Understanding somatic mosaicism in myotonic dystrophy type 1

A thesis submitted for the degree of Master of Science at the University of Glasgow

by

Grant Fraser Hogg B.Sc.

Division of Molecular Genetics
Institute of Biomedical and Life Sciences

December 2000

Abstract

Myotonic dystrophy type 1 is caused by an unstable CTG repeat expansion in the 3' UTR of the *DM1 protein kinase* gene on chromosome 19. A neuromuscular disease with a broad spectrum of symptoms, DM1 also exhibits anticipation whereby disease severity increases through successive generations. Increasing measured allele size between patients correlates with an increased severity of symptoms and an earlier age of onset. However, this correlation is not precise and therefore measured allele length cannot be used as an accurate indicator of age of onset. This suggests that repeat length may not be the major determinant of disease severity. There is a high level of somatic mosaicism shown by the mutation and failure to take into account age-dependent somatic mosaicism in patients may have compromised the accuracy of clinical correlations. The aim of this project was to investigate simple approaches for correcting for age-dependent somatic mosaicism and also to develop computer software to allow us to simulate the progression of age-dependent somatic mosaicism. We have demonstrated that employing alternative approaches in both molecular diagnoses and statistical comparison can yield significantly improved repeat length / age of onset correlations. This conclusively shows that repeat length is by far the major determinant in DM1 disease onset. Simulation software was also successfully developed and preliminary results suggest that DM1 repeat instability is amenable to mathematical modelling in the future.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Dr Darren G. Monckton for offering me the chance to attain an M.Sc. and for excellent and understanding supervision over the course of the past two years. I would also like to thank my assessors, Dr Richard Wilson and Dr Douglas Armstrong, for the constructive advice and the interesting and amusing anecdotes.

To everyone on level 5. I doubt there is not a single individual in the dynamic mutation group who did not contribute in some way to my work.

To Professor Tetsuo Ashizawa at the Baylor College of Medicine and Dr Montserrat Baiget at the Hospital du Sant Paul, Barcelona for the DM1 samples which made this study possible and the patients who provided them.

To Ela Hunt, David Jack and Iain Darroch for the advice and assistance developing the Java software.

Finally, to the Muscular Dystrophy Campaign for funding this research. A more emotional acknowledgements page will be inserted on acceptance of this thesis.

The research reported in this thesis is my own original work, except where otherwise stated, and has not been submitted for any other degree.

Grant Fraser Hogg

December 2000

Chapter 1

Introduction

1.1 Myotonic dystrophy type 1

Myotonic dystrophy type 1 (DM1) is a common hereditary disorder that was first recognised in 1909. It is an autosomal dominant disorder but difficult to diagnose as it shows remarkable phenotypic variability and progression of symptoms (Harper, 1989). DM1 shows a range of clinical symptoms, but is named after and most often identified by its effects on muscle. The disease causes weakening and wasting of a characteristic pattern of muscles. The most prominently affected muscles include facial and jaw muscles, sternomastoids and distal limb muscles. Other muscles are affected depending on the severity of the disease, while some muscles are almost always unaffected e.g. weight bearing muscles (Harper, 1989). The symptom of myotonia can be used to diagnose DM1 and is defined as the inability to relax a muscle after voluntary contraction. Myotonia has been studied at the molecular level in Thompson's disease and has been shown to be due to defects in chloride ion conductance across the muscle membrane (George *et al.*, 1992) although a different mechanism may be responsible in DM1. Myotonia tends to be more prominent in patients with less pronounced muscle weakening than those with severe muscular damage (Sarnat and Silbert, 1975). Other non-muscular symptoms include cataracts, which show progressive opacity characteristic of DM1, excessive daytime sleepiness, testicular atrophy and tumours in the gastro-intestinal tract (Brewster *et al.*, 1998) (Harper, 1989). Early deaths of DM1 patients are often due to cardiac conduction defects (Harper, 1989). The variability in the progression and age of presentation of these symptoms allow DM1 to be classified into 4 broad clinical forms: mild (late onset), classic adult onset, juvenile onset and finally congenital DM1 (CDM1), which is the severest form (IMDC, 2000). CDM1 shows many different symptoms e.g. facial dysmorphia, but these features can be explained by effects on foetal muscle – in other forms of the disease foetal muscles are normal and the adult muscles waste progressively in later life (Sarnat and Silbert, 1975).

1.2 DM1 Genetics

When DM1 presents itself, it can have a devastating effect on a family as frequently all clinical forms can be seen. This occurs in a pattern of increasing severity of symptoms and an earlier age of onset between generations- a phenomenon known as anticipation which has been associated with the disease since it was first described. A typical DM1 family will show a child with CDM1 whose mother exhibits the classic adult onset phenotype and whose grandparent has the mild form typified by cataracts. Clinicians had described anticipation in DM1 and other hereditary diseases such as Huntington disease but, at the time, the genetic mutation was thought to be stably inherited and could not explain this observation. A human geneticist named Lionel Penrose was able to dismiss anticipation as a series of errors in ascertainment for most diseases it had been observed in (Penrose, 1948). He had to concede that the pronounced anticipation observed in DM1 was a special case, suggesting that other genetic factors might have been involved. Ascertainment bias was taken to be the explanation for anticipation almost up to the date in which the mutation that caused DM1 was identified. In 1992, just as people were beginning to question Penrose's findings after some 40 years (Howeler *et al.*, 1989), three groups of scientists independently identified the mutation responsible for DM1 (Fu *et al.*, 1992) (Brook *et al.*, 1992) (Mahadevan *et al.*, 1992). It was found to be caused by an unstable tandem triplet repeat mutation, in this case a CTG repeat, in the 3' untranslated region of a gene called *DMPK.* Previously an unstable CGG repeat had been found to be responsible for FRAXA (Fu *et al.*, 1991) (Verkerk *et al.*, 1991) and an unstable CAG repeat was responsible for SBMA (La Spada *et al.*, 1991). With this a new area of research had opened up- the study of dynamic mutations. DM1 became one of an ever-growing list of genetic diseases caused by a triplet repeat mutation which is now numbered at 14 (Cummings and Zoghbi, 2000). The discovery of an unstable mutation being responsible for DM1 reconciled the clinical observations of anticipation by providing a plausible molecular explanation. Normal individuals have between 5 and 37 repeats at this locus whilst Southern blot analysis of those with the disease show repeat lengths from 50+ to several thousand (Harley *et al.*, 1993). There is a statistical correlation between the number of repeats an individual has and the clinical form of the disease. Those who show mild symptoms typically have approximately 100 repeats; those with classic adult onset have between 100 and 1000 repeats whilst those with CDM have 1000+ (IMDC, 2000). Although there is something of an overlap, a statistical technique known as Baye's theorem can be used to estimate the probability of an individual having any of the clinical forms. Only individuals with 200 or 800 repeats have equal probabilities of developing 2 different clinical forms (for example, an individual with 200 repeats has a 50:50 chance of developing mild or classic adult onset DM) (Gennarelli *et al.*, 1996).

The explanation of anticipation is that the offspring of affected individuals inherit a larger mutation than their parents inherited giving a more severe form of the disease, hence the term dynamic mutation. This intergenerational instability of the repeat poses an important question. DM1 has a prevalence of 1 in 8000 yet alleles are continually lost owing to the low reproductive fitness of those affected. Therefore, how is the mutation maintained in the population? Analysis of normal alleles world-wide reveals a tri-modal distribution with allele sizes of (CTG)$_5$, (CTG)$_{11-17}$, and (CTG)$_{19>}$, representing the peaks of the distribution (Zerylnick *et al.*, 1995). There is evidence that the mutation rate increases with the size of the repeat and it is thought the pool of normal alleles consisting of (CTG)$_{19+}$ mutate into disease ranges (Imbert *et al.*, 1993). This occurs at a rate between 10^{-3} and 10^{-4} maintaining the disease in the population. Allele repeat lengths between 50 and 80 CTG's only show mild symptoms at a later age and are termed proto-mutations. This proto-mutation range frequently shows intergenerational expansions to larger repeat sizes and severer disease (Barcelo *et al.*, 1993). The progression of the disease from mild to congenital through different generations is also influenced by the sex of the transmitting parent. There is an approximately equal sex ratio among those affected, but it was noted that the mutation was more likely to be inherited from an affected male than female (Bell, 1947). It was also noted that those individuals with CDM1 had almost always inherited the mutation from their mother. Repeats lengths <100 are more unstable in males than females therefore there is an excess of males transmitting the first clinically recognisable form of the disease (Barcelo *et al.*, 1993). It could be postulated that this is one of the many side effects of male fertility. Males might be mostly responsible for inherited mutations as the need for continual production of sperm and the extra cell divisions required to produce spermatoa (50 – 100 compared to 30 during oogenesis) increase the chances of mistakes during replication. The maternal bias towards CDM1 was suggested to be due to an intra-uterine maternal factor. Such a factor has never been identified, although it would seem plausible that CDM1 is a direct consequence of the effects on development of a DM1 foetus in a mother who also has DM1. The size of the repeat must be the most important factor as a handful of paternally inherited CDM cases have been identified (Bergoffen *et al.*, 1994) (Die Smulders *et al.*, 1997).

1.3 Somatic Mosaicism

Intergenerational instability of the CTG repeat could be postulated to be due to some mutational mechanism occurring in the germ cells. However, repeat instability is not solely restricted to sperm and ova. Southern blot analysis of genomic DNA containing an expanded CTG repeat allele often does not produce a single band, rather a diffuse hybridisation signal or 'smear'. This is due to a phenomenon called somatic mosaicism whereby individual cells have different repeat lengths (Ashizawa *et al.*, 1993). The mid-point of the smear in such cases is taken to be the average repeat length for that individual and is the value used in clinical studies. To better characterise somatic mosaicism a technique known as small-pool PCR can be utilised (Monckton *et al.*, 1995). This involves using template DNA at amounts equivalent to only a few cells worth of DNA (6pg is equivalent to one diploid cell). Analysis of patients of different ages (6 months to 62 years) but with the same average repeat length clearly showed that somatic mosaicism increased with the age of an individual (Martorell *et al.*, 1998). The explanation for age-dependent somatic mosaicism was provided by the fact that the repeat is unstable with time in the same individual. A study by (Martorell *et al.*, 1998) analysed the repeat length of 111 DM1 patients at two different time points separated by at least 2 years. A detectable increase in repeat length could be found in all patients with average repeat length >200 repeats. Hence, the DM1 mutation is a dynamic mutation in every sense- it is unstable through the generations and is also continuously unstable throughout the lifetime of an individual. It has also been shown that the longer the repeat tract the more unstable it is (Wong *et al.*, 1995), although this is not a straightforward correlation. In patients with CDM1 there is no correlation between repeat size and repeat heterogeneity as the repeat has been shown to be relatively stable in early development (instability has only been observed around 16 weeks and is still minimal around birth). However, when patients are separated into age groups it was found there was a significant positive correlation between size and heterogeneity in patients aged 21 years and over (Wong *et al.*, 1995). These are important observations as they provide an explanation for the progressive nature of the disease. That is, an individual who inherits a repeat length of 100 repeats will show a milder form of the disease with a slower progression than the severe disease and rapid progression of an individual with a 1000 repeats. While providing an explanation for the progression of the disease, failure to take into account age-dependent somatic mosaicism has probably hampered inter-generational transmission studies and clinical correlations (Monckton *et al.*, 1995). One study looking at repeat length transmissions found that 6.4% of all transmissions produced repeat contractions as opposed to expansions (Ashizawa *et al.*, 1994). Curiously, half these cases still showed clinical anticipation, which seemed to contradict the correlation between repeat length and severity of disease. This study involved different generations of the same family being sampled on the same day without any provision for the repeat instability that has occurred in the older patients. This will have affected the accuracy of their findings. It would be useful to eliminate these age dependent differences by estimating the progenitor allele received at birth by an individual. This could be used for more accurate pedigree studies and would allow a clearer index between repeat length and disease severity to be produced.

The size of the repeat tract and the age of the patient are not the only factors that influence somatic mosacism- analysis of different tissues show different average repeat lengths within the same individual (Jansen *et al.*, 1994). Of the few tissues studied, expansions are largest within muscle. This would appear to correlate with the symptoms of the disease, however unaffected muscles also have large expansions. A hypothesis to explain this would be that since repeat instability is tissue-specific perhaps the threshold whereby the repeat causes malfunction is also tissue-specific. It would be envisaged that as cataracts are usually the first symptom, these affected tissues would have either the lowest threshold or the most unstable DNA. Tissue specific effects again raise the question of the accuracy of using average repeat length for clinical and genetic studies. Genomic DNA is taken from blood, but there are many different populations of cells in blood, which can vary according to the immune status of an individual. If these populations were to vary in their repeat dynamics then this would be yet another variable that would have to be accounted for. Using progenitor allele of an individual would eliminate this variable.

1.4 Mechanisms of repeat instability

There are many factors governing repeat instability, what are the molecular mechanisms that could account for these observations? Using SP-PCR, multiple reactions using template DNA equivalent to single cells can be used to generate an allele distribution reflecting the mosaicism present. Allele distributions derived from leukocytes over different time periods can be used to build a picture of the type of mutations occurring during somatic mosaicism whilst distributions from sperm would indicate whether germline mutations are occurring. A typical distribution from leukocyte DNA shows a skewed shape with a sharp lower boundary below which no alleles are detected and a tail of larger rarer alleles (Monckton *et al.*, 1995). The progression of somatic instability appears to follow a defined pathway of small expansion biased mutations, starting from a progenitor allele to the characteristic skewed shape to a more normal shape in older patients. Distributions in sperm characterised show a normal distribution compared to the skewed somatic distribution in the same individual. This may be due to some additional meiotic effect. There are a number of proposed mechanisms for repeat expansion which could explain these observations but it is still not known whether any of these are correct. Most of these mechanisms concentrate on the involvement of the repeat itself in expansion, such as the tract forming secondary structure interfering with replication. Indeed, there is a correlation between repeat length and instability. However, if we look broadly at triplet diseases, there is a huge variety in the likelihood of the repeat tracts expanding (long tracts are very unstable in DM1, while in Fragile X syndrome long tracts that are methylated are stable)- yet we would expect there to be a shared mechanism to some extent. If we look at DM1 instability in more detail we see tissue-specific differences with muscle showing high instability and regions of the brain such as the cerebellar cortex showing low instability (Jansen *et al.*, 1994) (Ishii *et al.*, 1996). This can only be explained by the fact that repeat expansion is influenced by *cis/trans* acting factors as well as repeat length. Recently a possible *trans*-acting factor in yeast has been identified- FEN-1, a 5' endo/exonuclease (Freudenreich *et al.*, 1998). GC content has been suggested as a possible *cis*-acting factor (Brock *et al.*, 1999).

Research using cultured DM1 cells, yeast, bacteria and structural biology has provided theoretical mechanisms of repeat instability. It was originally thought that instability occurred in meiosis, possibly by unequal sister chromatid exchange (Smith, 1976), however the presence of somatic instability must mean there is the presence of a non-meiotic mechanism as well. Polymerase or DNA slippage is a model for repeat instability, which is widely recognised (Schlotterer and Tautz, 1992). Crystal studies provide evidence that unpairing of template and nascent strands occurs within the polymerase during replication, which could allow slippage of the polymerase back along the template (Hochtrasser *et al.*, 1994). It is proposed that hairpin or other secondary structure formation would facilitate slippage events, in order to minimise unfavourable energies. This is termed hairpin-mediated slippage (Pearson and Sinden, 1996). It has been shown that pure repeat tracts can form slipped strand DNA structures (S-DNA) on re-annealing *in vitro* (Pearson and Sinden, 1996). These secondary structures are stable and the propensity of S-DNA formation is dependent on the length of the repeat tract with lengths 50> repeats showing a high degree of structure formation (Pearson *et al.*, 1997; Pearson and Sinden, 1996). This model of hairpin-mediated slippage can result in expansions or deletions being produced in the next round of replication. A refinement on this model is the "lagging –strand" model of hairpin mediated DNA slippage (McMurray, 1995). This proposes that replication is blocked by hairpin formation. During this pause, additional DNA synthesis is initiated at single strand regions within the hairpin ultimately leading to expansion. Mismatch repair could play a role in instability as hMSH2 (a protein involved in repair) has been shown to bind to S-DNA structures (Pearson *et al.*, 1997). Other models of repeat expansion include re-iterative DNA synthesis (reviewed in Sinden and Wells, 1992), gene conversion events (which are likely responsible for infrequent reversions of disease alleles back into normal range) (McMurray, 1995) and the recombination gap repair model (Jansen *et al.*, 1994). However, data from yeast does not favour a role for recombination (Wierdl *et al.*, 1997). Another model that has support is the incomplete processing of Okazaki fragments during replication- a model which utilises FEN-1 and has the support of mathematical modelling (Leeflang *et al.*, 1999). The identification of further *cis/trans* acting factors will hopefully provide support from one particular model over the others. The creation of a mammalian model that appears to reflect the tissue-specific somatic instability observed in human patients by (Monckton *et al.*, 1997) (Fortune *et al.*, 2000) may go some way to identifying these factors.

1.5 Mechanism of disease

We have seen that the DM1 mutation is a dynamic mutation whose detrimental effect is influenced by many factors. The instability shown by the repeat provides an explanation to many phenomena- the disease's continued presence in the human population, anticipation and the progressive nature of the disease. We now turn our attention to how the repeat causes malfunction. DM1 is a progressive multi-systemic disorder and the role played by the repeat is being studied in many different fields of molecular biology yet the molecular etiology has largely remained elusive. Only recently, the convergence of results from a variety of different research angles is hinting at a plausible mechanism for the disease, which will be briefly touched upon here.

As the repeat is found in the 3' UTR of *DMPK*, studies have concentrated on assessing the function of *DMPK*, a serine-threonine protein kinase. *DMPK* has multiple transcription initiation sites which ultimately produce different protein isoforms, although only a small proportion have been shown to have functional significance (Jansen *et al.*, 1992). *DMPK* expression is limited to the tissues that are clinically affected by DM1 e.g. cardiac, skeletal muscle, nervous system and retina. DMPK has been shown to phosphorylate a variety of targets such as histones, RNA binding proteins and peptide substrates of cAMP-dependent kinases *in vitro* but does not modify their function (Brewster *et al.*, 1998). It has been suggested that DMPK acts as a Rho kinase in cell signalling cascades (Hall, 1998). This cascade is involved in a diversity of muscular processes, such as cytoskeletal organisation and calcium sensitisation of contractility (Hall, 1998). These findings suggest *DMPK* could be responsible for many of the clinical features of DM1. To investigate whether DM1 is due to loss of function of *DMPK*, knockout mouse were produced and analysed for symptoms of DM. Heterozygotes for *DMPK* appeared normal and unaffected. Closer scrutiny of these revealed minor muscle abnormalities (Jansen *et al.*, 1996) (Reddy *et al.*, 1996). This suggested that loss of function of *DMPK* could be contributing to the phenotype but is not fully responsible for all the symptoms seen in the disease. The fact that no classical mutations within *DMPK* have been identified that cause DM1 also suggest this.

The chromosomal segment, 19q13.3, which *DMPK* maps to is a gene rich area and 2 other candidate genes have been identified in this region which may be affected by the CTG repeat.

DMWD maps 1.1kb upstream of the start of *DMPK* and contains a conserved amino acid sequence (WD repeats) found in signal transduction proteins (Shaw *et al.*, 1993). It is also expressed readily in tissues clinically affected in DM1 such as brain and testes (Jansen *et al.*, 1995).

The other candidate gene identified is *SIX5*, located 1.2kb downstream of the final exon of *DMPK* (Boucher *et al.*, 1995). The CpG island for *SIX5* that contains regulatory sequences overlaps the CTG repeat region. *SIX5* is a member of the *SIX* family of proteins, involved in mammalian muscle, neural plate and eye development. For example *SIX4* is a transcription factor required for the maintenance of retina, muscle and kidney during development (Kawakami *et al.*, 1996) (Kawakami *et al.*, 1996). Hence, a role in the manifestation of symptoms such as cataracts could easily be played by *SIX5* (Winchester *et al.*, 1999). To explore this, the expression levels of *SIX5* in a variety of eye tissues has been studied by *in-situ* hybridisation, RT-PCR, western blot and immunocytochemistry. The expression of *SIX5* in normal eye matches the site of ocular pathology and is also predominantly expressed in adults and is not detected in foetal tissue (Winchester *et al.*, 1999). This corroborates with the fact that cataracts are only found in adults. *DMPK* expression was not detected in adult lens hence may not play a role in cataracts. It must be acknowledged there is difficulty dissecting fresh eyes without contaminating tissues with other tissues.

Studies have provided evidence for a number of pathogenic models, which are not mutually exclusive. The DM1 repeat expansion may affect chromatin structure, which in turn could detrimentally affect replication and transcription. As DM1 is a dominant disorder, the cell would have to be sensitive to changing dosage levels of those genes affected thus, disease may be caused by haploinsufficiency (50% of gene product not enough). The fact that *DMPK* is likely to be involved in signal transduction cascades mean that fluctuations in its production could have far reaching consequences. DNA-binding proteins which target CTG sequences have been identified and could be titrated from the cell by long expansions causing anomalies in chromatin topology (Groenen and Wieringa, 1998).

Studies on *DMPK* expression suggest DM1 could be classified as a *trans*-dominant RNA disorder, with mutant *DMPK* transcripts affecting the processing of normal *DMPK* and other RNA transcripts. An RNA-binding protein, hNab-50 has been identified which binds to CUG repeats (Timchenko *et al.*, 1996). This protein is normally cytosolic, but in DM1 cells is sequestered into the nuclei (Roberts *et al.*, 1997). A homolog of hNab-50 in *Xenopus* called EBEN-Bp is involved in mRNA deadenylation (Paillard *et al.*, 1998). Abnormal localisation of such a protein may severely affect RNA processing. Other ways the expansion could cause pathogenesis include irregular accumulation of mutant *DMPK* transcripts (Taneja *et al.*, 1995). This could cause mRNA transport congestion, which may indirectly inhibit other nuclear functions.

Although the mystery of DM1 is far from being unravelled, these models provide further avenues to be explored. The production of mouse models replicating the full phenotype of the disease, which are currently ongoing, would facilitate the illumination of the possible role of these pathways in DM1.

1.6 Mouse models of unstable DNA

Mouse models would facilitate more detailed analysis of repeat dynamics, answering questions such as whether expansion is age-dependent or time-dependent. Mouse models of DM1 have been developed with varying success. A mouse model containing a 45Kb human fragment consisting of *DMWD*, *SIX5* and *DMPK* with a 55 repeat CTG expansion has been created (Gourdon *et al.*, 1997). Instability and somatic mosaicism have been detected in most tissues but as the CTG is of proto-mutation length, rather than disease length, this is not an accurate model of DM1. The conclusions drawn from this mouse model suggest that flanking DNA is required for instability as other repeat models that have not shown instability have had little in the way of flanking DNA incorporated. The transgene expressed all 3 genes readily in most tissues allowing comparison between instability and expression. It was found that kidney showed the highest degree of somatic mosaicism but the lowest expression. Another question that was addressed was whether the most proliferative tissues show highest instability. This was not the case suggesting that instability is not solely caused by a mitotic mechanism such as replication slippage. In order to produce a model that more accurately reflected repeat instability in DM1 disease and to further explore whether the context of the repeat's location was important, (Monckton *et al.*, 1997) produced integrants containing 162 repeats and minimal flanking DNA from the human locus. Gross, tissue-specific, expansion-biased instability was observed in mice 20 months old- the pattern of which was reproducible in other mice

(Fortune *et al.*, 2000). By looking at other mice at different ages (2, 6 and 13 months), the study confirmed somatic mosaicism is age-dependent and that new mutations occur continually. Deletion mutations are also observed meaning that although expansion-biased, instability is a bi-directional pathway. Interestingly, of the 5 integrant lines generated, this instability is only observed in 1 of these lines, suggesting position effects are crucial. A third model derived from human cell lines, this time containing ~300 repeats has been produced (Seznec *et al.*, 2000). In contrast to the integrants produced by Monkton *et al* which had minimal flanking DNA surrounding the repeat, this includes 45kb of human flanking DNA including *SIX5* and *DMWD* and is the largest repeat expansion integrated. 3 different lines have been generated and 215 transgenic descendants analysed for inter-generational instability. It was observed that inter-generational expansions occurred in 86.5%, 88% and 95.5% of the offspring of each of the three lines, respectively. This ties in with the expansion rates observed in humans (Brunner *et al.*, 1993; Ashizawa *et al.*, 1994). The sizes of expansion vary from +1 repeat per generation to +60 with larger additions more likely on paternal transmission. As inter-generational contractions were more common in the model generated by Monckton *et al*, they suggest flanking DNA may influence the direction of mutations. This supports findings in a study comparing flanking DNA and expandibilty of triplet repeat loci (Brock *et al.*, 1999). Somatic mosaicism was observed in the 300 repeat lines and like humans was found to be age and size dependent. Somatic mosaicism was not detectable at birth but increased with age and somatic mosaicism was more dramatic in these lines than in the 55 repeat model which has the same flanking DNA. These models by Seznec *et al* and Monckton *et al* appear to capture almost all the characteristics of the human DM1 repeat with the exception of very large inter-generational expansions. Already they have provided insight into the importance of flanking DNA on the behaviour of the repeat and will allow experiments to be performed that will provide further answers on the nature of somatic mosaicism.

1.7 Other triplet repeat diseases

DM1 is one of an ever-growing list of genetic diseases that are caused by a triplet repeat expansion. Currently there are 14 triplet repeat diseases identified including Huntington disease(HD), Friedreich's ataxia and the spinocerebellar ataxias (Cummings and Zoghbi, 2000) (Table 1.1). These diseases are generally neurological diseases and are caused by a variety of different repeats. They vary in the position of the repeat with respect to the affected gene which can be in the 5'UTR (FRAXA), 3'UTR (DM1), coding (HD) or intronic regions (Freidriech's ataxia) (Figure 1.1). The also vary by methylation status, the purity of the tract and their relative instabilty. All these diseases show a characteristic repeat length associated with a normal phenotype and a threshold of repeat length which, if exceeded, is associated with the disease phenotype. Triplet repeat diseases can be separated into 2 groups according to whether the repeat tract is translated or not. Type 1 triplet repeat diseases in which the repeat is translated include HD, Machado-Joseph disease (MJD) and Spino-bulbar muscular atrophy (SBMA) and 5 others. These are caused by CAG repeats to give polyglutamine tracts when translated. It is thought that this causes aggregation of the affected protein leading to disease. The largest expansions found tend to be around 80 repeats in the spino-cerebellar ataxias and 120 repeats in HD. SCA7 shows the largest mutations with repeat tract lengths up to 306 having been observed (David *et al.*, 1997) (Benton *et al.*, 1998). Type 2 triplet repeat diseases are not translated and show large and variable repeat expansions- far larger than type 1 diseases. Expanded alleles have been detected which are greater than 1000 repeats (DM1, FRAXA). The repeat behaviour shown by the CTG repeat associated with DM1 is the most dramatic- it has the largest expansions and shows pronounced somatic instability. Other diseases also show somatic mosaicism but to a lesser extent. Tissue specific differences have been detected in HD (Telenius *et al.*, 1994), DRPLA , SBMA and MJD/SCA3 (Ito *et al.*, 1998). Contractions have been observed in FRAXA (Mornet *et al.*, 1996) and differing allele lengths have been observed in SCA7 (Gouw *et al.*, 1998). Anticipation is observed in triplet repeat diseases suggesting that they all are subject to germline instability. Single sperm analysis can be used to build a picture of germline repeat behaviour. Germline instability is pronounced in DM1 and has also been detected in HD (Leeflang *et al.*, 1995), DRPLA (Zhang *et al.*, 1994), FRAXA (Kunst and Warren, 1994), MJD/SCA3 (Takiyama *et al.*, 1997), SBMA (Zhang *et al.*, 1994) and SCA1 (Koefoed *et al.*, 1998). By taking measured allele length from blood to represent progenitor allele length, a mutation rate can be produced by comparing allele length from sperm with progenitor allele length. Mutation rate increases with measured allele length with allele lengths in HD of 38-40 repeats showing a mutation rate of 90% (Leeflang *et al.*, 1999). FRAXA repeats of length 100 show a mutation rate of 55% whilst DRPLA repeats of length 60-62 repeats show a mutation rate of 96-98%. Although there are differences between the various triplet repeat diseases there are many common phenomena particularly the fact that anticipation is always present. This suggests that any disease that exhibits anticipation would be a candidate triplet repeat disease.

Bipolar affective disorder (BPAD) and schizophrenia are 2 psychiatric disorders in which anticipation has been observed in some pedigrees (McInnis *et al.*, 1993) (Gotteman, 1991). BPAD affects 1% of the population and is characterised by symptoms including depression, low mood, sleep, appetite disturbance, psychomotor abnormalities, fatigue, diminished self attitude and cognitive slowing. The symptoms of schizophrenia include hallucinations, delusions, amotivation and paucity of thought. Both of these disorders are polygenic as the symptoms are highly variable and both have been linked to many regions of the genome. Samples from patients of these disorders have been investigated using the Repeat Expansion Detection (RED) method (Schalling *et al.*, 1993). This is a simple technique that involves annealing multimers of the triplet repeat of interest to target DNA then using ligase to anneal any adjacent oligonucleotides together. The products are then run on a gel and provide information about the size of the triplet repeat loci present (by the sizes of the products on the gel). It was found that in BPAD, the sex of the transmitting patient, symptoms and age of onset was linked with the size of the product size produced by RED (Verheyen *et al.*, 1999). In schizophrenia, larger RED products were found in comparison to controls in one study (Morris *et al.*, 1995) but not in any other. In follow up to these experiments a polymorphic CAG repeat was found in a gene linked to schizophrenia (Morris *et al.*, 1995) and alleles greater than 19 repeats were more common in patients than in controls (Chandy *et al.*, 1998).

Triplet repeat sequences have been identified that have not been attributed to any disease. Two of these sequences are CAG/CTG repeated sequences named CTG18.1 (Breschel *et al.*, 1997) and expanded repeat domain CAG/CTG 1(ERDA1) (Nakamoto *et al.*, 1997). Studies have shown that these sequences show the same behaviour as repeats associated with disease. CTG18.1 was mapped to the chromosomal region 18q21.1. Like the DM1 CTG repeat, it is highly polymorphic showing a range of observed allele lengths from 11 repeats up to 2000. Also, allele lengths less than 37 have been shown to be transmitted stably, whilst expansions had been noted in transmissions involving moderately enlarged alleles (53 to 250 repeats) (Breschel *et al.*, 1997). The CAG/CTG repeat named ERDA1 was mapped to 17q21.3 (Nakamoto *et al.*, 1997). It shows a range of alleles from 7 repeats to 92 thus far detected which ties in with the polymorphisms observed in disease associated CAG repeats such as HD. Large alleles at this locus have been shown to be unstable showing a maternal expansion bias and a parental contraction bias (Ikeuchi *et al.*, 1998). Individuals with large CTG18.1 or ERDA1 expansions are apparently healthy meaning they are not directly associated with a disease. They are however candidates for being involved in multigenic disorders which show anticipation, indeed CTG18.1 was identified during a search for candidate genes for BPAD which shows linkage to that chromosomal region. A study by (Verheyen *et al.*, 1999) analysed Belgian BPAD blood samples and a significant difference was found in the ERDA1 distributions of affected compared to controls and a negative correlation with ERDA1 allele size and age of onset was observed. CTG18.1 showed no significant differences between patients and controls.

These results have not been replicated- a further study by (Guy *et al.*, 1999) found no differences between affected and controls. Schizophrenia and BPAD are thought to be multigenic disorders- disease occurs when the contribution of many genes and the contribution of environmental factors exceeds a certain threshold. Therefore, although the presence of anticipation in these disorder suggests a role for triplet repeat mutations, a correlation will always be difficult to produce as the contribution a mutation makes towards the phenotype may not be very great.

1.8 Micro-satellite instability and hereditary cancer

Micro-satellite instability and disease is not solely restricted to triplet repeats, other types of repeats such as mono-nucleotide and di-nucleotide repeats are also disease related. Hereditary Non-Polyposis Colon Cancer (HNPCC) is one such condition. HNPCC can affect as many as 1 in 200 individuals and is distinguished from its sporadic counterpart by strict criteria. This requires 3 or more related family members in more than one generation to have been diagnosed with colorectal cancer before it is pronounced as a hereditary condition (Vasen *et al.*, 1999). HNPCC is caused by inherited mutations in any of 5 DNA mismatch repair enzymes so far identified (Bronner *et al.*, 1994) (Leach *et al.*, 1993) (Nicolaides *et al.*, 1994) (Akiyama *et al.*, 1997). These enzymes are involved in repair of mismatches during replication and have been well characterised in *Ecoli* but less successfully in humans. HNPCC is associated with differing (CA) dinucleotide lengths at many loci in tumours (Liu *et al.*, 1996) (Wijnen *et al.*, 1997) (Aaltonen *et al.*, 1993). A significant age effect was found with microsatellite instability in tumours of individuals 46 years old or younger (Chan *et al.*, 1999). HNPCC is a type of cancer that does not show mutations in the common cancer causing genes such as p53 or KRAS, nor does it show chromosomal rearrangements. This means that the tumour development is facilitated by the breakdown of the mis-match repair system. It could be postulated that as increasing micro-satellite instability is a direct consequence of this breakdown of the mismatch repair system then increased dinucleotide repeat length is playing a contributory role in bringing about tumourigenesis. Micro-satellite instability can also bee seen in other types of cancer such as hereditary ovarian cancer (Wiper *et al.*, 1998).

