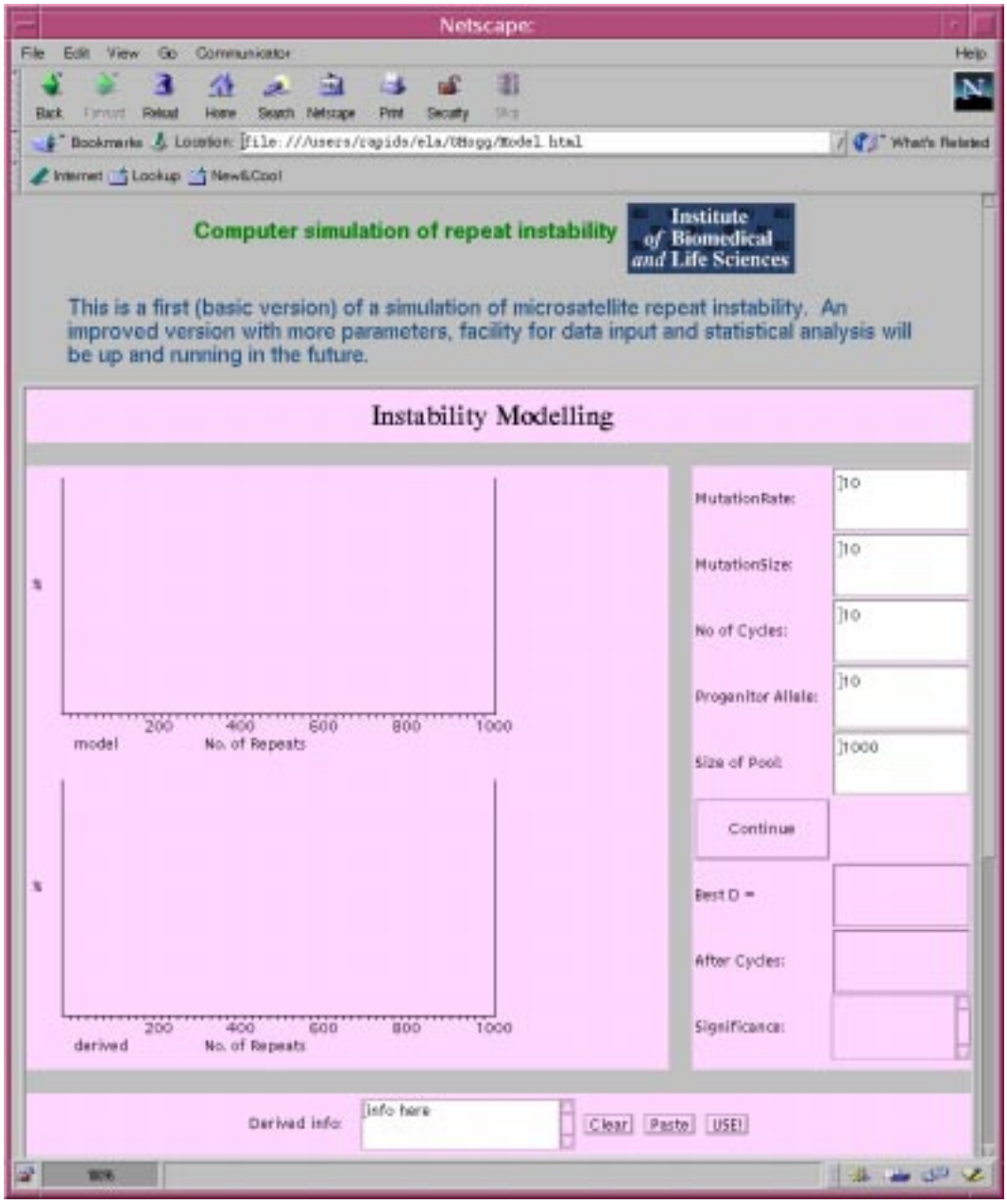

Figure 5: The instability modelling applet designed and implemented by DM researchers.

