# Augmenting Data Files with Semantics for Coherency, Extensibility, and Reproducibility

John McCloud, Subhasish Mazumdar
Dept. of Computer Science
New Mexico Institute of Mining and Technology
Socorro, NM, U.S.A
email: {amccloud, mazumdar}@cs.nmt.edu

***Abstract*—Data files have traditionally been thought of as the input and output of programs, as well as their intermediaries. When the need for usage of data files by a diverse set of consumers was recognized, it was addressed primarily by the addition of metadata. This metadata is structured data, providing guidance regarding the use of the data. Unfortunately, this approach has proven inadequate for the myriad applications of today. We posed two questions of a very common and popular data file standard in bioinformatics. First, are the conclusions presented in such a file verifiable? Second, can one use the data to test for alternative conclusions? Our answers for both questions were negative. In this paper, we outline the problems we found and propose a remedy. While we have used bioinformatics as a case study, our results are more general.**

***Keywords–Knowledge Representation; Comprehension; Semantics; Bioinformatics.***

## I. INTRODUCTION

How are data files designed? Traditionally, they have been designed to handle the needs of a computer program that consumes it, or as the output of a program for use by either humans or yet another program.

Thus, software systems exist that will take some particular data format and operate on it with satisfactory result. Such programs have canned understanding of the data and how it should provide some solution desired by a user. However, users themselves may neither be able to examine that data or make any sense of it at all. These individuals *must* rely on some software in order to express the high-level concepts that data tries to represent.

This problem exists because data usually lacks a kind of description of its content or guidance about its use. The solution to this problem has been *metadata*, but what amount of metadata is adequate? It is not clear where to draw the line; usually, metadata is constructed in an ad-hoc manner. We suggest that the answer be tied to an ontology, i.e., data files should be designed in terms of concepts defined in an ontology of the domain.

We consider bioinformatics data files in the very popular SAM file format and ask two questions: does the data format support (a) extensibility and (b) reproducibility? The first question asks whether or not researchers can use the included metadata to pose queries that pre-existing dedicated software (SAMTools) cannot answer.

Such extensibility is desired today as funding agencies are now insisting on the availability of created data sets for use by others; this has the promise of increasing research impact — perhaps exponentially. Reproducibility of results is a cornerstone of research. This too, we feel, must be supported by the published data sets.

In this paper, we report how we obtained negative answers to both questions of the SAM format. Further, we identify the gap as the lack of declarative functions or algorithms.

The issue of computational solutions in research has attracted renewed attention [1]. Currently, there is no accepted, standardized way for both code and data to be included alongside journal publications, and even the role a journal should play in vetting this additional information is not clear [2].

Research on data provenance is a promising line of attack: it attempts to bring lineage in the form of input, output, and so forth, presented as some workflow with clear beginning and end. myGrid, for example, creates such workflows from actionable steps in a process that can be saved and shared for re-use [3].

The Collaboratory for Multi-Scale Chemical Science (CMCS) is similar to myGRID, except that it has philosophical differences on what metadata means, and is able to present papers into related workflows. This is beneficial, since the scientific narrative and explanation of some methodology can be referenced and associated with previous, peer-reviewed research [4].

The solutions in myGrid and others, however, rely on middleware (such as Taverna [5]) to build and re-use the XML workflow constructions. A more light-weight approach is found in ESSW [6], which ties metadata and provenance to regular software by wrapping scripts around them. A script must be built for each actionable step (from input-to-output), and the script-writer is responsible for each piece of lineage information in the resulting provenance record.

myGrid and CMCS are both particularly interesting solutions, since they attempt to pair process workflow activities with domain-specific concepts. Tying such meaning to provenance records allows for researchers to understand, query, and make use of data more effectively.

The problem with all of these approaches is that they typically work on entire sets of data or files. What is needed is a solution that deals with sections or components of data *within* files. We imagine using the Resource Description Framework (RDF) [7] to annotate directly *within* data files themselves. RDF is a desirable choice for annotation, because its pairing with ontologies is understood [8], and RDF has been already been useful in provenance research, such as myGrid.

