Visual Cleaning of Genotype Data

Jessie Kennedy[1], Martin Graham[2]
Edinburgh Napier University, UK

Trevor Paterson[3], Andy Law[4]
The Roslin Institute, University of Edinburgh, UK

Abstract

While some data cleaning tasks can be performed automatically, many more require expert human guidance to steer the cleaning process, especially if erroneous or unclean data is a product of relationships between entities. An example is pedigree genotype data: inheritance hierarchies in which the correctness of genotype data for any individual is judged on comparison to their relations' genotypes, as individuals should inherit DNA from their assumed ancestors. Thus, cleaning this data must consider the relationships between individuals; sometimes this means more data must be cleaned than first assumed, while in other situations it means errors across many individuals can be remedied by cleaning the data of a shared relation. Such judgements require a domain expert to hypothesise the effect changing particular data has on the wider data set.

Using a visualization tool with the ability to undertake what-if interactions can assist a user in correctly cleaning such data. We achieve this by closely coupling an existing pedigree visualisation technique, VIPER, with a genotype cleaning algorithm, and then develop necessary extensions to the visualization to allow interactive data cleaning. A comparative user evaluation with biologists shows the advantages of this visualisation design over an existing cleaning tool and we discuss the challenges in the design of visual cleaning tools in which errors may be transitive.

Keywords: Pedigree, genotypes, data cleaning, user evaluation.

Index Terms: Applied computing~Population genetics; Human-centred computing~Visual Analytics.

1 Introduction

Data cleaning is a vital step in improving data accuracy and usability, with visual data cleaning being the use of visualisation tools to support data cleaning activity.

Pedigree genotypes are one example where clean data is particularly vital as analysis tools for these data sets break down when fed with erroneous data. A pedigree genotype is an animal breeding graph where each animal has up to two known parents and zero or more known children. On top of this graph is layered multivariate information, specifically genetic *markers*. In the most common scenario, these markers come in the form of SNPs (Single Nucleotide Polymorphisms) represented as a pair of letters from the (A, C, G, T) base alphabet. Mendelian inheritance states that each individual's value for a SNP marker should inherit one letter each from its parents' values for that marker. If that doesn't happen then there is an inconsistency, indicating either an error in the data set or, rarely, a mutation in the individual.

Additionally, there is frequently missing genotype information for some individuals, though this may be imputed to a limited, discrete number of choices. Due to this missing data, the cause and effect of an error may occur at different points, and thus fixing the manifestation does not always fix the cause of an error. One error may cause multiple difficulties further down the graph, and in turn many apparently separate errors can be solved by changing a shared, inherited piece of information.

In this paper, we begin by describing related work into visual data cleaning and the specific problem of transitive data masking. We then take an existing visualization tool, VIPER [1], with which biologists can view errors within pedigree genotype data, and fully integrate it with a genotype checking algorithm. We extend the VIPER interface with visual cleaning functionality for removal of suspect genotypes, markers and individuals, and for breaking troublesome pedigree relationships, so that it becomes an interactive data cleaning application rather than merely an error-viewing interface. We then describe and discuss a comparative user study with a group of biology professionals between this extended tool and an existing cleaning tool, GenotypeChecker [2]. Finally, we reflect on the conclusions we draw from this work.

2 Related Work

The work described in this paper intersects two main areas of visualisation research. Firstly, primarily as a visual data cleaning tool it has commonalities with the recent direction in visualisation research towards *data wrangling* [3]. This is the visually-aided process of transforming raw or problematic data into a more usable form, which includes data cleaning. Past work in this field has tackled the problems from one of two directions, either as a generalisable process on a datatype or, secondly, through a specifically targeted tool for a particular domain. As an example of the former approach, the Wrangler application [4] is a general framework that allows users to visually specify transformations on tabular data sets to improve data quality, which has various advantages over manual cleaning such as re-use of scripts and speed of operation. Of the latter approach, [5] developed a more specific application to clean and disambiguate data in social networks i.e. to merge instances that referred to the same person (e.g. "j.smith" vs. "John Smith") or on the other hand to mark similar names as distinct entities.

Our work fits the latter approach as we have a well-specified data domain with a central over-arching task: remove the errors from the pedigree genotype data structure. In their further work, [3] outlines a number of directions in which data wrangling can be applied, and amongst these are visualising raw data, removal of errors, and visualization of missing data, all of which apply to our situation. There is no requirement for any structural data transformation, as in our case the output data necessarily has to be in the same format as the input data.

The second area of visualisation research our work touches on is the representation of missing and erroneous data. From our perspective, the concepts of missing and erroneous data, though collected under the umbrella term of 'uncertainty data' and in turn 'uncertainty visualization' [6], can be seen as having somewhat

[1] e-mail: j.kennedy@napier.ac.uk
[2] e-mail: m.graham@napier.ac.uk
[3] e-mail: trevor.paterson@roslin.ed.ac.uk
[4] e-mail: andy.law@roslin.ed.ac.uk

opposing attributes. Erroneous data is present and definitely wrong, whereas missing data is absent and therefore we cannot know if it is wrong or not (or if any data exists). In our particular case we try to remove error data at the cost of introducing the lesser evil of missing data.

The challenge of visualizing missing data is not as troublesome here as it is for many other applications for two reasons: one relating to the data type and another to the layout technique. Firstly, our missing data, unlike in many other data sets, can be imputed to a limited and discrete number of choices. As an individual's values are inherited from their parents, we know the possible range that those values should take. Even with no ancestral data to work on, we know the values will be limited to a pairing of the (A,C,G,T) alphabet. Secondly, as [7] recognised, visualizations like scatterplots or graphs have difficulties with missing data as the data values are encoded in the layout's spatial attributes; as such missing data often translates directly to misleading or missing representations (e.g. replacing missing values in parallel coordinates with zero or minus one gives the impression the missing value still has a definite place on the coordinate, the alternative is to leave gaps in the representation). In our prototype though the missing data is layered on top of an existing visual representation. The pedigree visualization and table views described later always have placeholders for the individuals in the pedigree and coding for missing data can simply be overlaid on top.