The bottom applet panel is used for data entry. A file containing all the measured repeat lengths for a given patient is pasted in here, and three options are provided. One clears the data, another pastes data in, and the third called **USE!** draws a graph of the proportional distribution of repeat lengths in the lower part of the left panel. Figure 6 demonstrates the applet after some data has been pasted in and the **USE!** button was pressed. The input parameters

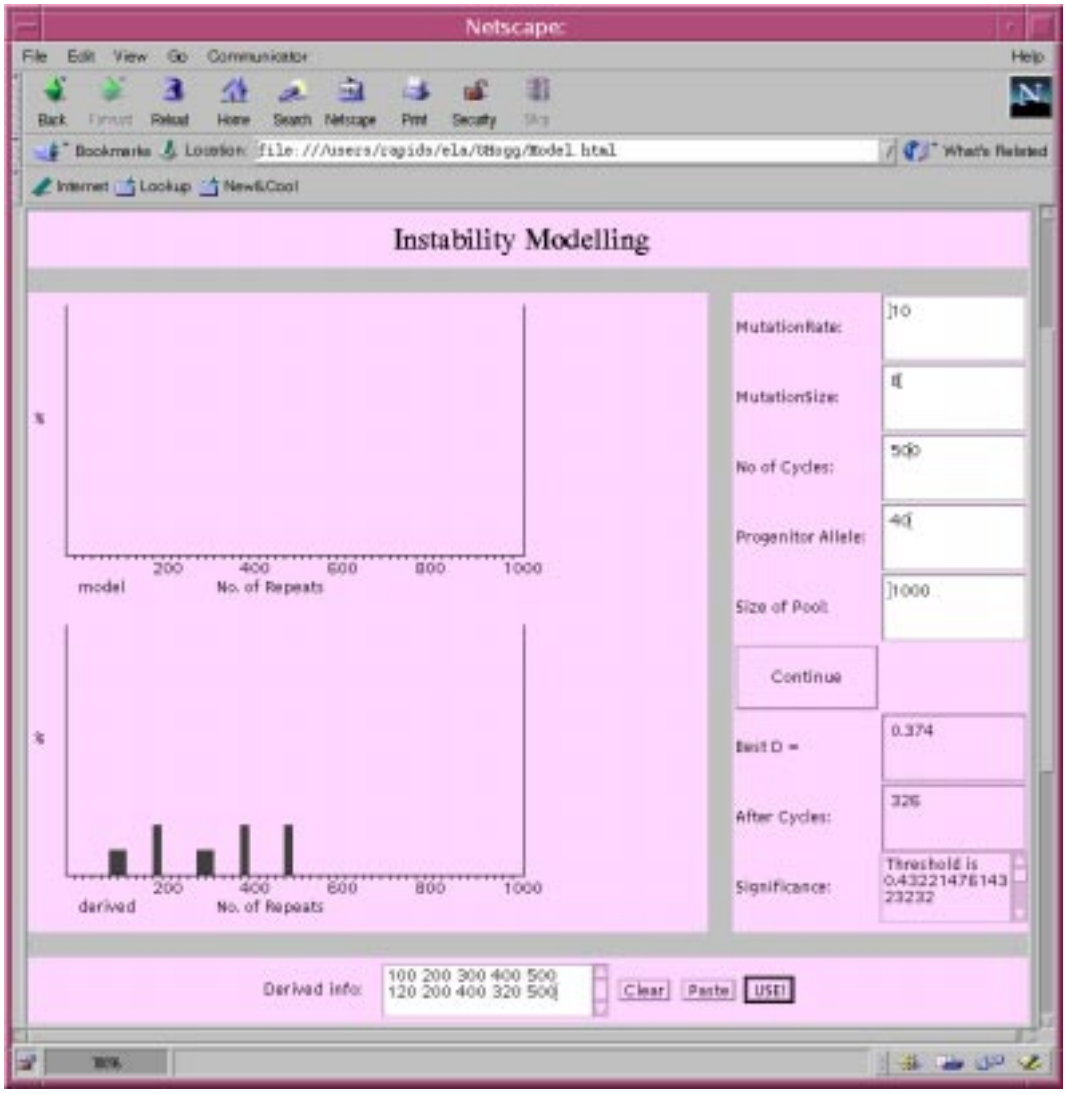

Figure 6: Applet displaying input data distribution expressed as a percentage of the input sample size.

top right are used in the simulation itself and are discussed further on.

A biologist can interact with the applet via the three buttons in the lower panel, as described, and via the parameter options and the **Continue** button in the right panel. Five simulation parameters can be edited, and the **Continue** button runs the simulation, calculates the statistics and draws a graph of simulated repeat lengths percentage distribution in the upper part of the left panel. The results of a simulation run are shown in Figure 7, and the parameters are discussed below.