 ISBN: 978-1-61208-415-2 

Furthermore, even with all this research, the entirety of *processes* that transform some initial data into some final result is not represented clearly in the data itself. While implementations of embeddable scripts into workflows are possible to view and maintain, internals of online web-services or programmed solutions can be opaque. The functions that are used in some computational solution may remain a black-box or exist in various languages that researchers of different fields will be unable to make sense of. What is needed is a description of these computational processes that is not tied to the computational language of some developer's choice. More clarity by way of *idealized* functions in the precise language of mathematics would alleviate this problem.

The rest of this paper is structured as follows. In the next section, we provide a short primer on biology, and subsequently, we examine the structure and contents of the SAM format. The following two sections investigate an appropriate formalization in the context of the questions of extensibility and reproducibility. We then offer concluding remarks.

## II. The Underlying Biology

Since we are using bioinformatics data, let us present a few related concepts from Biology. To motivate the reader, let us propose an experiment: we need to explore the differences between the DNA molecules obtained from two individuals of the same species: an 'experimental' individual affected by a disease versus a 'reference' individual without such a disease.

The DNA molecule is essentially a long sequence of nucleotide bases: adenine, guanine, cytosine, and thymine, represented by one of the letters *A, G, C*, and *T* respectively. For many organisms, a reference typically representing the entirety of its genetic material is already published on the web and is commonly referred to as a *genome*.

In the laboratory, there are methods of *DNA sequencing*, i.e., obtaining the exact order of those bases within a DNA molecule extracted from the 'experimental' individual. We will refer to such methods as *wet lab processes*. For organisms with short sequences, i.e., with length in the hundreds, it is a relatively straightforward process. Interesting organisms, however, have very long sequences (the human genome has over $3 \times 10^9$ bases). For those, the approach is to fragment the DNA randomly, replicate them, and determine the sequences for the short fragments (for which task there exist machines). The sequences obtained from the wet lab process are referred to as *reads*. Unfortunately, the wet lab methods are error-prone. Hence, the machines emit a probability of correctness (or quality) with each of the bases in the short sequence.

The remaining task is then to attempt to piece them together to infer the original, long sequence. This task is called *alignment*; it is a *dry lab process* in the sense that it is performed using computational methods.

Alignment (or *sequence alignment*, or *sequence mapping*) consists of *matching* each *read* from the experimental DNA against the pre-existing reference sequence. The task is challenging, since each obtained *read* could fit in many locations of the (huge) reference genome. The goal is to find the best match out of all possible matches.

As an analogy, one can think of sequencing as reading a book, attempting to commit its contents to memory, and then matching the memory against the actual contents of the book (reference). As one tries to recall from memory and match against the book, one will most likely realize that one lacks some words, mixes up the order of others, and so on; the result would be a series of 'close' matches, each with a notion of where it might properly fit in the actual book.

For example, suppose one wanted to align the sequence "TAAGCT" with the reference "TACGGT." There is more than one way they might align; two of them (as per the Needleman-Wunsch algorithm [9]) are shown in Tables I and II.

TABLE I. Sequence Alignment

| **Position** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **TACGGT** | T | A | _ | C | G | G | T | _ |
| **TAAGCT** | T | A | A | G | _ | _ | C | T |

One possible match between "TACGGT" and "TAAGCT" using Needleman-Wunsch for global alignment. Insertions and deletions are denoted by underscores.

TABLE II. Alternative Sequence Alignment

| **Position** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **TACGGT** | T | A | _ | C | G | G | _ | T |
| **TAAGCT** | T | A | A | _ | G | _ | C | T |

Another possible match between "TACGGT" and "TAAGCT" using Needleman-Wunsch for global alignment. Insertions and deletions are denoted by underscores.

In I, the character bases $TA$ match exactly; $A$ is an insert-error (requires it to be inserted into the reference string), $G$ is a mismatch error, then there are two delete errors (the next two characters of the reference $GG$ must be deleted), $C$ is a mismatch error, and finally $T$ is an insert-error.

Part of the task is to decide the best match. One way is to assign scores to each match/error; e.g., each insertion/deletion (*indel*) and mismatch error can be given negative values and matches positive values. Then the sum of the scores for each base in the experimental sequence can reflect the goodness of this match. The *best* match would be the one with the highest overall score.

It is possible to have a more complex scoring matrix in which each type of base match/mismatch/indel has a different value (e.g., $A$–$G$ mappings might be given a higher value than $A$–$C$).