Visualising erroneous data is not a topic that has been explored extensively in the visualization literature. There has been research into specific techniques for uncertainty visualization [8] and what uncertainty itself entails [9], but it tends to concentrate on how to communicate uncertainty parameters and data provenance rather than looking at how to communicate data that is clearly wrong. However, visualisations often act as ad-hoc data quality tools through serendipitous error discovery: Sánchez *et al* [10] discuss starfield (scatterplot) visualisations that uncover data quality issues by revealing erroneous outliers, and visualization researchers routinely discover errors in data during the course of application development, but the default state for the vast majority of visualisations is to assume that the data is correct. Often the only visualisations to explicitly expect incorrect data are those built for the purpose of cleaning it or those designed to assess data validity. An early example here is Visage [11] which allowed a user to purge a dataset of erroneous outliers after viewing. More recently, [12] explored how correlating data graphically in a table could reveal data quality issues, though they didn't include a method to then edit such data.

2.1 VIPER

Table 1. A simple matrix of individuals by markers. Some cells (genotypes) are coloured to indicate inheritance error.

	M1	M2	M3	M4	M5	Ind Errors
Ind 1	🟥					1
Ind 2		🟥	🟥		🟥	3
Ind 3			🟥			1
Ind 4			🟥			1
Ind 5						0
Marker Errors	1	1	3	0	1	6

Table 1 is a small example of individuals plotted against markers so each cell displays the error state of an individual genotype. Totalling the errors by column and row gives the errors per individual and per marker. While this tabular representation gives a useful overview for some tasks (here it reveals Individual 3 and Marker 3 both have a high proportion of errors), it doesn't show a) the pedigree structure of the individuals, necessary for determining the source of errors, or b) a visually compact representation of the scale of marker sets present in real-world data. The biologists on this project initially envisaged data sets of up to 10,000 markers - however this figure then leapt to 250,000 and will only rise as technology advances. Similarly, node-link representations are useful for showing simple pedigree structures, as exampled by one of the figures in this paper, but edge crossings proliferate if we look above a small number of relationships, making the view increasingly intractable for users [13].

In a previous response to these difficulties, a novel Java-based visualisation for representing errors in pedigree genotype data was developed - VIPER [14], upon which we build extensions for data masking. It combined a number of elements in a multiple view display, including histograms, tables and a novel pedigree representation. VIPER displays successive inter-generation relationships as in Figure 1: offspring are displayed as a layer of hexagonal glyphs sandwiched between layers of male and female parents (thus termed a 'sandwich' view). Within each glyph, triangles act as stylised up and down arrows to communicate the degree of errors to father (sire) and mother (dam) and the rectangle in-between shows errors that cannot be traced to either (novel alleles). A colour hue is used to convey the presence of erroneous data points, with the colour intensity representing the degree of error, thus large marker sets can be accommodated. Typically, grouping each sandwich by its male parents reduces clutter and as females rarely reproduce with multiple sires per generation, the dam glyphs only need repeating rarely.

A pair of tables gave detailed error information for every individual and marker in the data set, split into errors with sire, dam, and novel alleles, along with a total error count column. The tables and sandwich view were linked for reciprocal highlighting of selected individuals, whilst selecting a marker from its table put the application into single marker view, showing only errors for that marker. Two histograms were also used to show error distributions in the data set (termed 'errorgrams'): one showing error distribution for markers and one for individuals. This allowed users to see an overview of errors in the data and revealed high-level trends. In general, uncleaned pedigree data tended to have a) some individuals with lots of errors, suggesting a problem with those individuals, and b) some markers with lots of errors, again suggesting a problem with those markers. The challenge was then to further develop VIPER from an error visualisation into a data-cleaning tool.

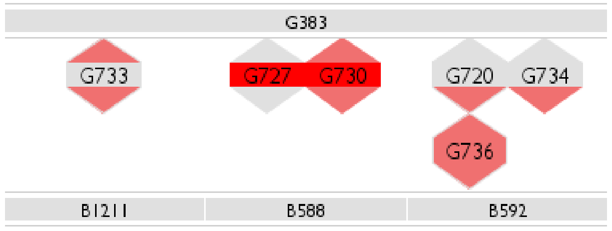


Figure 1. In VIPER, offspring are represented by hexagonal glyphs, coloured by error count: top triangle = no allele inherited from sire, central rectangle = novel alleles detected (present in neither parent), bottom triangle = no allele inherited from dam.

3 Pedigree Genotype Cleaning

The goal of cleaning error-ridden data sets is to remove errors. However to actually *correct* errors is usually an intractable task;

we might know which values have errors, but we do not know what the correct value should be. This is exacerbated in pedigree genotypes by the presence of missing data which shift the reporting of errors by genotype-checking algorithms away from their source in the pedigree. In fact, consistency checking of non-trivial pedigrees has proven to be NP-complete [15].

Removing errors within this domain is thus achieved by *masking* errors; manipulating the data by marking certain data points as having unknown values so explicit discrepancies between parents and offspring are removed. In short, we say "we don't know what it is, but it's definitely not that". The genotype-checking algorithm then treats these user-defined unknowns identically to missing values in the raw data: it infers the complete set of acceptable possible values based on analyzing the closest individuals in the pedigree that have present and correct values for the same marker. Thus, as well as discovering where errors occur, the algorithm uses structural information to help fill in the blanks. This is distinct from statistically-based methods that fill in gaps in multivariate data sets [16, 17], yet achieves the same ends.

In this particular problem domain, this is an acceptable solution, as downstream analysis tools can cope with missing data by inferring possible values based on Mendelian inheritance but are broken by incorrect data, or worse, process the incorrect data to produce incorrect findings that may be damaging further down the line. The effect of such bad data has previously been recognised in animal conservation [18] and human health research [19].