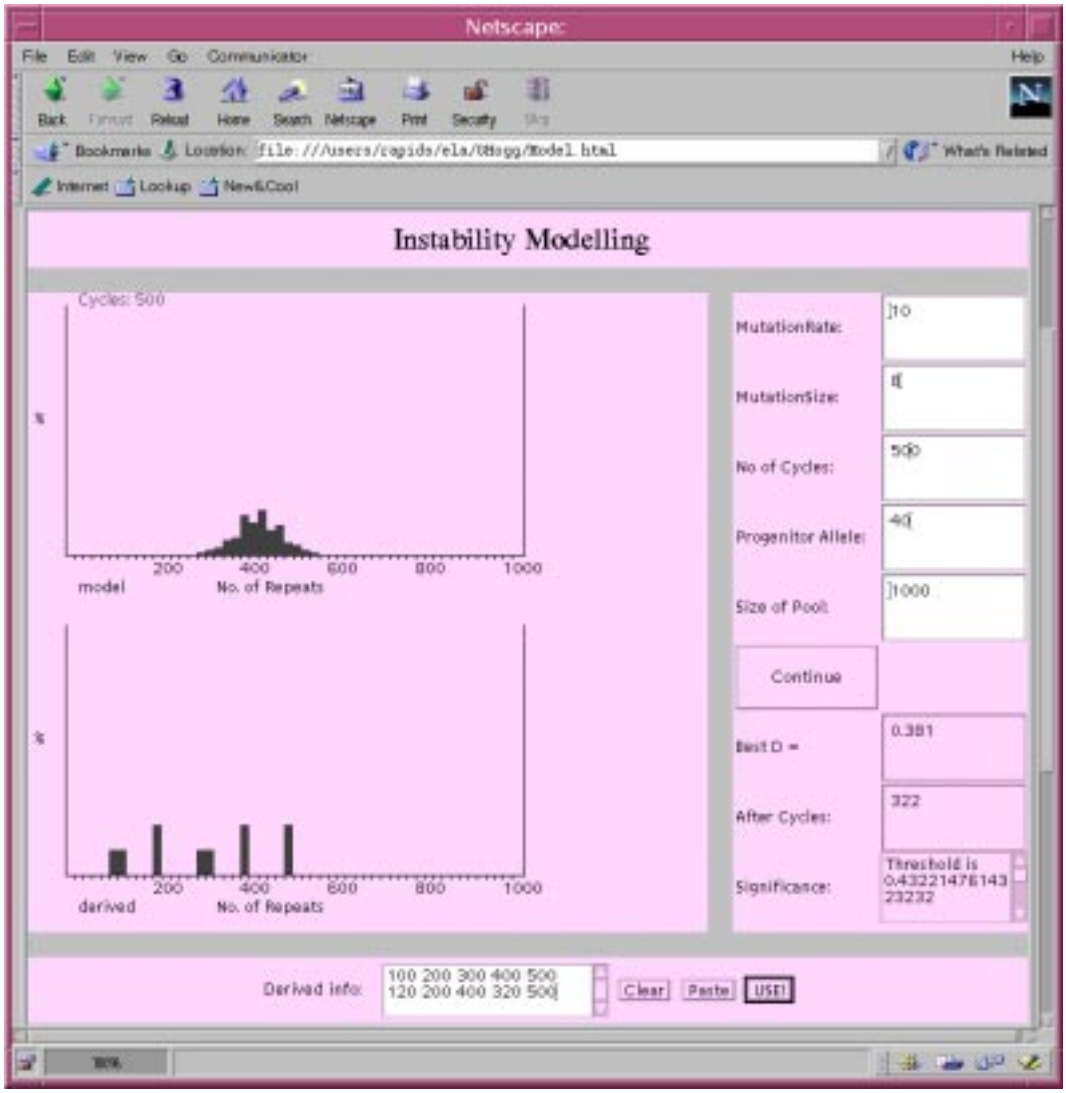

Figure 7: Applet running a simulation on a set of 10 values. Top left panel shows the simulated repeat distribution, bottom left panel shows the input data. The right panel displays the parameters and statistical measures.

4.4.2 Applet structure and logic

The structure and interactions within the applet software are summarised in Figure 8. User actions and applet calculations are controlled from the Model.actionPerformed method. Other methods are used to lay out the page and perform various services. Statistics are calculated using Model.frequency and Repeat.convert, while graph drawing is performed by Der.draw1 and Mod.-draw1 which are practically identical.