1.9 Research goals

Like all genetic diseases, when the mutation causing the disease is identified it gives great hope to those affected that a cure or effective treatment may suggest itself. When DM1 was found to be caused by an unstable triplet repeat it was expected that not only would an effective treatment be forthcoming but also information regarding the repeat would allow an accurate prognosis to be given to patients. An accurate prognosis would remove the spectre hanging over a patient that is caused by late-onset disease. Unfortunately, despite the enthusiasm of clinicians and scientists to provide these answers it must be acknowledged that elucidating the molecular etiology of DM1 is a complicated problem. The study of dynamic mutations is a new field and as only half of these mutations identified directly affect the protein product, the repeats must cause malfunction in some wider genomic context. The information from the Human Genome Project already suggests that the vast array of repetitive DNA found in the human genome must play some role in its functioning but we do not yet know how important its role is. We have already seen that triplet repeat sequences are being linked with other severe polygenic disorders such as schizophrenia. The recent completion of the draft Human Genome Sequencing Project will reveal how widespread sequences of triplet repeat are, each of which may have the possibility to expand. Thus, the research being performed on DM1 may not just provide answers for the mechanism of the disease itself but for numerous other diseases and even shed light on how the genome functions as a whole. Despite the complexity of DM1 it still remains the long term goal of those involved to identify a cure, treatment or prevention. The first stage in achieving this long term goal is to gain understanding of the mechanisms that govern repeat dynamics, such as how the repeat expands and the individual and environmental factors that affect this. This understanding would hopefully give rise to more accurate prognoses, which could then be used to help sufferers. Once the process of somatic instability is more clearly understood then therapies to control repeat length would have to be developed. A likely technique would be a variation on gene therapy but at the moment there is no clear mechanism that could be implemented in DM1. It is also important to realise that gene therapy has yet to be successfully implemented as a treatment for any genetic disease let alone a complex one such as DM1. Also, the success of any treatment will have to be monitored at the DNA level, a simple and rapid means to do this will also have to be developed. Understanding the mechanisms that govern repeat dynamics are the focus of the experiments that are presented in this thesis.

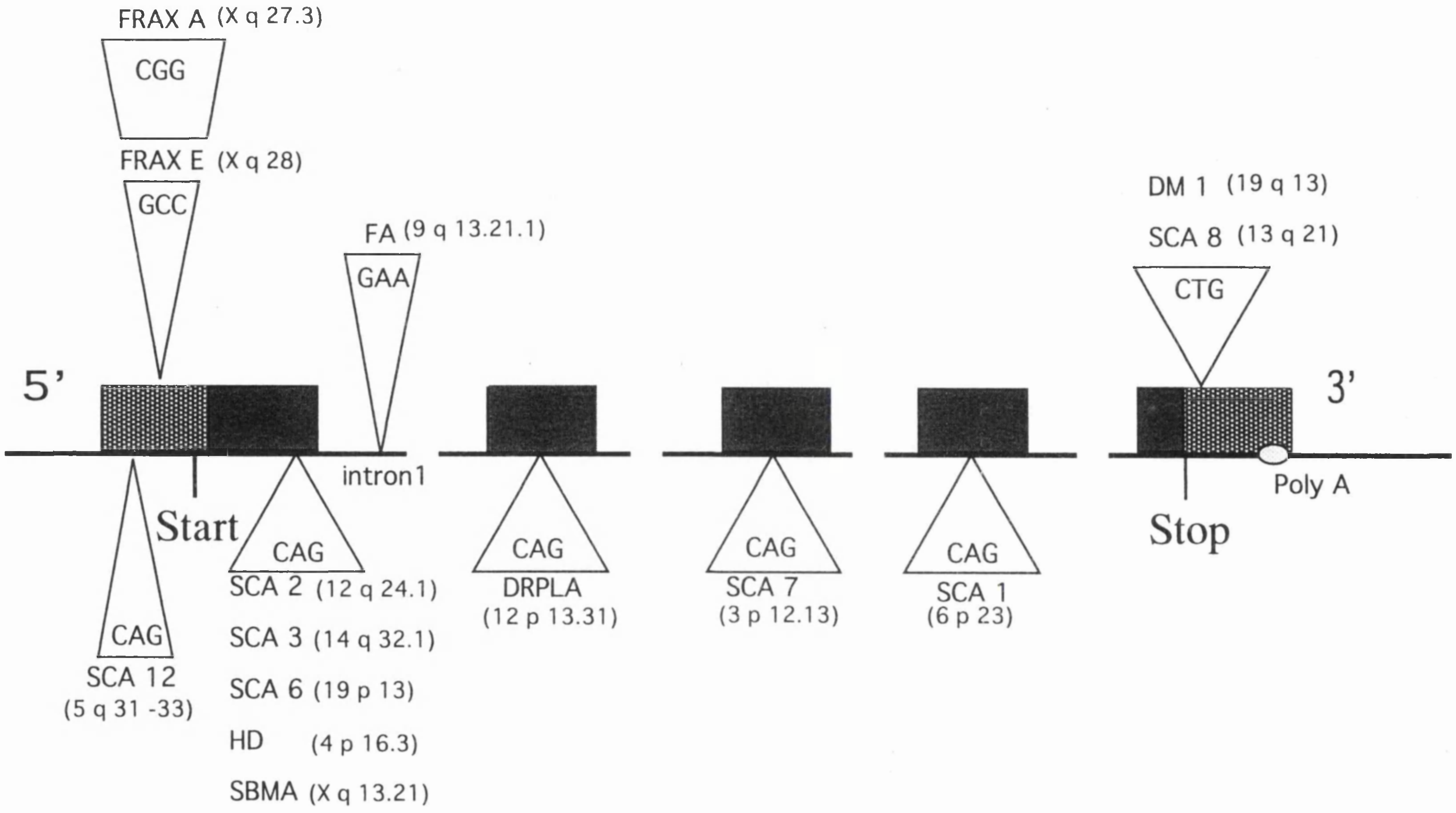


Figure 1.1. Relative positions of triplet repeat expansions. Shown schematically are the relative positions of triplet repeats associated with disease in the gene that they affect. and the chromosomal location of the affected gene. Black areas denote coding exons whilst shaded areas denote untranslated regions. Expansions are found in the 5' UTR, 3'UTR, introns and in the coding regions of their associated gene.

Chapter 2

Materials and methods

2.1 DNA samples

42 DM1 DNA samples that had been previously purified from leukocytes were provided by Dr. Tetsuo Ashizawa from the Baylor College of Medicine, Houston. Clinical information regarding each of the donor patients were also provided including age sampled, age of onset of DM1, CTG repeat length as determined by Southern blot hybridisation of genomic DNA, family pedigrees and brief descriptions of symptoms. DNA samples from 8 DM1 patients taken at 2 different time points were provided by Dr. Montserrat Baiget, Hospital de Sant Pau, Barcelona. These also had been previously purified from leukocytes and CTG repeat length determined by Southern blot. The samples were taken at time intervals ranging from 2 to 5 years.

2.2 Small-pool PCR

Previously purified DNA samples of 500ng/μl were prepared for SP-PCR by digestion with *Hin*dIII and dilution in 10mM Tris-HCl pH7.5, 1mM EDTA and 0.1μM carrier forward primer DM-A (Table 2.1). 40pg - 10ng of *Hin*dIII digested DNA was amplified in 7μl reactions using 1 x PCR Buffer, 0.2μM forward primer DM-A, 0.2μM reverse primer DM-BR (Table 2.1) and 0.05U/μl of Amplitaq. Reactions were cycled through 28 rounds of 96°C (45 seconds), annealing temperature 68°C (45 seconds), extension temperature 70°C (3 minutes) and a chase of 68°C (1 minute) and 70°C (10 minutes) in a Biometra UNO II 96 thermocycler. 3μl of the reaction products were electrophoresed through a 1% 40cm agarose gel in 0.5 x TBE buffer at 160 volts for 18 hours at 4°C. 250ng each of lambda *Hin*dIII digested marker and φX174 *Hae*III digested marker were electrophoresed down the side and middle of each gel. Gels were placed in depurinating solution (10 minutes), denaturing solution (30 minutes) and neutralising solution (30 minutes) and then were Southern blotted overnight using Hybond-N membrane. Blots were hybridised with 20ng of a $(CTG)_{56}$ probe and 4ng each of λ *Hin*dIII and φX174 *Hae*III markers. Alleles were detected by autoradiography (16 hours - 4 days) with exposure to an intensifying screen at room temperature.

2.3 Determination of expanded allele length

Expanded allele length was determined by one of 2 techniques. Densitometry was used via Kodak 1-D Digital Science to determine the average expanded allele of 6 SP-PCR reactions. The alternative approach was to again utilise Kodak 1-D Digital Science to determine the size of every individual expanded allele from multiple SP-PCR reactions equivalent to single cell dilutions.

2.4 Statistical Analysis

Linear regression analysis and data transformations were performed using Microsoft Excel software. Multiple linear regression was performed according to previously described procedure (Armitage and Berry, 1971).

2.5 Reagents

1X TBE buffer

Tris	0.09M
Boric Acid	0.09M
EDTA pH 8.0	0.002M

1X PCR buffer

Tris HCl pH 8.8	45mM
$(NH_4)_2SO_4$	11mM
$MgCl_2$	4.5mM
β-Mercaptoethanol	6.7 mM
EDTA	4.4μM
dATP	1mM
dCTP	1mM
dGTP	1mM
dTTP	1mM
BSA	113μg/ml

1X Te

Tris Hcl pH 7.5	10mM
EDTA pH 8.0	0.1mM

Blue Loading Dye

SDS	0.5% (w/v)
Ficoll	15% (w/v)
Bromophenol Blue	0.25% (w/v)
Xylene Cyanol	0.25% (w/v)
TBE	3x

Denaturing solution

NaCl	0.5M
NaOH	0.4M

Depurinating solution

HCl	0.25M

Neutralising Solution

Tris (pH 7.5)	0.5M
NaCl	3.0M

Primer Name	Sequence (5'-3')
DM-A	CAGTTCACAACCGCTCCGAGC
DM-BR	CGTGGAGGATGGAACACGGAC

Table 2.1 Sequences of PCR primers. Depicted are the primer sequences used to perform SP-PCR.

Chapter 3

Utilising Small-pool PCR to improve clinical correlation in DM1

3.1 Introduction

3.1.1 DM1 clinical correlation

On discovery that the underlying mutation of DM1 was an expanded CTG tract, it was noted there was an association with disease severity and the length of the CTG repeat in the patients used to confirm the mutation (Brook *et al.*, 1992). Analysis of further patients confirmed this- expansion of the CTG repeat shows a positive correlation with disease severity and a negative correlation with age of disease onset (Harley *et al.*, 1993). These analyses were performed by Southern blot of restriction digested genomic DNA obtained from leukocytes and hybridised to a probe from the 3' region of DMPK. This is the standard technique for molecular diagnosis used today (IMDC, 2000). Southern blot analysis has detected expansions in the region of many thousands of repeats. Due to the large size of the restriction digested genomic fragment used in Southern blotting, it is not possible to accurately size smaller expansions and lengths of normal alleles. For this, a single PCR reaction can be utilised, using template DNA in the magnitude of 100ng (IMDC, 2000). Frequently, a diffuse smear rather than discrete band is produced by both techniques. This is explained by somatic mosaicism of the repeat and the mid-point of the smear is taken to represent the average repeat length. Despite the presence of somatic mosaicism, using average repeat length has provided an indicator of disease severity. In most cases patients with late onset disease have between 50 and 150 repeats, classic adult onset patients have between 100 and 500 repeats, patients who show a juvenile onset of disease have between 300 and 1500 repeats and congenital patients have between 700 and 4000 repeats (Redman *et al.*, 1993) (IMDC, 2000). Due to these broad overlaps no prognosis is offered to asymptomatic individuals based on their measured repeat size. Correlation coefficients of repeat length against disease severity and age of onset reflect these broad overlaps giving only a weak correlation. This is puzzling as the behaviour of the repeat mutation can adequately explain most of the phenomena associated with DM1, namely anticipation and the progressive nature of the disease. Yet, statistical analysis suggests that repeat length may not be the major contributor to disease severity and that other genetic/environmental factors must play a greater role in tandem. As we gain further insight into the disease mechanism and repeat instability it is possible this disparity may be due to factors involving the special nature of the repeat which must be accounted for.

3.1.2 Age-dependent somatic mosaicism

The extent and nature of somatic mosaicism has been investigated with the smear of expanded alleles produced by Southern blot analysis used to detail somatic variability (Wong *et al.*, 1995). This was performed by measuring the intensity of the hybridisation signal. Signal intensity differences between normal and expanded alleles were accounted for by subtracting the background signal and taking the mid-peak width ratio to the normal allele. As the normal allele and the expanded allele have the same molar quantity this took into account discrepancies in signal intensity. A correction factor was then used to take into account the inverse logarithmic relationship between molecular weight and distance migrated on the gel. The corrected mid-peak width ratio value was produced for 173 patients with DM1. This data-set was correlated with age sampled and gave a correlation coefficient r=0.81. This strong correlation between age of the individual and the extent of somatic variation suggested somatic mosaicism is age-dependent. This study also demonstrated that there is a size effect on repeat heterogeneity shown in age groups 20 -50 years with heterogeneity increasing with age. Thus from looking at the smear on Southern blot this study has provided evidence for the extent of somatic mosaicism being dependent on the age of the individual and on the size of the progenitor repeat. They propose that this could be explained by continuous expansion or by a small window of expansion followed by preferential proliferation of larger alleles. In order to quantify somatic variation in DM1 patients using a more accurate technique, SP-PCR was used to quantify repeat length variability in leukocyte, sperm and muscle DNA in DM1 patients (Monckton *et al.*, 1995). This involved amplifying multiple aliquots of a DNA concentration from a patient that would give a low number of amplifiable expanded molecules per lane on a gel such that each allele can be individually quantified. The allele distributions in leukocytes showed a skewed shape with a lower boundary below which no alleles were found. The level and variation increased with allele size. DNA from muscle showed increased variation but with the same lower boundary, whilst sperm distributions were notably different to those witnessed in leukocytes of the same patient in particular there is the presence of rare smaller expanded alleles which may account for germ-line reversions. The differences in sperm and blood distributions have implications for genetic counselling. There were a number of observations that provide evidence that these distributions accurately reflect *in-vivo* populations and that they are not PCR artifacts such as heteroduplexes. They noted that no variation was ever present around the normal allele. It was also observed that variation increased with the number of template molecules used and the variations seen corresponded with the Southern blot analysis. The findings of this study allowed the authors to speculate that if as suggested somatic expansion is continuous through life (Wong *et al.*, 1995) then the process must occur in a step-wise addition of small mutations. A study confirmed that somatic mosaicism was age-dependent by utilising the same sensitive SP-PCR approach to 111 DM patients at two different time points separated by 1-7 years (Martorell *et al.*, 1998). Every patient with an average repeat length greater than 200 showed a detectable change in repeat length after 1 year. This phenomenon is likely to be the main confounding factor in age of onset/repeat correlation. Different generations of the same family are measured around the same time with no account taken for an extra 20-30 years of instability occurring between the generations. A more accurate index of correlation may be the inherited allele length which may be measured on young patients as it appears that somatic mosaicism is minimal around birth (Martorell *et al.*, 1998). As somatic mosaicism is age and size dependent (Wong *et al.*, 1995) inherited allele length must be predicted in older patients. The distribution of somatic mosaicism often shows a lower boundary. Somatic mosaicism is tissue-specific (e.g. muscle shows higher instability than blood (Anvret *et al.*, 1993)) which may play a role in the disease manifestation. This lower boundary is present in most tissues with the notable exception of sperm (Monckton *et al.*, 1995). It is therefore conceivable that this lower boundary may correspond to an approximation of the progenitor allele. The creation of a mouse model which mirrors the features of somatic mosaicism observed in humans has indicated that lower boundary does present a reliable estimation of progenitor allele (Fortune *et al.*, 2000).

Information from studies on the mechanism of the disease suggest that there may also be other factors which may hinder clinical correlation. A variety of mechanisms have been suggested which may work together to cause disease. The expanded CTG repeat may cause disease by affecting the expression levels of *DMPK* and neighbouring genes leading to haploinsufficiency. Abnormal mRNA localisation has also been reported with *DMPK* transcripts showing focal accumulation which may in turn inhibit other nuclear functions (Davis *et al.*, 1997) (Taneja *et al.*, 1995). Timchenko and co-workers have identified proteins that bind specifically *DMPK* mRNA CUG repeats (Timchenko *et al.*, 1996) (Roberts *et al.*, 1997). One protein, CUG-BP/hNab-50 which is normally cytosolic is found to be sequestered into the nuclei of cells with largely expanded repeats. These mechanisms suggest that it is feasible that a repeat expansion could be the sole cause of the broad symptoms exhibited by DM1. Furthermore, the models of nuclear clogging and CUG binding protein sequesteration suggest a direct relationship between repeat length and extent of malfunction. The model of haploinsufficiency, if correct, may require that a different approach is taken toward producing clinical correlations. The haploinsufficiency model has support from studies on both *DMPK* and *SIX5*. *DMPK* knockout mouse show some of the symptoms of DM1 (Reddy *et al.*, 1996) (Jansen *et al.*, 1996) (Berul *et al.*, 1999) and there is evidence for nuclear clogging of expanded transcripts which affects transcript levels in the cytoplasm by 50% (Krahe *et al.*, 1995) (Davis *et al.*, 1997) (Taneja *et al.*, 1995). Expression of *SIX5* is reduced in individuals with expanded repeats and mouse knockouts of *SIX5* develop cataracts in a similar fashion to human patients. The model of haploinsufficiency poses the question of how much of a reduction of expression will cause disease and what is the threshold of repeat length that will cause this necessary reduction? There is also the subsidiary question that given DM1 shows somatic mosaicism, what is the proportion of cells that need to cross this threshold to give disease? It is suggested that the repeat size for reduction in *SIX5* expression is between 200 and 400 repeats (Klesert *et al.*, 1997). Whilst it has been noted that the threshold for *DMPK* transcript nuclear accumulation is between 50 and 400 repeats (Hamshere *et al.*, 1997). It has bee subsequently shown that in clinical correlation between repeat length and age of onset there is no apparent correlation between repeat lengths >400 and age of onset. This suggests that once this threshold is crossed the repeat length plays no further contribution to disease severity (Hamshere *et al.*, 1999).

Thus there are a number of considerations that may explain the disparate relationship between repeat length and disease severity. The simplest explanation which does not take into account any of the observations discussed is that the statistical analysis is correct and that there is only a loose relationship between repeat length and disease severity, with other factors being the major determinant. Alternatively, repeat length could be the major determinant in one of two ways: there could be a relationship between small repeat lengths and disease severity but no relationship between repeat lengths that exceed a certain threshold. The other possibility is there is a direct relationship between repeat length and disease severity but the accuracy of measured allele length has been compromised by age-dependent somatic mosaicism and other factors. If the latter were the case then we could expect that progenitor allele (the allele size received at conception) would be the major indicator of disease severity.

Although genetic background will be a factor in repeat instability, we believe repeat length should be the major determinant in age of onset and the reason that this has not been demonstrated in clinical correlations is failure to take into account age-dependent somatic mosaicism. We believe that age-dependent expansion provides the explanation for the progressive nature of the disease. Our hypothesis that age-dependent somatic mosaicism occurs in a step-wise expansion biased mutational pathway that is largely genetically determined would mean that progenitor allele length is the major determinant of the rate and degree of age-dependent somatic mosaicism and therefore age of onset and disease severity. It is interesting to note that a study in which repeat length in patients with minimal somatic mosaicism was correlated with age of onset provided a strong LOG_{10} association (Gennarelli *et al.*, 1996). Patients with minimal somatic mosaicism will either be older patients with small repeat lengths or younger patients with larger repeat lengths that have not had time to undergo significant age-dependent somatic mosaicism. This means that these repeat values used would be not that far off the value representing the progenitor allele.

To address whether using progenitor allele as an index of age of onset will improve clinical correlation we have characterised somatic mosaicism in blood DNA in 42 DM1 patients with a variety of severity, age of onset and repeat number. We have investigated several approaches to determining progenitor allele and it is our aim to apply a simple correction for age dependent somatic mosaicism to improve clinical correlations such that the technique could be applied in any laboratory.

3.2 Results

Repeat length at the DM1 locus was assessed in 49 patients with DM1. The patients came from 8 apparently unrelated families. From these patients 42 were used as the basis of the study who presented symptoms from all DM1 severity classes. Information regarding age, clinical symptoms and family pedigrees were also provided (Table 3.1, Figure 3.1). Repeat expansion was previously determined by Southern blot of restriction digested DNA using the standard technique (Wong *et al.*, 1995) (Table 3.1) as confirmation of clinical diagnosis or as pre-symptomatic test. In order to evaluate the relationship between this data set's Southern blot derived repeat length and age of onset, linear regression analysis was performed between the two variables. This statistical analysis gave a significant negative correlation, $r= 0.60, P<0.0001$ (Figure 3.2A). The value r^2 is taken to mean the amount of variation in age of onset attributable to Southern blot repeat length, which in this case is 36%. This analysis confirmed previous correlations that suggested that repeat length was not the major determinant of the main symptoms of DM1 (Melacini *et al.*, 1995) (Jaspert *et al.*, 1995).

To investigate whether repeat length/age of onset correlation could be improved by attempting to correct for age-dependent somatic mosaicism we performed more detailed analyses of the repeat heterogeneity of the 42 DM1 patients' leukocyte DNA using small-pool PCR (Table 1). The genomic DNA was previously prepared from peripheral blood leukocytes using standard procedures. SP-PCR was performed on dilutions equivalent to ~50 molecules and for each individual at least 6 reactions were performed. Multiple zero DNA control reactions were performed for each set of PCR amplifications to detect contamination. The 42 samples showed a wide variety of somatic mosaicism spread consistent with somatic mosaicism being age and size dependent (Figure 3.3). To investigate whether Southern blot analysis of restriction digested DNA gave an accurate indicator of average repeat length we determined average repeat size by densitometry of the SP-PCR reactions using Kodak 1-D scientific imaging software (Figure 3.3A). We found there were clear differences between Southern blot analysis and small-pool PCR analysis. In particular there was a trend that alleles in the high range (1000+) were distinctly smaller when measured by SP-PCR (see discussion), the most dramatic example being an individual (Sample 137-3666) who had ~1400 repeats when measured by Southern blot but only 739 when densitometry of SP-PCR was used. Using linear regression, the magnitude of correlation between age of onset and average as determined by SP-PCR increased (r=0.64, p<0.0001) (Figure 3.2B). An r^2 value of 0.41 still suggested average repeat length is not the major determinant of age of onset. In order to investigate whether correcting for age-dependent somatic mosaicism could improve these values we then investigated simple ways of determining lower boundary of the allele distribution of a sample. The approaches we favoured would allow an approximation of lower boundary to be determined from any sample regardless of the repeat size or extent of somatic mosaicism. The first approach was to assume that the lowest band observed represented the lower boundary (Figure 3.3B). This was measured for all the patients and regressed against age of onset. The magnitude of correlation was lower than average repeat length. It is likely that this approach does not offer a true representation of the lower boundary as both the presence of deletions which have been shown to occur (Fortune *et al.*, 2000) and spurious bands will severely confound the measured allele length. To account for this the average of the lowest band for the SP-PCR reactions of each sample was taken to represent lower boundary as this would iron out discrepancies caused by spurious bands (Figure 3.3B). This gave a magnitude of correlation of r=0.68 (p<0.0001) – an improvement from 0.60 to 0.68. This simple regression analysis assumes the relationship between age of onset and repeat length is linear. It has been noted that larger repeats (1000+) have a tendency to show greater instability than smaller repeats in some cases (Martorell *et al.*, 1998) suggesting that the relationship between repeat length and repeat instability is a non-linear relationship. To account for this, a simple transformation of Log_{10}(lower boundary+1) was used which gave a correlation coefficient of r=0.79 against age of onset

(Figure 3.2D). This suggests that 62% of the variation in the age of onset is attributable to Log_{10}(lower boundary +1). Regression analysis between Log_{10}(average repeat length) gave a correlation coefficient of *r* =0.77 (*p*<0.0001). These higher *r* values using the Log_{10} transformation suggest the non-linear relationship between repeat length and age of onset is the most important variable with the role of progenitor allele, although providing a stronger correlation in conjunction with the transformation, less important. Other reasons for this may be that the method of determining lower boundary does not represent a good approximation of the progenitor allele. A size effect on the repeat tract's expandibility has been observed and it is possible that this is a non-linear relationship but it could also be postulated that this could be due to the presence of a threshold effect. If there is a threshold then larger alleles would have no bearing on the correlation. The $\log_{10}$ transformation means that larger alleles do not have as great a bearing on the correlation as they would in a linear relationship hence the improvement in correlation.

It is clear that both the average repeat length and the age of the individual when sampled affect predicted age of onset. Therefore, the next logical step was to perform multivariate analysis which would determine the "weight" that average repeat length and the age of the individual sampled conferred upon age of onset. This analysis was performed according to the procedure in (Armitage and Berry, 1971). This gives an equation in the form of $y = a + b_1x_1 + b_2x_2$ where x_1 and x_2 are the two variables: age sampled and repeat length. We would expect that the most weight would come from repeat length with the age sampled variable providing an indication of the amount of age-dependent somatic mosaicism which will have occurred. However, it was found that repeat length had no significant effect on the correlation with age of onset and that the main component was age sampled. It is clear that the age an individual was sampled on should not have the main bearing on age of onset of the disease and that this represents an ascertainment bias. This is explained by the fact that most patients are sampled as diagnostic confirmation of the disease meaning that the age sampled tends to be on or around the age disease symptoms present.

3.3 Discussion

At present, no prognosis is offered to DM1 patients due to the imprecise nature of the clinical correlations. A factor in this observation is likely to be the failure to take into account expansion-biased age-dependent somatic mosaicism. Using the traditional molecular diagnosis of Southern blot of restriction digested genomic DNA correlated with age of onset gives a correlation coefficient of 0.60. This suggests that only 36% of the variation in age of onset is attributable to repeat length and that the length of repeat only plays a minor role with other genetic/environmental factors being responsible. This led us to question the accuracy of clinical correlations. Using lower boundary of somatic mosaicism to account for this phenomenon and a simple mathematical transformation has improved the repeat length/ age of onset correlation from 0.60 to 0.79. An r^2 value of 0.62 suggests that repeat length accounts for 62% of the variation in age of onset. This demonstrates that repeat length is the major determinant in DM1 disease severity. Consequently elucidation of how the repeat causes malfunction will likely explain how the disease arises. Our explanation of poor clinical correlations is due to problems in ascertainment of data. Our revised technique is also subject to problems in ascertainment that must be acknowledged. We have observed that no alleles of greater length than 1300 repeats have been amplified by small pool PCR despite these being detected by Southern blot analysis. This could be due to a variety of factors; perhaps the Southern blot analysis may be inaccurate and the PCR analysis offers a true reflection of the alleles present *in vivo*, however smaller alleles may have a selective advantage in PCR reactions due to competition. We had previously performed experiments to determine the accuracy of PCR amplification on a difficult template such as a CTG repeat which can form secondary structure *in vitro* (Pearson and Sinden, 1996) and could find no evidence of spurious bands (unpublished data). An experimental strategy devised by Zhang *et al* also suggests PCR artifacts are minimal (Zhang *et al.*, 1994). There are also problems in ascertainment concerning the measurement of age of onset. As DM1 is highly pleiotropic, age of onset will be determined according to different criteria depending on what clinical form of the disease is manifest. There is also likely to be a tendency to diagnose patients earlier if there is a family history of the disease. It would be hoped that a standardised diagnosis of age of onset would further improve the correlations accounting for yet more variation. This may be too impractical to use in every clinical laboratory, indeed our primary consideration of the study is that findings should be able to be utilised by clinicians hence the use of leukocyte DNA and standard laboratory techniques. We have noted differences between Southern blot and PCR based detection of expanded allele length. Southern blot of genomic DNA continues to be the technique of choice for molecular diagnosis despite possible inaccuracies and the requirement for a relatively large amount of DNA. This is because of reported problems amplifying CTG repeats, possibly because of their high G-C content. We have experienced little difficulty in amplification of CTG repeats using our protocol (our laboratory has performed several hundred thousand successful amplifications). A protocol has been described which allows successful amplification of expanded alleles from as few as 10 cells (Hsiao *et al.*, 1999). They propose that this protocol could be used for pre-natal diagnosis and population screening studies. We recognise the usefulness of this technique, which only requires limited quantities of DNA, as it increases the avenues of source material for screening. Patients with several hundred repeats will show age-dependent somatic mosaicism and we have observed that allele lengths can show an approximate normal distribution. This means that determining the average allele length is subject to the same rules that govern determining the mean in a population. A sample size of 10 cells would have a large standard error value compromising the confidence of the measured mean representing the true mean. This is not the case for pre-natal diagnosis as somatic mosaicism is minimal at birth.

It has been noted that there is no correlation with repeat length and age of onset with repeats greater than 400 and mRNA localisation experiments suggest an apparent threshold size which may correspond to 400 repeats (Hamshere *et al.*, 1999). *DMPK* mRNA with repeats above this threshold show different cellular localisation. This may provide an explanation for the lack of correlation – the size of the repeat effects the localisation of mRNA and this effect reaches a maximum at 400 repeats. We would suggest that age-dependent somatic mosaicism could also offer an explanation for this observation. Patients with repeat lengths >400 will show significant somatic mosaicism in adult life and if this is not accounted for would confound clinical correlations. Both of these effects could operate in tandem. Also, CDM1 may have to be treated as a special case as it may be reliant on how severe the form of disease shown by the mother. We have demonstrated than an adaptation of clinical diagnosis can significantly improve disease correlations in DM1. We are confident that the repeat itself is the major determinant of the clinical severity and it is the characteristic features of both the genetics of the disease and the wide symptomology are the main confounding factors. Our simple findings provide the foundation for a logical mathematical approach to correct for these factors which would in turn yield even superior correlations. If this is the case then there may be hope of prognostic information being made available to patients in the forseeable future.

identifier	Age of onset	Age collected	Average repeat (Southern blot)	Average repeat (SP-PCR)
101-3001	65	70	100	70.4
101-3008	41	60	78	81.3
101-3071	15	46	770	410.5
101-3080	21	42	470	487.7
101-3082	19	43	830	717.8
101-3057	25	63	1100	840.3
101-3059	51	51	470	421.9
101-3061	25	44	530	484.0
101-3065	29	45	1230	1141.5
101-4010	17	31	1170	999.9
101-5025	11	27	800	649.4
101-4033	20	34	670	525.6
101-4035	20	34	670	457.8
101-4205	17	26	1070	903.7
101-4222	15	18	260	261.9
101-4223	0	17	860	904.3
101-4177	1	34	1360	1035.2
101-4180	18	27	330	271.4
101-3056	39	50	430	280.1
101-4174	18	25	900	1048.3
114-2005	30	38	530	626.4
114-3673	7	8	560	552.6
114-3008	12	14	530	442.4
114-1001	68	68	75	78.3
119-2004	68	68	81	85.3
119-2005	67	68	170	118.0
119-3011	17	40	560	620.6
119-3017	18	36	960	818.5
125-4014	0	12	430	303.0
137-1558	62	72	360	318.5
137-2004	18	32	1460	812.6
137-2664	25	34	530	441.3
137-3666	10	10	1360	739.1
152-2007	13	47	100	636.7
152-3021	19	20	470	530.1
152-3633	12	21	500	518.9
164-3015	21	31	660	701.9
164-2009	60	62	78	90.0
175-1001	63	62	56	60.2
175-2007	31	36	300	225.3
175-2009	16	40	700	343.1
175-2012	26	31	400	557.9

Table 3.1. Details of the 42 DM1 patients analysed in this study. Patients are grouped according to which of the 8 families they come from.