3.1 Fundamental Cleaning Operations

The original visualization [14] had two errorgrams that supported some simple filtering operations. One allowed removal of markers above a given error count, and another supported a visual filtering on individuals (greying out individuals with less than a given error count). While this provided opportunities to study basic error patterns, cleaning the data requires a more involved set of interactions, necessitating an iterative communication with the underlying data cleaning algorithm rather than just relying on an initial supply of error count information.

Investigating with the biologists the most effective way to remove errors revealed that they would first remove markers with many errors, then mask the most erroneous individuals and then iteratively work through the remaining errors, leading to a set of three fundamental cleaning operations:

1. Mask a marker
2. Mask an individual
3. Mask a genotype (pairing of marker and individual)

3.2 Cleaning Bad Markers

Markers are independent of each other, such that removing one marker does not affect the error state of other markers, making it cognitively the easiest operation and a natural starting point for cleaning. Therefore our first interactive cleaning task was to support the masking of bad markers, which required hooking up the existing errorgram slider to the genotype checking algorithm through an existing API (as would all subsequent masking tasks.)

This proved to be an effective technique for the biologists who then asked to visually exclude markers with only a few errors as they were unimportant in the overall cleaning process. It was initially proposed to replace the errorgram slider with a range slider to set two values (low & high), however the biologists stated that ignoring these markers for the time being was conceptually different from masking the most erroneous markers and therefore another errorgram was added to allow them to control the display of low-error markers. This new errorgram was also filtered so that the high-error markers masked in the first errorgram were not shown, making it a zoomed view of the first errorgram, as shown in Figure 2.

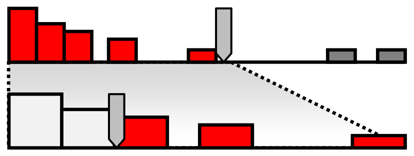


Figure 2. The second marker errorgram is an expanded view of the unfiltered portion of the first errorgram. In this new errorgram markers are visually excluded with the slider but not excised from the data set, unlike the first errorgram.

We also set both marker errorgrams to always reflect the data as it was upon loading. This gelled with the biologists' approach to cleaning the data: masking the worst markers first, and then tackling the more involved and smaller units of data such as individuals and genotypes, so revisiting the marker errorgram was not envisaged as necessary.

3.3 The Transitive Error Problem

The expectation was that, as with the markers, we could use an errorgram to mask the most error-prone individuals. However, this proved naïve as it did not account for the major problem with errors in pedigree genotype data sets – namely their transitive nature due to the relationships of individuals within the pedigree.

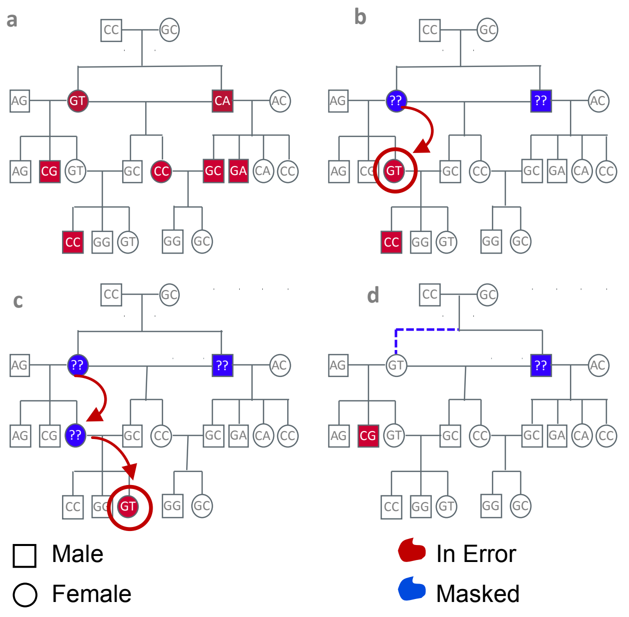


Figure 3. The transitive effect of masking an individual genotype. Given the errors shown in red in part a) we mask the two error reporting individuals in the second generation in b). This removes many errors but also produces a new transitive error (circled). In part c) we mask this new error but it is just pushed down the pedigree (again circled). In part d) we alternatively break one of the pedigree relationships at the top (the dashed line), removing many existing errors and, crucially, we don't introduce new errors.

Figure 3 demonstrates a simple example of this effect. Figure 3a) shows a small single-marker pedigree with errors shown in red. Figure 3b) shows the result of naively masking the two individuals in error highest up in the pedigree. Here, due to masking a female (now labeled '??' in blue), errors in two of its children are removed, however another child (GT - circled) now cannot match its 'T' allele with either parent. With its mother now masked, it tries to reconcile with its maternal grandparents (the two individuals in the top generation) but neither carry the 'T' allele as well and thus it now reports an error. When in turn this individual is masked in Figure 3c) the problem is just pushed down again to the next generation with another circled individual containing a 'T' allele now reporting an error.

The underlying error in Figure 3 was between the first and second generations i.e. the lack of a T allele, but trying to mask the problem pushed the error down the pedigree and another subsequent masking just pushed it even further down. This is in fact the essential problem of transitive errors in pedigree structures: errors may be chased blindly down to the bottom of the pedigree, masking otherwise useful data along the way, and moving further from the source of the error at the same time.

Therefore we realised the necessity for an operation that, rather than masking individuals or genotypes, masked the *relationship* between a child and parent. In terms of seeing the pedigree as a graph, this is like masking an edge between two nodes rather than masking information carried in the node itself. The effect of this is shown in Figure 3d) where the relationship from the erroneous individual in a) to its parents is removed instead of the individual being masked, which results in no new errors being caused. One of its offspring still remains in error, but this can be directly masked itself. This results in a fourth fundamental masking operation for cleaning the data set:

1. Mask a marker
2. Mask an individual
3. Mask a genotype (pairing of marker and individual)
4. *Break a child/parent relationship in the pedigree.*