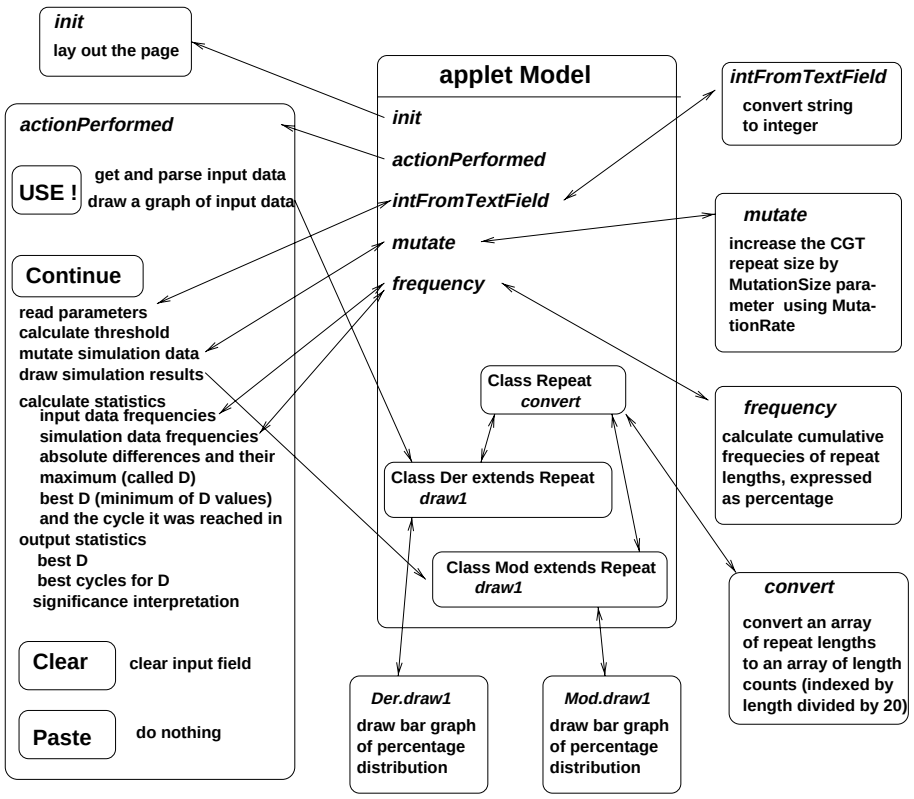

Figure 8: Applet software structure. The applet Model has five methods and three inner classes. The methods handle page layout, user interaction, graph drawing and statistics. Inner classes are responsible for graph generation. Communication between methods and classes is represented by lines, where arrows indicate the direction of information flow. The method Model.actionPerformed handles user interactions related to the four buttons: **USE!**, **Continue**, **Clear** and **Paste**

4.4.3 The model

The simulation itself is based on a model of random mutation in the CTG repeat. We refer here to simulation parameters as shown in Figure 7. A cell pool of a given size (*Size of Pool*) is created where all cells contain the same allele (*Progenitor Allele*, usually taken to be the shortest expanded allele present in patient sample). To give an example: if the diseased progenitor allele has 200 CTG repeats, all cells in the pool are initialised to carry that allele. Then this pool of cells is randomly mutated for a number of cycles (*No of Cycles*). A random number generator is used, and for each cell this number is converted to a percentage scale (division modulo 100). If the random percentage is smaller than the mutation rate *MutationRate*, the allele size is increased by *MutationSize*. Otherwise, it is left unchanged. So, if the mutation size is 10, and the cell is chosen for mutation, the allele length will go from

200 to 210. An example will illustrate this method:

Initial length	200	200	200	200	200	200	200	200	200	200
Random number	22	39	88	7	5	33	92	37	11	48
After mutation	200	200	200	210	210	200	200	200	200	200

Because mutations are random, there is no guarantee of a mutation happening or not. If for each cell the probability of mutation is p (here 10% - corresponding to p=0.1), and we consider those individual mutations to be independent, then in this mutation of n cells the expected number of mutations is p $\times$ n. Taking k cycles, we can expect k $\times$ p $\times$ n mutated cells. For a population of 1,000 cells, p=0.1 and 10 cycles, we expect 1,000 (10 $\times$ 0.1 $\times$ 1,000) mutations in the simulation. This simple model will have to be refined, and it is expected that mutation probability is influenced by the repeat size, and some other factors.