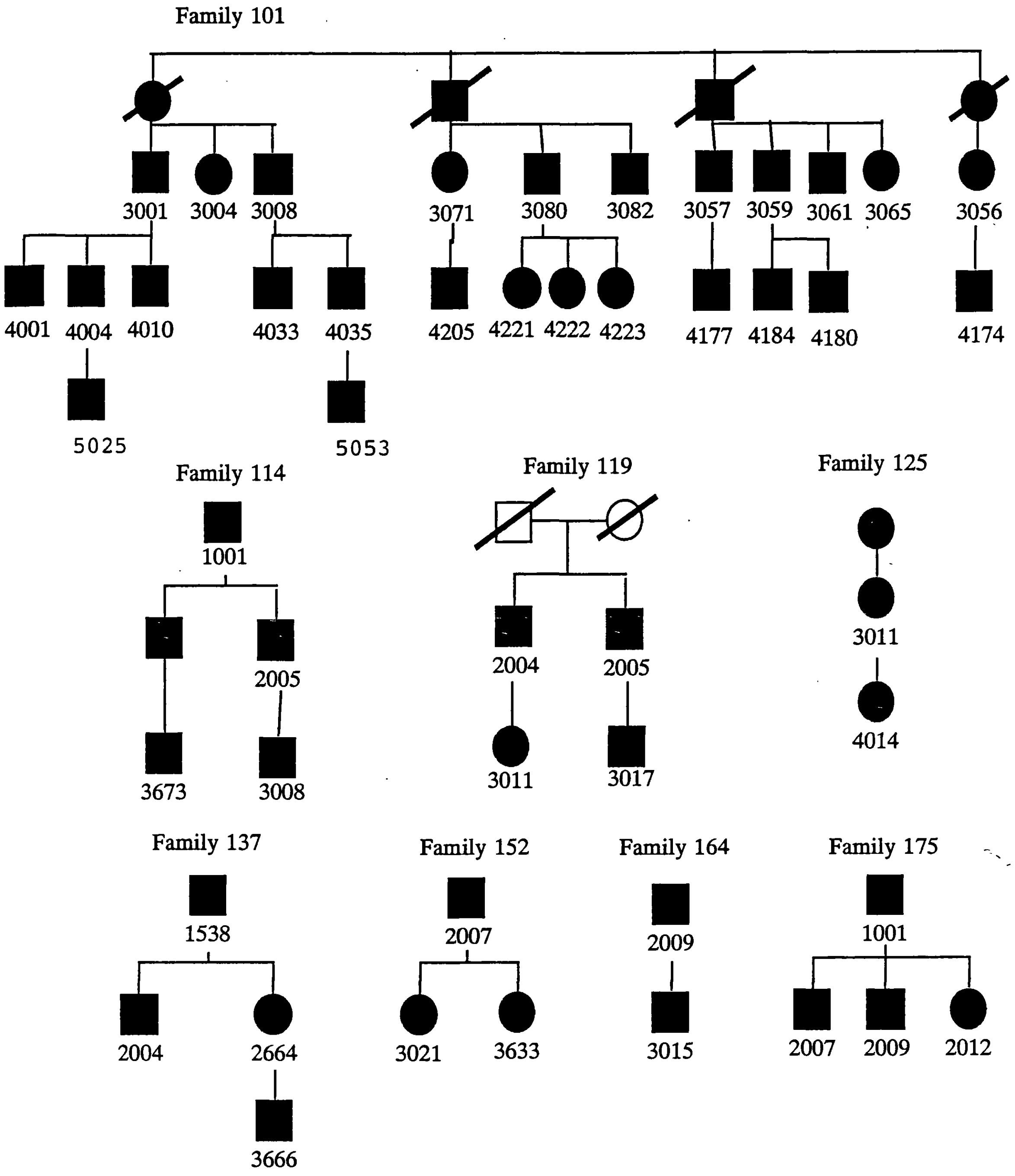


Figure 3.1. Family pedigrees. Pedigrees show relationships between the 8 different families who form the basis of the study.

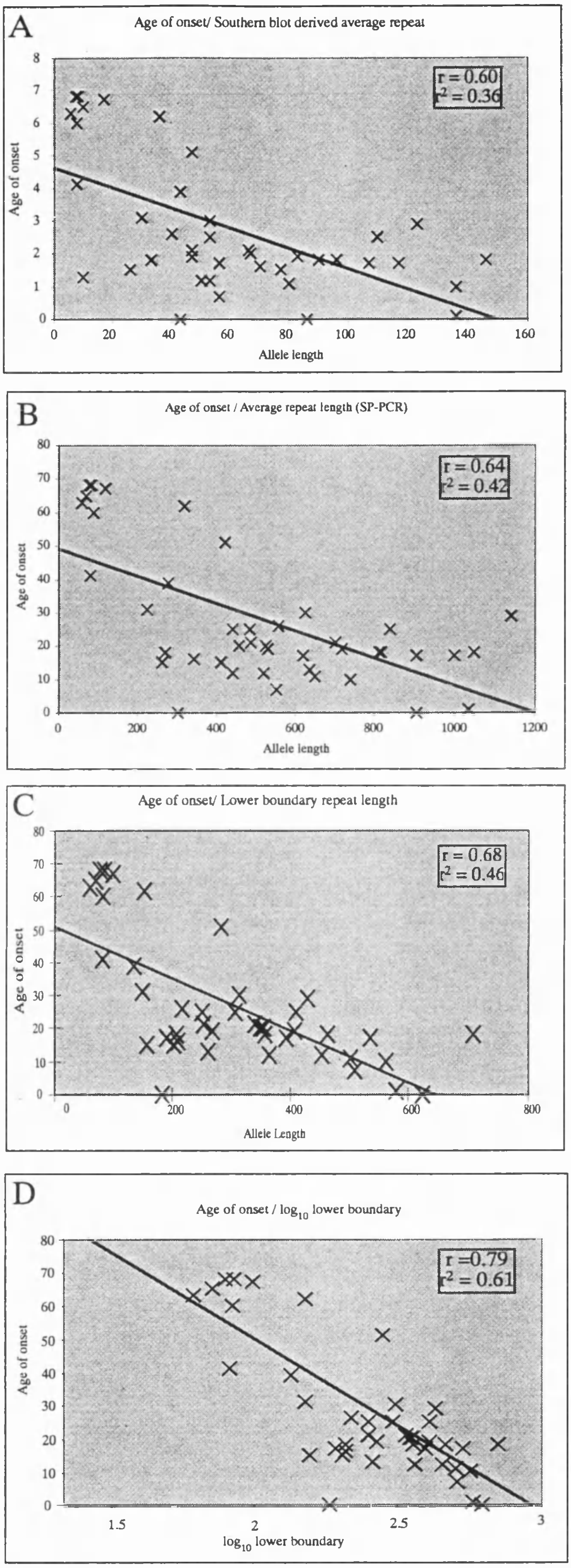


Figure 3.2. Correlation of (CTG)$_n$ repeat length with age of onset. Shown are scatter plots for correlation: (A) average repeat length determined by genomic Southern blot versus age of onset (r =0.60, P<0.0001, n=42), (B) average repeat length determined by SP-PCR versus age of onset (r =0.64, P<0.0001, n=42), (C) lower boundary versus age of onset (r =0.68, P<0.0001, n=42), (D) Log$_{10}$ of lower boundary +1 versus age of onset (r =0.79, P<0.0001, n =42). For each scatter plot the derived linear regression line is shown.

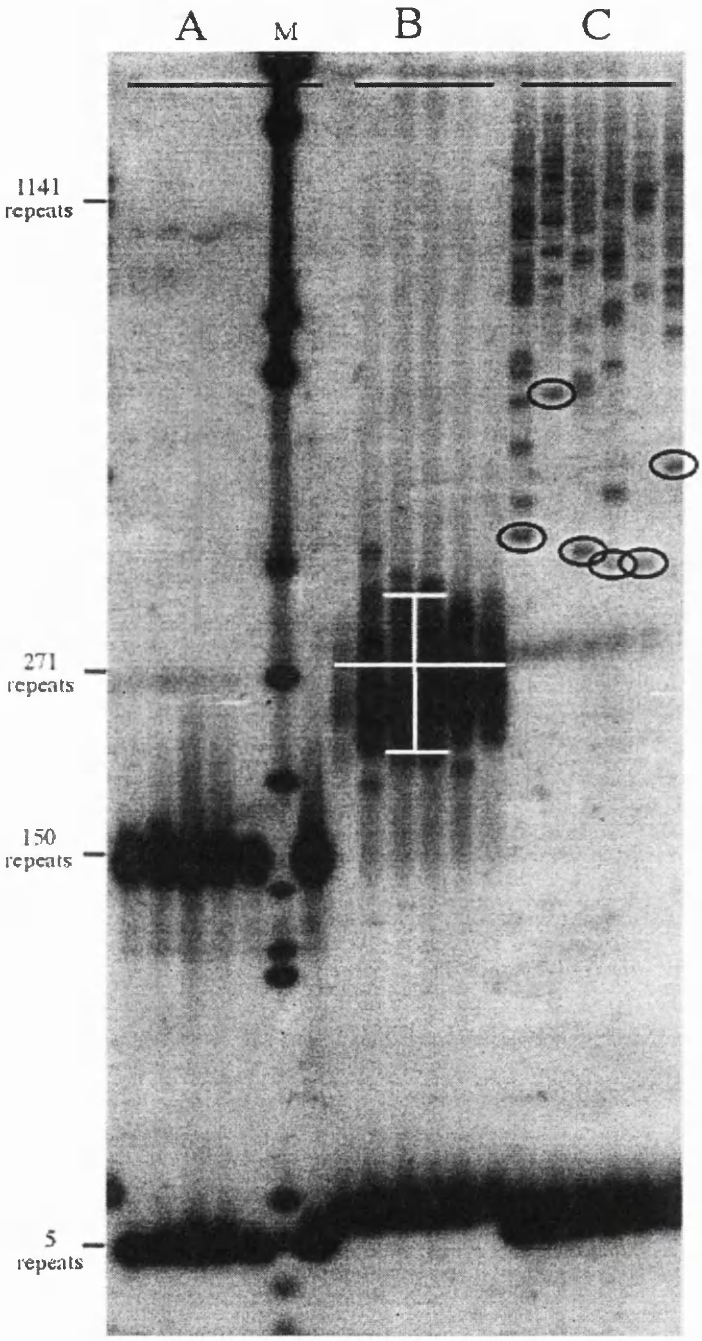


Figure 3.3. Characterisation of DM1 patients by small pool PCR. Samples show a range of repeat length and heterogeneity. 1A shows an asymptomatic individual with 150 repeats. 1B shows an individual with an average of 271 repeats and onset of symptoms aged 18 years. 1C shows an individual with an average of 1141 repeats and onset of symptoms aged 29. Average was determined by densitometry of 6 reactions as depicted in 1B. Lower boundary was determined as the average of the lowest bands per reaction as depicted in 1C.

Chapter 4

Progression of somatic mosaicism in 8 DM1 patients

4.1 Introduction

The expanded repeat in DM1 has been shown to be unstable over time in the same individual. This expandibility is detectable in all patients who have an average repeat length >200 over a time period >2 years. This continuous expansion may play some role in the progression of DM1 symptoms. We have seen that SP-PCR provides a sensitive approach to characterising and quantifying the extent of somatic mosaicism (Chapter 1, Chapter 3) but this technique has not been widely utilised to investigate repeat expansion progression in the same individual. A previous study (Leeflang *et al.*, 1999) used multiple single sperm analyses of CAG repeat length at the HD locus to build a picture of germ-line mosaicism. They compared samples of a patient taken two years apart and could not detect any differences in allele length over this period. This individual had a progenitor allele length of 39 repeats which is close to the disease threshold range in HD. It would be expected that this allele length would not show the same expandibility as larger repeat lengths. By looking at different individuals using SP-PCR a model for the progression of somatic expansion has been proposed (Figure 4.1) (Monckton *et al.*, 1995). It is thought that somatic mosaicism progresses through an expansion-biased bi-directional pathway of small mutations. In distributions that show a skewed shape the lower boundary may correspond to the progenitor allele. In markedly progressed distributions that show a normal distribution it may be impossible to determine progenitor allele as even the smallest alleles present have expanded beyond the initial progenitor length. Using SP-PCR to produce an allele distribution reflecting the somatic mosaicism in an individual at different time points would both confirm whether this explanation for expansion is accurate and whether the lower boundary gives an accurate approximation of the progenitor allele. Being able to accurately quantify individual alleles will allow us to treat the different repeat lengths as a population thus allowing us to determine using statistics whether observed increments reflect a genuine increment or differences due to sampling. If enough individuals were studied it may be possible to detect familial effects upon expandibility. The simplest case would be that all individuals expand at a rate concurring with their repeat length and age. The alternative would be that there were individual specific differences affecting mutation rate. There are mechanisms to explain repeat instability such as replication based slippage and mismatch repair (see Chapter 5). These are complicated pathways involving many stages and many proteins and would widen the opportunity for individual specific effects. To observe whether any simple patterns of mutation rate emerge we analysed 8 patients at 2 different time points in detail using SP-PCR.

4.2 Results

Leukocyte DNA was previously purified in 8 patients with a variety of DM1 clinical forms. They were sampled at different time points with a time interval ranging from 1.75 - 5 years. These samples were taken from patients referred to the Hospital de Sant Pau, Barcelona. Southern blot analysis had previously been performed on the samples and the findings had formed the basis of a study on progression of repeat expansion (Martorell *et al.*, 1998). A preliminary SP-PCR analysis of all the samples was performed in order to determine the concentrations of DNA that would allow optimum quantification of expanded alleles. Our typical procedure when quantifying somatic mosaicism is to use a template concentration of DNA that will allow us to detect 4 expanded alleles per reaction. If it is not possible to size all expanded alleles in a lane then that reaction is disregarded. Individual expanded alleles were sized using Kodak 1-D Digital Imaging software (Figure 4.2) and quantified as allele frequency distributions. The number of molecules quantified per sample ranged from 100 – 558 (Table 4.1). Descriptive statistics of the 8 patients' distributions at both time points were also produced (Table 4.1). All patients showed a detectable increase in the average repeat length in the time interval, although for SDM-B2, SDM-B3 and SDM-B4, the large 95% confidence limits mean that we cannot say that this is a statistically significant increment. All the distributions show a skewed shape apart from SDM-B2, which shows a more normal shape at both time points. These observations are consistent with the proposed model of progression of repeat expansion. The medians also show an increment suggesting that expansion is a result of a general shift as opposed to just alleles in the higher range expanding. These samples had been measured before using Southern blot and showed an increment of between 100 and 267 repeats. The measured increment was markedly smaller using SP-PCR in all cases except SDM-B5. The mutation rate (given in repeats per year) shows a general trend of increasing rate with repeat length except for SDM-B5 (Figure 4.4). SDM-B5 has shown dramatic expansion over a period of 3 years. The mutation rate produced cannot be a constant figure, as in all cases would give a higher than detected average repeat length (using the equation *mutation rate * age + lower boundary*). Thus there must be some kind of fluctuation in mutation rate possibly by age or size.

A value for the lower boundary has also been produced. In order to minimise the influence of spurious bands the 5th percentile value was taken to correspond to the lower boundary. In the case of SDM-B4, SDM-B6 and SDM-B10 the lower boundary values correspond reasonably well at both time points (they both lie within 10 repeats) suggesting that it could approximately represent the progenitor allele. For the other patients the lower boundary figures at both time points do not correspond as well (For SDM-B9 the lower boundary figure is 208 repeats at time point 1 and 237 repeats at time point 2). The shapes of the distributions show a skewed shape with SDM-B2 having the most normal shape distribution suggesting the progression of somatic mosaicism is most advanced in this individual. This corresponds with the large average allele size observed and hence greater effect on repeat expansion. Due to the limited number of patients and information it is not possible to discern any specific patterns of repeat expansion from this data although SDM-B5 appears to be a special case compared to the other patients. The individual has shown marked expansion over a 3-year period. It is interesting to note that this patient had unusually severe symptoms for his measured allele length, and perhaps this represents differences in rates of expansion between repeat length in leukocytes and affected tissues. Some of the distributions produced appear to show more than one peak suggesting a multi-modal distribution. This is particularly evident in SDM-B4, which shows 2 clear peaks (Figure 4.3F). A multi-modal distribution suggests that the measured alleles are from more than one population of cells.

4.3 Discussion

The somatic mosaicism in leukocyte DNA of 8 patients has been characterised at 2 different time points using SP-PCR. This has allowed the quantification of individual alleles to produce allele distributions and statistical analyses to be performed. These samples had been previously characterised by Southern blot and their average allele length determined (Martorell *et al.*, 1998). It had been found that all patients showed an increment of between 100 and 267 repeats over the course of the time interval. We characterised samples by building a distribution of expanded alleles using SP-PCR. This allowed us to take an accurate average allele length and also statistically examine each population of expanded alleles. Making the assumption that the distribution follows a normal distribution we can state that the mean has shown a statistically significant increment in patients SDM-B5, SDM-B6, SDM-B8, SDM-B9 and SDM-B10. The other patients have shown a measured allele increase but we cannot rule out the null hypothesis that no change has taken place. The change in length detected using SP-PCR is markedly smaller than that detected using

Southern blot analysis in all cases except SDM-B5. This may reflect inaccuracies in either method of detection. As the fragments measured in the Southern blot analysis are of larger size, inaccurate measurement of migration distance could result in a greater or lower increment being recorded. It is interesting to note the non-continuous nature of the Southern blot data with SDM-B5, SDM-B6, SDM-B8 and SDM-B9 all showing increments of 167 repeats. A possible inaccuracy of SP-PCR is a bias towards amplification of smaller alleles which would account for a smaller increment being detected. However, we noted that alleles > 1000 repeats were detected in 7 of the 16 samples characterised. Given this observation and the fact that quantification of individual alleles allows statistical comparison to be performed, we would suggest that SP-PCR represents the more accurate measure of average allele length and hence change over time.

We also investigated whether the lower boundary corresponded to the progenitor allele by determining its value at both time points. All the distributions characterised had sharp lower boundaries and the 5th percentile of the repeat data was used to correspond to the lower boundary. In only 3 patients did the lower boundary value remain fairly constant over the time interval - in the other patients the lower boundary figure increased over time. An explanation for this would be that the lower boundary does in fact correspond to the progenitor allele but that taking the 5th percentile is not an accurate index of the lower boundary and that some other statistical technique must be used. From our data, however it would seem that all alleles appear to undergo a general shift of expansion meaning that progenitor allele may be a figure that is less than the lower boundary. This would lend support to one of the mutation mechanisms involving a high mutation rate of very small repeat length changes. We have calculated a simple mutation rate as *rate of change per year* in units of repeats. In all cases this mutation rate could not be a constant figure as it would mean the patients received a progenitor allele of less than zero. This confirms the age and size effect noted on repeat expansion. It may not be possible to produce any useful conclusions from this value as a patient has cells of differing repeat lengths, each of which may be governed by its own specific size-dependent mutation rate. The individual SDM-B5 has shown a dramatic expansion of 158 repeats in 3 years showing a far higher mutation rate than the other individuals. Given that we propose that this individual received a progenitor allele of 130 repeats this change must represent a recent increase in mutation rate possible due to age, repeat length or an environmental effect. It would be interesting to know whether this individual has developed cancer or is a high risk for hereditary cancer.

There is a possibility that some of the distributions show multi-modality as there are several distinctive peaks (e.g. Figure 4.3F, 4.3H, 4.3J). Although, these peaks are not present in the distributions produced at the earlier time point (Figure 4.3E, 4.3G, 4.3I). Large pool PCR could be used to determine whether these are real differences or due to sampling. Multi-modality would be caused by several different cell populations with different repeat distributions being amplified. These samples in this study were taken from DNA purified from blood. There are many different populations of cells found in blood including granulocytes, monocytes and lymphocytes – each of which may have specific differences in their repeat profile. The immune status of an individual varies from day to day and therefore the repeat distribution of an affected individual might as well. Whether these fluctuations are significant enough to affect the average repeat length has not been deduced as patients have not been characterised over the course of several days. Even if this were to be a problem there may be little we can do about it. Although it is possible to separate cells by flow cytometry and use a single cell type as the basis of measured repeat length, this would be too complicated a procedure for most diagnostic laboratories to implement. The alternatives to leukocyte DNA would be muscle biopsy or using sperm DNA. To date, correlation between muscle repeat length and disease has been worse than that found in blood. This fact coupled with the difficulty of obtaining muscle DNA means that the focus of study has primarily remained on blood. Multi-modal distributions have been observed in tissues other than blood such as kidney (Fortune *et al.*, 2000), thus cell populations that perform related functions can have widely differing repeat profiles.

It is difficult to make any specific conclusions regarding this study of the change in repeat distributions over time in the same individual due to the small number patients used for observation (n=8). It appears that even this small data-set demonstrates the wide variety of individual specific differences in repeat profile which can be detected. Environmental factors are being studied by applying chemicals which may mimic the day to day stresses that are conferred upon cells and measuring effects on repeat length by using cell culture models (Gomes-Pereira, Personal communication). Other environmental factors such as diet may represent other variables that could be used to improve clinical correlation although there is no evidence to suggest this will affect instability.

Identifier	Age	Sex	Time Point	Mean S.B.	Mean SP-PCR	Median	No. of Molecules	Increment	Time (Years)	Mutation Rate (rpts per year)	Lower boundary
SDM-B2	29	M	1	400	447±13	452.87	457	26	1.75	14.86	203.4
			2	667	473±21	461.35	228				224.8
SDM-B3	31	M	1	333	311±16	285.76	229	24	2.17	11.06	165.1
			2	433	335±12	317.45	344				180.5
SDM-B4	?	M	1	400	341±14	323.05	424	32	2.75	11.64	158.4
			2	500	373±38	351.56	115				150.4
SDM-B5	38	M	1	167	173±12	154.8	142	154	3.00	51.3	130.6
			2	333	327±22	306.1	128				174.5
SDM-B6	35	F	1	333	285±12	274.87	177	28	4.00	7	158.8
			2	500	313±14	298.34	191				167.9
SDM-B8	34	F	1	333	311±12	295.1	254	52	4.60	11.3	186.6
			2	500	363±24	362.9	100				202.2
SDM-B9	?	F	1	333	335±14	313.25	234	55	5.00	11	208
			2	500	390±10	368.4	558				236.8
SDM-B10	?	F	1	267	286±12	245.7	465	33	5.00	6.6	151.7
			2	400	319±210	264.7	275				149.7

Table 4.1. Patient data at two different time points. Table shows information about each of the 8 patients studied. Information shown is identifier, age of patient, sex, mean for both time points determined by Southern blot, mean for both time points and 95% confidence limits for SP-PCR, median for both time points, the number of molecules quantified by SP-PCR, detected increment in repeat length, time period over which this change was detected, mutation rate in repeats gained per year and lower boundary value for both time points (given as the 5th percentile of the data).

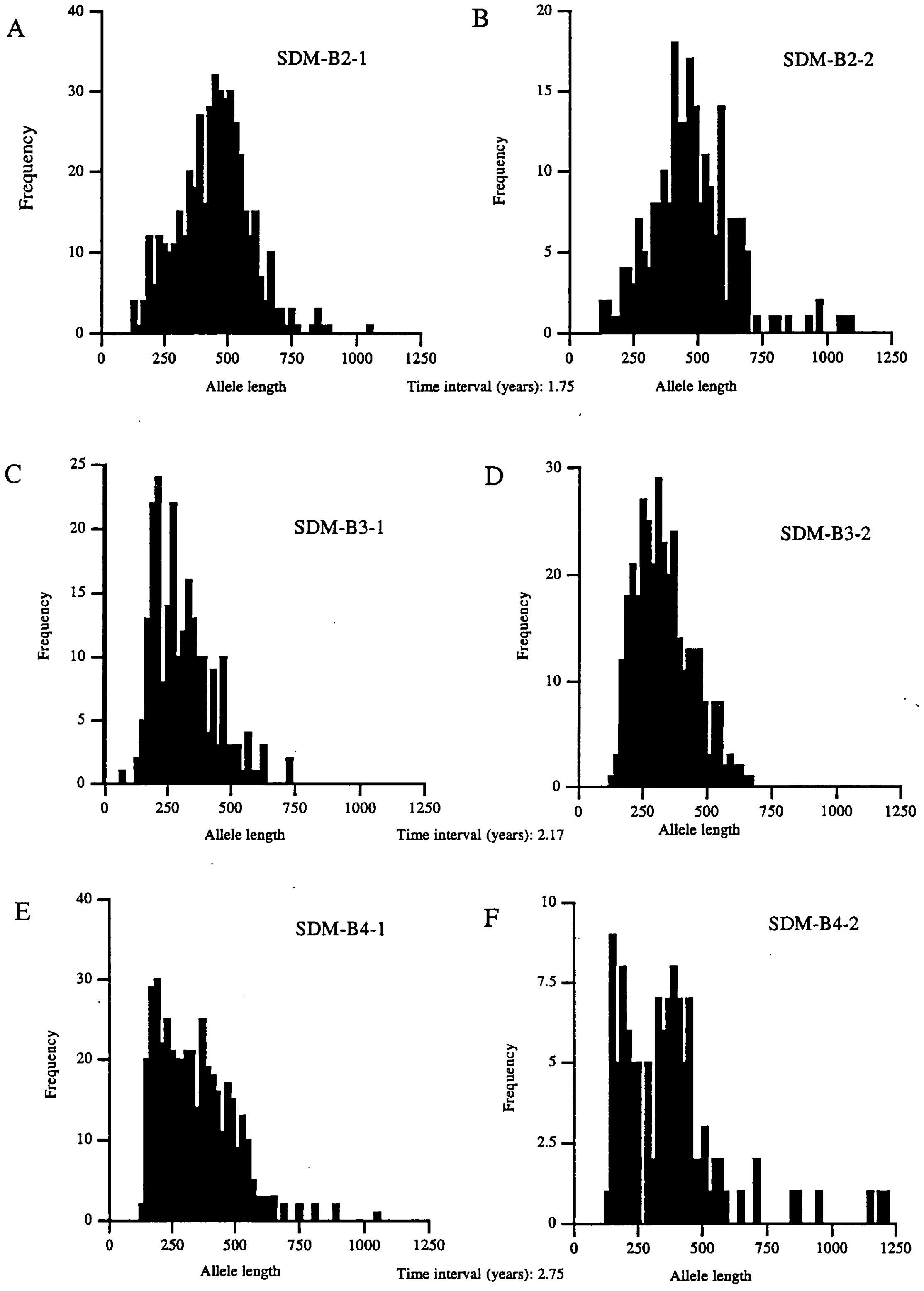


Figure 4.3. Allele length distributions. Shown are histograms of allele lengths (CTG repeats) for the 8 patients taken at two different time points.

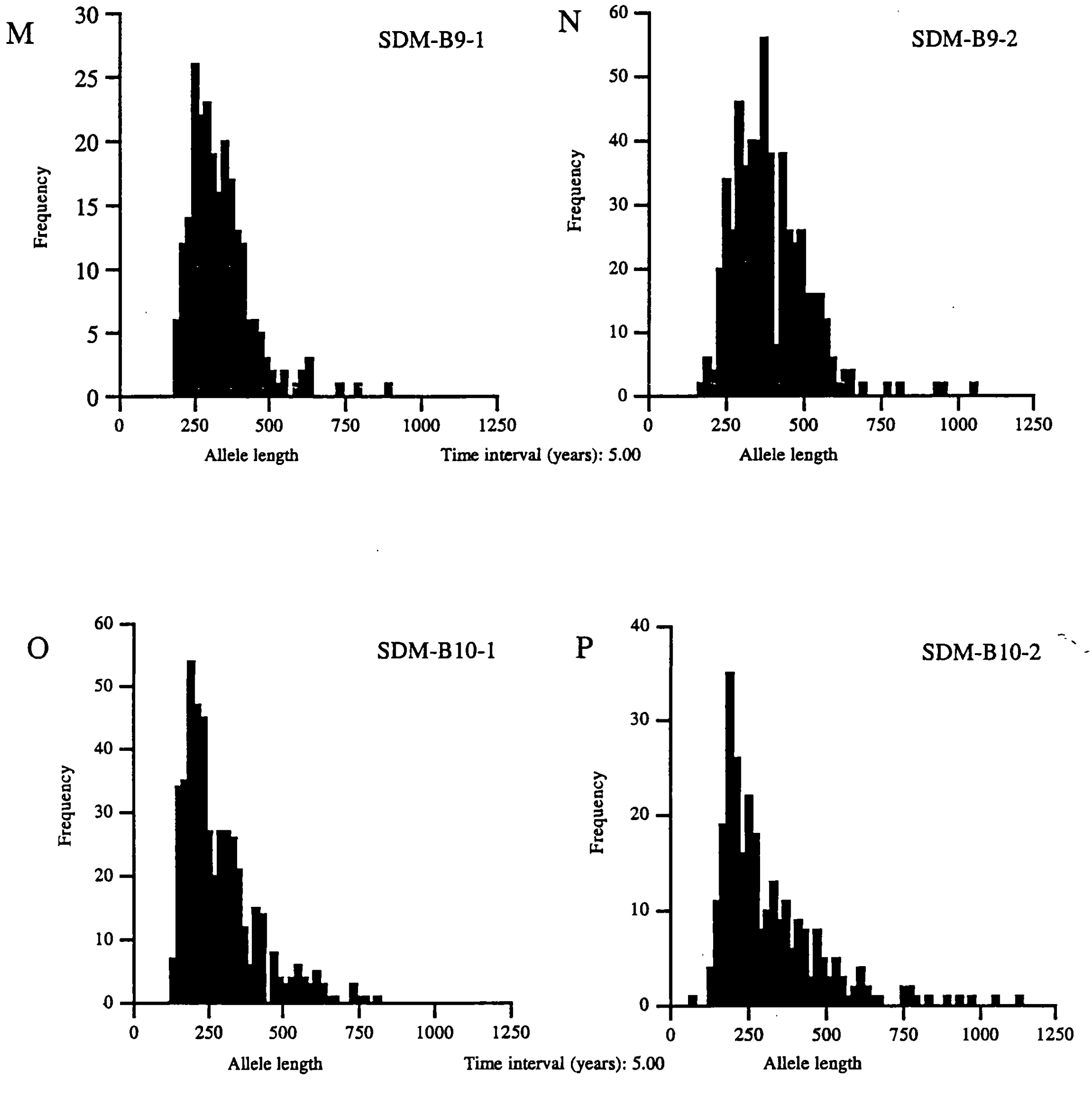


Figure 4.3. Allele length distributions continued.

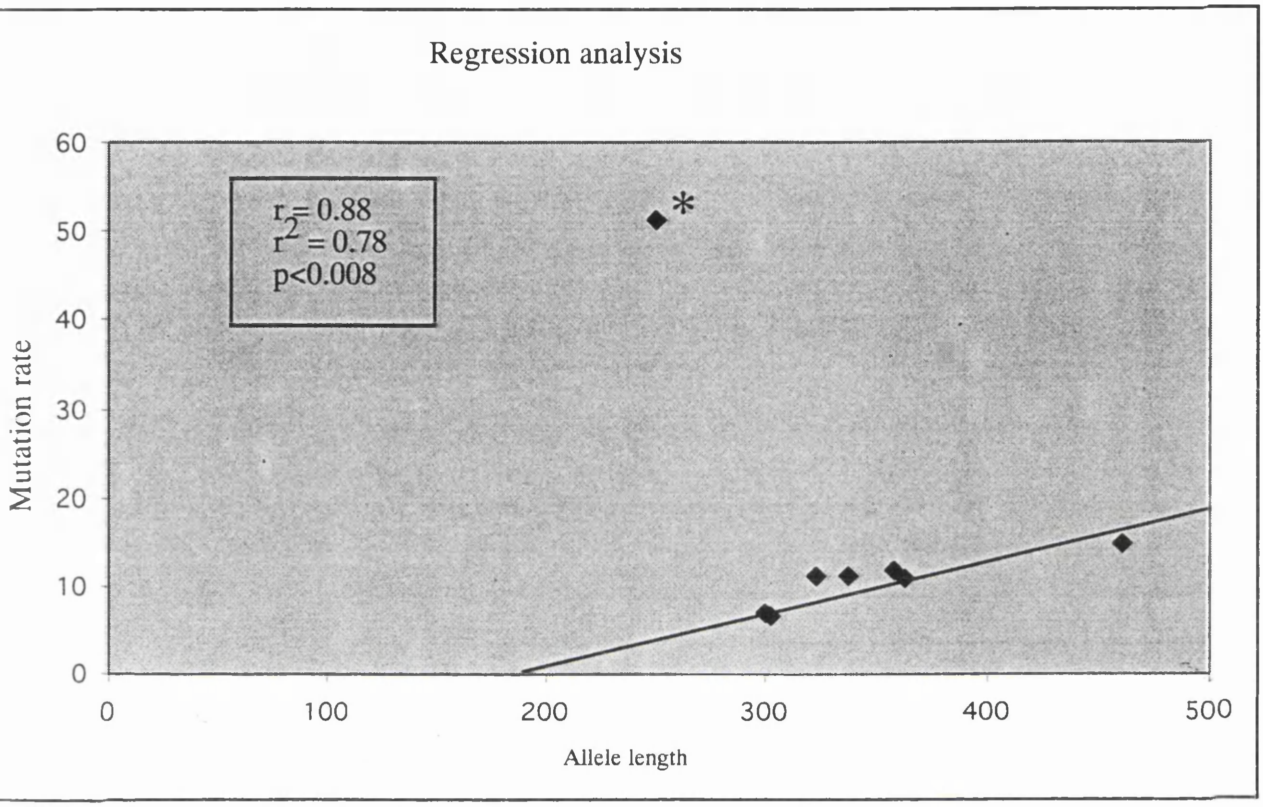


Figure 4.4. Regression analysis of allele length against mutation rate. This shows a positive correlation of mutation rate with allele length (r=0.88, r^2=0.78, p<0.008). The analysis was performed on 7 data points, the other data point corresponding to SDM-B5 (indicated by an asterisk) was not included in the analysis owing to its curiously high mutation rate.