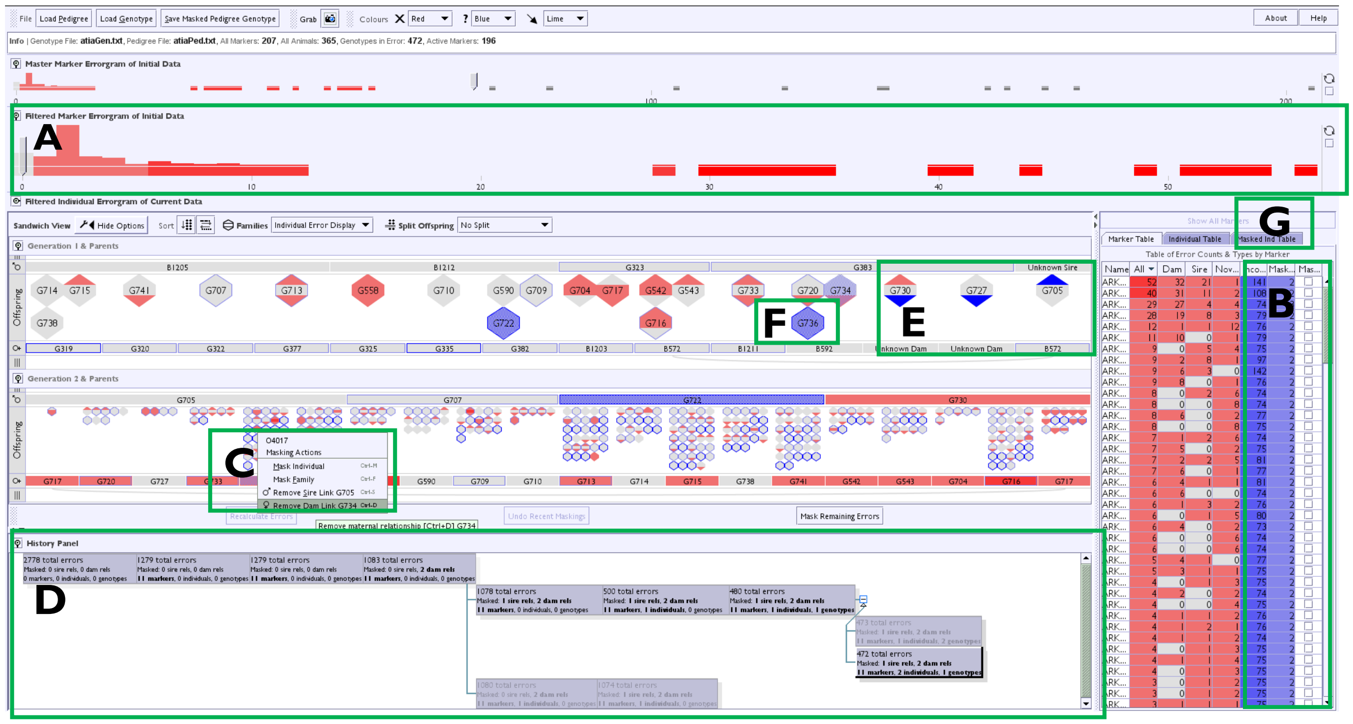
Of the four operators, 1 and 4 cannot add further errors into the data, but 2 and 3 do introduce this possibility.

3.4 Cleaning Bad Individuals & Genotypes

On a practical note, transitive errors meant error totals for related individuals would increase or decrease when other individuals were masked, making an errorgram and slider mechanism for masking individuals inelegant as individuals would sometimes jump from one side of the slider 'cut-off' to the other, and the shape of the histogram would also markedly change during the process. It would also change the shape of the marker histograms if we populated them with the current error counts, which formed part of our reasoning to set the marker histograms to show only the initial state.

To therefore support cleaning operations beyond the broad filtering of the marker errorgram, extra functionality was incorporated into the existing visualization. In both the sandwich view and the individual table we added a right-click context menu to individuals which gave a selection of masking operations. E.g. in the full marker mode, bringing up the menu on an individual would show the option to mask or unmask that particular individual - the second operation in our list of four. In single marker mode (where errors related to only one particular marker are shown), it would offer the choice to mask or unmask the genotype for that individual - the third of the four operations.

There are also options to break or restore the pedigree relationships of an individual to its parents (the fourth basic masking operation). In the sandwich view the additional option to mask all individuals in an entire family is provided as a shortcut to doing the same masking operation on each individual. Again, in the single marker mode, the same action is available but instead of masking individuals it masks sets of genotypes. To allow markers



Info | Genotype File: atiaGen.txt, Pedigree File: atiaPed.txt, All Markers: 207, All Animals: 365, Genotypes in Error: 472, Active Markers: 196

2778 total errors
Masked: 0 sire rels, 0 dam rels
0 markers, 0 individuals, 0 genotypes

1279 total errors
Masked: 0 sire rels, 0 dam rels
11 markers, 0 individuals, 0 genotypes

1279 total errors
Masked: 0 sire rels, 0 dam rels
11 markers, 0 individuals, 0 genotypes

1083 total errors
Masked: 0 sire rels, 2 dam rels
11 markers, 0 individuals, 0 genotypes

1078 total errors
Masked: 1 sire rels, 2 dam rels
11 markers, 0 individuals, 0 genotypes

500 total errors
Masked: 1 sire rels, 2 dam rels
11 markers, 1 individuals, 0 genotypes

480 total errors
Masked: 1 sire rels, 2 dam rels
11 markers, 1 individuals, 1 genotypes

473 total errors
Masked: 1 sire rels, 2 dam rels
11 markers, 1 individuals, 2 genotypes

472 total errors
Masked: 1 sire rels, 2 dam rels
11 markers, 2 individuals, 1 genotypes

1082 total errors
Masked: 1 sire rels, 2 dam rels
11 markers, 1 individuals, 0 genotypes

1074 total errors
Masked: 1 sire rels, 2 dam rels
11 markers, 1 individuals, 0 genotypes

Figure 4. The extended VIPER interface with newly introduced data cleaning elements highlighted. Filterable errorgrams (A) (with the bottom errorgram being a zoomed view of the top errorgram), expanded individual and marker tables showing incomplete and masked data points, along with individual marker masking (B), context menus for masking in the sandwich display (C) and a history tree for reversing masking decisions (D) - all connected to an underlying genotype checking algorithm. Masked data (in blue) is represented in various ways: masked pedigree relationships are indicated by solid triangles within glyphs (E), masked individuals and genotypes are indicated by stippling across an individuals' glyph (F). Masked individuals' details are also accessible in an extra table (G).