4.5 Statistics

The point of the simulation is to find out the relationship between the inherited allele and the measured allele distribution and to understand the laws governing repeat expansions. At present, patient data taken at a particular point in the disease development is compared to simulated expansion data. If the distribution of the simulated alleles is significantly similar to the patient's alleles' distribution, the cycle number when this happened is displayed. Possibly a conclusion may be then drawn regarding the patient's prognosis, based on the trend in repeat expansion. Statistics are used to compare the repeat distribution data with the simulation data, and to find the point in the simulation where the input data approximated simulation data closely. A measure called significance [4] is used, based on the Kolmogorov-Smirnov two-sample test and it is calculated as follows, based on the sample sizes. First, the *significance threshold* is calculated using the formula:

$$1.36 * \sqrt{\frac{n_1 + n_2}{n_1 n_2}}$$

where n_1 is the size of the input data vector (containing observed repeat lengths) and n_2 is the number of cells used in the simulation (size of the output data vector). The factor 1.36 corresponds to the largest difference between the two data sets being 5 per cent significant (1.63 would correspond to 1 per cent significance, and 1.95 to 0.1 per cent).

Next, a D value (difference) is calculated for each simulation cycle, as the maximum of the absolute differences of cumulative frequencies for the two compared data sets. For instance, if in patient data only 3 repeat lengths were present (100, 120, 150) and the input data contained the values: 100, 150, 120, 100, 100, 150, 150, 150, 150, 150, the frequencies of those values would be:

Allele length	100	120	150
Allele frequency	0.30	0.10	0.60
Cumulative frequency	0.30	0.40	1.00

If a simulation cycle on 10 cells resulted in the following allele length data: 100, 110, 130, 160, 100, 120, 140, 100, 100, 130, the frequency table would show:

Allele length	100	110	120	130	140	160
Allele frequency	0.40	0.10	0.10	0.20	0.10	0.10
Cumulative frequency	0.40	0.50	0.60	0.80	0.90	1.00

D will then be calculated as the maximum of the absolute values of differences between the cumulative frequences of both samples. These differences are then:

Allele length	100	110	120	130	140	150	160
Absolute difference	0.10	0.50	0.20	0.80	0.90	0.10	1.00

In this cycle the maximum difference D is 1.00. The *best D* value is calculated as the minimum of D's over all cycles. The cycle number, at which *best D* is reached, and the *best D* are then output. If the *best D* exceeds the previously calculated threshold value, the two data sets are regarded to be significantly different, and the simulation does not reflect a biological process. However, if *best D* is below the threshold, the input and simulation data at *best D* cycle are similar, and the simulation might reflect random repeat mutations.

4.6 Other data required for the full assessment of patient status

The modelling applet delivers the mathematical and statistical analysis required for patient assessment. It can only process one data set at a time and needs to be used in conjunction with spreadsheets holding other patient information and summarising the simulation runs and graphs. The following additional input data are used in the analysis [2]:

[2]It is worth observing that some data interpretation takes place at this point, as the estimated base length uses human judgement to select the value that is considered to be the inherited allele length and bears an influence on the results of the simulation.

Data	Comment
blood sample identifier	unique
sample name	patient identifier
patient sex	F or M
age of disease onset	-
patient age when sampled	-
base repeat number by Southern blot	provided by the clinic
base repeat number by PCR	produced for this study
PCR gel pictures	TIFF, 1.3MB, several
repeat data array	observed lengths
estimated base length	adjusted value
other clinical data	-

During the initial processing of PCR results data for each patient, following values will be derived in Excel [3] and need to be kept:

Data
count of different lengths observed
mean
standard deviation
standard error
minimum
maximum
mode (peak in histogram)
different percentiles for minimum length
a graph of data distribution

After the applet has been used to simulate the mutation process, following results will be filed:

Data	Comment
input data array and sample ID	-
best D	-
threshold	-
cycle number	at *best D*
input/simulation data distribution	screendump (image)
simulation parameters	as described before
comments	-

5 Discussion

5.1 Applet potential use

The CTG repeat modelling applet is part of an ongoing research project which provides a very elegant solution to the mutation simulation problem. If the

[3] This calculation performed in Excel is already automated as a macro. It is envisaged that that calculation and graph production could be trivially performed entirely in Java at the applet or server level

simulation algorithm provides consistent results for a large enough sample of patients, it could potentially be used by geneticists investigating a number of diseases which are due to expanding DNA repeat sequences including Friedriech's ataxia, Huntington's disease, Fragile-X syndrome and spinobulbar muscular atrophy. It could also be useful to clinicians treating patients suffering from those diseases, by providing a patient's prognosis. The applet can be seen as a foundation for a more complex tool which could let cooperating clinicians and genetics researchers share their patient data, clinical observations, simulation results and the prognosis, and help with the disease development tracking. At the moment it implements a small part of the clinical data assessment process, and the way it is used is illustrated in Figure 9.