Chapter 5

Computer simulations to model DM1 repeat instability

5.1 Introduction

Analysis of somatic mosaicism at different time points using SP-PCR has led to a broad explanation of the dynamics of repeat instability, namely that it is expansion-biased and involving small repeat length changes. As the role of DNA is to provide the information of heredity, the DNA message is always presumed to be fixed and inherited stably. It is quite alarming then to see the dramatic changes at the DM1 locus occurring within the lifetime of an individual. There are a number of mechanisms that have evolved over the years to maintain DNA fidelity, such as the many DNA repair systems and also the recombination systems that promote variation. It is therefore not difficult to imagine that when these systems go awry, it would have drastic consequences for the genome. However, in the case of DM1, no further mutations have been identified (the transmission of the disease is single gene autosomal dominant) therefore the expansions observed must be due to the inability of healthy DNA replication and repair systems to deal with the CTG repeat itself. There are 2 metres of DNA in a cell, 10^9 nucleotides and only 4 bases to choose from. A single genome will have every combination of trinucleotide sequences by chance so why does a CTG repeat cause problems? Studies from structural biology, computational studies and bacterial/yeast models are providing answers and it appears CAG/CTG repeats have special properties.

Pure CAG/CTG repeat tracts have been shown to form secondary structure *in vitro* (Pearson and Sinden, 1996). When linear polynucleotides consisting of pure uninterrupted CAG/CTG repeats of the same length are run on a polyacrylamide gel they produce a single band. If these polynucleotides are denatured, re-annealed and electrophoresed as before then many bands are seen. If the same DNA is subsequently run under denaturing conditions only the single original band is seen again. The explanation for this was that the CAG/CTG tracts were forming secondary structure termed slipped strand DNA (S-DNA) which was eradicated when the DNA was denatured (Pearson and Sinden, 1996). Interestingly, the threshold of length for observed S-DNA formation is ~50 repeats, the threshold observed for disease. Thus, the CTG tract has a propensity to form secondary structure and this can be brought about by separating and re-annealing strands- a process that occurs in replication and transcription.

Computational studies suggest the presence of an expanded CTG repeat will affect chromatin structure. The sequence of DNA influences its 3-D structure as some sequences will favour certain conformations over others. A study comparing the possible conformation of the different classes of triplet repeats found that CAG/CTG repeats were the most flexible class of repeat (Baldi *et al.*, 1999). This was measured by the extent of DNase1 cutting and a value corresponding to bendability was computed. CAG/CTG repeats have the highest affinity for histones (Wang *et al.*, 1994), the proteins required for chromatin packing. Altered chromatin structure could affect the processes of DNA replication and repair and this has led to a number of possible mechanisms of repeat expansion being proposed. The most favoured theories will be touched upon here.

Replication slippage:

Replication slippage has been proposed as a mechanism for repeat instability (Wells, 1996). This mechanism involves the polymerase dissociating from the replication complex whilst it is synthesising the repeat tract. As the repeat tract can form secondary structure, hairpins or loop-outs can be formed. When the polymerase re-associates it does so upstream of the dissociation point and an extra number of repeat units are eventually synthesised (Figure 5.1). This is a simple model but there is a lack of evidence suggesting the polymerase will dissociate from the replication complex on the leading strand. A more plausible variation on the model (McMurray, 1995) is that hairpin formation occurs on the lagging strand as this strand exists as a single stranded molecule for periods during replication. This would lead to replication stalling when the polymerase is blocked by the base of the hairpin. Replication could initiate from these hairpins leading to expansion or the hairpins could persist until repaired by DNA mismatch repair system.

Re-iterative DNA synthesis:

Secondary structure formation ahead of the replication complex could lead to reiterative DNA synthesis – another mechanism that could explain expansion (Sinden and Wells, 1992). The polymerase could stall and synthesise many repeat units presumably with no limit to the extra repeats added which could lead to very large expansions at the next round of replication. The formation of hairpin structures within the repeat tract could invoke post-replicative DNA repair systems. The hairpins themselves may be recognised and repaired leading to expansion or the mismatches within the hairpin (a hairpin produced from CTG repeats will contain T-T mismatches) may be recognised and repaired causing expansion (Figure 5.2). This model would suggest an equal chance of a deletion or expansion occurring.

Mismatch repair:
If, as suggested, hairpins can form by re-annealing DNA strands (a process that occurs in transcription as well as replication) then mismatch repair could represent a non-replication based method of repeat expansion (figure 5.2). In order for expansion to occur, hairpins of equal size would form on both strands. The hairpin formed on the 3'-5' strand would have to form in a different region of the CTG tract in respect to the hairpin on the 5'-3' strand. This would allow part of the CTG tract to exist as a heteroduplex and depending on which strand is used as the parent strand during repair, expansion or deletion could take place.

Okazaki fragment processing model:
Another model of expansion is the Okazaki fragment processing model (Gordenin *et al.*, 1997) (figure 5.3). A consequence of lagging strand synthesis during replication is that synthesis of the current Okazaki fragment may displace the 5' end of the previously synthesised Okazaki fragment. In order to combat this there are 5' to 3' exo/endonucleases such as FEN-1 that remove any displaced flaps of DNA. As we have seen, CAG/CTG repeats have the ability to form secondary structure therefore the flaps produced whilst replicating a repeat tract could form hairpins which would be resistant from processing by FEN-1. This would lead to expansion mutations at the next round of replication or after DNA repair.

5.1.2 Mathematical modelling

The mechanisms described in this chapter are only a few of the many proposed. The next stage is to try to find evidence that favours one model over another or rules one out of the picture altogether. For example, studies on yeast that have mutations that dramatically reduce recombination result in little change in trinucleotide instability (Miret *et al.*, 1997). This experimental observation lessens the support for repeat instability involving a recombination system. An alternative to experimental observation is to use mathematical modelling to test the viability of mutation mechanisms. This involves generating data detailing the mutations occurring *in vivo* and then determining the probability that any particular model of a mutation mechanism could explain the findings. It is not possible to account for all the factors controlling the mechanism to be tested and there will always be unknown parameters. Thus, a number of biological assumptions must be made which can compromise the accuracy of a model. It is common practise to use the most parsimonious model to explain observed data and the fewer parameters there are in a model the easier it is to perform the statistics. If a particular model does not provide a good fit of observed data then it would suggest other mechanisms are involved. It can also mean that a certain feature of the model such as a particular biological assumption is inadequate. The use of computer simulations of synthetic data can often shed light on where the particular model is going wrong and it can be adjusted accordingly. The final confirmation that a model is accurate is to successfully use it for prediction. Mathematical modelling has been used to give credence to the Okazaki processing model of repeat instability as being responsible for germ-line variation at the HD locus (Leeflang *et al.*, 1999). Among the biological assumptions that are made in this study are the number of cell divisions that have occurred in an individual according to their age (as according to this model mutations can only occur during replication). Examples of the parameters that could be altered are the probability of displacement of an Okazaki fragment and the probability of ligation of a displaced flap to the nascent strand. A mathematical technique called maximum likelihood was used to find the "best-fit" for the observed data. This model has not yet been used to predict future mutation spectra.

The generation of detailed descriptions of somatic mosaicism present in leukocytes at two different time points would allow the possibility of a mathematical model of DM1 repeat instability to be generated. Having touched upon the possible mechanisms that could lead to expansion, a mathematical model of DM1 instability may shed light on which of these is actually involved. Our approach to this would be to use computer simulations to produce a computer-generated distribution of expanded allele based on mutation parameters and compare this distribution to the *in vivo* data. Having data at two different time points in the same individual would allow a more accurate model to be produced as a model could only be deemed correct if it explained both sets of data. In order to simulate the possible mutation mechanisms computer software had to be developed which could incorporate the possible parameters that each mechanism would require. If we look at the 4 mechanisms of repeat instability that we have discussed we can propose the types of mutations that may be occurring.

Replication Slippage:

When slippage occurs, unfavourable energies must be minimised by the formation of secondary structure. Depending on which strand is then subsequently repaired between 1 and *n* repeats are either lost or added (*n* is the maximum number of repeats that can form hairpin structures on the repeat tract and is probably constrained in some way by the length of the repeat tract). The number or repeats added/lost could be a random number between 1 and *n*. Alternatively, there could be a relationship between a particular mutation size occurring and its probability with a mutation size of 1 repeat showing the highest probability and larger mutations showing a lower probability. Slippage could occur more than once depending on the length of the repeat tract with the probability increasing in longer tracts. This may explain the size effect seen on repeat instability.

Re-iterative DNA synthesis:
Mutation size changes observed could be of 3 types: A fixed repeat could be added, some function of the length of repeat tract could be added or between 1 repeat and an assumed maximum could be added. Re-iterative DNA synthesis would allow for many repeat units to be added but there would have to be a theoretical maximum. Again, the length of the repeat tract would increase the probability of this happening at all and the chance of it happening more than once. Whether the expansion persists would depend on hairpin excision and repair or the extra repeats remaining intact to the next round of replication.

Incomplete processing of Okazaki fragments:
Length of mutation added or lost would be between 1 and the number of repeats in an Okazaki fragment (thought to be 150bp therefore 50 repeats). This could be a random number between this range or a probability curve with smaller length changes more likely to occur than larger lengths. A mutation would be more likely to occur on a tract which spans the length of more than one Okazaki fragment. The physical change in repeat length on both strands would again be due to excision of hairpins that had formed to minimise unfavourable energies or persistence to the next round of replication.

Mismatch repair model:
Pure tracts of CAG/CTG repeats have show a propensity to form hairpin structures (S-DNA) *in vitro*. The formation of S-DNA can be facilitated by denaturing and re-annealing (termed re-duplexing) DNA *in vitro*. It is possible the formation of S-DNA could occur spontaneously or during transcription which would allow a non-replication based method of expansion. Hairpins would be formed on both DNA strands and if mismatches were recognised in only one of the strands then this could to lead to additions or deletions occurring after excision repair. This observation has been found *in vitro* with mismatch repair proteins only binding to hairpins on one of the strands (Pearson *et al.*, 1997). The types of mutation that would occur would be additions or deletions of between 1 repeat and the length of the repeat tract. There may be particular sizes of hairpin which are more stable which would lead to specific size mutations being more common. Again the greater the

length of the repeat tract the greater the propensity of S-DNA formation so the greater the probability of mutation.

In all 4 of these mechanisms, some kind of excision repair is required for the change in repeat length to be permanent on both strands and all mechanisms suggest a possible size effect on mutation rate. These are simplified versions of the actual mechanisms with few parameters to alter. We have made the assumption that as the mechanisms allow for a mutation to occur more than once on the repeat tract there should be no constraints according to the number of cell divisions that have taken place.

5.2 Development of Computer Software

Software to model repeat instability in DM1 was developed as a type of application which is nested within a web-site called an applet. Applets are developed using a programming language called Java and have the advantage that they can run on any type of computer that can run a web-browser allowing PC and Mac users to access exactly the same software. The applet was developed on an Apple Mac using the Codewarrior programme development environment. Java code consists of small re-usable pieces of code called classes and methods. The classes and methods were developed using step-wise refinement and how they are linked together (their inheritance) is depicted schematically (Figure 5.4). Each class was developed and tested independently using extreme values. The source code can be found in the appendix.

5.2.1 Description of Computer Software

As the development of the software took up a significant portion of research time a description of the program is presented here. The software produced allows a user to input values corresponding to *in vivo* expanded allele lengths which will then be converted to an allele frequency distribution and displayed. The user can then adjust a variety of mutational parameters which can be used to simulate the progression of repeat instability of a group of cells of a given progenitor allele. This simulated distribution is then statistically compared with the input *in vivo* data and a value representing the similarity between the two distributions is output. The modelling software was developed as an applet to be embedded in a web browser. It utilises a standard graphical user interface of menus and buttons to allow future use by anyone regardless of their computing knowledge (Figure 5.5). The layout of the applet is divided into 3 panels. The left panel shows a graphical display of 2 allele distributions, the derived distribution of *in vivo* data and the model distribution produced by simulation. The graphs are displayed as a histogram of the repeat values with a bin value of 20. The right panel is the control panel. It contains the mutational parameters that can be adjusted by the user. These parameters are:

- Mutation size: This is the number of whole repeats added or lost per mutation event during the simulation. There are 3 options available. A whole negative or positive number can be added per mutation event. A random number of repeats can be added or lost between 2 boundaries given by the user or a mutation spectrum can be used. The mutation spectrum allows certain types of mutations to occur at differing rates and can be selected by the user (Figure 5.6).
- Mutation rate: This is the rate at which the chosen mutations occur during the simulation and can be selected in one of 2 ways. A fixed mutation rate whereby the user enters a whole number corresponding to the percentage mutation rate. The other option is to select a size effect on the mutation rate whereby increasing repeat length causes the mutation rate to increase in a linear relationship. The user is presented with a line graph and is asked to alter the gradient to the one that suits (Figure 5.7).
- Number of Cycles: As the software would be required to test non-replication based mutation mechanisms, this parameter does not represent cell divisions, rather it is the number of times a cell is subjected to the mutational parameters. If the mechanism were to involve transcription we could imagine that in certain cells this value could be far higher than the number of cell divisions (which would be around 50 depending on the tissue and age of an individual). During each cycle a random number will be generated which will determine whether a mutation event will occur according to the selected mutation rate.
- Progenitor allele: This is the starting allele length for the model distribution before any mutation events occur.
- Size of Pool: This is the number of cells in the model. The larger the number the lower the sampling error.

Also found on the right panel is a textbox that allows the user to input the *in vivo* repeat data and the "continue" button that executes the simulation. The lower panel contains the output from the statistical comparison.

5.2.2 Program structure and logic

The simulation utilises random numbers in attempt to mirror the events that are occurring *in vivo*. A cell pool of user-determined length is created with all cells containing the estimated progenitor allele length. The progenitor allele for the *in vivo* distribution to be modelled is estimated using the lower boundary of the allele distribution (See Chapter 3) – this is the main biological assumption we make. Then this pool is subject to the mutational criteria selected for the number of cycles chosen. A random number generator is used and this number is converted to a percentage by dividing by 100 and using the remainder. If a simple fixed mutation rate is selected and if the random number is smaller than the selected mutation rate then the allele length is changed by the mutation size, otherwise it is left unchanged. This occurs for every cell. Using this simple model with a mutation rate of 10% (probability, p=0.1) and assuming each mutation is independent there will be $k * p * n$ mutations where k is the number of cycles and n is the number of cells. For a model population containing 1000 cells we would therefore expect 1000 mutations to occur but as the random number means there will be a stochastic element there is no guarantee of this. As there are many possible values for each parameter, the simulation software has been developed to allow a range of values for each parameter to be tested by iteration and the values are recorded that produce the best fit (see below).

5.2.3 Statistics

The question the simulation software wishes to address is when a population of cells of a progenitor allele length are subjected to simple mutational parameters whether this reflects *in vivo* data. This is achieved by comparing the model population to the *in vivo* population after each cycle of the simulation using the Kolmogorov-Smirnov (K-S) two-sample test. The K-S test provides a value over which we can say 2 distributions are significantly different. The maximum absolute difference of the 2 cumulative frequency distributions of our samples is the value compared to a significance threshold. The K-S test provides a more accurate comparison than comparison of mean and median values.

5.3 Results

The mechanisms of repeat instability we have introduced suggest possible models to test. Our approach was to start with a simple model and further refine it according to results of the simulation until an adequate and parsimonious model was found. We had previously characterised 8 DM1 patients at two different time points (Chapter 4). These individuals show a range of different repeat lengths but are all patients who have shown an adult age of onset. We attempted to model the data generated at the first time point and if a successful model was produced then use it to predict the data we observe at the second time point. If a model could accurately describe the data at both time points then this would give it more credibility. We began testing the simplest model of repeat instability. This model states that changes in repeat length are due to mutations occurring at a fixed mutation rate. When a mutation occurs x repeat units are added (no deletions are taking place)- the simplest case being when x =1. The parameters therefore which could be altered were the mutation rate and the estimated progenitor allele. This model was used to simulate the data generated for patient SDM-B6 at time point 1. The simulation quickly revealed that the range of expanded alleles from minimum to maximum produced using the model would never be large enough to adequately describe any of the *in vivo* distributions.

The model was refined to test whether other values of the mutation size x could explain the *in-vivo* distributions. All possible mutation sizes were tested along with all possible mutation rates and progenitor allele lengths. It was found that a fixed mutation size of 39 repeats at different mutation rates and progenitor alleles could provide a statistical explanation to the *in-vivo* data sets for SDM-B6 at both time points (Table 5.1, Figure 5.8). Although these values provide an explanation to the data, when the simulations were repeated only a few gave a statistical similarity to the *in-vivo* data (~1 in 20). This suggests there is a high stochastic component to this particular model. When this model was used to simulate patient SDM-B5 an adequate fit to the data could not be produced using any combination of parameters. No other patients have yet been tested with this model.

The model was then refined to incorporate a size effect on mutation rate. The simplest scenario is that size effect is a linear relationship with mutation rate increasing as the length of repeat tract according to the straight line equation $y = mx + c$. The possible values for this parameter would range from 0.01 to infinity (a straight vertical line has a gradient equal to infinity). Due to time constraints only a small range of values were tested. Simulations were performed on *in vivo* data from SDM-B5, SDM-B6 and SDM-B9. For SDM-B5 an addition mutation size of 10 repeats, at a range of values for the progenitor allele and the gradient m gave statistical fits at both time points (Table 5.2). This was also the case for SDM-B6 (Table 5.3) and SDM-B9 (Table 5.4) with statistical fits also being produced. It must be stressed that owing to time constraints the full range of parameters were not tested and also the simulations were not repeated to gauge the stochastic variation provided by the model. The main conclusion that can be drawn from these limited results is that the instability observed in DM1 is amenable to mathematical modelling. Neither of the models tested are models of biological mechanisms of repeat instability. The software has been developed that will allow this analysis to take place in the near future.

5.4 Discussion

Computer simulations have therefore ruled out the simplest hypothesis of repeat instability (1 repeat additions at a fixed mutation rate) as being the sole explanation of DM1 repeat instability (we cannot rule out more than one mechanism being involved). Preliminary data suggest a model of a fixed 10 repeat addition with a size effect on mutation rate can explain 3 patients looked at so far at both time points although not every single parameter permutation has been tested yet. Time constraints have meant only 2 of the many models that the software would allow us to test have been investigated. The problem of producing a model of repeat instability can be stated in simple mathematical terms- an equation needs to be produced that explains the values found in an array after time *t* has passed. Because it is a simple mathematical problem, it is possible that there will be more than one possible explanation of the *in vivo* data. Although, having data at several time points for the same individual will limit the possible models, nevertheless it is important to reconcile possible mutation models with experimental observations. For example the 4 possible theoretical mutation mechanisms touched upon in this chapter each suggest there is a likelihood of deletion mechanisms occurring, indeed deletion mutations have been shown to occur in cell culture (Gomes-Pereira, personal communication) and in mouse models (Fortune *et al.*, 2000). A mathematical model is a model of the real world, so these possible models which have no facility for deletion mutations may have to be discounted. It is interesting that in our model accounting for a positive size effect on mutation rate makes a dramatic improvement suggesting this may be a key component. Software has been developed and placed on a web-site, which will allow further models to be tested in the future. There are a number of adjustments that could be made to the software that would improve both the speed of generating results and the robustness of the results. Every possible parameter within realistic ranges has been tested for each model using iteration, which is time consuming but produces water-tight results. The software could be refined to utilise an algorithm that tested values within a range using a step (e.g. testing all the values within the range 1 –10 with a step of 5 would see the values 1, 5 and 10 being tested). The next stage would be to home in on values that gave a better statistical fit by using smaller step values. The time saved adjusting the software would be minimal compared to the time saved running simulations. Currently the software utilises the KS paired test, a statistical test that gives a comparison between the model distribution and the *in-vivo* distribution. A better technique would be to utilise likelihood, a mathematical procedure that answers the same question but also allows different models to be compared with each other so even if several models give a good statistical fit we can determine the best model. As it is clear from the few models tested that there may be a number of models that describe the data, it is imperative that likelihood calculations are used so different models can be compared. Therefore the computer software will have to be modified to incorporate this test before further simulations can continue.

Progenitor allele.	Size (rpts)	Rate %	Poolsize	Sig. Thresh. (time 1)	Sig thresh. (time 2)	*D*(time 1)	*D*(time 2)
131	39	2	1000	0.11	0.11	0.0748	0.0893
138	39	12	1000			0.0644	0.1

Table 5.1. Fixed mutation rate/ fixed mutation size modelling results of SDM-B6. Shown are the parameter values of a simple model that provided a good fit to observed DM1 allele distributions from individual SDM-B6. The *D* values produced for both time points are below the calculated significance threshold therefore the model allele distributions and the observed *in-vivo* allele distributions are not statistically different. The values of the progenitor allele correspond to the measured value of the lower boundary.

Progenitor allele	Size	m	Poolsize	Sig. thresh. (time 1)	Sig. thresh. (time 2)	D(time 1)	D(time 2)
60	10	0.01	1000	0.12	0.13	0.0812	0.0816
70	10	0.01	1000			0.082	0.072
80	10	0.01	1000			0.062	0.0801
90	10	0.01	1000			0.059	0.091
100	10	0.01	1000			0.06	0.108
80	10	0.02	1000			0.071	0.101
90	10	0.02	1000			0.062	0.1216
70	10	0.05	1000			0.112	0.0746

Table 5.2. Size effect on mutation rate (linear relationship)/ fixed mutation size modelling results of SDM-B5. Shown are the parameter values of a model that gave a good fit to observed DM1 allele distributions from individual SDM-B5. The *m* value represents the gradient of the size effect. The *D* values produced for both time points are below the calculated significance threshold therefore the model allele distributions and the observed *in-vivo* allele distributions are similar. However, the values of the progenitor allele do not correspond to the measured value of the lower boundary.

Progenitor	Size	M	Pool Size	Sig. thresh. (time 1)	Sig. thresh. (time 2)	D(time 1)	D(time 2)
50	5	0.01	1000	0.110	0.107	0.0315	0.099
50	5	0.02	1000			0.041	0.05
50-60	5	0.05	1000			0.0264	0.0834
50-110	10	0.01	1000			0.0504	0.047
50-110	10	0.02	1000			0.038	0.056 (progenitor = 80)
50-90	10	0.05	1000			0.0486	0.0516 (progenitor = 70)
50-90	10	0.1	1000			0.0514	0.0514 (progenitor = 70)
60-70	10	0.16	1000			0.069	0.0271
70-90	10	0.2	1000			0.0363	0.0726

Table 5.3. Size effect on mutation rate (linear relationship)/ fixed mutation size modelling results of SDM-B6. Shown are the parameter values of a model that gave good fit to observed DM1 allele distributions from individual SDM-B6. The *m* value represents the gradient of the size effect. The *D* values produced for both time points are below the calculated significance threshold therefore the model allele distributions and the observed *in-vivo* allele distributions are similar. However, only in a few cases do the values of the progenitor allele correspond to the measured value of the lower boundary.

Progenitor allele	Size	M	Poolsize	Sig. thresh. (time 1)	Sig. thresh. (time 2)	D(time 1)	D(time2)
50	10	0.01	1000	0.0987	0.0718	0.07	0.059
100	10	0.01	1000			0.031	0.031
120	10	0.01	1000			0.0538	0.071

Table 5.4. Size effect on mutation rate (linear relationship)/ fixed mutation size modelling results of SDM-B9. Shown are the parameter values of a model that gave good fit to observed DM1 allele distributions from individual SDM-B9. The *m* value represents the gradient of the size effect. The *D* values produced for both time points are below the calculated significance threshold therefore the model allele distributions and the observed *in-vivo* allele distributions are similar. However, only in a few cases do the values of the progenitor allele correspond to the measured value of the lower boundary.

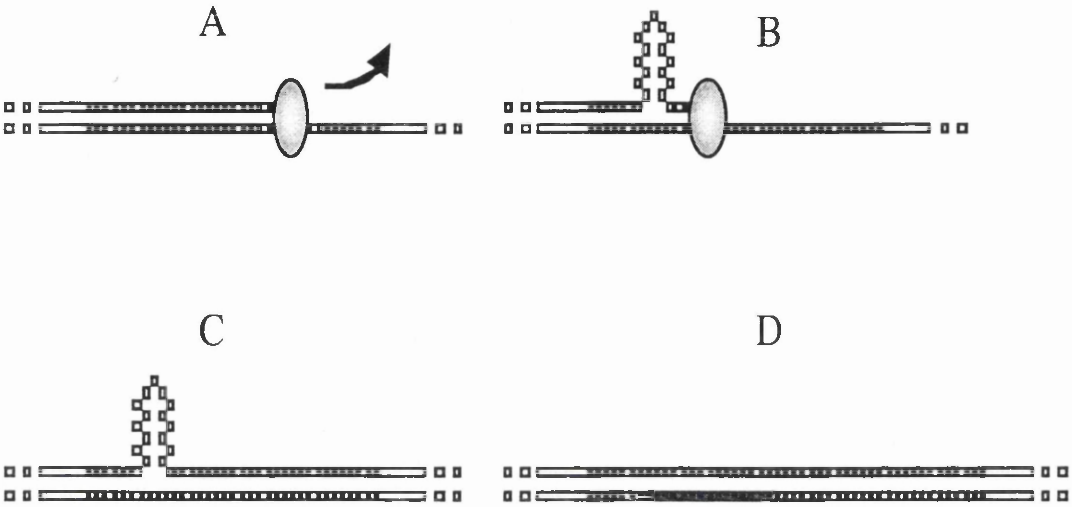


Figure 5.1. Replication slippage. Shown schematically is the mechanism of replication slippage. As replication proceeds the polymerase stalls in the repeat tract (A) allowing the free strand to loop out and form a hairpin (B). Extension continues (C) and the loop-out is repaired, in this case resulting in an expansion (D).

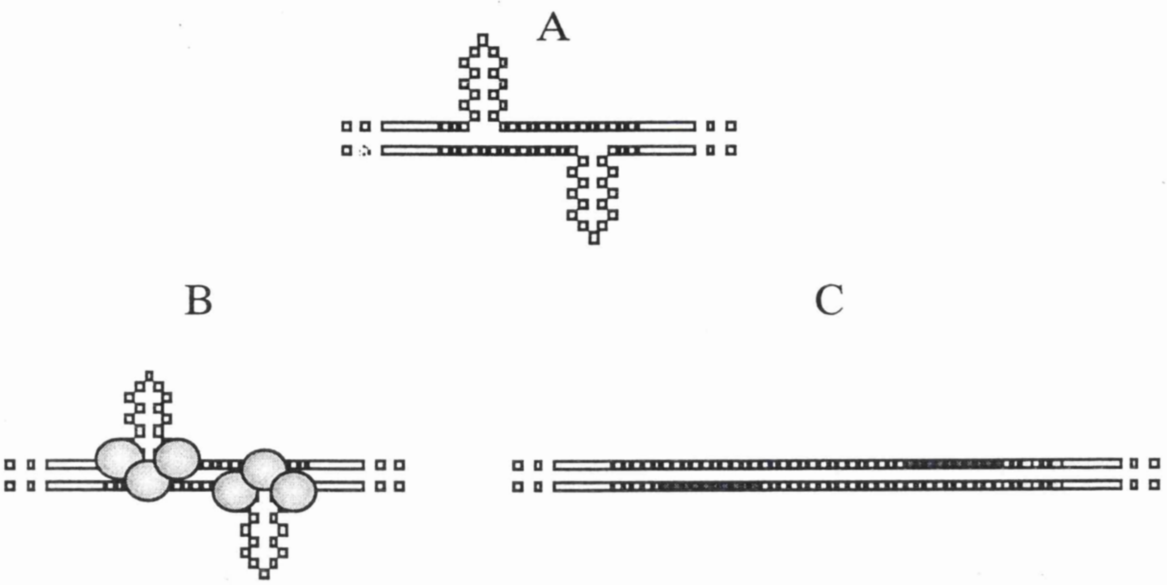


Figure 5.2. DNA repair mutation mechanism. DNA strands re-anneal out of register producing loop-outs (A), which are subsequently recognised as aberrant by the DNA mismatch repair system (B). Loop-outs are repaired which can either lead to deletions or, in this case, an expansion (C).

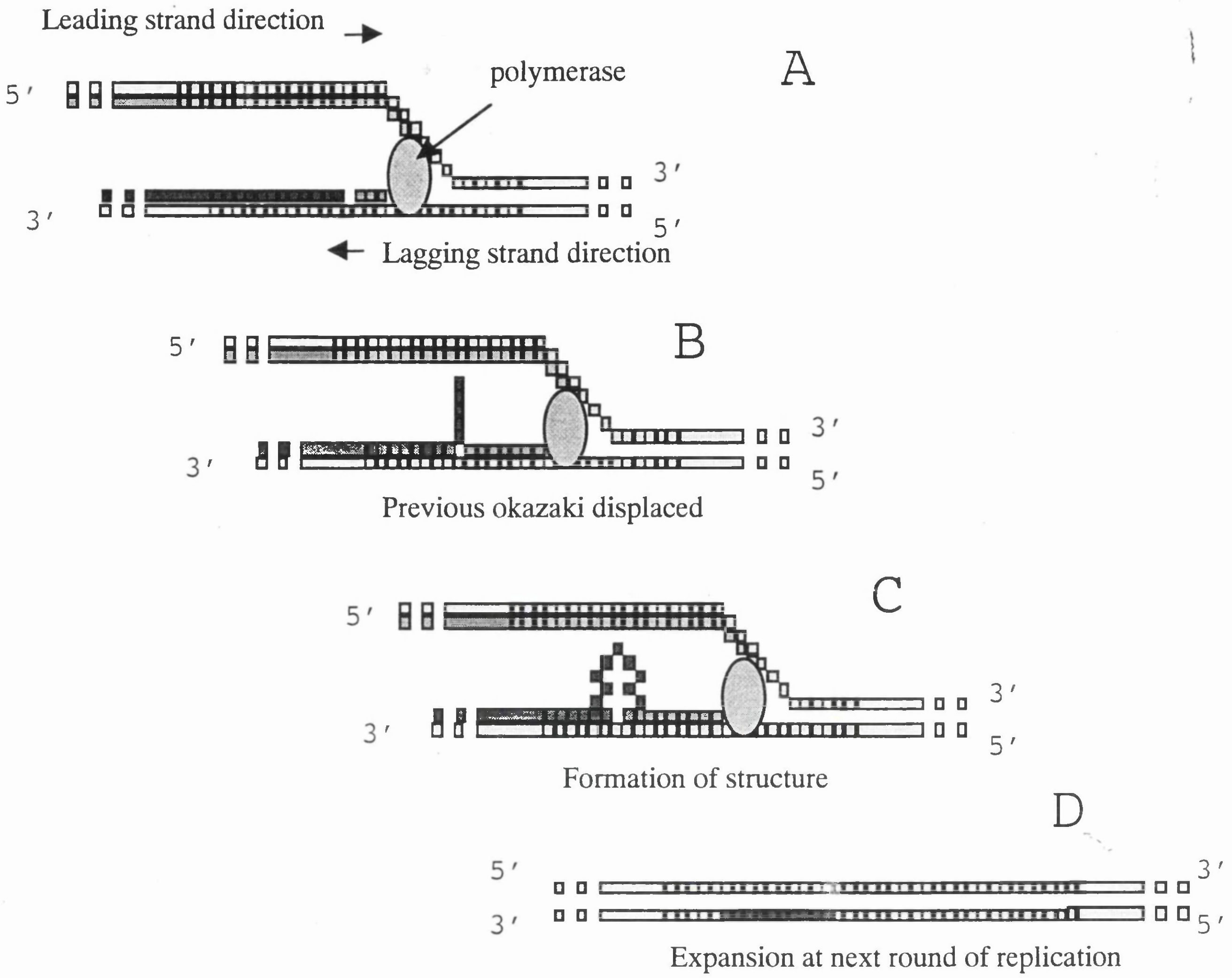


Figure 5.3. Mechanism of incomplete processing of okazaki fragments leading to expansion. During replication the polymerase synthesises DNA on both strands in the 5’-3’ direction (A). This allows the possibility of displacement of the previous okazaki fragment during synthesis of the lagging strand (B). Displaced flaps are usually excised but CTG repeats can form hairpins which are resisistant to excision (C). The hairpin persists to thenext round of replication resulting in an expansion (D).