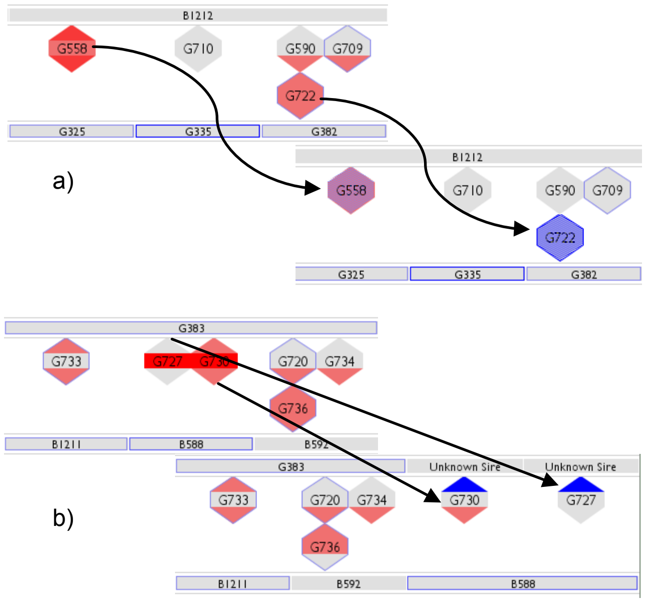


Figure 5. How masked data is represented in the visualisation. a) A full individual masking on G722 stipples the whole hexagon blue, and a single genotype masking on G558 stipples a lighter blue over the remaining errors shown in red. b) Masked relationships (G727 & G730) are shown as solid blue triangles pointing in the direction of the parent gender they have disassociated from, and they are realigned with new, separate "unknown sire" (or dam) parents elements. The presence of originally incomplete data is shown as a blue border around individuals.

to be removed singly, or be removed independently of the errorgram filtering, a column of checkboxes was added to the marker table to allow masking or inclusion on a marker-by-marker basis. This proved to be invaluable when dealing with sex-linked marker problems. Lastly, an extra table that lists all current individual-level maskings (masking operations 2 & 4) was provided to give users a place to undo such operations. Figure 4 shows these additions in the interface, along with examples of individuals having undergone various aspects of masking.

The biologists stated that with these fine-grained maskings they wanted control of when the checking algorithm ran, so that the effect of several logically-related maskings could be tried together. This approach also aided the application's responsiveness as recalculating the errors is CPU-intensive, and also helped when masking pedigree relationships, as re-layouts of individuals under dummy parents would not happen until users were prepared to view the effects of recalculation. So, automatic recalculation was kept only for marker errorgram interactions which rarely involve small numbers of maskings in any case.

3.5 Masking Representations

Initially, masking simply called the error checking algorithm for recalculation and redisplayed the new error output in the visualisation. However, the biologists stressed the importance of keeping track of what data they had masked during the cleaning process, and so a visual indication of masking was deemed necessary. As discussed, masking of errors is equivalent to setting values to unknown, therefore for visual consistency we decided to show masked data in the same hue as used for indicating missing data. Unknown data can therefore be thought of as divided into incomplete - missing from the initial data set - and masked - data deliberately hidden by the user to remove errors.

Accordingly, this data was added to the marker and individual tables as two separate columns, masked and incomplete, and within the sandwich display deliberately masked individuals and genotypes were indicated with a stippled infill of the individual with the current "missing" hue - missing data was already indicated with a border of the same colour around the individual, so infilling served to distinguish the two conditions.

Broken pedigree relationships were displayed as solid infills of either the up or down triangles in an individual's glyph, indicating in which direction(s) the pedigree relationships had been broken.

Figure 5 shows examples of how masking is conveyed within the sandwich view, both genotype and complete individual masks and the breaking of pedigree relationships between individuals.

3.6 Visual History

With such an explorative method of error removal, it is imperative that users can undo masking operations and then restart exploration from a prior position. Such a facility beyond a simple single undo/redo is known as a history function, and in this particular case a branching history function. After any user interaction that changes the set of masked markers, individuals or genotypes, a new history state composed of the current states of these values is saved and appended to this history.

The history tree's visualization is shown at the bottom of Figure 4, similar to that in [20], with each history state being represented by a node showing the overall error counts and different maskings applied at each state. The details in each node are kept as textual, rather than visual, representations to keep to a design choice that the only colour in the interface should represent the current state of erroneous and missing data. The path from the start of the cleaning process to the current error state is highlighted in the tree structure, and conversely, the greyed-out nodes represent masking attempts by the user that were then revoked, and act as a visual memory aid for hypotheses that have been tried and rejected.

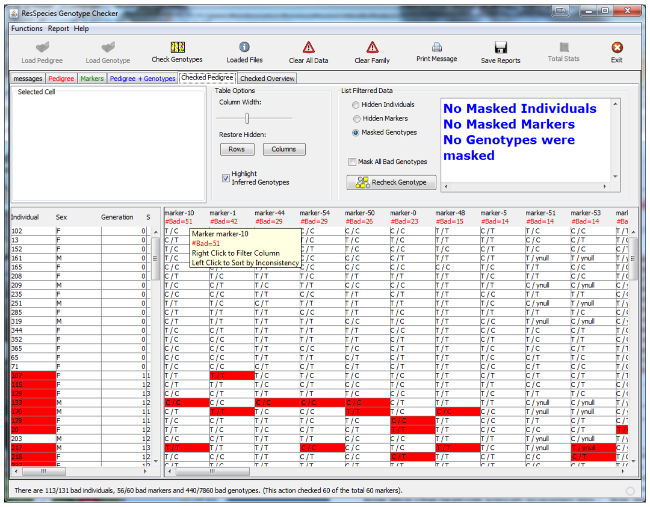
To return to a particular state the user simply clicks on the corresponding item in the history tree, and the saved maskings are reapplied to the data set. Without such a facility the onus falls on the user to remember what actions must be redone, not ideal when they are focusing on a challenging problem with the data. Indeed, the information-seeking mantra [21] includes the idea of history explicitly.