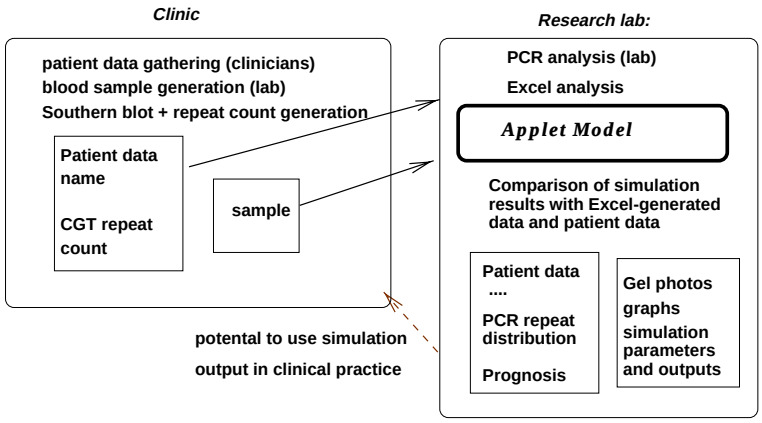

Figure 9: The role of the modelling applet in the context of clinical practice and DM research. Data flows are triggered by the cooperating researchers, and include the exchange of paper reports, and electronic files.

5.2 Functionality extensions

We consider two types of functionality extensions. First of all, extensions to the mutation model and parameters used in the simulation, secondly to the environment the applet is run in. Any extensions to the mutation model may require changes to the applet interface. New input and output fields may need to be added, and the display may require a different layout. On the other hand, extensions to the way data is stored, accessed and used may require changes to the interface, to provide buttons and menus for using added database functions.

To realise its full potential, this applet needs to overcome the limitations inherent, amongst others, in applet security restrictions. It would also have to provide access to a data store and suitable data import procedures, data export routines, data query facilities, user authentication and user profiles. All possible extensions to this applet require a carefully thought-out system architecture which will be able to accept the addition of new facilities, and

evolution of those functions. Figure 10 illustrates the possible components which could turn this applet into a versatile and truly useful application. We

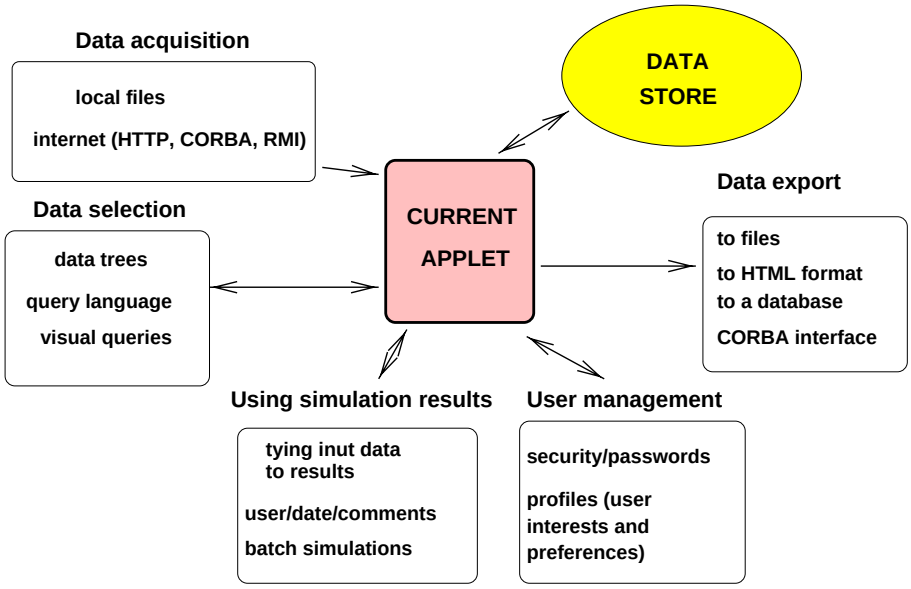

Figure 10: Possible extensions to the CTG modelling applet. Possible communication channels between system parts are shown as thick lines.

think that the future functionality of this applet could subsume all of the data import, storage, analysis and output functions. A possible extended applet could be used as shown in Figure 11.