Another possible approach is to take into account the quality values of each base in the sequence obtained from the wet lab experiment. For example, a mismatch error may be forgiven if the base in question was already flagged with a poor quality score [10].

Yet another approach may penalize each *beginning* of a series of gaps (inserts or deletes) since larger runs of gaps are usually more biologically plausible (as opposed to many small insertions/deletions peppered throughout a sequence). Using this approach, the alignment in Table II would yield a higher score than that in Table I, since the former has two runs of two contiguous gaps (at positions 3–4 and 6–7), while the latter has only one contiguous gap (at 5–6), and two individual gaps (at 3 and 8).

The algorithms typically involve some type of dynamic programming [11]. Since this is time consuming for large sequences, modern approaches use heuristics to reduce the time necessary. Two well-known examples are FASTA [12] and BLAST [13].

 ISBN: 978-1-61208-415-2 

```
gi|110640213|ref|NC_008253.1|... 16
gi|110640213|ref|NC_008253.1| 611 42 70M *
0 0 TTTCGTCGACCAGGAATTTGCCCAAATAAAACATGT...
22222222222222222222222222222222222... AS:i:0
XN:i:0 XM:i:0 XO:i:0 XG:i:0 NM:i:0 MD:Z:70 YT:Z:UU

gi|110640213|ref|NC_008253.1|... 0
gi|110640213|ref|NC_008253.1| 215 42 70M *
0 0 CCACCCCCATCAGCATTACCACAGGTAACGGTGCGG...
22222222222222222222222222222222222... AS:i:-6
XN:i:0 XM:i:2 XO:i:0 XG:i:0 NM:i:2 MD:Z:5A6C57
YT:Z:UU

gi|110640213|ref|NC_008253.1|... 16
gi|110640213|ref|NC_008253.1| 706 40 70M *
0 0 CCCGTGGCGAGAAAAGGTCGATAGCCATTAGGGCCG...
22222222222222222222222222222222222... AS:i:-12
XN:i:0 XM:i:4 XO:i:0 XG:i:0 NM:i:4 MD:Z:0G14T6C7T39
YT:Z:UU
```

Figure 1. Three alignment lines from a SAM file. Fields are delimited by whitespace. These lines are copied from a larger dataset [16].

- **QNAME:** *gi|110640213|ref|NC_008253.1|...*
- **FLAG:** *16*
- **RNAME:** *gi|110640213|ref|NC_008253.1|*
- **POS:** *706*
- **MAPQ:** *40*
- **CIGAR:** *70M*
- **RNEXT:** ***
- **PNEXT:** *0*
- **TLEN** *0*
- **SEQ:***CCCGTGGCGAGAAAAGGTCGATAGCCAT...*
- **QUAL:** *22222222222222222222222222...*
- **TAG:** *AS:i:-12 XN:i:0 XM:i:4 XO:i:0 XG:i:0 NM:i:4 MD:Z:0G14T6C7T39 YT:Z:UU*

Figure 2. The third alignment line from Figure 1 decomposed into component fields. The ellipsis are used here to denote that field's value continues on. The TAG field itself has several subfields of tags.

### A. Interpreting the Alignment

When the locations and lengths of the reads are overlaid, one on top of another at their aligned locations on the reference, the ones with a great deal of overlap (called *pileup*) lead to stronger confidence in the fragment of the genome they cover.

Interestingly, this can lead one to conclude that while most of the genome matches, a part of it definitely has been altered in the experimental sequence. Suppose that one finds that no matter how a specific *read* was aligned, a handful of particular mismatches are manifested consistently. To return to the example we started with in this section, one might conclude that those consistent mismatches exhibit a mutation that contributed to the experimental individual's risk for the disease in question.

## III. The SAM Format

The Sequence Alignment Format (SAM) [14] captures the result of both the dry and wet lab processes described in Section II. It is designed to be used with a specific software package, called SAMTools [15]. It uses the alignment information contained in SAM files to perform various taks such as visualizing overlapping reads against the reference and examining pileups in a binned fashion (e.g., how many reads start at a given position of the reference).

The SAM format is similar to that of a Comma-Separated Value (CSV) file with implicit component names (an Extensible Markup Language (XML) equivalent with named fields (components) can easily be conceived).