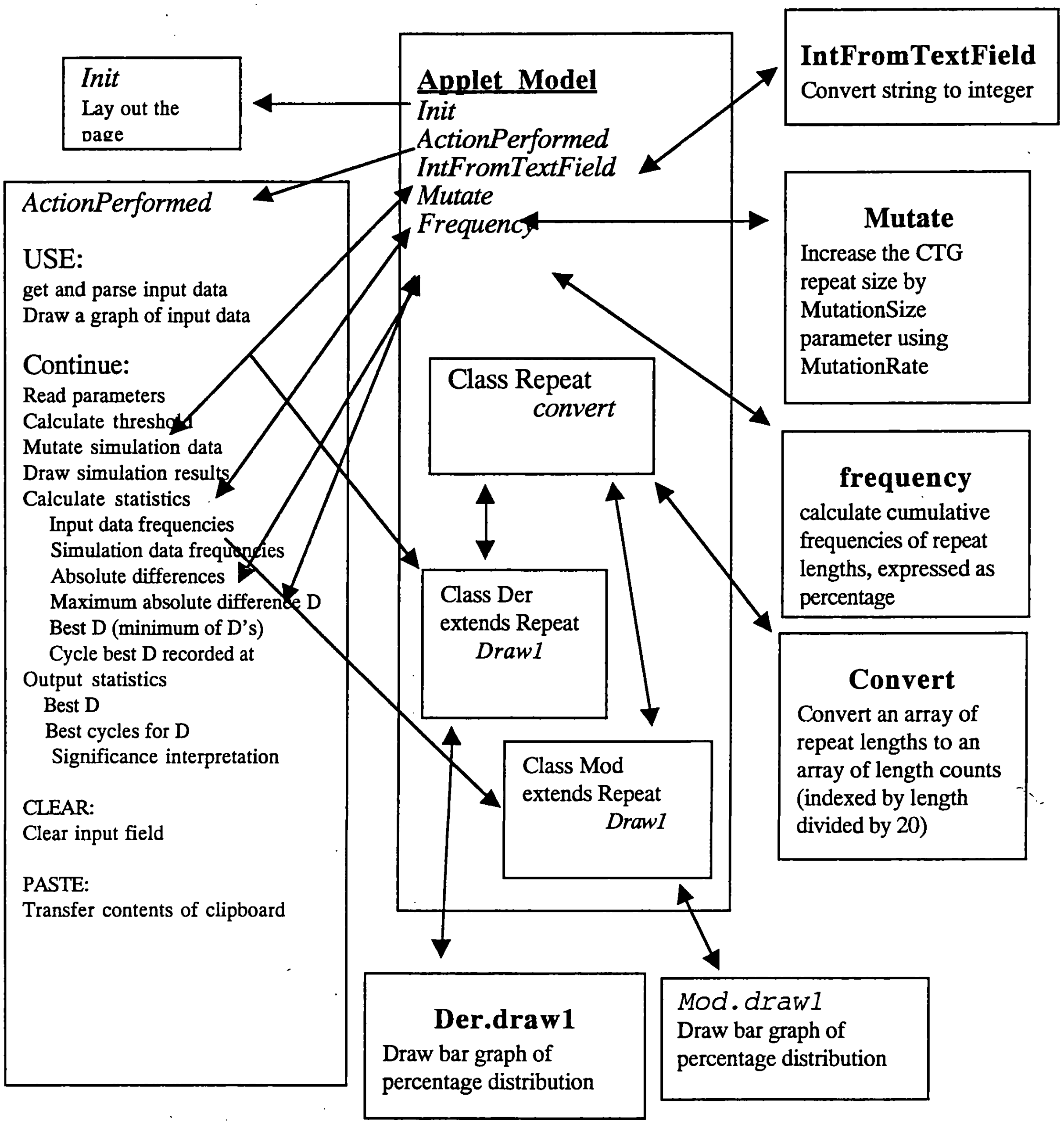
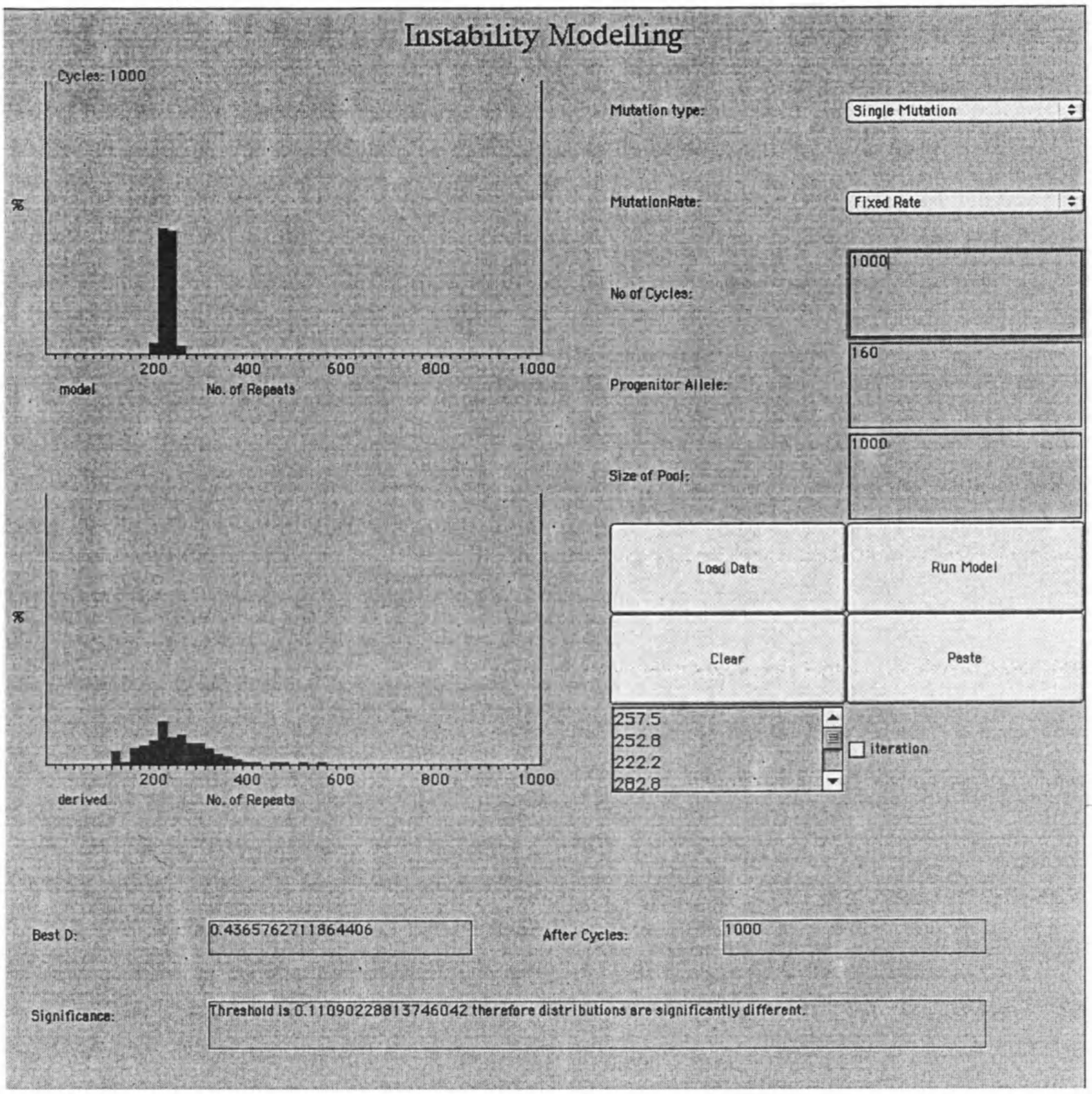


Figure 5.4. Shows inheritance of the classes and methods of the modelling software. Direction of arrow denotes data transfer.



Mutation type: Single Mutation

MutationRate: Fixed Rate

No of Cycles: 1000

Progenitor Allele: 160

Size of Pool: 1000

257.5
252.8
222.2
282.8

□ iteration

Best D: 0.4365762711864406

After Cycles: 1000

Significance: Threshold is 0.11090228813746042 therefore distributions are significantly different.

Figure 5.5. The layout of the modelling software is shown here. The left panel displays graphical output, right panel is the user control section and the lower panel displays the results of the statistical analysis.

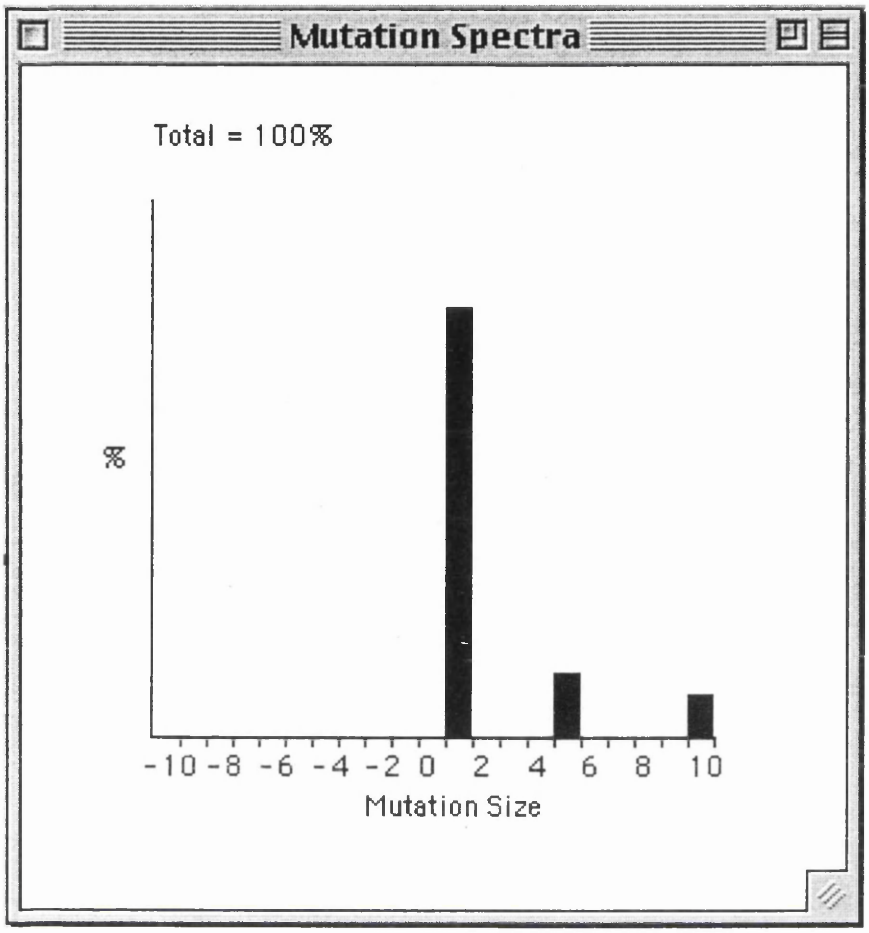
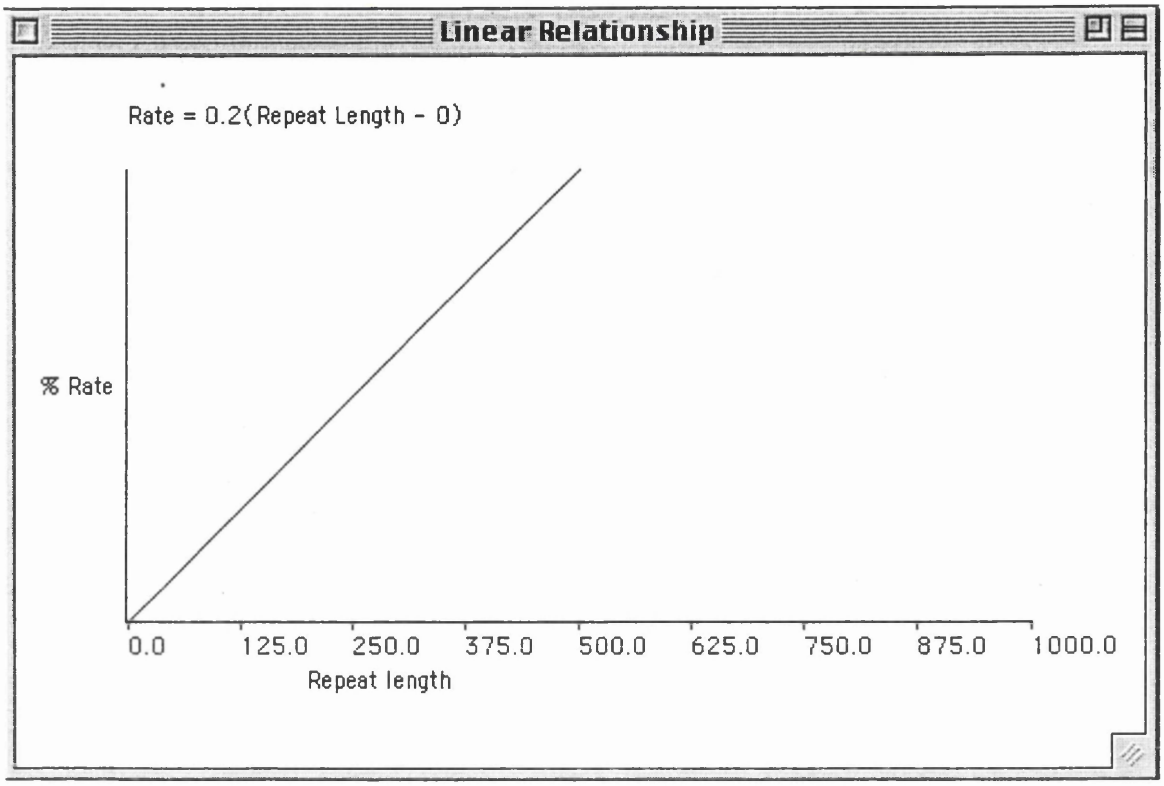


Figure 5.6. Mutation spectrum. A spectrum of different mutation sizes can be selected by the user by means of this adjustable bar chart. In this particular example the user has selected (by means of point and click) a 1 repeat mutation to occur 80% of the time, 5 repeat additions to occur 12% of the time and 10 repeat additions to occur 8% of the time.



Rate = 0.2(Repeat Length - 0)

Figure 5.7. Linear relationship between repeat length and mutation rate. A gradient corresponding to the size effect can be selected by the user by bringing up this window. In this particular case the user has selected a gradient of 0.2 with an intercept of 0.

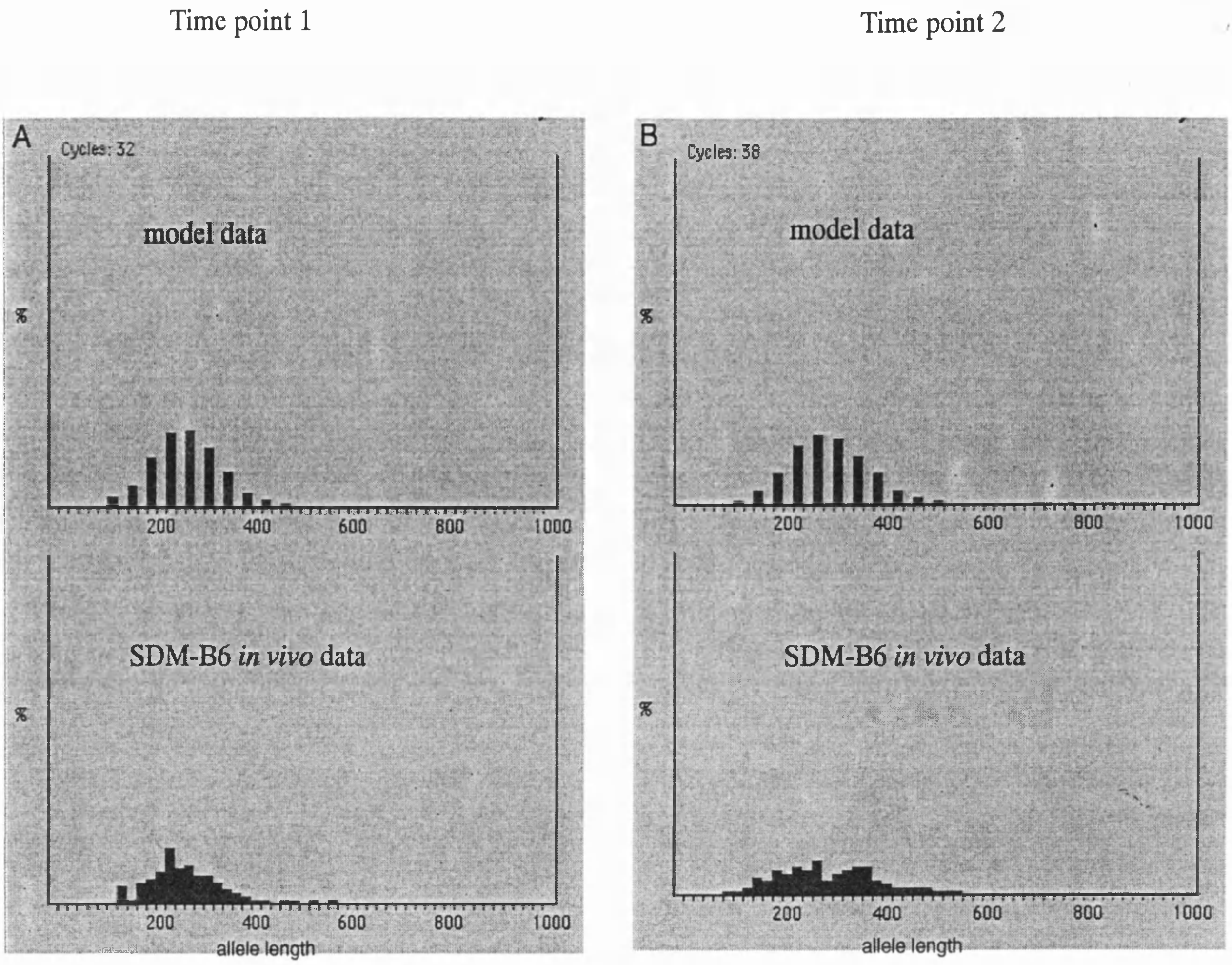


Figure 5.8. Example of attempts to model in vivo data at two different time points. (A) shows computer simulation of SDM-B6-1 using a fixed mutation size of 39 repeats and a mutation rate of 12%. After 32 cycles of these parameters the model distribution of 1000 alleles and the *in vivo* distribution were not statistically different (threshold = 0.11, $D = 0.064$). Using the same parameters the simulation was compared with the *in vivo* data at the second time point (B). The best comparison occurs after 38 cycles but for this particular simulation the 2 distributions are significantly different (threshold = 0.11, D =0.12). When the simulation is repeated a small proportion of model distributions are not significantly different at both time points.

Chapter 6

Discussion

This investigation into the dynamics of triplet repeat instability in DM1 has allowed us to make a number of observations. We have investigated correlation between measured repeat length and age of onset in 42 individuals diagnosed with DM1. Using the traditional technique of Southern blot derived average repeat length against age of onset gave a poor correlation. However, using SP-PCR to measure average repeat length and to predict progenitor allele length each gave successfully better correlations with age of onset (Chapter 3). Using a simple $\log_{10}$ transformation of the data improved the correlation further still. We also investigated change in repeat length over time in 8 different patients. Using the sensitive SP-PCR technique to quantify repeat number has not only allowed us to measure change in average repeat length but also allowed us to observe how time affected the progression of somatic mosaicism as a whole (Chapter 4). These observations suggested that small repeat length changes are responsible for the progression of somatic expansion. Finally, a web-based software application was developed to provide facilities to mathematically model DM1 repeat instability using computer simulations. The current version of the software allows parameters to be altered that could be used to test mutation models of replication slippage, re-iterative DNA synthesis, mismatch repair and incomplete processing of Okazaki fragments. We have used the software to rule out a model of 1 repeat additions at a fixed mutation rate as being responsible for observed *in vivo* somatic mosaicism (Chapter 5).

These observations suggest a number of considerations that have to be addressed. DM1 is a late onset disease whose underlying mutation can provide an explanation for the progression of the symptoms yet no prognosis is offered to patients, due to poor correlation between repeat length and age of onset. We have shown that this correlation can be improved, but the next question is whether the correlation can be improved enough to give us an accurate prediction of age of onset. The first stage would be to determine whether using SP-PCR to predict progenitor allele would improve correlation in a second data set of DM1 patients. We have the possibility of studying west of Scotland DM1 patients referred to the Medical Genetics Department at Yorkhill Hospital but owing to strict ethical practise in the U.K., it is a slow process gaining permission from patients. We have obtained some samples from our counterparts in Costa Rica (16 samples in total) which may allow us to replicate our first set of results. The next stage would be to ascertain how accurate our prediction of progenitor allele is. Data from mouse models suggest that the lower boundary of the distribution of somatic mosaicism does represent a reasonable approximation of the progenitor allele (Fortune *et al.*, 2000). To confirm this in humans, allele length could be measured at birth when somatic mosaicism is minimal and an individual followed and measured throughout their lifetime. A more realistic approach would be to obtain Guthrie cards for current adult patients. Guthrie cards are blood spots taken at birth which are used to test for phenylketonuria and then stored. There are a number of protocols available that can successfully amplify DNA from cells embedded in Guthrie cards e.g. (Hsiao *et al.*, 1999). Thus it should be possible to obtain a measure of repeat length an individual had at birth which would give an indication of progenitor allele. A drawback is that Guthrie cards are much sought after for genetic studies and may be difficult to obtain for patients. A further drawback is that although protocols are available for the amplification of triplet repeats, our attempts to amplify patient repeat lengths from Guthrie cards were not successful. We were able to successfully amplify repeat length from normal individuals' blood dried onto Guthrie cards, although not very efficiently. Due to the low numbers of Guthrie cards we had obtained from patients, we postponed this approach until we could efficiently amplify and reproduce repeat lengths from normal individuals. It may be that large repeat lengths may not be amenable to amplification perhaps due to alternative structure formation. It would be hoped that if this problem were to be overcome there would be a relationship between progenitor allele length deduced from Guthrie cards and somatic mosaicism and that this would allow us to predict progenitor allele length in other patients. We have already seen that the lower boundary value may also increase with time (Chapter 4), therefore a better indicator of progenitor allele may be required which would in turn provide a better clinical correlation. We have also noted that a $\log_{10}$ transformation of the data has markedly improved the correlation. This suggests that there is a non-linear relationship between repeat length and age of onset. A $\log_{10}$ transformation is a simple transformation and there may be other mathematical approaches to deducing the exact relationship between repeat length and age of onset, which would allow us to perform a transformation to account for this relationship. A final improvement would be to standardise the measurements of both the repeat length and age of onset values. We attempted to provide constant conditions for each of the 42 patients amplified and repeated several measurements in order to assess reproducibility. The repeated measurements of average repeat length varied slightly (within 10 repeats) but the protocol we were using did not allow for all alleles to be measured individually, therefore the measurements could not be compared statistically. Standardising measurements of age of onset is a task for clinicians, but it is not made easy by the wide and progressive symptomology shown by DM1. Questions such as when does a weak muscle become a diseased muscle are difficult to answer.

There are obstacles to these improvements but they are achievable. It would have to be decided how accurate the correlation would have to be in order for it to be used by clinicians as a poor prognosis could severely affect the quality of life of an individual. Also, if the correlation was to be used for prenatal diagnosis then the requirement for accuracy is even greater as a decision to terminate a pregnancy may be based on the findings. A particular problem would be where the boundary of repeat length lies between the congenital form and the other clinical forms. CDM1 is associated with its own subset of serious symptoms including developmental defects. It is likely that the desire to bring a CDM1 infant into the world would be less than infants who will develop other forms of the disease but may have many years of normal life. It is suggested that repeat lengths >700 are associated with CDM1, yet there are many patients when measured have lengths >700 but do not have CDM1. Again, this is likely due to age-dependent expansion with the progenitor allele actually being lower than the measured repeat length. The best indicator for the likelihood of CDM1 is maternal transmission. A probable explanation for the lack of paternal CDM1 may be an effect on reproductive fitness. A symptom of DM1 is testicular atrophy, which suggests there may be reproductive problems. Determining the boundary repeat length for CDM1 may be the most likely avenue of success when you consider the obvious symptomatic differences between CDM1 and the other clinical forms.

If the suggested improvements did not yield an appropriate increase in the correlation coefficient then this would suggest some other factor is playing an important role. This may be environmental factors or possibly due to the proposed threshold between repeat length and age of onset. There are many possible genetic and environmental factors that could affect age of onset. We have mentioned genetic factors that could affect the stability of the repeat that act either in *cis* (G-C content) or *trans*(mismatch repair genes) whilst specific chemicals have been shown to affect repeat stability of cultured cells (Pereira, personal communicaton). It is also important to remember there are factors that could influence the pathophysiology of the disease. DM1 is primarily a disease of muscles and if we take a look at ourselves, it is clear that individuals vary in their natural muscle bulk and muscle strength. This is due to genetics and also due to their environment such as the amount of protein in their diet and the physical exercise that they perform. All these observations are likely to play a role in the rate of progressive weakening of the muscles seen in DM1.

Our data cannot rule out the presence of a threshold repeat length above which repeat length makes no further contribution to age of onset. This threshold is thought to be 400 repeats which does not explain how the CDM, the severest form of the disease, is only associated with repeat lengths >700. The presence or absence of a threshold must be reconciled in order for the correlation to reach usable values. The suggestions put forward here could all be currently implemented and an accurate prognosis could improve the quality of life of patients.

A continued observation in these experiments was the differences in measurements obtained by Southern blot and those obtained by SP-PCR. In order to back up the results provided by SP-PCR an investigation exploring the differences between the two techniques on the same patient should be performed. In particular how the measure of somatic mosaicism by Southern blot, given as the mid-peak width ratio, compares to the measure of somatic mosaicism of individually quantified alleles by SP-PCR. The maximum allele length that can be detected by SP-PCR is also a very important figure that needs to be deduced. Due to the very nature of SP-PCR with a period of extension time there will be a maximum, but it needs to be determined whether any of the differences in Southern blot measurement reflect the bias towards smaller alleles or are an artefact of inaccurate measurement by Southern blot.

It would be hoped that in the future the disease could be prevented or treated. As there is the absence of an obvious supplementation as there is for diabetes, any treatment for DM1 would require a novel mechanism. If the disease is shown to be the result of haploinsufficiency, gene therapy could be used in order to hopefully introduce another functional copy of the region of DNA affected by the triplet repeat. As more than one gene is affected, the size of the region that would have to be replaced in order for the loss of function to be overcome would have to be deduced. It is also important to realise that gene therapy has not been successfully implemented in any genetic disease as a treatment let alone a complex one such as DM1. This system of "DNA supplementation" also has the drawback that if treated patients were healthy they would also be genetically fit and able to pass on large repeat lengths to their offspring resulting in more cases of CDM1 with developmental symptoms that would not respond to treatment.

This may be hypothetical as current research suggests DM1 may operate as a *trans*-dominant RNA disorder and supplementing a healthy copy of DNA may not produce the desired effects. A novel mechanism would have to be developed in order to combat this type of malfunction. The most obvious treatment would be to attempt to control repeat expansion as a prevention and to cause repeat contraction as a possible treatment. Attempts to mathematically model DM1 repeat instability along with other fields of study may shed light onto the mechanisms of repeat instability. Once the underlying mechanisms are established, a treatment may become apparent that would prevent the repeat from expanding. For example, it has been noted that contraction mutations do take place but at a lower level than addition mutations. If there were a simple technique to tip the bias in favour of contraction mutations then a potential treatment or prevention could be yielded. Preliminary results on cell culture suggest that specific chemicals can increase the rate of expansion, therefore it is plausible that chemicals may exist that reduce expansion, although the overall effect on the genome would also have to be determined. As DM1 would require a new approach for treatment, these are very much long term goals.

The features of somatic instability seen in DM1 are also seen in Huntington's disease (HD). Expansion-biased age-dependent somatic mosaicism has been observed in an accurate mouse model of HD instability (Kennedy and Shelbourne, 2000). The somatic mosaicism observed follows a different pattern to the progression seen in DM1 with only certain tissues showing increased expansion such as the striatum. The authors suggest that increased severity of symptoms is caused by increased CAG repeat number. Expansion may arise due to the consequence of cells carrying an expansion being vulnerable to further expansion due to age-related or disease-related DNA damage. If this were the case, then it may be possible to predict the degree of somatic mosaicism an individual will have over the time course of their life based upon their progenitor allele length as we are trying to do with DM1 and subsequently predict symptoms. An interesting feature of the observations in this HD mouse model is that striatal tissue primarily consists of post-mitotic neurons, thus increasing the support for a non-replication based method of expansion in HD. In other HD mice, absence of a specific component of the mis-match repair pathway reduces length variability in the striata (Manley *et al.*, 1999). This provides compelling evidence that the mechanism of repeat expansion, at least in HD, involves the mis-match repair system. A possible way of reconciling these observations in humans would be to investigate incidence of HNPCC in patients with triplet repeat diseases. HNPCC is associated with mutations in the mis-match repair system – the lack of DNA repair is eventually thought to lead to cancer. Observations that showed a decreased incidence of HNPCC in triplet repeat disease patients and vice versa would lend support to a mis-match repair mechanism of triplet expansion.

Thus, we have seen how that the preliminary experiments presented in this thesis are the first steps in the long-term approach to producing a prognosis and cure for DM1 in the future and may also increase our understanding of other triplet repeat diseases.

References

Aaltonen, L. A., Peltomaki, P., Leach, F. S., Sistonen, P., Pylkkanen, L., Mecklin, J. P., Jarvinen, H., Powell, S. M., Jen, J., Hamilton, S. R., and *et al.* (1993). Clues to the pathogenesis of familial colorectal cancer. Science *260*, 812-6.

Akiyama, Y., Tsubouchi, N., and Yuasa, Y. (1997). Frequent somatic mutations of hMSH3 with reference to microsatellite instability in hereditary nonpolyposis colorectal cancers. Biochemistry and Biophysics Research Communication *236*, 248-52.

Anvret, M., Ahlberg, G., Grandell, U., Hedberg, B., Johnson, K., and Edstrom, L. (1993). Larger expansions of the CTG repeat in muscle compared to lymphocytes from patients with myotonic dystrophy. Human Molecular Genetics *2*, 1397-1400.

Armitage, P., and Berry, G. (1971). Statistical methods in medical research (Oxford: Blackwell Scientific Publications).

Ashizawa, T., Anvret, M., Baiget, M., Barcelo, J. M., Brunner, H., Cobo, A. M., Dallapiccola, B., Fenwick Jr., R. G., Grandell, U., Harley, H., Junien, C., Koch, M. C., Korneluk, R. G., Lavedan, C., Miki, T., Mulley, J. C., Lopez de Munain, A., Novelli, G., Roses, A. D., Seltzer, W. K., Shaw, D. J., Smeets, H., Sutherland, G. R., Yamagata, H., and Harper, P. S. (1994). Characteristics of intergenerational contractions of the CTG repeat in myotonic dystrophy. American Journal of Human Genetics *54*, 414-423.

Ashizawa, T., Dubel, J. R., and Harati, Y. (1993). Somatic instability of CTG repeat in myotonic dystrophy. Neurology *43*, 2674-2678.

Baldi, P., Brunak, S., Chauvin, Y., and Pedersen, A. G. (1999). Structural basis for triplet repeat disorders: a computational analysis. Bioinformatics *15*, 918-29.

Barcelo, J. M., Mahadevan, M. S., Tsilfidis, C., MacKenzie, A. E., and Korneluk, R. G. (1993). Intergenerational stability of the myotonic dystrophy protomutation. Human Molecular Genetics *2*, 705-709.

Bell, J. (1947). Dystrophia myotonica and allied diseases, L. S. Penrose, ed. (Cambridge: Cambridge University Press).

Benton, C. S., de Silva, R., Rutledge, S. L., Bohlega, S., Ashizawa, T., and Zoghbi, H. Y. (1998). Molecular and clinical studies in SCA-7 define a broad clinical spectrum and the infantile phenotype. Neurology *51*, 1081-1086.

Bergoffen, J., Kant, J., Sladky, J., McDonald-McGinn, D., Zackai, E. H., and Fischbeck, K. H. (1994). Paternal transmission of congenital myotonic dystrophy. Journal of Medical Genetics *31*, 518-520.

Berul, C. I., Maguire, C. T., Aronovitz, M. J., Greenwood, J., Miller, C., Gehrmann, J., Housman, D., Mendelsohn, M. E., and Reddy, S. (1999). DMPK dosage alterations result in atrioventricular conduction abnormalities in a mouse myotonic dystrophy model. Journal of Clinical Investigation *103*, R1-7.

Boucher, C. A., King, S. K., Carey, N., Krahe, R., Winchester, C. L., Rahman, S., Creavin, T., Meghji, P., Bailey, M. E. S., Chartier, F. L., Brown, S. D., Siciliano, M. J., and Johnson, K. J. (1995). A novel homeodomain-encoding gene is associated with a large CpG island interrupted by the myotonic dystrophy unstable $(CTG)_n$ repeat. Human Molecular Genetics *4*, 1919-1925.

Breschel, T. S., McInnis, M. G., Margolis, R. L., Sirugo, G., Corneliussen, B., Simpson, S. G., McMahon, F. J., MacKinnon, D. F., Xu, J. F., Pleasant, N., Huo, Y., Ashworth, R. G., Grundstrom, C., Grundstrom, T., Kidd, K. K., DePaulo, J. R., and Ross, C. A. (1997). A novel, heritable, expanding CTG repeat in an intron of the SEF2-1 gene on chromosome 18q21.1. Human Molecular Genetics *6*, 1855-1863.