5.3 Persistent data storage options

One of the prerequisites for an application which will interface to considerable amounts of data is to provide an automatic connection between the application and persistent data storage. For a Java applet this can be done in several ways. These possibilities include:

- reading files from the file system. This can provide read-only access to data, as it is inadvisable to allow unidentified web users to write to the file system directly.

- reading and writing data from and to a database (relational or object-oriented) via CGI scripts or servlets. User authentication can then be performed in the database or by the web server. CGI scripts or server scripts and servlets communicating with a database are an expensive way of providing secure data storage because they involve high initial (programming) and ongoing costs (database administration) as well as database license costs.

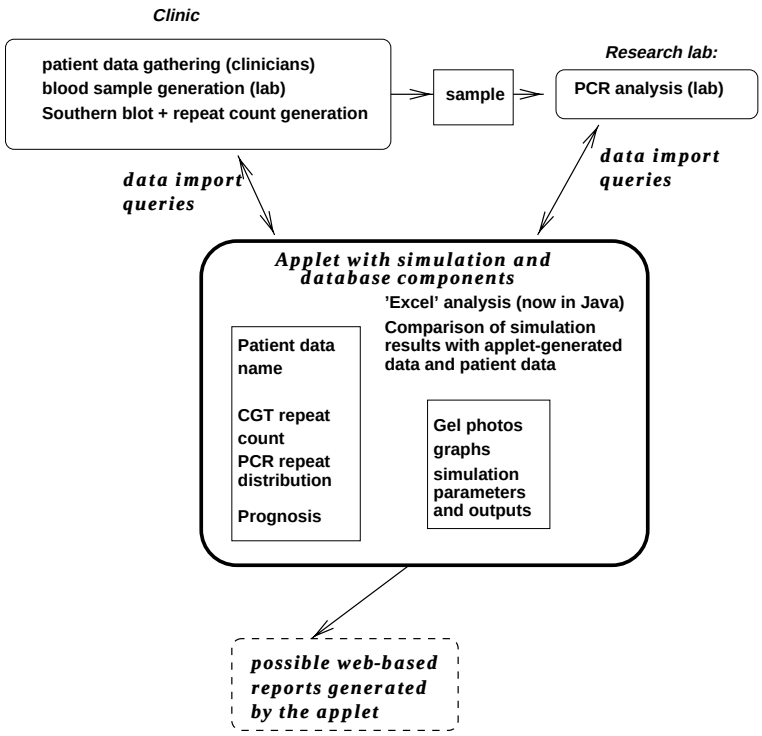

Figure 11: The proposed applet would provide a complete extensible solution to data storage and analysis. It would be used by both the clinicians and other CTG researchers, and provide data access functions to all authorised project members, and selected reporting for public use, via a web interface.

- use novel PJama technology and HTTP protocol to send binary data between the applet and the server. A Java web server and servlets will be run as part of PJama to provide transparent data storage for existing Java programs.
- run any servlet-capable web server independently of PJama and use Remote Method Invocation (RMI) to transfer data between the applet and the servlet embedded in a PJama store.

Since the combination of a persistent web server and an applet communicating over HTTP protocol has been tested in several applications [3], and entails low programming and support costs, we have decided to test it in the context of this applet. The steps undertaken to provide PJama storage are described in next section. We envisage exploring the combination of PJama and RMI in the near future as well and are planning to compare both solutions.

5.4 Persistent Java

PJama is an on-going research project working on support for providing data persistence in Java [7, 2, 1]. Persistent data stores built using PJama maintain data according to the same principles as traditional databases. That is, they allow for application data structures to be created and maintained independently from the application's life span. In case of Java applications and applets minimum programming is required to make their data persist, and a programmer can add persistent storage to an application with just a few lines of code. In contrast, using a database like ORACLE^{TM} or MSQL would require considerable programming effort, database installation skills and SQL programming experience. Applications using PJama do not explicitly require to handle connections to persistent data store. Instead, the PJama interpreter software, a modified Java interpreter program, launches the application with the persistent store, allowing for transparent data storage. Data objects are made persistent by registering them with a persistent root. This can be done directly by declaring the object as a new persistent root, or indirectly by referencing the object from an existing root. Persistent roots are the 'handles' by which data can be accessed.