An optional header section contains general information relevant to the entirety of reads in the file, e.g., the SAM format version number, the specifics of the alignment lines at various locations, and the order in which data will appear.

### A. The Alignment Section and Lines

Immediately following the header section (should it exist at all) are the rows of alignment lines corresponding to each *read*. (It should be noted that sometimes the reported sequences correspond to smaller pieces of reads called *read segments*). Predictably, the bulk of the SAM file is taken up by the alignment section, filled with alignment lines.

A sample of three alignment lines from an example SAM file is given in Figure 1. The third alignment alignment line is decomposed into its component fields in Figure 2. There are at least eleven distinct fields, delimited by whitespace, in each alignment line. The twelfth field, TAG, is optional and slightly different, since there can be multiple tags within that field or even none at all (multiple tags are also delimited by whitespace). These are numbered 1 through 12; here and always in discussing the SAM file format, enumeration starts at one (i.e., all sequences are *1-indexed*).

The **SEQ** field contains the sequence corresponding to the read as obtained by the wet lab process.

The **QUAL** field (holding a quality string) captures the probability of incorrectness of each base in that sequence reflecting the error-prone nature of the wet lab processes. Each base is associated with an ASCII character, which can be transformed into a probability of correctness by different functions depending on the scoring method originally used to encode it. One equation to interpret characters of the QUAL field is given in (1).

$$P = 10^{\frac{QUAL-33}{-10}} \tag{1}$$

where QUAL is the decimal value of the ASCII character and P is the probability of the associated base being incorrect. As an example, the ASCII value of "2" is 50. When used with (1), the resulting probability $P$ for its associated base being wrong is 0.0199 (or about 1-in-50).

The **RNAME** field gives the reference. It is typically the value of (or contains the value of) some genetic identifier (e.g., an NCBI genetic code [17]) with a code representing which organism the reference comes from. For instance, in the example SAM line in Figure 2, RNAME is an NCBI code with value *gi|110640213|ref|NC_008253.1|*. The portion "NC_008253.1" is an accession number, giving a value for a particular sequence record. The number "1" after the period says that this is the first version of the sequence, which happens to be the entire genome of *Escherichia coli 536*.) The "gi" code given in "gi|110640213" is another version number [18].

The **POS** field provides the offset (1-indexed) into the reference sequence where the experimental sequence was best matched. In Figure 2, POS has the value *706*, meaning that

 ISBN: 978-1-61208-415-2 

the read segment on this alignment line starts at offset 706 of the *Escherichia coli 536* genome according to the best match.

The **CIGAR** field contains a string of the same name, CIGAR (Compact Idiosyncratic Gapped Alignment Report), which SAMTools uses as a coded abbreviation for how one sequence matches to a reference [14]. It contains an ordered series of numbers and letters, which tells one precisely how the sequence maps. Numbers correspond to the length of a subsequence while the letters imply an edit. An example of such mapping is given in Table III.

Referring back to Table I, its CIGAR string would read: $2M1I1M2D1M1I$. It would mean a match obtained in the following way:

> *The first two characters match or mismatch, the next character was missing in the reference and had to be inserted; the next character is a match or mismatch; the next two characters had to be deleted from the reference; this is followed by a match or mismatch; and the last character is an insertion into the reference.*

In Figure 2, the CIGAR field reads "70M", meaning the read aligned to the reference sequence with some combination of 70 matches and mismatches. Some of the extended CIGAR codes and their meanings are given in Table IV.

TABLE III. SEQUENCE ALIGNMENT WITH CIGAR

| **Position** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **AACTG** | A | A | C | T | G | _ |
| **ACTGG** | A | _ | C | T | G | G |

Best match between the two strings "AACTG" and "ACTGG" using Needleman-Wunsch for global alignment. Insertions and deletions are denoted by an underscore. The CIGAR string for "ACTGG" would read 1M1D3M1I.

TABLE IV. CIGAR STRING CODES

| **Op. Letter Code** | **Meaning** |
|---|---|
| M | Alignment Match (sequence match or mismatch) |
| I | Insertion to the reference |
| D | Deletion from the reference |
| N | Skipped region from reference |

Some (but not all) lettered operation codes and their meanings present in the extended CIGAR format.