Brewster, B. S., Groenen, P., and Wierenga, B. (1998). Myotonic dystrophy: clinical and molecular analysis. In Neuromuscular Disorders: Clinical And Molecular Genetics, A. E. H. Emery, ed. (New York: John Wiley and Sons), pp. 323-364.

Brock, G. J. R., Anderson, N. H., and Monckton, D. G. (1999). Cis-acting modifiers of expanded CAG/CTG triplet repeat expandibility: associations with flanking GC content and CpG islands. Human Molecular Genetics *8*, 1061-1067.

Bronner, C. E., Baker, S. M., Morrison, P. T., Warren, G., Smith, L. G., Lescoe, M. K., Kane, M., Earabino, C., Lipford, J. and Lindblom, A. (1994). Mutation in the DNA mismatch repair gene homologue hMLH1 is associated with hereditary non-polyposis colon cancer. Nature *368*, 258-61.

Brook, J. D., McCurrach, M. E., Harley, H. G., Buckler, A. J., Church, D., Aburatani, H., Hunter, K., Stanton, V. P., Thirion, J.-P., Hudson, T., Sohn, R., Zemelman, B., Snell, R. G., Rundle, S. A., Crow, S., Davies, J., Shelbourne, P., Buxton, J., Jones, C., Juvonen, V., Johnson, K., Harper, P. S., Shaw, D. J., and Housman, D. E. (1992). Molecular basis of myotonic dystrophy: expansion of a trinucleotide (CTG) repeat at the 3' end of a transcript encoding a protein kinase family member. Cell *68*, 799-808.

Brunner, H. G., Bruggenwirth, H. T., Nillesen, W., Jansen, G., Hamel, C. J., Hoppe, R. L. E., de Die, C. E. M., Howeler, C. J., van Oost, B. A., Wieringa, B., Ropers, H. H., and Smeets, H. J. M. (1993). Influence of sex of the transmitting parent as well as of parental allele size on the CTG expansion in myotonic dystrophy (DM). American Journal of Human Genetics *53*, 1016-1023.

Chan, T. L., Yuen, S. T., Chung, L. P., Ho, J. W., Kwan, K. Y., Chan, A. S., Ho, J. C., Leung, S. Y., and Wyllie, A. H. (1999). Frequent microsatellite instability and mismatch repair gene mutations in young Chinese patients with colorectal cancer. Journal of the National Cancer Institute *91*, 1221-6.

Chandy, K. G., Fantino, E., Wittekind, O., Kalman, K., Tong, L. L., Ho, T. H., Gutman, G. A., Crocq, M. A., Ganguli, R., Nimgaonkar, V., Morris-Rosendahl, D. J., and Gargus, J. J. (1998). Isolation of a novel potassium channel gene hSKCa3 containing a polymorphic CAG repeat: a candidate for schizophrenia and bipolar disorder? Molecular Psychiatry *3*, 32-37.

Cummings, C. J., and Zoghbi, H. Y. (2000). Fourteen and counting: unraveling trinucleotide repeat diseases. Human Molecular Genetics *9*, 909-916.

David, G., Abbas, N., Stevanin, G., Dürr, A., Yvert, G., Cancel, G., Weber, C., Imbert, G., Saudou, F., Antoniou, E., Drabkin, H., Gemmill, R., Giunti, P., Benomar, A., Wood, N., Ruberg, M., Agid, Y., Mandel, J. L., and Brice, A. (1997). Cloning of the SCA7 gene reveals a highly unstable CAG repeat expansion. Nature Genetics *17*, 65-70.

Davis, B. M., McCurrach, M. E., Taneja, K. L., Singer, R. H., and Housman, D. E. (1997). Expansion of a CUG trinucleotide repeat in the 3' untranslated region of myotonic dystrophy protein kinase transcripts results in nuclear retention of transcripts. Proceedings of the National Academy of Sciences USA *94*, 7388-7393.

Die Smulders, C. E. M., Smeets, H. J. M., Loots, W., Anten, H. B. M., Mirandolle, J. F., Geraedts, J. P. M., and Howeler, C. J. (1997). Paternal transmission of congenital myotonic dystrophy. Journal Of Medical Genetics *34*, 930-933.

Fortune, M. T., Vassilopoulos, C., Coolbaugh, M. I., Siciliano, M. J., and Monckton, D. G. (2000). Dramatic, expansion-biased, age-dependent, tissue-specific somatic mosaicism in a transgenic mouse model of tripler repeat instability. Human Molecular Genetics *9*, 439-445.

Freudenreich, C. H., Zakian, V. A., and Kantrow, S. M. (1998). Expansion and length dependent fragility if CTG repeats in yeast. Science *279*, 853-856.

Fu, Y.-H., Kuhl, D. P. A., Pizzuti, A., Pieretti, M., Sutcliffe, J. S., Richards, S., Verkerk, A. J. M. H., Holden, J. J. A., Fenwick, R. G., Jr., Warren, S. T., Oostra, B. A., Nelson, D. L., and Caskey, C. T. (1991). Variation of the CGG repeat at the fragile X site results in genetic instability: resolution of the Sherman paradox. Cell *67*, 1047-1058.

Fu, Y. H., Pizzuti, A., Fenwick, R. G., King, J., Rajnarayan, S., Dunne, P. W., Dubel, J., Nasser, G. A., Ashizawa, T., de Jong, P., Wieringa, B., Korneluk, R., Perryman, M. B., Epstein, H. F., and Caskey, C. T. (1992). An unstable triplet repeat in a gene related to myotonic muscular dystrophy. Science *255*, 1256-1258.

Gennarelli, M., Novelli, G., Andreasi Bassi, F., Baiget, M., and Dallapiccola, B. (1996). Prediction of myotonic dystrophy clinical severity based on number of intragenic CTG trinucleotide repeats. American Journal of Medical Genetics. *65*, 342-347.

George, A. L., Crackower, M. A., Abdall, J. A., Hudson, A. J., and Ebers, G. C. (1992). Molecular basis of Thomsen's disease (autosomal dominant myotonica congenita). Nature Genetics *3*, 305.

Gordenin, D. A., Kunkel, T. A., and Resnick, M. A. (1997). Repeat expansion--all in a flap? [news]. Nature Genetics *16*, 116-8.

Gotteman, I. I. (1991). Schizophrenia Genesis: The Origins Of Madness (New York: WH Freeman).

Gourdon, G., Radvanyi, F., Lia, A. S., Duros, C., Blanche, M., Abitbol, M., Junien, C., and Hofmann Radvanyi, H. (1997). Moderate intergenerational and somatic instability of a 55 CTG repeat in transgenic mice. Nature Genetics *15*, 190-192.

Gouw, L. G., Castañeda, M. A., McKenna, C. K., Digre, K. B., Pulst, S. M., Perlman, S., Lee, M. S., Gomez, C., Fischbeck, K., Gagnon, D., Storey, E., Bird, T., Jeri, F. R., and Ptácek, L. J. (1998). Analysis of the dynamic mutation in the SCA7 gene shows marked parental effects on CAG repeat transmission. Human Molecular Genetics *7*, 525-32.

Groenen, P., and Wieringa, B. (1998). Expanding complexity in myotonic dystrophy. BioEssays *20*, 901-912.

Guy, C. A., Bowen, T., Jones, I., McCandless, F., Owen, M. J., Craddock, N., and O'Donovan, M. C. (1999). CTG18.1 and ERDA-1 CAG/CTG repeat size in bipolar disorder. Neurobiological Disorders *6*, 302-307.

Hall, A. (1998). Rho GTPases and the actin cytoskeleton. Science *279*, 509-514.

Hamshere, M. G., Harley, H., Harper, P., Brook, J. D., and Brookfield, J. F. Y. (1999). Myotonic dystrophy: the correlation of (CTG) repeat length in leucocytes with age at onset is significant only for patients with small expansions. Journal of Medical Genetics, 59-61.

Hamshere, M. G., Newman, E. E., Alwazzan, M., Athwal, B. S., and Brook, J. D. (1997). Transcriptional abnormality in myotonic dystrophy affects *DMPK* but not neighboring genes. Proceedings Of the National Academy Of Sciences Of the United States Of America *94*, 7394-7399.

Harley, H. G., Rundle, S. A., MacMillan, J. C., Myring, J., Brook, J. D., Crow, S., Reardon, W., Fenton, I., Shaw, D. J., and Harper, P. S. (1993). Size of the unstable CTG repeat sequence in relation to phenotype and parental transmission in myotonic dystrophy. American Journal of Human Genetics *52*, 1164-1174.

Harper, P. S. (1989). Myotonic dystrophy, 2nd Edition (London: WB Saunders Co.).

Hochtrasser, R. A., Carver, T. E., Sowers, L. C., and Millar, D. P. (1994). Melting of a DNA helix terminus within the active site of a DNA polymerase. Biochemistry *33*.

Howeler, C. J., Busch, H. F. M., Geraedts, J. P. M., Neirmeijer, M. F., and Staal, A. (1989). Anticipation in myotonic dystrophy: fact or fiction? Brain *112*, 779-797.

Hsiao, K. M., Lin, H. M., Pan, H., Li, T. C., Chen, S. S., Jou, S. B., Chiu, Y. L., Wu, M. F., Lin, C. C., and Li, S. Y. (1999). Application of FTA sample collection and DNA purification system on the determination of CTG trinucleotide repeat size by PCR-based Southern blotting. Journal of Clinical Laboratory Analysis *13*, 188-193.

Ikeuchi, T., Sanpei, K., Takano, H., Sasaki, H., Tashiro, K., Cancel, G., Brice, A., Bird, T. D., Schellenberg, G. D., Pericak-Vance, M. A., Welsh-Bohmer, K. A., Clark, L. N., Wilhelmsen, K., and Tsuji, S. (1998). A novel long and unstable CAG/CTG trinucleotide repeat on chromosome 17q. Genomics *49*, 321-326.

Imbert, G., Kretz, C., Johnson, K., and Mandel, J.-L. (1993). Origin of the expansion mutation in myotonic dystrophy. Nature Genetics *4*, 72-76.

IMDC (2000). New nomenclature and DNA testing guidelines for myotonic dystrophy type 1 (DM1). Neurology *54*, 1218-1221.

Ishii, S., Nishio, T., Sunohara, N., Yoshihara, T., Takemura, K., Hikiji, K., Tsujino, S., and Sakuragawa, N. (1996). Small increase in triplet repeat length of cerebellum from patients with myotonic dystrophy. Human Genetics *98*, 138-40.

Ito, Y., Tanaka, F., Yamamoto, M., Doyu, M., Nagamatsu, M., Riku, S., Mitsuma, T., and Sobue, G. (1998). Somatic mosaicism of the expanded CAG trinucleotide repeat in mRNAs for the responsible gene of Machado-Joseph disease (MJD), dentatorubral-pallidoluysian atrophy (DRPLA) and spinal and bulbar muscular atrophy (SBMA). Neurochemical Research *23*, 25-32.

Jansen, G., Bachner, D., Coerwinkel, M., Wormskamp, N., Hameister, H., and Wieringa, B. (1995). Structural organization and developmental expression pattern of the mouse WD-repeat gene DMR-N9 immediately upstream of the myotonic dystrophy locus. Human Molecular Genetics *4*, 843-52.

Jansen, G., Groenen, P. J. T. A., Bächner, D., Jap, P. H. K., Coerwinkel, M., Oerlemans, F., van den Broek, W., Gohlsch, B., Pette, D., Plomp, J. J., Molenaar, P. C., Nederhoff, M. G. J., van Echteld, C. J. A., Dekker, M., Berns, A., Hameister, H., and Wieringa, B. (1996). Abnormal myotonic dystrophy protein kinase levels produce only mild myopathy in mice. Nature Genetics *13*, 316-324.

Jansen, G., Mahadevan, M., Amemiya, C., Wormskamp, N., Segers, B., Hendriks, W., O'Hoy, K., Baird, S., Sabourin, L., Lennon, G., Jap, P. L., Iles, D., Coerwinkel, M., Hofker, M., Carrano, A. V., de Jong, P. J., Korneluk, R. G., and Wieringa, B. (1992). Characterization of the myotonic dystrophy region predicts multiple protein isoform-encoding mRNAs. Nature Genetics *1*, 261-266.

Jansen, G., Willems, P., Coerwinkel, M., Nillesen, W., Smeets, H., Vits, L., Howeler, C., Brunner, H., and Wieringa, B. (1994). Gonosomal mosaicism in myotonic dystrophy patients: Involvement of mitotic events in (CTG)n variation and selection against extreme expansion in sperm. American Journal of Human Genetics *54*, 575-585.

Jaspert, A., Fahsold, R., Grehl, H., and Claus, D. (1995). Myotonic dystrophy: correlation of clinical symptoms with the size of the CTG trinucleotide repeat. Journal of Neurology *242*, 99-104.

Jeffreys, A. J., Tamaki, K., MacLeod, A., Monckton, D. G., Neil, D. L., and Armour, J. A. L. (1994). Complex gene conversion events in germline mutation at human minisatellites. Nature Genetics *6*, 136-145.

Kawakami, K., Ohto, H., Ikeda, K., and Roeder, R. G. (1996). Structure, function and expression of a murine homeobox protein AREC3, a homologue of Drosophila sine oculis gene product, and implication in development. Nucleic Acids Research. *24*, 303-10.

Kawakami, K., Ohto, H., Takizawa, T., and Saito, T. (1996). Identification and expression of six family genes in mouse retina. FEBS Letters *393*, 259-63.

Kennedy, L., and Shelbourne, P. F. (2000). Dramatic mutation instability in HD mouse striatum: does polyglutamine load contribute to cell-specific vulnerability in Huntington's disease? Human Molecular Genetics *9*, 2539-44.

Klesert, T. R., Otten, A. D., Bird, T. D., and Tapscott, S. J. (1997). Trinucleotide repeat expansion at the myotonic dystrophy locus reduces expression of DMAHP. Nature Genetics *16*, 402-406.

Koefoed, P., Hasholt, L., Fenger, K., Nielsen, J. E., Eiberg, H., Buschard, K., and Sorensen, S. A. (1998). Mitotic and meiotic instability of the CAG trinucleotide repeat in spinocerebelllar ataxia type 1. Human Genetics, 564-569.

Krahe, R., Ashizawa, T., Abbruzzese, C., Roeder, E., Carango, P., Giacanelli, M., Funanage, V. L., and Siciliano, M. J. (1995). Effect of myotonic dystrophy trinucleotide repeat expansion on *DMPK* transcription and processing. Genomics *28*, 1-14.

Kunst, C., and Warren, S. T. (1994). Cryptic and polar variation of the fragile X repeat could result in predisposing normal alleles. Cell *77*, 853-861.

La Spada, A. R., Wilson, E. M., Lubahn, D. B., Harding, A. E., and Fischbeck, K. H. (1991). Androgen receptor gene mutations in X-linked spinal and bulbar muscular atrophy. Nature *352*, 77-79.

Leach, F. S., Nicolaides, N. C., Papadopoulos, N., Liu, B., Jen, J., Parsons, R., Peltomaki, P., Sistonen, P., Aaltonen, L. A., Nystrom-Lahti, M., and *et al.* (1993). Mutations of a mutS homolog in hereditary nonpolyposis colorectal cancer. Cell *75*, 1215-25.

Leeflang, E. P., Tavare, S., Marjoram, P., Neal, C. O. S., Smidhi, J., MacDonald, M. E., de Young, M., Wexler, N. S., Gusella, J. F., and Arnheim, N. (1999). Analysis of germline mutation spectra at the Huntington's disease locus supports a mitotic mutation mechanism. Human Molecular Genetics *8*, 173-183.

Leeflang, E. P., Zhang, L., Tavaré, S., Hubert, R., Srinidhi, J., MacDonald, M. E., Myers, R. H., de Young, M., Wexler, N. S., Gusella, J. F., and Arnheim, N. (1995). Single sperm analysis of the trinucleotide repeats in the Huntington's disease gene: quantification of the mutation frequency and spectrum. Human Molecular Genetics *4*, 1519-1526.

Liu, B., Parsons, R., Papadopoulos, N., Nicolaides, N. C., Lynch, H. T., Watson, P., Jass, J. R., Dunlop, M., Wyllie, A., Peltomaki, P., de la Chapelle, A., Hamilton, S. R., Vogelstein, B., and Kinzler, K. W. (1996). Analysis of mismatch repair genes in hereditary non-polyposis colorectal cancer patients. Nature Medicine *2*, 169-74.

Mahadevan, M., Tsilfidis, C., Sabourin, L., Shutler, G., Amemiya, C., Jansen, G., Neville, C., Narang, M., Barcelo, J., O'Hoy, K., Leblond, S., Earle-Macdonald, J., de Jong, P. J., Wieringa, B., and Korneluk, R. G. (1992). Myotonic dystrophy mutation: an unstable CTG repeat in the 3' untranslated region of the gene. Science *255*, 1253-1255.

Manley, K., Shirley, T. L., Flaherty, L., and Messer, A. (1999). Msh2 deficiency prevents in vivo somatic instability of the CAG repeat in Huntington disease transgenic mice. Nature Genetics *23*, 471-3.

Martorell, L., Monckton, D. G., Gamez, J., Johnson, K. J., Gich, I., Lopez de Munain, A., and Baiget, M. (1998). Progression of somatic CTG repeat length heterogeneity in the blood cells of myotonic dystrophy patients. Human Molecular Genetics *7*, 307-312.

McInnis, M. G., McMahon, F. J., Chase, G., Simpson, S. G., Ross, C. A., and DePaulo, J. R. J. (1993). Anticipation in bipolar affective disorder. American Journal of Human Genetics, 385-390.

McMurray, C. T. (1995). Mechanisms of DNA expansion. Chromosoma *104*, 2-13.

Melacini, P., Villanova, C., Menegazzo, E., Novelli, G., Danieli, G., Rizzoli, G., Fasoli, G., Angelini, C., Buja, G., Miorelli, M., and *et al.* (1995). Correlation between cardiac involvement and CTG trinucleotide repeat length in myotonic dystrophy. Journal of the American College of Cardiology *25*, 239-45.

Miret, J. J., Pessoa-Brandao, L., and Lahue, R. S. (1997). Instability of CAG and CTG trinucleotide repeats in Saccharomyces cerevisiae. Molecular Cell Biology *17*, 3382-7.

Monckton, D. G., Coolbaugh, M. I., Ashizawa, K., Siciliano, M. J., and Caskey, C. T. (1997). Hypermutable myotonic dystrophy CTG repeats in transgenic mice. Nature Genetics *15*, 193-196.

Monckton, D. G., Wong, L.-J. C., Ashizawa, T., and Caskey, C. T. (1995). Somatic mosaicism, germline expansions, germline reversions and intergenerational reductions in myotonic dystrophy males: small pool PCR analyses. Human Molecular Genetics *4*, 1-8.

Mornet, E., Chateau, C., Hirst, M. C., Thepot, F., Taillandier, A., Cibois, O., and Serre, J. (1996). Analysis of germline variation at the FMR1 CGG repeat shows variation in the normal-premutated borderliine range. Human Molecular Genetics *5*, 821-825.

Morris, A. G., Gaitonde, E., McKenna, P. J., Mollon, J. D., and Hunt, D. M. (1995). CAG repeat expansions and schizophrenia: association with disease in females and with early age-at-onset. Human Molecular Genetics *4*, 1957-1961.

Nakamoto, M., Takebayashi, H., Kawaguchi, Y., Narumiya, S., Taniwaki, M., Nakamura, Y., Ishikawa, Y., Akiguchi, I., Kimura, J., and Kakizuka, A. (1997). A CAG/CTG expansion in the normal population. Nature Genetics *17*, 385-386.

Nicolaides, N. C., Papadopoulos, N., Liu, B., Wei, Y. F., Carter, K. C., Ruben, S. M., Rosen, C. A., Haseltine, W. A., Fleischmann, R. D., Fraser, C. M., and *et al.* (1994). Mutations of two PMS homologues in hereditary nonpolyposis colon cancer. Nature *371*, 75-80.

Paillard, L., Omilli, F., Legagneux, V., Bassez, T., Maniey, D., and Osborne, H. B. (1998). EDEN and EDEN-BP, a cis element and an associated factor that mediate sequence-specific mRNA deadenylation in Xenopus embryos. EMBO Journal *17*, 278-87.

Pearson, C. E., Ewel, A., Acharya, S., Fishel, R. A., and Sinden, R. R. (1997). Human MSH2 binds to trinucleotide repeat DNA structures associated with neurodegenerative diseases. Human Molecular Genetics *6*, 1117-1123.

Pearson, C. E., and Sinden, R. R. (1996). Alternative structures in duplex DNA formed within the trinucleotide repeats of the myotonic dystrophy and fragile X loci. Biochemistry *35*, 5041-5053.

Penrose, L. S. (1948). The problem of anticipation in pedigrees of dystrophia myotonica. Annals of Eugenics *14*, 125-132.

Reddy, S., Smith, D. B. J., Rich, M. M., Leferovich, J. M., Reilly, P., Davis, B. M., Tran, K., Rayburn, H., Bronson, R., Cros, D., Balice-Gordon, R. J., and Housman, D. (1996). Mice lacking the myotonic dystrophy protein kinase develop a late onset progressive myopathy. Nature Genetics *13*, 325-335.

Redman, J. B., Fenwick, R. G., Fu, Y.-H., Pizzuti, A., and Caskey, C. T. (1993). Relationship between parental trinucleotide GCT repeat length and severity of myotonic dystrophy in offspring. Journal of the American Medical Association *269*, 1960-1965.

Roberts, R., Timchenko, N. A., Miller, J. W., Reddy, S., Caskey, C. T., Swanson, M. S., and Timchenko, L. T. (1997). Altered phosphorylation and intracellular distribution of a $(CUG)_{(n)}$ triplet repeat RNA binding protein in patients with myotonic dystrophy and in myotonin protein kinase knockout mice. Proceedings Of the National Academy Of Sciences Of the United States Of America *94*, 13221-13226.

Sarnat, H. B., and Silbert, S. W. (1975). Maturational arrest of fetal muscle in neonatal myotonic dystrophy. Archives of Neurol. *33*, 466.

Schalling, M., Hudson, T. J., Buetow, K. H., and Housman, D. E. (1993). Direct detection of novel expanded trinucleotide repeats in the human genome. Nature Genetics *4*, 135-9.

Schlotterer, C., and Tautz, D. (1992). Slippage synthesis of simple sequence DNA. Nucleic Acids Research *20*, 211-215.

Seznec, H., Lia-Baldini, A. S., Duros, C., Fouquet, C., Lacroix, C., Hofmann-Radvanyi, H., Junien, C., and Gourdon, G. (2000). Transgenic mice carrying large human genomic sequences with expanded CTG repeat mimic closely the DM CTG repeat intergenerational and somatic instability. Human Molecular Genetics *9*, 1185-94.

Shaw, D. J., McCurrach, M., Rundle, S. A., Harley, H. G., Crow, S. R., Sohn, R., Thirion, J. P., Hamshere, M. G., Buckler, A. J., Harper, P. S., Housman, D. E., and Brook, J. D. (1993). Genomic organization and transcriptional units at the myotonic dystrophy locus. Genomics *18*, 673-679.

Sinden, R. R., and Wells, R. D. (1992). DNA structure, mutations, and human genetic disease. Current Opinion in Biotechnology *3*, 612-622.

Smith, G. P. (1976). Evolution of repeated DNA sequences by unequal crossover. Science *191*.

Takiyama, Y., Sakoe, K., Soutome, M., Namekawa, M., Ogawa, T., Nakano, I., Igarashi, S., Oyake, M., Tanaka, H., Tsuji, S., and Nishizawa, M. (1997). Single sperm analysis of the CAG repeats in the gene for Machado-Joseph disease (MJD1): evidence for non-Mendelian transmission of the MJD1 gene and for the effect of the intragenic CGG/GGG polymorphism on the intergenerational instability. Human Molecular Genetics *6*, 1063-1068.

Taneja, K. L., McCurrach, M., Schalling, M., Housman, D., and Singer, R. H. (1995). Foci of trinucleotide repeat transcripts in nuclei of myotonic dystrophy cells and tissues. Journal Of Cell Biology *128*, 995-1002.

Telenius, H., Kremer, B., Goldberg, Y. P., Theilmann, J., Andrew, S. E., Zeisler, J., Adam, S., Greenberg, C., Ives, E. J., Clarke, L. A., and Hayden, M. R. (1994). Somatic and gonadal mosiacism of the Huntington disease gene CAG repeat in brain and sperm. Nature Genetics *6*, 409-414.

Timchenko, L., Timchenko, N. A., Caskey, C. T., and Roberts, R. (1996). Novel proteins with binding specificity for DNA CTG repeats and RNA CUG repeats: implications for myotonic dystrophy. Human Molecular Genetics, 115-121.

Vasen, H. F., Watson, P., Mecklin, J. P., and Lynch, H. T. (1999). New clinical criteria for hereditary non-polyposis colorectal cancer(HNPCC, Lynch syndrome) proposed by the international Collaborative group on HNPCC. Gastroenterology *116*, 1453-1456.

Verheyen, G. R., Del-Favero, J., Mendlewicz, J., Lindblad, K., Van Zand, K., Aalbregste, M., Schalling, M., Souery, D., and Van Broeckhoven, C. (1999). Molecular interpretation of expanded RED products in bipolar disorder by CAG/CTG repeats located at chromosomes 17q and 18q. Neurobiological disorders *6*, 424-432.

Verkerk, A. J. M. H., Pieretti, M., Sutcliffe, J. S., Fu, Y.-H., Kuhl, D. P. A., Pizzuti, A., Reiner, O., Richards, S., Victoria, M. F., Zhang, F., Eusen, B. E., van Ommen, G.-J. B., Blonden, L. A. J., Riggins, G. J., Chastain, J. L., Kunst, C. B., Galjaard, H., Caskey, C. T., Nelson, D. L., Oostra, B. A., and Warren, S. T. (1991). Identification of a gene (FMR-1) containing a CGG repeat coincident with a breakpoint cluster region exhibiting length variation in fragile X syndrome. Cell *65*, 905-914.

Wang, Y. H., Amirhaeri, S., Kang, S., Wells, R. D., and Griffith, J. D. (1994). Preferential nucleosome assembly at DNA triplet repeats from the myotonic dystrophy gene. Science *265*, 669-671.

Wells, R. D. (1996). Molecular basis of genetic instability of triplet repeats. Journal Of Biological Chemistry *271*, 2875-2878.

Wierdl, M., Dominska, M., and Petes, T. D. (1997). Microsatellite instability in yeast: dependence on the length of the microsatellite. Genetics *146*, 769-779.

Wijnen, J., Khan, P. M., Vasen, H., van der Klift, H., Mulder, A., van Leeuwen-Cornelisse, I., Bakker, B., Losekoot, M., Moller, P., and Fodde, R. (1997). Hereditary nonpolyposis colorectal cancer families not complying with the Amsterdam criteria show extremely low frequency of mismatch-repair-gene mutations. American Journal of Human Genetics *61*, 329-35.

Winchester, C. L., Ferrier, R. K., Sermoni, A., Clark, B. J., and Johnson, K. J. (1999). Characterization of the expression of DMPK and SIX5 in the human eye and implications for pathogenesis in myotonic dystrophy. Human Molecular Genetics *8*, 481-92.

Wiper, D. W., Zanotti, K. M., Kennedy, A. W., Belinson, J. L., and Casey, G. (1998). Analysis of allelic imbalance on chromosome 17p13 in stage I and stage II epithelial ovarian cancers. Gynecological Oncology *71*, 77-82.

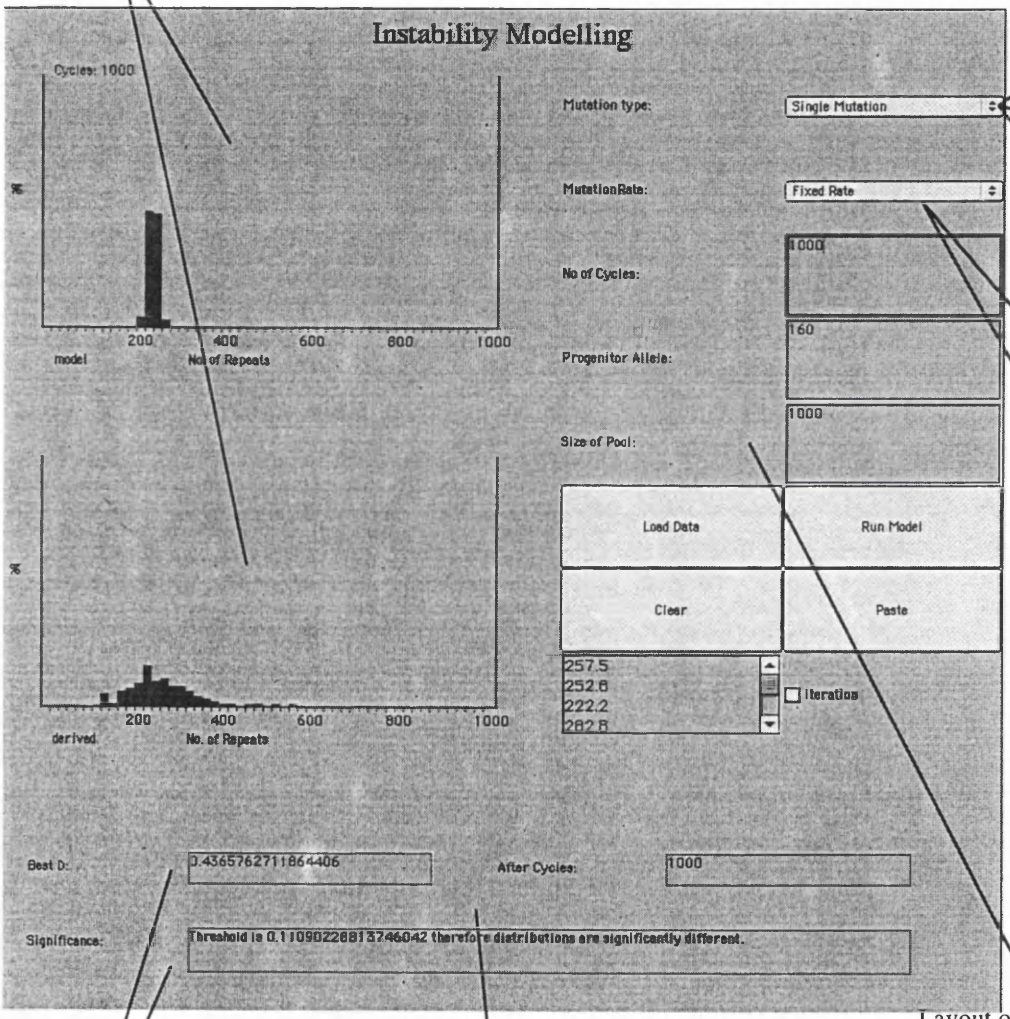
Wong, L.-J. C., Ashizawa, T., Monckton, D. G., Caskey, C. T., and Richards, C. S. (1995). Somatic heterogeneity of the CTG repeat in myotonic dystrophy is age and size dependent. American Journal of Human Genetics *56*, 114-122.

Zerylnick, C., Torroni, A., Sherman, S. L., and Warren, S. T. (1995). Normal variation at the myotonic dystrophy locus in global human populations. American Journal of Human Genetics *56*, 123-130.

Zhang, L., Leeflang, E. P., Yu, J., and Arnheim, N. (1994). Studying human mutations by sperm typing: instability of CAG trinucleotide repeats in the human androgen receptor gene. Nature Genetics *7*, 531-535.

Appendix

Presented here is the source code for the simulation software. The source code consists of small pieces of code called classes, each of which perform a specific task. Classes are arranged in a hierarchy which is the overall program. The function that each class performs is depicted here:



Mutation type: Single Mutation

MutationRate: Fixed Rate

No of Cycles: 1000

Progenitor Allele: 160

Size of Pool: 1000

257.5
252.8
222.2
282.8

Best D: 0.4365762711864406

After Cycles: 1000

Significance: Threshold is 0.11090228813746042 therefore distributions are significantly different.

Visual display of allele distributions controlled by class GraphCanvas

Class MutationFrame

Class RandomFrame

Class SpectrumFrame

Class FixedRateFrame

Class RateFrame

Layout of mutation parameters and input data functions controlled by class ModelControlPanel.

Layout of results controlled by class ModelResultsPanel.

All calculations of statistics and generation of model data controlled by class Model.