5.5 Using PJama

The first step taken towards providing persistent data storage for the CTG modelling applet was to create the persistent store itself. This was achieved by creating an empty store using the PJama utility, opjcs, and populating it with data using a Java class, PopulateStore, which reads in the application data from a flat file, stores it as a Java Vector, and registers it as a new persistent root.

```
// Using a PJama store

// get data
Vector data = readFile(dataFile);

// open store
String root = "Genetic Objects";
PJStore pjs = PJStoreImpl.getStore();

// associate data with the store
pjs.newPRoot(root, data);
```

After adding a persistent data store, the next step involves extending the applet (in a predictable way which is amenable to automation) so that it can access the data which is now kept in the Vector structure accessible from the root. Due to Java's applet security restrictions which protect the integrity of the data server, the applet can not access the store directly. This could allow unidentified users to damage the store and data. The standard solution to this

problem was to develop a Java servlet as the middle-tier of the application. The applet makes a request to the servlet for data, and the servlet makes the connection with the data store and returns the appropriate data to the applet.

The Java servlet is embedded within the Jigsaw web server [5], which has in-built user authentication filters in order to provide application security. The resulting three-tier application structure is presented in Figure 12.

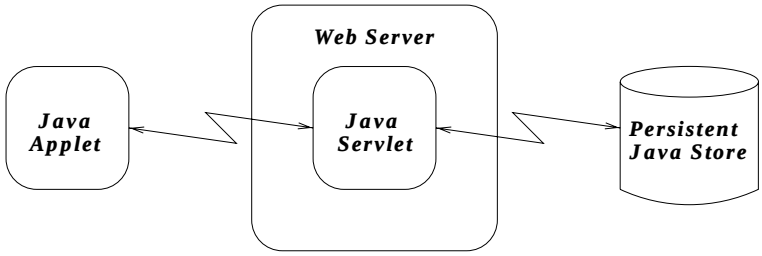

Figure 12: A three-tier application structure consisting of a web server with an integrated servlet which provides a secure connection between a PJama data store and an internet user using the applet.

6 Conclusions

We have analysed a novel solution to the CTG repeat modelling problem, and described its Java interface and the mathematical and statistical elements of the simulation. We found the solution to be elegant, robust and pleasing to use. We have found that it is possible to add database storage to the applet using PJama orthogonal persistence, and that the proposed solution using a servlet embedded in a Java web server is viable. However, we discovered that managing the Jigsaw web server running under PJama causes problems at the time of system evolution, as the persistence models of Jigsaw and PJama clash. As the requirement to store and access more data, and change the applet itself is becoming more urgent now, we will explore the ways of enriching the applet with additional capabilities as outlined above, and we will pursue the implementation option based on PJama and RMI communication, as outlined in Section 5.3.

7 Acknowledgements

We would like to thank Sir Graeme Davies, the Principal of the University of Glasgow for funding this research and the Department of Computing Science, in particular Professor Malcolm Atkinson and Professor John Coggins of the Institute of Biomedical and Life Sciences for providing intellectual, financial

and institutional framework which enabled this co-operation. We thank Doctor Richard Wilson for reviewing this manuscript.

References

[1] M. Atkinson and M. Jordan. Providing Orthogonal Persistence for Java. *Lecture Notes in Computer Science*, 1445, 1998.

[2] M.P. Atkinson, L. Daynes, M.J. Jordan, T. Printezis, and S. Spence. An Orthogonally Persistent Java. *ACM Sigmod Record*, 25(4), 1996.

[3] M.P. Atkinson and M.J. Jordan. Issues Raised by Three Years of Developing PJama. In *Database Theory - ICDT'99*, volume 1540, 1999.

[4] R. C. Campbell. *Statistics for Biologists.* Cambridge University Press, 1989.

[5] World Wide Web Consortium. http://www.w3c.org/Jigsaw.

[6] Patricia Groenen and B*é* Wieringa. Expanding complexity in myotonic dystrophy. *BioEssays*, 20:901–912, 1998.

[7] Department of Computing Science PJama Project, Glasgow University and Sun Microsystems. http://www.dcs.gla.ac.uk/pjama/.

[8] R. D. Wells, S. T. Warren, and Marion Sarmiento (eds). *Genetic Instabilities and Hereditary Neurological Diseases.* Academic Press London, 1998.