The inclusion of an interactive history is designed to encourage an explorative approach to masking data, as users have the option to undo the maskings reflected in the interface to a previous state. Users can form hypotheses as to what actions would clean certain portions of the data set, carry out the actions to put those hypotheses into practice, and safely return to a previous known state if the presupposed effect on the error count did not occur.

4 User Study

With the extended version of VIPER now incorporating data cleaning functionality it was decided to run a comparative empirical evaluation against an existing data cleaning tool for pedigree genotypes: GenotypeChecker (GC) [2]. The aim was to find out what objective and subjective differences occurred when both applications were used by expert biologists, and which of the features of VIPER, both the existing sandwich representation developed in [14] and the new cleaning interactions outlined in this paper, differed in respect to GC.

GC is a Java application that shows a table-based display of markers (columns) by individuals (rows) in which genotypes that cannot be the product of an individual's assumed parents are coloured, as shown in Figure 6. As with VIPER, individuals, markers and genotypes can be masked and the effect on the remaining errors then recalculated. Rows (individuals) can be sorted on the basis of error count, gender, name or generation



No Masked Individuals
No Masked Markers
No Genotypes were masked

Marker marker-10
#Bad=51
Right Click to Filter Column
Left Click to Sort by Inconsistency

There are 113/131 bad individuals, 56/60 bad markers and 440/7860 bad genotypes. (This action checked 60 of the total 60 markers).

Figure 6. The GenotypeChecker interface.

whereas the column ordering is fixed. However, beyond a simple ability to list the children and parents of a given individual, it cannot represent or explore the relationships in a pedigree, which is seen as a drawback to finding, understanding and fixing errors. As GC's underlying checking algorithm is the same as VIPER's, any differences in performance or preference should be due solely to the different interfaces.

4.1 Participants

Eleven biology professionals from The Roslin Institute, unconnected with this work, volunteered to take part in the evaluation for which we offered a prize draw of a £50 voucher. All were qualified to degree level, eight had PhDs and four had at least 25 years experience in their field. They variously described themselves as geneticists, bioinformaticians or biologists.

All but one stated they used pedigree data in the course of their work, though that individual stated they understood Mendelian inheritance. Three stated they had infrequent experience of previously using GC when they needed to clean pedigree data, but for most work used pedigree data they assumed was consistent. Three others stated they had seen some form of visually represented pedigree data, and the other five said they'd never used any application that showed a pedigree visually.

To explain the lack of visualisation exposure, raw pedigree data is often just simple "sire-dam-child-gender" format CSV files that are typically loaded into the de facto data manipulation tool of these biologists: Microsoft Excel. One of the developers of GC stated the reason that tool was so strongly based on a table-based representation was due to the biologists' familiarity with Excel.

4.2 Test Procedure

The experiment was a within-subject test where each participant was faced with 12 questions categorised into two principal tasks, one that posed questions about the interface and its representations, and another that asked them to perform cleaning on the data; the rationale being that before masking can be performed on any data, the representation of that data must be understandable. The participants tackled these questions using both the GC and VIPER interfaces, with the order flipped between successive participants to counter learning effects across the user group. Before they answered the questions on each interface they were given a demonstration and allowed to experiment and clarify issues regarding the interface.

The twelve questions (listed in Table 2, along with the correct results per question) in the two tasks were a mixture of finding/masking data that did not require any pedigree exploration and those that would require a user to analyse individuals in the context of their family members - e.g. a question such as 'which individual has the most errors overall' required no pedigree exploration, whereas another question such as 'Marker-32 has 5 errors - remove the three errors that occur in the same family' would require the user to relate the genotypes to the pedigree. Of the twelve questions, 5-8 inclusive were considered not to need pedigree exploration.

The questions in the masking task in particular were organized such that it resembled a typical cleaning scenario: bad markers would be removed first, followed by the masking of problematic individuals. After that, sex-linked markers and genotypes would be targeted for masking.

The tests were scored with each question receiving a Boolean right/wrong mark according to whether the correct answer was found, giving each user a total out of 12 for the tasks on each interface. Further, after the users completed the tasks on both interfaces an attitude scale with 12 items was used to elicit user preferences between the two interfaces for given operations.

The data set used was a relatively small clean dataset of 131 animals measured across 59 markers, making for a total of 7,729 genotype data points, into which errors of varying types had been introduced for the participants to discover and clean. The tests were performed at the Roslin Institute on a laptop with a 19-inch widescreen display and each user was scheduled to take

Table 2. Task questions, and correct answers per question by interface. VP = Viper, GC = Genotype Checker.

Task 1 – Interface Representation	**GC**	**VP**
1. How many sires are there in the original generation?	**5**	**10**
2. What is the biggest family in the pedigree?	**2**	**5**
3. How many full siblings does Individual 39 have?	**10**	**7**
4. How many different dams does the sire Individual 203 produce offspring with?	**4**	**8**
5. Which individual has the most errors overall?	**9**	**9**
6. Which marker has the most errors overall?	**10**	**10**
Task 2 – Data Cleaning (Masking)		
7. Mask all markers that have more than 30 errors.	**9**	**8**
8. Mask the individual that reports the most remaining errors.	**6**	**5**
9. One of the three markers 5, 10 and 50 is sex-linked and contains only errors to sires and novel alleles. Find it	**3**	**10**
10. Marker-4 has 10 errors. Remove these errors by masking as few individual genotypes as possible.	**5**	**4**
11. Marker-32 has 5 errors. Remove the three errors that occur in the same family.	**6**	**9**
12. The offspring of one of sire 203's families has many errors to 203 across all markers. Identify this family and mask the offspring in the family.	**2**	**2**

approximately one hour overall to complete.