Appendix Contents:

Class Model:

```
/**
 * Class :   Model
 *
 * @author   Grant Hogg (ghogg@molgen.gla.ac.uk)
 *           David Jack (davidj@dcs.gla.ac.uk)
 * @version  0.3
 */






import java.applet.Applet;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.Panel;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.Transferable;
import java.awt.datatransfer.DataFlavor;
import java.awt.Choice;
import java.awt.event.ActionEvent;
import java.awt.event.ItemEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemListener;
import java.io.InputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;


import java.net.URL;
import java.net.URLConnection;

import java.util.Date;
import java.util.Enumeration;
import java.util.Random;
import java.util.Vector;
import java.util.StringTokenizer;



/**
 * Applet sub-class.  Performs a CTG repeat instability simulation, taking
 * patient data and parameter settings and returning results of statistical
 * analysis.
 */

public class Model2  extends Applet  implements ActionListener,ItemListener, ModelConstants  {



  /**
   * Applet control panel and results panel.
   */

  private ModelControlPanel controlPanel;
  private ModelResultsPanel resultsPanel;
  Clipboard clipboard = null;
```

```
/**
 * Represents the model and derived data.
 */

private Mod model = new Mod();
private Der derived = new Der();



/**
 * Distribution histograms for the model and derived data.
 */

private GraphCanvas modelGraph = new GraphCanvas("model");
private GraphCanvas derivedGraph = new GraphCanvas("derived");



/**
 * Initialised parameters.
 */

private Color myColor = new Color(200, 150, 250);
private boolean isDerived = false;
private int cycleCounter = 0;



/**
 * Variables required for statistical analysis.
 */

private double threshold;
private double D;
private double bestD;
private int bestCycles;
private double[] derivedFreq;
private double[] modelFreq;
private double[] absoluteDiff;


/**
 * Variables required for iteration
 */

        private int overallBestCycles;
        private double overallBestD;
        private int bestSize;
        private int bestRate;
        private int bestProgenitor;
        private int iterationCounter;



/**
 * URL of the servlet which returns the data.
 */

private final String servletString = "http://mars.dcs.gla.ac.uk:8001/servlet/geneprobe/RepeatInstabilityServletV0_3";



/**
 * Applet's init() method.  Constructs the Applet interface.
 */

public void init() {
```

```
// Sets up a BorderLayout with no gaps between components.
// -------------------------------------------------------------

setLayout(new BorderLayout());


// Constructs the side control panel.
// -------------------------------------------------------------

controlPanel = new ModelControlPanel();
controlPanel.setBackground(myColor);
controlPanel.addActionListener(this);
controlPanel.addItemListener(this);
add(controlPanel, "East");


// Construct the bottom results panel.
// -------------------------------------------------------------

resultsPanel = new ModelResultsPanel();
resultsPanel.setBackground(myColor);
add(resultsPanel, "South");


// Construct the center graph panel.
// -------------------------------------------------------------

Panel graphPanel = new Panel ();

graphPanel.setBackground(myColor);
graphPanel.setLayout(new GridLayout(2, 1));


// Set the size of the two GraphCanvas components, and add them.
// -------------------------------------------------------------

modelGraph.setSize(410, 260);
graphPanel.add(modelGraph);

derivedGraph.setSize(410, 260);
graphPanel.add(derivedGraph);


// Construct the top panel, for displaying the Applet title.
// -------------------------------------------------------------

Panel titlePanel = new Panel();

titlePanel.setBackground(myColor);
titlePanel.setFont(new Font("Serif", Font.PLAIN, 24));
titlePanel.add(new Label("Instability Modelling"));


// Add the four panels to the Applet.
// -------------------------------------------------------------

add(graphPanel,"Center");
add(titlePanel,"North");

}
```

```
/**
  * ActionListener callback method.
  * Called when any of the buttons on the Applet are selected.
  *
  * @param e the ActionEvent generated.
  */

 public void actionPerformed(ActionEvent e) {



    // Get the action command from the ActionEvent.
    // -----------------------------------------------------------------

    String command = e.getActionCommand();



    // If the 'Load Data' button was selected: load the appropriate data.
    // If the 'Run Model' button was selected: run the model.
    // -----------------------------------------------------------------
    if (clipboard == null)
                                   clipboard = getToolkit().getSystemClipboard();


    if (command.equals("Load Data")) {
       resultsPanel.clearResults();

      loadData();

    } else if (command.equals("Run Model")) {
       resultsPanel.clearResults();

       if ((controlPanel.getCheckboxState() == true) &&
                                   (controlPanel.selectedItem.equals("Single Mutation"))) {
       iterateModel();
       } else
       {
       runModel();
       }


    }       else if (command.equals("Clear")) {
                                   controlPanel.clearArea();

                      }            else if (command.equals("Paste")) {
                                   Transferable clipData = clipboard.getContents(this);
                                   String s;
                                   try {
                                              s = (String) (clipData.getTransferData(
                                                        DataFlavor.stringFlavor));
                                   } catch (Exception ee) {
                                              s = ee.toString();
                                   }// end try catch

                                   controlPanel.setArea(s);

                      }
  } // end method

public void itemStateChanged( ItemEvent e)
           {
                      // only one menu so all itemEvents will come from this
                      Choice choice = (Choice) e.getItemSelectable();
                      if (choice == controlPanel.rateOptions)
                      {
                      controlPanel.selectedRate = e.getItem().toString();
		if (controlPanel.selectedRate.equals("Fixed Rate")){
			// set spectrum frame to be visible
			controlPanel.fixedRateFrame.setVisible(true);


			} else if (controlPanel.selectedRate.equals("Linear")){
			// set spectrum frame to be visible
			controlPanel.rateFrame.setVisible(true);


			}
		}


		if (choice == controlPanel.mutationOptions)
		{
		controlPanel.selectedItem = e.getItem().toString();

		if (controlPanel.selectedItem.equals("Select Spectrum")){
			// set spectrum frame to be visible
			controlPanel.specFrame.setVisible(true);


			} else if (controlPanel.selectedItem.equals("Single Mutation")){
			// set spectrum frame to be visible
			controlPanel.mutFrame.setVisible(true);


			}

		else if (controlPanel.selectedItem.equals("Random Mutation")) {
			// set spectrum frame to be visible
			controlPanel.ranFrame.setVisible(true);


			}
		}
	} // end method


  /**
   * Sets up the connection to the servlet. Sends a dummy Vector to the
   * servlet (sendData) in order to work-around a possible bug(?) to do
   * with the ObjectInputStream. Then posts a request to the servlet
   * which returns the data, read in from the file, as a Vector. This
   * derived data is then displayed on the lower histogram on the applet.
   */

  protected void loadData() {

  // I have removed David's stuff here for the time being

/*   Vector data = null;



    // Try ... getting the derived data from the servlet and displaying it.
    // -----------------------------------------------------------------

    try {


      // Open the connection to the servlet URL.
      // -----------------------------------------------------------

      URL servletURL = new URL(servletString);
      URLConnection servletConnection = servletURL.openConnection();
      // Set the servlet connection parameters.
      // -----------------------------------------------------------

      servletConnection.setDoInput(true);
      servletConnection.setDoOutput(true);
      servletConnection.setUseCaches(false);
      servletConnection.setRequestProperty("Content-Type",
            "application/octet-stream");



      // Send data to the servlet: << presently sends dummy data >>
      // Get the derived data returned from the servlet.
      // -----------------------------------------------------------

      sendData(servletConnection);
      data = getData(servletConnection);



      // Display the derived data in the distribution histogram.
      // -----------------------------------------------------------

      displayDerivedData(data);


    } catch(IOException e) {
      System.err.println("<<<<<<"+e.getMessage()+">>>>>>");
    }
*/

  // this is my simple alternative
          isDerived = true;
                                String s = controlPanel.getInfo();
                                StringTokenizer t = new StringTokenizer(s);
                                derived.alleles = new int[t.countTokens()];
                                derived.size = t.countTokens();

                                int i = 0;
                                int value;
                                Double tempValue;
                                // get numbers from our derived string
                                while(t.hasMoreTokens()){
                                        try{
                                                tempValue = (Double.valueOf(t.nextToken()));
                                                value = (int)tempValue.doubleValue();
                                                } catch (Exception ee){
                                                isDerived = false;
                                                value = 0;
                                                // need to notify user
                                                } // end try catch
                                        derived.alleles[i] = value;
                                        i++;
                                } // end while

                                // draw graph for user
                                Graphics g = derivedGraph.getGraphics();
                                g.setColor(myColor);
                                g.fillRect(31,1,357,209);
                                g.setColor(Color.darkGray);
                                derived.draw1(g);



  }



  /**
```

```
            objectInputStream = new ObjectInputStream(inputStream);
            vect = (Vector)objectInputStream.readObject();
            objectInputStream.close();


        } else {
            throw new IOException("Null InputStream from URLConnection:<"+
                conn.toString()+">");
        }



    // Catches exception if the incoming data is not a serialized Vector.
    // ------------------------------------------------------------

    } catch(ClassNotFoundException e) {
        System.err.println("ClassNotFoundException: "+e.getMessage());
        e.printStackTrace();
    }



    // Check the incoming vector before returning it.
    // NB: Not sure whether this is really necessary, since a null Vector
    //     may be caught by the ClassNotFoundException.
    // ------------------------------------------------------------

    if (vect != null)
        return vect;
    else
        return new Vector();
}




/**
 * Displays the derived data in the lower applet histogram.
 *
 * @param Vector the derived data Vector.
 */

private void displayDerivedData(Vector data) {


    // Set isDerived flag, and initialise other parameters.
    // ------------------------------------------------------------

    isDerived = true;
    int index = 0;
    Double tmp;
    int value;



    // Construct a array of integers from the Vector of Strings.
    // ------------------------------------------------------------

    derived.alleles = new int[data.size()];
    derived.size = data.size();



    // For each String in the data Vector ....
    //    Extract the value as an integer and add it to the array.
    // ------------------------------------------------------------

    Enumeration derivedElements = data.elements();
    while(derivedElements.hasMoreElements()) {
        String str = (String)derivedElements.nextElement();

        try {
           tmp = Double.valueOf(str);
           value = tmp.intValue();

        } catch(Exception ex) {
           isDerived = false;
           value = 0;
           System.err.println(ex.getMessage());
        }

        derived.alleles[index++] = value;
     }



     // Draw the derived data set as a graph, using the derived
     // GraphCanvas's Graphics object. (First clear the graph area).
     // -----------------------------------------------------------

     Graphics g = derivedGraph.getGraphics();

     g.setColor(myColor);
     g.fillRect(31, 1, 357, 209);
     g.setColor(Color.darkGray);

     derived.draw1(g);
  }




  /**
   * Run the model using the parameters entered into the applet.
   * Calculates the model array and displays it in the upper histogram.
   * Calculates the statistical results from the model and derived data
   * and displays them in the lower results panel.
   */

  protected void runModel() {


     // Reset the cycle counter.
     // ----------------------------------------------------------------

     cycleCounter = 0;



     // Read the int values from the control panel textfields.
     // ----------------------------------------------------------------

     int mutationRate = 10;
     int cycles = controlPanel.getCycles();
     int progenitor = controlPanel.getProgenitor();
     int poolSize = controlPanel.getPoolSize();
                        int mutationSize = 1;
                        int[] randomRange = new int[2];
                        // get mutationSize value

                         if (controlPanel.selectedItem.equals("Single Mutation")){
                                   mutationSize = controlPanel.mutFrame.getMutationSize();

                                   }

                        else if (controlPanel.selectedItem.equals("Random Mutation")) {
                                  randomRange = controlPanel.ranFrame.getMutationSizeRange();
                                  }
```

```
                // get mutationRate value
                if (controlPanel.selectedRate.equals("Fixed Rate")){
                        mutationRate = controlPanel.fixedRateFrame.getMutationRate();

                        }
                else {
                // will be using linear relationship information contained in rateFrame
                }


// Use the size of the pool to set up an empty integer array for
// the model data.
// ---------------------------------------------------------------

model.alleles = new int[poolSize];
model.size = poolSize;



// Calculate the threshold<?> using the sizes of the two data sets.
// ---------------------------------------------------------------

if (isDerived == true)  {

   int N1 = model.size;
   int N2 = derived.size;

   threshold = (Math.sqrt((double)(N1 +N2)/(N1*N2)))*1.36;
}



// For each element in the model data array, fill with the value
// given for the progenitor allele<?>.
// ---------------------------------------------------------------

for (int i=0;  i<model.size;  i++)  {
   model.alleles[i] = progenitor;
}



// Get the model GraphCanvas's Graphics object.
// ---------------------------------------------------------------

Graphics g = modelGraph.getGraphics();



// Process the model array for the set number of cycles ....
// ---------------------------------------------------------------

for (int i=1;  i<=cycles;  i++)  {


   // Increment the cycle counter.
   // ------------------------------------------------------------

   cycleCounter++;



   // mutate the model array adding the correct mutation size
   // ------------------------------------------------------------
                        if (controlPanel.selectedItem.equals("Select Spectrum")){
                                // mutate accordingly
                                model.alleles = spectrumMutate(model.alleles,model.size,mutationRate

     ,controlPanel.specFrame.spectrumArray,controlPanel.specFrame.intervalno);
```

```
                                             } else if (controlPanel.selectedItem.equals("Single Mutation")){
                                             // need to check whether linear relationship selected
                                                         if (controlPanel.selectedRate.equals("Fixed Rate"))
                                                         {
                                              model.alleles = mutate(model.alleles, model.size, mutationRate,
                                        mutationSize);
            }

                        if (controlPanel.selectedRate.equals("Linear"))
                                                         {
                                                          model.alleles = linearMutate(model.alleles, model.size,
controlPanel.rateFrame.m

           ,controlPanel.rateFrame.diseaseThreshold,mutationSize);
                                                         }

                                             } else if (controlPanel.selectedItem.equals("Random Mutation")) {
                                                         if (controlPanel.selectedRate.equals("Fixed Rate"))
                                                         {
                                                                    model.alleles = randomMutate(model.alleles, model.size,
mutationRate,
                   randomRange[0],randomRange[1]);
             }

                                                          if (controlPanel.selectedRate.equals("Linear"))
                                                          {
                                                          model.alleles = randomLinearMutate(model.alleles, model.size,
controlPanel.rateFrame.m

           ,controlPanel.rateFrame.diseaseThreshold,randomRange[0],randomRange[1]);
                                                          }
                                                          // probably a more efficient way of doing this
                                  }

       // Clear graph area, and draw the model array as a graph.
       // ----------------------------------------------------------

       g.setColor(myColor);
       g.fillRect(31, 1, 357, 209);
       g.setColor(Color.darkGray);

       model.draw1(g);



       // Calculate the statistical results.
       // --------------------------------------------------------------

       if (isDerived == true)  {

          modelFreq = new double[maxDist];
          derivedFreq = new double[maxDist];
          absoluteDiff = new double[maxDist];



          // Calculate arrays of model and derived frequencies.
          // ----------------------------------------------------------

          modelFreq = frequency(model.graphArray, model.size,
                          modelFreq);
          derivedFreq = frequency(derived.derivedArray, derived.size,
                          derivedFreq);



          // Calculate array of absolute differences.  Get first
          // absolute difference and set that as the maximum.
          // -- ---------------------------------------------------------
```

```
         int j = 0;
         absoluteDiff[j] = Math.abs(modelFreq[j]-derivedFreq[j]);
         D = absoluteDiff[j];


         // Check rest of the absolute differences to find the max.
         // ------------------------------------------------

         for (j=1; j<maxDist; j++) {

            absoluteDiff[j] = Math.abs(modelFreq[j]-derivedFreq[j]);

            if (absoluteDiff[j] > D)
               D = absoluteDiff[j];
         }


         // Now, we have a D<?> for this particular cycle. In order
         // to find the best cycle, set the first D to be the best
         // and then compare against subsequent values.
         // ------------------------------------------------

         if (cycleCounter == 1) {
            bestD = D;
            bestCycles = cycleCounter;

         } else {
            if (D < bestD) {
               bestD = D;
               bestCycles = cycleCounter;
            }
         }
      }
   }

   // Finished mutating, now want to output statistic results.
   // ------------------------------------------------------------

   if (isDerived == true) {


      resultsPanel.setBestD(String.valueOf(bestD));
      resultsPanel.setBestCycle(String.valueOf(bestCycles));



      // Output results of comparison between the best value of D
      // and the threshold<?>.
      // --------------------------------------------------------

      if (bestD < threshold)
         resultsPanel.setSignificance("Threshold is "+
            String.valueOf(threshold)+" therefore distributions "+
            "are not significantly different.");
      else
         resultsPanel.setSignificance("Threshold is "+
            String.valueOf(threshold)+" therefore distributions "+
            "are significantly different.");
   }
}


          protected void iterateModel() {

                        // method that will read in a range of parameters and try every combination
                        // noting the best one.
// need to define array iterateGraphArray.  This array contains the same
// information as graphArray but is defined separately here to complicate things.
// ---------------------------------------------------------------
int[] iterateGraphArray = new int[maxDist];

// can only do this method if we have derived information

if (isDerived == true) {

// reset iteration counter
// ---------------------------------------------------------------

iterationCounter =0;

// read the values from the control panel textfields
// ---------------------------------------------------------------
int[] mutationRateRange = new int[2];
int[] mutationSizeRange = new int[2];
int[] progenitorRange = new int[2];

 mutationRateRange = controlPanel.getMutationRateRange();
 mutationSizeRange = controlPanel.mutFrame.getMutationSizeRange();
 progenitorRange = controlPanel.getProgenitorRange();
int cycles = controlPanel.getCycles();
int poolSize = controlPanel.getPoolSize();


// initialise the best parameters to the ones we have.
// ---------------------------------------------------------------
 bestRate = mutationRateRange[0];
 bestSize = mutationSizeRange[0];
 bestProgenitor = progenitorRange[0];
overallBestCycles = 0;

// use the size of the pool to set up an empty integer array
// ---------------------------------------------------------------

model.alleles = new int[poolSize];
model.size = poolSize;

// initialise stuff for statistical comparison
// ---------------------------------------------------------------
int N1 = model.size;
int N2 = derived.size;
threshold = (Math.sqrt((double)(N1 + N2)/(N1*N2))) * 1.36;

// Get the model GraphCanvas Graphics object
// ---------------------------------------------------------------
Graphics g = modelGraph.getGraphics();



// lets begin iterations
// ---------------------------------------------------------------

for (int param1 = progenitorRange[0]; param1 <= progenitorRange[1]; param1++) {
		// param1 = current progenitor allele

for (int param2 = mutationSizeRange[0]; param2 <= mutationSizeRange[1]; param2++) {
		// param2 = current mutation size

for (int param3 = mutationRateRange[0]; param3 <= mutationRateRange[1]; param3++) {
		// param3 = current mutation rate

// now have all the parameters I need. Lets go.
// increment iteration Counter
// ---------------------------------------------------------------

iterationCounter ++;
g.setColor(myColor);
```

```
g.fillRect(31, 1, 357, 209);
g.setColor(Color.darkGray);
g.drawString("calculations: " + iterationCounter,100,150);

            // initialise model.alleles with progenitor
            // -----------------------------------------------------------

            for (int i = 0; i< model.size; i++) {
                    model.alleles[i] = param1;
            }



            // reset cycleCounter
            // -----------------------------------------------------------
            cycleCounter = 0;

            // go through array for each of the cycles
            // -----------------------------------------------------------

            for (int i=1; i<=cycles;i++) {

                    cycleCounter ++;
                    model.alleles = mutate(model.alleles, model.size,param3, param2);
                    iterateGraphArray = model.convert(model.alleles,model.size,iterateGraphArray);
                    // now have a mutated array. Lets do the stats.
                     modelFreq = new double[maxDist];
  derivedFreq = new double[maxDist];
  absoluteDiff = new double[maxDist];

     // Calculate arrays of model and derived frequencies.
    // -------------------------------------------------------

    modelFreq = frequency(iterateGraphArray, model.size,
                   modelFreq);
    derivedFreq = frequency(derived.derivedArray, derived.size,
                   derivedFreq);

                            // Calculate array of absolute differences. Get first
     // absolute difference and set that as the maximum.
   // --------------------------------------------------------

    int j = 0;
    absoluteDiff[j] = Math.abs(modelFreq[j]-derivedFreq[j]);
    D = absoluteDiff[j];



    // Check rest of the absolute differences to find the max.
    // -------------------------------------------------------

    for (j=1; j<maxDist; j++) {

       absoluteDiff[j] = Math.abs(modelFreq[j]-derivedFreq[j]);

       if (absoluteDiff[j] > D)
          D = absoluteDiff[j];
    }



    // Now, we have a D<?> for this particular cycle. In order
    // to find the best cycle, set the first D to be the best
    // and then compare against subsequent values.
    // -------------------------------------------------------

    if (cycleCounter == 1) {
       bestD = D;
       bestCycles = cycleCounter;
    }
    if (D < bestD)  {
        bestD = D;
        bestCycles = cycleCounter;
      }
                                  // end else
                        // once D is clearly not going to get any better then we may as well break from the loop
                        // -------------------------------------------------------
                        if (D> (bestD + 0.15)) break;

                        } // end mutate cycles

              // now have a BestD is it overall Best D?
              // ---------------------------------------------------



              if (iterationCounter == 1) {
                        overallBestD= bestD;
                        overallBestCycles = bestCycles;
                        // set other parameters
                        // -------------------------------------------------------
                        bestProgenitor = param1;
                        bestSize = param2;
                        bestRate = param3;
              } // end if

              else {
              // already have overall best D so lets compare best D to it.
              // ------------------------------------------------------

                        if (bestD < overallBestD) {
                                 overallBestD= bestD;
                                 overallBestCycles = bestCycles;
                                 // set other parameters
                                 // -------------------------------------------------------
                                 bestProgenitor = param1;
                                 bestSize = param2;
                                 bestRate = param3;
                        } // end if
              } // end else



              } // end for

              } // end for

              } // end for

              // output results.  need to update class Mod such that results are updated when
              // iteration is selected
              // ------------------------------------------------------
              // need to clear area for drawing
               g.setColor(myColor);
g.fillRect(31, 1, 357, 209);
g.setColor(Color.darkGray);

              model.draw1(g);

              // output stats stuff
              // ------------------------------------------------------
              resultsPanel.setBestD(String.valueOf(overallBestD));
              resultsPanel.setBestCycle(String.valueOf(overallBestCycles));

              // output results of comparison between the best value of D and the threshold

              if (overallBestD < threshold)
                          resultsPanel.setSignificance("Threshold is "+
      String.valueOf(threshold)+" therefore distributions "+
          "are not significantly different.");
    else
        resultsPanel.setSignificance("Threshold is "+
          String.valueOf(threshold)+" therefore distributions "+
          "are significantly different.");


  } // end if at the beginning

 } // end method




/**
  * Utility method for returning an integer array.
  *
  * @param mutateArray the array of integers to mutate.
  * @param size the size of the mutation array.
  * @param rate the percentage mutation rate.
  * @param mutation the mutation factor<?>.
  * @returns the mutation array.
  */

private int[] mutate(int[] mutateArray, int size, int rate, int mutation)  {
                         // need to take into account whether linear relationship being used

    // Get intial random number generated using the current date and time.
    // ------------------------------------------------------------------

    Date d = new Date();
    Random r = new Random(d.getTime());

    int ran = r.nextInt();



    // For each cell ....
    //     Get a random number between 0 and 100, and compare it against
    //     the mutation rate. If mutation rate is higher, then add the
    //     mutation factor to the mutation array value for that cell.
    // ------------------------------------------------------------------

    for (int i=0; i < size; i++)  {

      ran = Math.abs(ran);
      ran = ran%100;

      if (ran < rate)
        mutateArray[i] = mutateArray[i] + mutation;



      // Get next random number for the next cell.
      // ------------------------------------------------------------

      ran = r.nextInt();
    }


    return mutateArray;
  } // end mutate method

private int[] linearMutate(int[] mutateArray, int size, double m,int threshold, int mutation)  {


    // Get intial random number generated using the current date and time.
    // ------------------------------------------------------------------

    Date d = new Date();
    Random r = new Random(d.getTime());

    int ran = r.nextInt();



    // For each cell ....
    //     determine mutation rate based on equation rate = m(repeat length- threshold)
    // mutation rate must be a whole number
    // generate random number
    // perform mutation event
                    // ------------------------------------------------------------------
    for (int i=0; i < size; i++)  {

      ran = Math.abs(ran);
      ran = ran%100;

      if (ran < ((int)(m*(mutateArray[i]-threshold)+0.5)))
        mutateArray[i] = mutateArray[i] + mutation;



      // Get next random number for the next cell.
      // ------------------------------------------------------------

      ran = r.nextInt();
    }


    return mutateArray;
  } // end mutate method

private int[] randomMutate(int[] mutateArray, int arraySize, int rate,int randomLower, int randomUpper)  {

                    // method takes an array of arraySize and mutates
                    // it according to rate adding a mutation between randomLower and randomUpper

    // Get intial random number generated using the current date and time.
    // ------------------------------------------------------------------

    Date d = new Date();
    Random r = new Random(d.getTime());

    int ran = r.nextInt();



    // For each cell ....
    //     Get a random number between 0 and 99, and compare it against
    //     the mutation rate.  If mutation rate is higher, then add a
    //    random  mutation size to the mutation array value for that cell.
    // ------------------------------------------------------------------

    for (int i=0; i < arraySize; i++)  {

      ran = Math.abs(ran);
      ran = ran%100;

      if (ran < rate){
        // a mutation must take place
        // get a second random number using the given range
        ran= r.nextInt();
        ran = Math.abs(ran);
        int mutation = ran%((randomUpper-randomLower)+1);

        // add random lower
        mutation = mutation + randomLower;

        mutateArray[i] = mutateArray[i] + mutation;
                              } // end if
```

```
          // Get next random number for the next cell.
      // -----------------------------------------------------------------

      ran = r.nextInt();
    }


    return mutateArray;
  }

private int[] randomLinearMutate(int[] mutateArray, int arraySize, double m,int Threshold,int randomLower, int randomUpper)  {

                        // method takes an array of arraySize and mutates
                        // it according to  a rate equation adding a mutation between randomLower and randomUpper

    // Get intial random number generated using the current date and time.
    // -----------------------------------------------------------------

    Date d = new Date();
    Random r = new Random(d.getTime());

    int ran = r.nextInt();



          // For each cell ....
    //     determine mutation rate based on equation rate = m(repeat length- threshold)
    //  mutation rate must be a whole number
    // generate random number
    // perform mutation event by adding a random mutation between upper and lower
                        // -----------------------------------------------------------------
    for (int i=0; i < arraySize; i++)  {

      ran = Math.abs(ran);
      ran = ran%100;

      if (ran <((int)(m*(mutateArray[i]-threshold)+0.5))){
        // a mutation must take place
        // get a second random number using the given range
        ran= r.nextInt();
        ran = Math.abs(ran);
        int mutation = ran%((randomUpper-randomLower)+1);

        // add random lower
        mutation = mutation + randomLower;

        mutateArray[i] = mutateArray[i] + mutation;
                                   } // end if

          // Get next random number for the next cell.
      // -----------------------------------------------------------------

      ran = r.nextInt();
    }


    return mutateArray;
  }

 private int[] spectrumMutate(int[] mutateArray, int arraySize, int rate,
                                                                                        int[] mutationSizeArray, int
mutationSizeArrayLength)  {



                        // method takes an array of arraySize and mutates
                        // it according to rate, mutationSizeArray
    // Get intial random number generated using the current date and time.
    // ------------------------------------------------------------

    Date d = new Date();
    Random r = new Random(d.getTime());

    int ran = r.nextInt();



    // For each cell ....
    //    Get a random number between 0 and 99, and compare it against
    //    the mutation rate.
    // ------------------------------------------------------------

    for (int i=0; i < arraySize; i++) {

       ran = Math.abs(ran);
       ran = ran%100;

       if (ran < rate){
          // a mutation must take place
          // get a second random number between 0 and 99
          ran= r.nextInt();
          ran = Math.abs(ran);
          ran = ran%100;

          // the contents of mutationSizeArray are added up. When the contents
          // is greater than (not equal!) the random number the index for the array
          // denotes the size
          int sum = 0;
          for (int j=0; j < mutationSizeArrayLength; j++)
            {
                        sum = sum + mutationSizeArray[j];

                        if (sum > ran) {
                                    mutateArray[i] = mutateArray[i] + (j-10);
                                    // should be self contained
                                    break; // fromfor loop
                        } // end if

            } // end for

       } // end if

            // Get next random number for the next cell.
       // ------------------------------------------------------------

       ran = r.nextInt();
    }


    return mutateArray;
  } // end method


  /**
   * Utility method for returning an array of frequencies<?>
   *
   * @param startArray initial array of frequencies.
   * @param n  <?>
   * @param freqArray array of frequency values.
   * @returns the array of frequency values.
   */

  private double[] frequency(int[] startArray, int n, double[] freqArray) {


    // Initialise the frequency array.
    // ------------------------------------------------------------
    int i = 0;
    freqArray[i] = startArray[i];

    for (i=1;  i<maxDist;  i++)  {
       freqArray[i] = startArray[i] + freqArray[i-1];
    }



    //  Calculate the frequencies.
    //  ---------------------------------------------------------------

    for (int j=0;  j<maxDist;  j++)  {
       freqArray[j] = (double)(freqArray[j]/n);
    }


    return freqArray;
 }




/**
  *  Class  :  Repeat
  *
  *  Inner class used as a superclass for the Der and Mod classes.
  */

public class Repeat  implements ModelConstants  {


    protected int[] alleles;
    protected int size;



    /**
      *  Method to convert an array of integer values into an array
      *  of their graphical distribution, which can then be used to
      *  draw a histogram of the data.
      *
      *  @param source initial array of integers.
      *  @param size initial size of array.
      *  @param target final array of integers.
      *  @returns the final array of integers.
      */

    protected int[] convert(int[] source, int size, int[] target)  {


       //  First make sure the target array is empty.
       //  ------------------------------------------------------------

       for (int j=0;  j<maxDist;  j++)  {
          target[j] = 0;
       }



       //  Convert the source array into an array of frequencies.
       //  ie. depending upon the value of source[i], increment the target
       //  counter for that value.
       //  ------------------------------------------------------------

       for (int i=0;  i<size;  i++)  {

          if (source[i] > maxPoolSize)
             target[(maxDist-1)]++;
```

```
        else if (source[i] <= 0)
          target[0]++;
        else
                    target[source[i]/20]++;
      }


      return target;
    }
}




/**
 * Class : Mod
 *
 * Subclass of Repeat class - Used for model data.
 */

class Mod  extends Repeat {


   // Initialise an integer array which represent the model graph data.
   // ----------------------------------------------------------------

   int[] graphArray = new int[maxDist];




   /**
    * Method used to draw the graph data as histogram bars.
    *
    * @param g the Graphics object to use.
    */

   void draw1(Graphics g) {
          // method will draw results of iteration if iteration check box has bee selected
          // else will draw graph
          // ----------------------------------------------------------------

          if (controlPanel.getCheckboxState() == true) {

          // Convert alleles to an array for statistical distribution
      // ----------------------------------------------------------------

      graphArray = convert(this.alleles, this.size, graphArray);


          // draw results of iteration
          // ----------------------------------------------------------------

          g.drawString("Best parameters are: ", 40,20);
          g.drawString("Progenitor: " + bestProgenitor,40,30);
          g.drawString("Mutation Rate: " + bestRate, 40,40);
          g.drawString("Mutation Size: " + bestSize, 40, 50);

          } else {



      // Draw the number of cycles selected.
      // ----------------------------------------------------------------

      g.drawString("Cycles: "+cycleCounter, 40, 10);



      // Convert alleles to an array for graphical distribution.
    // ---------------------------------------------------------------

    graphArray = convert(this.alleles, this.size, graphArray);


    // Draw the histogram of model data.
    // ---------------------------------------------------------------

    for (int i=0;  i<maxDist;  i++)  {

      double tempPercent = ((double)graphArray[i]/this.size) * 100;
      int percent = (int)(tempPercent + 0.5);

      g.fillRect((30+(i*widthPix)), (heightPix-(percent*percentPix)),
                  widthPix, (percent*percentPix));
    }
    } // end else
  }
}



/**
 *  Class  :  Der
 *
 *  Subclass of Repeat class  -  Used for derived data.
 */

class Der  extends Repeat  {


  // Initialise an integer array which represent the derived graph data.
  // ---------------------------------------------------------------

  int[] derivedArray = new int[maxDist];



  /**
   *  Method used to draw the graph data as histogram bars.
   *
   *  @param g the Graphics object to use.
   */

  void draw1(Graphics g)  {


    // If there is derived data  .....
    // ---------------------------------------------------------------

    if (isDerived == true)  {


      // Convert alleles to an array for graphical distribution.
      // -------------------------------------------------------------

      derivedArray = convert(this.alleles, this.size, derivedArray);


      // Draw the histogram of model data.
      // -------------------------------------------------------------

      for (int i=0;  i<maxDist;  i++)  {
        double tempPercent = ((double)derivedArray[i]/this.size)*100;
        int percent = (int)(tempPercent + 0.5);
```

```
                    g.fillRect((30+(i*widthPix)),
                              (heightPix-(percent*percentPix)), widthPix,
                              (percent * percentPix));
                }
            }


            // Else .... no derived data.
            // --------------------------------------------------------

            if (isDerived == false)
                g.drawString("No Derived information.", 150, 100);
        }
    }
}
```

Class ModelControlPanel:

```
/**
 * Class :   ModelControlPanel
 *
 * @author   Grant Hogg (ghogg@molgen.gla.ac.uk)
 *           David Jack (davidj@dcs.gla.ac.uk)
 * @version  0.3
 */




import java.awt.Button;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.Panel;
import java.awt.TextField;
import java.awt.TextArea;
import java.awt.Checkbox;
import java.awt.event.ActionListener;
import java.awt.Choice;
import java.awt.event.ItemListener;


/**
 * Panel sub-class which represents the CGT repeat instability simulation
 * applet's parameter controls.
 */

public class ModelControlPanel  extends Panel  implements ModelConstants  {



  /**
   * User interface components.
   */
  public FixedRateFrame fixedRateFrame;
          public MutationFrame mutFrame;
          public SpectrumFrame specFrame;
          public RandomFrame ranFrame;
          public RateFrame rateFrame;

  private TextField mutationRateTextField;
  //private TextField mutationSizeTextField;
  private TextField cyclesTextField;
  private TextField progenitorTextField;
  private TextField poolSizeTextField;

          public Choice rateOptions;
          public Choice mutationOptions;
  private Button runButton;
  private Button loadButton;
  private Button clearButton;
  private Button pasteButton;
          private TextArea derivedInfo;
          public String selectedItem;
          public String selectedRate;
          private Checkbox iterationCheckbox;


  /**
   * Initialised settings.
   */

  private Color myColor = new Color(200, 150, 250);
private boolean isDerived = false;
private int cycleCounter = 0;



/**
 * Simulation parameters.
 */

//private int mutationRate = 10;
//private int mutationSize = 10;
private int cycles = 100;
private int progenitor = 160;
private int poolSize = 1000;



/**
 * Constructs the controls panel for the model simulator.
 */

public ModelControlPanel() {


  // Constructs the panel, with a grid layout manager.
  // ---------------------------------------------------------------

  super();

  setBackground(myColor);
  setLayout(new GridLayout(9, 2));

                 specFrame = new SpectrumFrame();
                 mutFrame = new MutationFrame();
                 ranFrame = new RandomFrame();
                 rateFrame = new RateFrame();
                 fixedRateFrame = new FixedRateFrame();

                 selectedItem = ("Single Mutation");
                 selectedRate = ("Fixed Rate");

  // Add the control settings as a series of label/textfield rows.
  // ---------------------------------------------------------------
                 mutationOptions = new Choice();
                 mutationOptions.add("Single Mutation");
                 mutationOptions.add("Random Mutation");
                 mutationOptions.add("Select Spectrum");

  // Add the control settings for the rate choice menu
  // ---------------------------------------------------------------
  rateOptions = new Choice();
  rateOptions.add("Fixed Rate");
  rateOptions.add("Linear");



                 // add label
                 add(new Label("Mutation type: "));
                 add(mutationOptions);

  add(new Label("MutationRate:"));
  add(rateOptions);

  add(new Label("No of Cycles:"));
  cyclesTextField = new TextField(""+cycles, 6);
  add(cyclesTextField);

  add(new Label("Progenitor Allele:"));
  progenitorTextField= new TextField(""+progenitor, 6);
  add(progenitorTextField);

  add(new Label("Size of Pool:"));
  poolSizeTextField = new TextField(""+poolSize, 6);
  add(poolSizeTextField);



  // Add the user interface buttons: 'Load Data' and 'Run Model'.
  // ------------------------------------------------------------------

  loadButton = new Button ("Load Data");
  runButton = new Button ("Run Model");

  add(loadButton);
  add(runButton);

  // add the buttons and text field for cutting and pasting data
  // ------------------------------------------------------------------

  derivedInfo = new TextArea("info here",3,20,derivedInfo.SCROLLBARS_VERTICAL_ONLY);
  clearButton= new Button("Clear");
                  pasteButton= new Button("Paste");

                  add(clearButton);
                  add(pasteButton);
                  add(derivedInfo);

                  // add the checkbox for iteration here I think
                  // ------------------------------------------------------------------

                  iterationCheckbox = new Checkbox("iteration", false);
                  add (iterationCheckbox);

}



/**
 * Delegates the control buttons to the given ActionListener.
 *
 * @param l the ActionListener.
 */

public void addActionListener(ActionListener l) {

  loadButton.addActionListener(l);
  runButton.addActionListener(l);
  clearButton.addActionListener(l);
  pasteButton.addActionListener(l);


  // not sure if I have to add an actionlistener for the checkbox

}

        public void addItemListener(ItemListener k) {

                  mutationOptions.addItemListener(k);
                  rateOptions.addItemListener(k);

        }

/**
 * Returns the mutation rate setting.
 *
 * @returns the mutation rate setting.
 */
```

```
  public int getMutationRate() {

    return intFromTextField(mutationRateTextField);
  }



  /**
   * Returns the mutation size setting.
   *
   * @returns the mutation size setting.
   */

 /* public int getMutationSize() {

    return intFromTextField(mutationSizeTextField);
  }
*/


  /**
   * Returns the cycles setting.
   *
   * @returns thc cycles setting.
   */

  public int getCycles() {

    return intFromTextField(cyclesTextField);
  }



  /**
   * Returns the progenitor setting.
   *
   * @returns the progenitor setting.
   */

  public int getProgenitor() {

    return intFromTextField(progenitorTextField);
  }



  /**
   * Returns the pool size setting.
   *
   * @returns the pool size setting.
   */

  public int getPoolSize() {

    return intFromTextField(poolSizeTextField);
  }

  /**
   * Returns array containing range of mutation rates
   *
   * @returns array containing range of mutation rates.
   */

  public int[] getMutationRateRange() {
          return arrayFromTextField(mutationRateTextField);
  }

  /**
   * Returns array containing range of mutation sizes
   *
   * @returns array containing range of mutation sizes.
   */

/* public int[] getMutationSizeRange() {

          return arrayFromTextField(mutationSizeTextField);
  }
*/

  /**
   * Returns array containing range of progenitors
   *
   * @returns array containing range of progenitor alleles.
   */

  public int[] getProgenitorRange() {

          return arrayFromTextField(progenitorTextField);
  }

  public boolean getCheckboxState() {

          return iterationCheckbox.getState();
  }



  public void clearArea() {
          derivedInfo.setText("");
  }

  public void setArea(String s) {
          derivedInfo.setText(s);
  }

          public String getInfo() {
                  String s;
                  s = derivedInfo.getText();

                  return s;
          }


  /**
   * Utility method for obtaining the integer representation of a number
   * entered into a TextField.
   *
   * @param tf the TextField.
   * @returns the integer representation of the TextField's String.
   */

  private int intFromTextField(TextField tf)  {


    String s;
    int value;



    // Get the text from the TextField.
    // --------------------------------------------------------------------

    s = tf.getText();
    // Try ... parsing it into an int.
    // ---------------------------------------------------------------

    try{
       value = Integer.parseInt(s);
    } catch(Exception e) {
       value = 0;
    }


    return value;
}

/**
 * Utility method for obtaining an array integer representing the range
 * entered into a TextField.
 *
 * @param tf the TextField.
 * @returns the integer array representation of the TextField's String.
 */


        private int[] arrayFromTextField(TextField tf) {
                // this method reads in a string containing 2 numbers seperated by a "-"
                // and converts them to integers in an array
                // -----------------------------------------------------------------

                // define variables
                // -----------------------------------------------------------------
                String s,min,max;
                StringBuffer first = new StringBuffer("");
                StringBuffer second = new StringBuffer("");
                int[] values;
                values = new int[2];

                // get string from test fields
                // -----------------------------------------------------------------
                s= tf.getText();

                // now have a string
                // lets churn through it
                // -----------------------------------------------------------------
                int position = 0;

                while (s.charAt(position) != '-') {
                        first.append(s.charAt(position));
                        position ++;
                } // end while

                // have read to a '-' which separates the two numbers
                // -----------------------------------------------------------------

                if (s.charAt(position) == '-'){
                        while ((position+1) < s.length()) {
                        second.append(s.charAt(position+1));
                        position ++;
                        } // end while

                } // end if

                // now have 2 string buffers
                // convert to strings
                // -----------------------------------------------------------------
                min = first.toString();
                max = second.toString();

                // convert strings to integers
                // -----------------------------------------------------------------
```

```
                    try{
                              values[0] = Integer.parseInt(min);
                    }catch (Exception e) {
                              values[0] = 1;
                    }

                    try{
                              values[1] = Integer.parseInt(max);
                    } catch (Exception e) {
                              values[1] = 1;
                    }

          // return array
          // -----------------------------------------------------------
          return values;
     } // end method

}
```

Class ModelResultsPanel:/**

```
 * Class :   ModelResultsPanel
 *
 * @author    Grant Hogg (ghogg@molgen.gla.ac.uk)
 *           David Jack (davidj@dcs.gla.ac.uk)
 * @version  0.3
 */



import java.awt.Label;
import java.awt.Panel;
import java.awt.TextArea;
import java.awt.TextField;



/**
 * Panel sub-class which is used to display the statistical analysis results
 * from the CGT repeat instability simulation.
 */

public class ModelResultsPanel  extends Panel  implements ModelConstants  {



  /**
   * User interface components.
   */

  private TextField bestDTextField;
  private TextField bestCycleTextField;
  private TextArea significanceTextArea;



  /**
   * Constructs the model simulator's results panel.
   */

  public ModelResultsPanel()  {


    // Construct the panel.
    // -------------------------------------------------------------------

    super();
    setSize(800, 150);
    setLayout(null);



    // Add statistical analysis components as a series of label/textfields.
    // 1)  'Best D'
    // -------------------------------------------------------------------

    Label bestDLabel = new Label("Best D:");
    bestDLabel.setBounds(20, 20, 90, 30);
    add(bestDLabel);

    bestDTextField = new TextField(12);
    bestDTextField.setEditable(false);
    bestDTextField.setBounds(150, 20, 200, 30);
    add(bestDTextField);
   // 2) 'After Cycles'
   // -----------------------------------------------------------

   Label bestCycleLabel = new Label("After Cycles:");
   bestCycleLabel.setBounds(400, 20, 90, 30);
   add(bestCycleLabel);

   bestCycleTextField = new TextField(12);
   bestCycleTextField.setEditable(false);
   bestCycleTextField.setBounds(530, 20, 200, 30);
   add(bestCycleTextField);



   // 3) 'Significance'
   // -----------------------------------------------------------

   Label significanceLabel = new Label("Significance:");
   significanceLabel.setBounds(20, 80, 100, 30);
   add(significanceLabel);

   significanceTextArea = new TextArea("", 4, 12,TextArea.SCROLLBARS_NONE);
   significanceTextArea.setEditable(false);
   significanceTextArea.setBounds(150, 80, 580, 40);
   add(significanceTextArea);
}



/**
 * Clears the text from the three result fields.
 */

public void clearResults() {

   bestDTextField.setText("");
   bestCycleTextField.setText("");
   significanceTextArea.setText("");
}



/**
 * Sets the Best D statistic text.
 *
 * @param str the value of the Best D statistic.
 */

public void setBestD(String str) {

   bestDTextField.setText(str);
}



/**
 * Sets the Best Cycle statistic text.
 *
 * @param str the value of the Best Cycle statistic.
 */

public void setBestCycle(String str) {

   bestCycleTextField.setText(str);
}
```

```
  /**
   * Sets the significance statistic text.
   *
   * @param str the value of the significance statistic.
   */

  public void setSignificance(String str) {

    significanceTextArea.setText(str);
  }
}
```

Class GraphCanvas:

```
/**
 * Class :   GraphCanvas
 *
 * @author   Grant Hogg (ghogg@molgen.gla.ac.uk)
 *          David Jack (davidj@dcs.gla.ac.uk)
 * @version  0.3
 */




import java.awt.Canvas;
import java.awt.Color;
import java.awt.Graphics;

import java.util.Vector;




/**
 * Canvas sub-class used to display a distribution histogram, given a Vector
 * of Strings, representing floating point numbers.
 */

public class GraphCanvas  extends Canvas  implements  ModelConstants  {



  /**
   * Private parameters:  graph title and data resepectively.
   */

  private String title;
  private Vector data;



  /**
   * Constructs a GraphCanvas with the given title and an empty data set.
   *
   * @param title the graph's title.
   */

  public GraphCanvas(String title)  {

    super();
    this.title = title;
    data = new Vector();
  }



  /**
   * Canvas paint method.
   * Draws the graph title, axes, labels, and tick markers.
   * Uses the 'Mod' and 'Der' classes to paint the actual data points.
   *
   * @param g the Canvas' Graphics object.
   */

  public void paint (Graphics g)  {
    // Set graph color to black.
    // -----------------------------------------------------------------

    g.setColor(Color.black);


    // Draw the graph axes, labels, indentations and tick markers.
    // -----------------------------------------------------------------

    drawGraphOutline(g, "No. of Repeats", "%");


    // Draw the graph data.
    // -----------------------------------------------------------------

    drawGraphData(g);
}



/**
 * Draws the graph axes, labels, indentations and tick markers.
 *
 * @param g the Graphics object.
 * @param xLabel the x-axis label.
 * @param yLabel the y-axis label.
 */

private void drawGraphOutline(Graphics g, String xLabel, String yLabel) {


    // Draw the outline of the graph and the axes labels.
    // -----------------------------------------------------------------

    g.drawLine (397, 10, 397, 210);
    g.drawLine (30, 10, 30, 210);
    g.drawLine (30, 210, 397, 210);

    g.drawString (title, 40, 240);
    g.drawString (xLabel, 150, 240);
    g.drawString (yLabel, 5, 105);


    // Draw indentations and numbers.
    // -----------------------------------------------------------------

    for (int i=1; i<maxDist+1; i++) {
        g.drawLine (30+(i*widthPix), 210, 30+(i*widthPix), 213);

        if ((i%10) == 0) {
            g.drawString(String.valueOf(i*20),30+(i*widthPix),225);
        }
    }
}



/**
 * Draws the graph data as a series of histogram bars.
 *
 * @param g the Graphics object.
 */

private void drawGraphData(Graphics g) {
    // If there is no data .... draw 'no data' string.
    // ------------------------------------------------------------

    if (data.isEmpty()) {

      g.drawString("No Data.", 150, 100);


    // Else, for each data point ....
    // ------------------------------------------------------------

    } else {

      for (int i=0; i<data.size(); i++) {


        // Calculate the percentage distribution of the data point.
        // ------------------------------------------------------

        int tmp = ((Integer)data.elementAt(i)).intValue();
        double tmpPercent = (tmp/data.size()) * 100;
        int percent = (int)(tmpPercent + 0.5);



        // Draw the histogram bar representing that distribution.
        // ------------------------------------------------------

        g.fillRect((30+(i*widthPix)), (heightPix-(percent*percentPix)),
          widthPix, (percent*percentPix));
      }
    }
  }
}
```

Class MutationFrame:

```
import java.awt.*;
import java.awt.event.*;

public class MutationFrame extends Frame implements ItemListener {
          // define variables

          private int mutationSize = 1;
          private TextField mutationSizeTextField;


          public MutationFrame()
          {
                    // call base class constructor
                    super ("Mutations Size");
                    setSize(200,50);
                    addWindowListener(new CloseWindow());

                    setLayout( new FlowLayout(1,0,0));
                    // set up layout
                    // default is border layout

                    add(new Label("Mutation Size: "));
                    mutationSizeTextField = new TextField(""+mutationSize,6);
                    add(mutationSizeTextField);
          } // end constructor


          // utility methods

          public int getMutationSize()  {

    return intFromTextField(mutationSizeTextField);
  }

  public int[] getMutationSizeRange() {

          return arrayFromTextField(mutationSizeTextField);
  }


  private int intFromTextField(TextField tf)  {


    String s;
    int value;



    // Get the text from the TextField.
    // ------------------------------------------------------------------

    s = tf.getText();



    // Try ... parsing it into an int.
    // ------------------------------------------------------------------

    try{
      value = Integer.parseInt(s);
    } catch(Exception e)  {
      value = 0;
    }


    return value;
  }
public void itemStateChanged (ItemEvent e)
        {
                repaint();
        }


        private int[] arrayFromTextField(TextField tf) {
                // this method reads in a string containing 2 numbers seperated by a "-"
                // and converts them to integers in an array
                // -----------------------------------------------------------------

                // define variables
                // -----------------------------------------------------------------
                String s,min,max;
                StringBuffer first = new StringBuffer("");
                StringBuffer second = new StringBuffer("");
                int[] values;
                values = new int[2];

                // get string from test fields
                // -----------------------------------------------------------------
                s= tf.getText();

                // now have a string
                // lets churn through it
                // -----------------------------------------------------------------
                int position = 0;

                while (s.charAt(position) != '-') {
                        first.append(s.charAt(position));
                        position ++;
                } // end while

                // have read to a '-' which separates the two numbers
                // -----------------------------------------------------------------

                if (s.charAt(position) == '-'){
                        while ((position+1) < s.length()) {
                        second.append(s.charAt(position+1));
                        position ++;
                        } // end while

                } // end if

                // now have 2 string buffers
                // convert to strings
                // -----------------------------------------------------------------
                min = first.toString();
                max = second.toString();

                // convert strings to integers
                // -----------------------------------------------------------------
                        try{
                                values[0] = Integer.parseInt(min);
                        }catch (Exception e) {
                                values[0] = 1;
                        }

                        try{
                                values[1] = Integer.parseInt(max);
                        } catch (Exception e) {
                                values[1] = 1;
                        }

                // return array
                // -----------------------------------------------------------------
                return values;
        } // end method
}// end class
```

Class RandomFrame:

```
import java.awt.*;
import java.awt.event.*;


public class RandomFrame extends Frame implements ItemListener {
        // define variables

        private TextField mutationSizeTextField;

        public RandomFrame()
        {
                // call base class constructor
                super ("Random Mutation Size Range");
                setSize(250,50);
                addWindowListener(new CloseWindow());

                setLayout( new FlowLayout(1,0,0));
                // set up layout
                // default is border layout

                add(new Label("Random size range: "));
                mutationSizeTextField = new TextField("-10:10",8);
                add(mutationSizeTextField);
        } // end constructor


        public void itemStateChanged (ItemEvent e)
                {
                        repaint();
                }
 /**
   * Returns array containing range of mutation sizes
   *
   * @returns array containing range of mutation sizes.
   */

  public int[] getMutationSizeRange() {

        return arrayFromTextField(mutationSizeTextField);
  }

  private int[] arrayFromTextField(TextField tf) {
                // this method reads in a string containing 2 numbers seperated by a ":"
                // and converts them to integers in an array
                // -----------------------------------------------------------------

                // define variables
                // -----------------------------------------------------------------
                String s,min,max;
                StringBuffer first = new StringBuffer("");
                StringBuffer second = new StringBuffer("");
                int[] values;
                values = new int[2];

                // get string from test fields
                // -----------------------------------------------------------------
                s= tf.getText();

                // now have a string
                // lets churn through it
                // -----------------------------------------------------------------
                int position = 0;

                while (s.charAt(position) != ':') {
                        first.append(s.charAt(position));
                        position ++;
                } // end while
        // have read to a ':' which separates the two numbers
        // ----------------------------------------------------------------

        if (s.charAt(position) == ':'){
            while ((position+1) < s.length()) {
            second.append(s.charAt(position+1));
            position ++;
            } // end while

        } // end if

        // now have 2 string buffers
        // convert to strings
        // ----------------------------------------------------------------
        min = first.toString();
        max = second.toString();

        // convert strings to integers
        // ----------------------------------------------------------------
            try{
                values[0] = Integer.parseInt(min);
            }catch (Exception e) {
                values[0] = 1;
            }

            try{
                values[1] = Integer.parseInt(max);
            } catch (Exception e) {
                values[1] = 1;
            }

        // return array
        // ----------------------------------------------------------------
        return values;
    } // end method

}// end class
```

Class SpectrumFrame:

```
// next class
// class produces a frame which will display a graph


import java.awt.*;
import java.awt.event.*;
import java.awt.event.ItemListener;

public class SpectrumFrame extends Frame implements ItemListener,MouseListener,MouseMotionListener {
        // define variables
        public int[] spectrumArray;
        private int total;
        private int xpos, ypos = -10;
        private int currentX, currentY = -10;
        // define constants
        public static final int heightpix =200; // no of pixels in graph square
        public static final int widthpix = 10; // width of each histogram interval
        public static final int boundary = 50; // border size
        public static final int intervalno = 21; // no of intervals histogram will be split into
        public static final int ypix = 2; // no of pixels reprseting a single percentage
        public static final int repeatInterval = -10; // no of repeats represented by

        public SpectrumFrame ()
        {
                // call base class constructor
                super( "Mutation Spectra");
                setSize( boundary + (widthpix*intervalno) + boundary, boundary +heightpix + boundary);
                addWindowListener( new CloseWindow());

                // need to set up layout and event listeners here!
                // add mouse listener
                addMouseListener(this);
                addMouseMotionListener(this);


                // define spectrumArray and total
                spectrumArray = new int[intervalno];

                // initialise spectrumArray
                for ( int i=0; i<intervalno; i++) {
                        spectrumArray[i] =0;
                }

                total = getTotal(spectrumArray);
        } // end constructor

        public void mouseClicked( MouseEvent e )
                {
                                        xpos = e.getX();
                                        ypos = e.getY();

                                        // now have the coordinates
                                        // is mouse position within graph boundary?
                                        // something is wrong here
                                        if ((xpos >= boundary && xpos < (boundary +  (intervalno*widthpix))) &&
                                                        (ypos >= boundary && ypos < (boundary + heightpix)))
                                        {
                                                // update spectrumArray
                                                spectrumArray[((xpos-boundary)/widthpix)] =
                                                                                        (int)((heightpix-(ypos-
boundary))/ypix);

                                                total = getTotal(spectrumArray);

                                                // need to check mutation spectra is not greater than 100%
                                                if (total > 100)
                                                {
                                                        // this mutation spectra can only ever add up to 100
                                                        spectrumArray[((xpos-boundary)/widthpix)] =
                                                                                (int)((heightpix-(ypos
- boundary))/ypix) -(total-100);
                                                // make sure this has been cast properly
                                        }

                                        // have updated spectrumArray so draw graph
                                        // get graphics object

                                        Graphics g = this.getGraphics();
                                        draw(g);
                                } // end if

                } //  end mouseClicked

                // need to call other method in mouselistener interface
        public void mousePressed ( MouseEvent e) {};
        public void mouseReleased ( MouseEvent e) {};
        public void mouseEntered ( MouseEvent e) {};
        public void mouseExited ( MouseEvent e) {};

        public void mouseMoved( MouseEvent e)
                {
                currentX = e.getX();
                currentY = e.getY();
                Graphics g = this.getGraphics();

                if ((currentX >= boundary && currentX < (boundary + (intervalno*widthpix))) &&
                                                (currentY >= boundary && currentY < (boundary + heightpix)))
                        {

                                draw(g);
                                g.setColor(Color.magenta);
                                g.drawString(String.valueOf((int)((heightpix-(currentY-
boundary))/ypix))+"%",currentX,currentY);
                        } // end if
                else
                        {
                        //redraw graph so ther are no numbers hanging around!

                                draw(g);
                        } // end else
                } // end method

        // call other event handler in MouseMotionListener interface

        public void mouseDragged( MouseEvent e ) {};


        public int getTotal(int[] source)
        {
                int sum = 0;

                // method returns the sum of a source array's elements
                for (int i=0; i< intervalno; i++) {
                        sum= sum + source[i];
                }
                return sum;
        }

        public void paint (Graphics g)
        {
                // paint method for spectrumframe
                // set color to black
                g.setColor(Color.black);

                // draw graph outline
                g.drawLine(boundary-1,boundary,boundary-1,heightpix+boundary);
                g.drawLine(boundary,heightpix + boundary,
                                                        boundary + (widthpix*intervalno),
heightpix+boundary);
```

```
                // draw labels
                g.drawString("Mutation Size",boundary + 80, boundary+heightpix + 30);
                g.drawString("%",boundary-20,boundary+(int)(heightpix/2));

                // draw indentations and numbers
                g.drawString(String.valueOf(repeatInterval), boundary-4, boundary+heightpix+15);
                for ( int i =1; i <(intervalno +1); i++)
                {
                                g.drawLine(boundary+(i*widthpix),heightpix+boundary,

        boundary+(i*widthpix),boundary+heightpix+3);
                                if ((i%2)==0) {
                                g.drawString(String.valueOf(i + repeatInterval), boundary + (i*widthpix),
boundary+heightpix+15);


                        }
                } // end for

                // now draw graph
                draw(g);
        } // end paint

        public void draw(Graphics g)
        {
                // draws the graph
                // get total
                total = getTotal(spectrumArray);
                // first clear the area
                g.setColor(Color.white);
                g.fillRect(boundary,boundary-30,(widthpix*intervalno)+30,heightpix+30);

                g.setColor(Color.black);
                // draw the total % produced
                g.drawString("Total = " + total     + "%", boundary,boundary-20);

                // use spectrumArray to calculate positions of graphs
                for (int i=0; i<intervalno; i++){
                        // fill rect uses X,Y,width, Height
                        g.fillRect(boundary + (i*widthpix),boundary + (heightpix- (spectrumArray[i]*2)), widthpix,
(spectrumArray[i]*2));
                }  // end for
        }// end method

        public void itemStateChanged (ItemEvent e)
        {
                repaint();
        }
}// end class
```

Class FixedRateFrame:

```
import java.awt.*;
import java.awt.event.*;

public class FixedRateFrame extends Frame implements ItemListener {
            // define variables

            private int mutationRate = 10;
            private TextField mutationRateTextField;


            public FixedRateFrame()
            {
                        // call base class constructor
                        super ("Mutation Rate");
                        setSize(200,50);
                        addWindowListener(new CloseWindow());

                        setLayout( new FlowLayout(1,0,0));
                        // set up layout
                        // default is border layout

                        add(new Label("Mutation Rate: "));
                        mutationRateTextField = new TextField(""+mutationRate,6);
                        add(mutationRateTextField);
            } // end constructor


            // utility methods

            public int getMutationRate() {

      return intFromTextField(mutationRateTextField);
  }

  public int[] getMutationRateRange() {

            return arrayFromTextField(mutationRateTextField);
  }


  private int intFromTextField(TextField tf) {


      String s;
      int value;



      // Get the text from the TextField.
      // ------------------------------------------------------------------

      s = tf.getText();



      // Try ... parsing it into an int.
      // ------------------------------------------------------------------

      try{
         value = Integer.parseInt(s);
      } catch(Exception e) {
         value = 0;
      }


      return value;
  }
public void itemStateChanged (ItemEvent e)
        {
                repaint();
        }


        private int[] arrayFromTextField(TextField tf) {
                // this method reads in a string containing 2 numbers seperated by a "-"
                // and converts them to integers in an array
                // ---------------------------------------------------------------

                // define variables
                // ---------------------------------------------------------------
                String s,min,max;
                StringBuffer first = new StringBuffer("");
                StringBuffer second = new StringBuffer("");
                int[] values;
                values = new int[2];

                // get string from test fields
                // ---------------------------------------------------------------
                s= tf.getText();

                // now have a string
                // lets churn through it
                // ---------------------------------------------------------------
                int position = 0;

                while (s.charAt(position) != '-') {
                        first.append(s.charAt(position));
                        position ++;
                } // end while

                // have read to a '-' which separates the two numbers
                // ---------------------------------------------------------------

                if (s.charAt(position) == '-'){
                        while ((position+1) < s.length()) {
                        second.append(s.charAt(position+1));
                        position ++;
                        } // end while

                } // end if

                // now have 2 string buffers
                // convert to strings
                // ---------------------------------------------------------------
                min = first.toString();
                max = second.toString();

                // convert strings to integers
                // ---------------------------------------------------------------
                        try{
                                values[0] = Integer.parseInt(min);
                        }catch (Exception e) {
                                values[0] = 1;
                        }

                        try{
                                values[1] = Integer.parseInt(max);
                        } catch (Exception e) {
                                values[1] = 1;
                        }

                // return array
                // ---------------------------------------------------------------
                return values;
        } // end method
}// end class
```

Class RateFrame:

```
import java.awt.*;
import java.awt.event.*;
import java.awt.event.ItemListener;

public class RateFrame extends Frame implements ItemListener,MouseListener
{
// a frame that will aloow a user to alter linear rate relationship

// variables

public int diseaseThreshold = 0;
public double m =0.105263;

private int xpos,ypos = -10;
private int x1,y1;
private int x2,y2;

// define constants

public static final int heightpix=200;
public static final int widthpix=400;
public static final int boundary = 50;

public static final double pixPerRepeat = 0.4;
public static final int pixPerRate = 2;

public RateFrame()
        {

                // call base class constructor
                super( "Linear Relationship");
                setSize( boundary + widthpix + boundary, boundary +heightpix + boundary);
                addWindowListener( new CloseWindow());

                // need to set up layout and event listeners here!
                // add mouse listener
                addMouseListener(this);

                // initialise x1, y1,x2,y2

                x1 = diseaseThreshold;
                y1 = 0;
                x2 = 1000;
                y2 = 100;

        }


public void mouseClicked ( MouseEvent e)
        {

                xpos = e.getX();
                ypos = e.getY();


                // now have the coordinates

                // is the mouse position within graph boundary?

                if ((xpos >= (boundary + (int)(diseaseThreshold*pixPerRepeat)) && xpos < (boundary + widthpix))
                        && (ypos >= boundary && ypos < (boundary + heightpix)))
                {
                        // update graph and calculate m

                        Graphics g = this.getGraphics();
                        y2 = convertY(ypos);
                        x2 = convertX(xpos);
                        m = (double)(y2-y1)/(x2-x1);

                        draw(g);
                }
        } // end method


        // need to call other method in mouselistener interface
public void mousePressed ( MouseEvent e) {};
public void mouseReleased ( MouseEvent e) {};
public void mouseEntered ( MouseEvent e) {};
public void mouseExited ( MouseEvent e) {};


public void paint (Graphics g)
        {
                // paint method for rateframe
                // set color to black
                g.setColor(Color.black);

                // draw graph outline
                g.drawLine(boundary-1,boundary,boundary-1,heightpix+boundary);
                g.drawLine(boundary,heightpix + boundary,
                                                        boundary +widthpix, heightpix+boundary);

                // draw labels
                g.drawString("Repeat length",boundary + 80, boundary+heightpix + 30);
                g.drawString("% Rate",boundary-40,boundary+(int)(heightpix/2));

                // draw indentations and numbers

                for ( int i =boundary; i <(boundary+widthpix+1); i+= 50)
                {
                                g.drawLine(i,heightpix+boundary,
                                                                        i,boundary+heightpix+3);

                                g.drawString(String.valueOf((int)(i-boundary)/pixPerRepeat),i, boundary+heightpix+15);


                } // end for

                // now draw graph
                draw(g);
        } // end paint


public void draw(Graphics g)
        {
                // draws the line

                // clear area
                g.setColor(Color.white);
                g.fillRect(boundary,boundary-30,widthpix+30,heightpix+30);

                g.setColor(Color.black);

                // calculate x and y coordinates using equation
                // rate =m(repeatLength- threshold)

                // only want to draw graph to the edge of the graph area
                // if gradient small x will be maximum

                y2 = (int)(m * (1000-diseaseThreshold));

                if ( y2 <= 100) {

                // drawline using x2 = 1000
                x2=1000;
                g.drawLine( convertRepeat(x1),convertRate(y1),convertRepeat(x2),convertRate(y2));
                }
                else
                {
                // drawline using y2 = 100
                y2 = 100;
                // rearranged equation
                x2 = (int)(y2/m) + diseaseThreshold;
                g.drawLine( convertRepeat(x1),convertRate(y1),convertRepeat(x2),convertRate(y2));
                }

                // update equation
                g.drawString("Rate = " + m +"(Repeat Length - " + diseaseThreshold + ")"
                                                                , boundary,boundary-20);

        // end method
        }

        public void itemStateChanged(ItemEvent e)
                {
                        repaint();
                }

        // methods

        public int convertX(int x)
        {
        // convert X coordinate into repeat number
        int repeat;
        repeat = (int)(((x-boundary)/pixPerRepeat) + 0.5);
        return repeat;
        }

        public int convertY(int y)
        {
        // convert Y coordinate into mutation Rate
        int rate;
        rate = (int)(((boundary + heightpix - y)/pixPerRate) + 0.5);
        return rate;
        }

        public int convertRepeat(int repeat)
        {
        // convert Repeat into a usable x coordinate
        int x;
        x = (int)((boundary + repeat*pixPerRepeat) + 0.5);
        return x;
        }

        public int convertRate(int rate)
        {
        // converts rate into a usable y coordinate
        int y;
        y = (heightpix+boundary) - (rate*pixPerRate);
        return y;
        }

}
```

Class CloseWindow:

```
// next class
import java.awt.event.*;

public class CloseWindow extends WindowAdapter {
            public void windowClosing ( WindowEvent e)
            {
                        e.getWindow().setVisible( false );
            }
} // end class
```

Interface ModelConstants

```
/**
 * Interface : ModelConstants.
 *
 * @author            Grant Hogg (ghogg@molgen.gla.ac.uk)
 *       David Jack (davidj@dcs.gla.ac.uk)
 * @version  0.3
 */




/**
 * Interface used to define all of the constants used in the Model classes.
 */

public interface ModelConstants  {


  public static final int widthPix = 7;
  public static final int heightPix = 210;
  public static final int percentPix = 2;

  public static final int maxDist = 51;
  public static final int maxPoolSize = 1000;
}
```