4.3 Results (Performance)

The results per user, shown in Table 3, revealed that when all the answers were considered there was no significant difference between the interfaces (two-tailed t-test, p=0.12). However, when only the eight questions that required users to explore within the pedigree were analysed, a significant effect came to light (two-tailed t-test, p=0.03) showing that users obtained more correct answers using VIPER than with GC for these questions.

Table 3. Correct answers per user by interface, and by subset of pedigree-biased questions.

	All Q's, correct answers per user										
VP	7	11	8	6	6	9	7	4	11	10	8
GC	7	10	2	6	3	6	8	8	11	6	4
	Q1-4 & 9-12, correct answers per user										
VP	5	7	5	3	4	6	4	2	7	6	6
GC	4	6	1	2	2	2	5	4	7	2	2

There was no significant correlation between the time taken and correctness for either tool, and nor was there a significant difference in time taken by users between the two tools.

4.4 Results (Preference)

From the post-test questionnaire we obtained the results shown in Table 4. They showed a strong subjective preference for VIPER overall; all median values across the set of items were either a preference for VIPER or 'No Preference'. Performing a Wilcoxon signed rank test on the summative values in the questionnaire showed a statistical preference (p<0.05, two-tailed) for VIPER compared to the 'No Preference' rating on the scale. The same test on the questions individually showed the same preference for VIPER in questions 1-3, 6-9 & 11, with no significant preference in the remaining questions.

4.5 Observations

In qualitative terms it was clear to see where users were led astray at some points in VIPER, the most common was an inability to remember being in the single marker view, and when asked to judge families on errors across the full marker set they would give an answer based on one marker's errors. This led to a particularly poor performance by all users with question 12 (though it was relatively no worse than the answers achieved with GC). Furthermore, the fact that roughly a third or more of questions were either skipped or answered incorrectly with both VIPER (45/132) and GC (61/132) reflects that cleaning flawed pedigree data is a difficult and skilful process; even though our 11 test subjects were professional biologists with an understanding of pedigrees most of them had not attempted any previous cleaning of pedigree data.

What was interesting to note was how conditioned some users were to a spreadsheet mode of working. Even though the 'sandwich' visualisation was explained and demonstrated, took up the majority of VIPER's screen space, and is decidedly table-esque (it is essentially a nested table) itself compared to most other pedigree visualisations, the first instinct of some users was to head over to the tables of marker and individual error information and try and answer the questions from there. Only when this proved fruitless did they investigate the sandwich view. Some users stated GC's "mask row/column" functionality for individuals and markers was something they were accustomed to from using Excel. This 'table-centredness' has been the case with other groups we have worked with, such as taxonomists, who replicated tree structures within a spreadsheet. Sairaya *et al* [22] also report that the bioinformaticians they studied performed extensive data pre-processing and cleaning in Excel before they used further visualisation tools on the data, so this attachment is not just unique to our users (and also indicates that when users visualize data, they expect it to be error-free.)

5 Conclusion

We have extended an existing pedigree genotype visualization tool, VIPER, with interactive cleaning capabilities, and then evaluated it against a table-based pedigree genotype cleaning application. Our evaluation revealed a significant subjective preference for VIPER's family-centred visualization as opposed to GC's table-based display, and reflected a common finding [23, 24] with visualization evaluations in that subjective user preference for interfaces tends to be more strongly pronounced than the effect found in objective task performance.

Another major influence on users' performance with interfaces is the similarity to other tools they commonly use, with one tree browsing evaluation [25] showing users performed better with the most familiar widget. Sedlmair *et al* [26] in their discussion of testing InfoVis applications in large companies also note the effects of users' attachment to conventional tools. It may be no performance difference was found in this study as while GenotypeChecker reflected the biologists' familiarity with

Table 4. Frequency distribution of user preference answers - (1 = Strongly Prefer Viper, 2 = Slightly Prefer VIPER, 3 = No Preference, 4 = Slightly Prefer GenotypeChecker, 5 = Strongly Prefer GenotypeChecker). Medians asterisked and shown in bold italics.

Preference Item	1	2	3	4	5
1. Finding structural information on a pedigree	*7	1	2	1	0
2. Finding descendents of an individual	*8	2	0	1	0
3. Finding ancestors of an individual	*7	3	1	0	0
4. Finding error information on a single individual	4	1	*1	4	1
5. Finding error information on a single marker	3	*3	2	3	0
6. Distinguishing between different types of error	*7	2	2	0	0
7. Tracing errors to a shared parent	*8	0	2	1	0
8. Finding error information on a single family	*7	1	2	1	0
9. Comparing errors between related families (one shared parent)	*8	1	1	1	0
10. Masking errors	1	2	*4	3	1
11. Overall understanding of errors	5	*1	4	1	0
12. Overall ease of use	5	*2	3	0	1

spreadsheets it was at the cost of not representing the underlying pedigree structure, whereas VIPER represented the structure of the underlying data more faithfully but users were less accustomed to its visual style. Both these points, along with the fact that pedigree cleaning is a cognitively tasking operation, perhaps explain why roughly 30% of the questions in both conditions were answered incorrectly, showing that there is still room for further research into pedigree cleaning interfaces.

During the course of this work, it has become clear that it is crucial to understand the nature of the errors that are to be cleaned. In the case of pedigree genotypes the error is a function of the relation between two entities (i.e. incompatible genotypes), rather than of a particular entity in isolation. In such situations, removal of existing errors can cause new errors to appear, or remove other related errors at the same time. Interfaces for cleaning data where such transitive effects occur must be designed so that, optimally, these dependencies are clear to the users, and at the very least do not confuse the users. For instance, when cleaning involves structural alterations to data, as with the relationship masking operation here, the consequent re-layouts of the data often need to be delayed until the user is ready to view the result. Similarly, a history function is vital to a hypothesizing and exploratory cleaning of such data. When the underlying cause of error is not known, often due to the propagating effects of error and missing data, then the ability to undo and return to previous states is crucial. These points are of relevance not just to animal pedigree data but to a larger family of ontological data sets where entity properties are inherited from other entities, or when an object's correctness can only be judged in comparison to other related entities.

ACKNOWLEDGEMENTS

We would like to acknowledge the BBSRC (Biotechnology and Biological Sciences Research Council) for funding this project (project refs: BB/H023879/1 & BB/H023909/1), and the biologists at the Roslin Institute for their valuable time in contributing to the evaluation. Software is available for download at http://www.bioinformatics.roslin.ed.ac.uk/viper/.

REFERENCES

[1] T. Paterson, M. Graham, J. Kennedy, and A. Law, "Evaluating the VIPER pedigree visualisation: detecting inheritance inconsistencies in genotyped pedigrees," Proc. 1st IEEE Symposium on Biological Data Visualization, pp. 119-126, 2011.

[2] T. Paterson and A. Law, "GenotypeChecker: An interactive tool for checking the inheritance consistency of genotyped pedigrees.," Animal Genetics, 42(5):560-562, 2011.

[3] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. van Ham, N. Henry-Riche, C. Weaver, B. Lee, D. Brodbeck, and P. Buono, "Research directions in data wrangling: Visualizations and transformations for usable and credible data," Information Visualization, 10(4):271-288, October 2011.

[4] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer, "Wrangler: Interactive Visual Specification of Data Transformation Scripts," Proc. ACM Human Factors in Computing Systems, pp. 3363-3372, 2011.

[5] M. Bilgic, L. Licamele, L. Getoor, and B. Shneiderman, "D-Dupe: An Interactive Tool for Entity Resolution in Social Networks," Proc. IEEE Symposium on Visual Analytics Science and Technology, pp. 43-50, 2006.

[6] H. Griethe and H. Schumann, "The Visualization of Uncertain Data: Methods and Problems," Proc. Simulation und Visualisierung 2006 pp. 143-156, 2006.

[7] C. Eaton, C. Plaisant, and T. Drizd, "Visualizing Missing Data: Classification and Empirical Study," Proc. IFIP International Conference on Human-Computer Interaction, pp. 861-872, 2005.

[8] J. Sanyal, S. Zhang, G. Bhattacharya, P. Amburn, and R. J. Moorhead, "A User Study to Compare Four Uncertainty Visualization Methods for 1D and 2D Datasets," IEEE Transactions on Visualization and Computer Graphics, 15(6):1209-1218, November 2009.

[9] M. Skeels, B. Lee, G. Smith, and G. G. Robertson, "Revealing uncertainty for information visualization," Information Visualization, 9(1):70-81, 2010.

[10] J. A. Sánchez, M. B. Twidale, D. M. Nichols, and N. N. Silva, "Experiences with starfield visualizations for analysis of library collections," Proc. SPIE Visualisation and Data Analysis, pp. 215-225, 2005.

[11] M. Derthick, J. Kolojejchick, and S. F. Roth, "An Interactive Visualization Environment for Data Exploration," Proc. Third International Conference on Knowledge Discovery and Data Mining, pp. 2-9, 1997.

[12] W. A. Malik, A. Unwin, and A. Gribov, "An Interactive Graphical System for Visualizing Data Quality - Tableplot Graphics," Proc. 11th IFCS Biennial Conference - Classification as a Tool for Research, pp. 331-339, 2009.

[13] H. C. Purchase, R. F. Cohen, and M. James, "Validating Graph Drawing Aesthetics," Proc. Graph Drawing, pp. 435-446, 1995.

[14] M. Graham, J. Kennedy, T. Paterson, and A. Law, "Visualising Errors in Animal Pedigree Genotype Data," Computer Graphics Forum, 30(3):1011-1020, June 2011.

[15] L. Aceto, J. A. Hansen, A. Ingólfsdóttir, J. Johnsen, and J. Knudsen, "The Complexity of Checking Consistency of Pedigree Information and Related Problems," Journal of Computer Science and Technology, 19(1):42-50, January 2004.

[16] J. L. Schafer, Analysis of Incomplete Multivariate Data. Chapman & Hall:London, UK, 1st ed, 1997.

[17] M. Ward, Z. Xie, D. Yang, and E. Rundensteiner, "Quality-aware visual data analysis," Computational Statistics, 26(4):567-584, December 2011.

[18] P. A. Oliehoek and P. Bijma, "Effects of pedigree errors on the efficiency of conservation decisions," Genetics Selection Evolution, 41, 14 January 2009.

[19] G. Brush and L. Almasy, "Pedigree and genotype errors in the Framingham Heart Study," BMC Genetics, 4(Suppl 1), December 2003.

[20] M. Derthick and S. F. Roth, "Enhancing data exploration with a branching history of user operations," Knowledge-Based Systems, 14(1-2):65-74, March 2001.

[21] B. Shneiderman, "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations," Proc. IEEE Visual Languages Symposium, pp. 336-343, 1996.

[22] P. Saraiya, C. North, V. Lam, and K. A. Duca, "An Insight-Based Longitudinal Study of Visual Analytics," IEEE Transactions on Visualization and Computer Graphics, 12(6):1511-1522, Nov/Dec 2006.

[23] K. Andrews and J. Kasanicka, "A Comparative Study of Four Hierarchy Browsers using the Hierarchical Visualisation Testing Environment (HVTE)," Proc. IEEE IV Conference, pp. 81-86, 2007.

[24] C.-A. Julien, J. E. Leide, and F. Bouthillier, "Controlled User Evaluations of Information Visualization Interfaces for Text Retrieval: Literature Review and Meta-Analysis," Journal of the American Society for Information Science and Technology, 59(6):1012-1024, April 2008.

[25] A. Kobsa, "User Experiments with Tree Visualization Systems," Proc. IEEE InfoVis, pp. 9-16, 2004.

[26] M. Sedlmair, P. Isenberg, D. Baur, and A. Butz, "Information Visualization Evaluation in Large Companies: Challenges, Experiences and Recommendations," Information Visualization, 10(3):248-266, 20 July 2011.