The **TAG** field can contain many tags (minimum zero), even user-specified ones. All tags have the format of NAME:TYPE:VALUE and each can only appear once in any given alignment line [14]. 44 unique tags are established in the current SAM format, and each provides additional information from textual comments to the original alignment score generated by some aligner. (These tags are essentially additional pieces of metadata.)

The **MAPQ** field gives a mapping quality (higher number implies higher quality) for the alignment which, as we have stated earlier, is a difficult task. The MAPQ value of the SAM line given in Figure 2 is 40, denoting a high-quality match.

## IV. FORMALIZING SAM-FORMATTED DATA

SAM data is very useful: it says a lot about what occurred in sequence alignment and also something about sequencing. It is possible, however, that the software used to generate the data could change with time. For example, the change could be in an underlying alignment algorithm or the interpretation of read quality values. It would be beneficial if one could take such improved data and compare it with older SAM data. We argue that a formalization that shows *how* to represent and compute information would address this problem.

Such a formalization should have three parts: an ontology and two kinds of functions; we describe them below.

Such a formalization should live alongside the current data as supplementary metadata. In this way, all data can be understood more readily by domain experts while also revealing the specific functionality of that data (e.g., how to interpret an alignment sequence's structure using the CIGAR field).

### A. Ontology

We need an ontology that codifies concepts of the knowledge domain appropriate to the data. The ontology needs to contain the domain concepts arranged (possibly) in a generalization/specialization hierarchy; plus mappings among them; The full ontology necessary to capture all the semantic concepts of a SAM file would be far too large to present here. Instead, we furnish the reader with those essential constructions that clarifies the point of formalization: some of those entities and relationships that are components of the *sequence alignment* concept. This subset of semantic concepts are given in Figure 3.

The ontology in Figure 3 is the minimal amount needed to describe the concepts involved in sequence alignment. A parent class for many of the concepts here is *Sequence*, which is why many other concepts, such as experimental and reference sequences are its child classes. Any concept that has *Sequence* as a parent in the hierarchy will inherit from its definitions. For example, anything that is a *Sequence* will have a "character_seq_string" (the actual string representation of characters in the sequence).

*ExperimentalSequence* is distinct from *ReferenceSequence* (though they are both child classes of Sequence). *ExperimentalSequence*s have both quality scores (denoted by the "quality_scores_string" attribute) and is composed of at least 1 *Read* (with the *minimum 1* cardinality restriction). *ReferenceSequences*, instead, have an *OrganismName* attribute.

Reads, as we have said, can be split up into *ReadSegment*s, and these *ReadSegment*s form a chain from one *ReadSegment* to the next (the *hasNextReadSegment maximum 1* cardinality restriction reflects this). A *Read*, then, is composed of at least one *ReadSegment*, and *ReadSegment*s are components of *Read*s.

*SequenceAlignment* is the concept related to actually doing something with these various sequences. It acts as a function signature for performing sequence alignment in general. It has two inputs listed, which are an *ExperimentalSequence* and *ReferenceSequence*. The output is *minimum 0 AlignedSequence*s, because it is possible no alignment can be found. This *minimum 0* also means that many *AlignedSequence*s can exist for some SequenceAlignment. The "FunctionInternal" attribute is explained in Subsection IV-B, and describes "how" this *SequenceAlignment* concept computes its output.

The *AlignedSequence* is the output of some *SequenceAlignment* (as clear in the *isOutputOf* relation). This concept has an

**Sequence** (1)

Attribute character_seq_string (String)
Attribute length (Integer)

**ExperimentalSequence** (2)

Attribute quality_scores_string (String)
isA Sequence
isComposedOf *minimum 1* Read

**ReferenceSequence** (3)

Attribute OrganismName (String)
isA Sequence

**Read** (4)

isA Sequence
isComposedOf *minimum 1* ReadSegment
isComponentOf *minimum 1* ExperimentalSequence

**ReadSegment** (5)

isA Sequence
isComponentOf Read
hasNextReadSegment *maximum 1* ReadSegment

**Sequence Alignment** (6)

Attribute FunctionInternal (XML Literal)
hasOutput *minimum 0* AlignedSequence
hasInput *some* ExperimentalSequence
hasInput *some* ReferenceSequence

**AlignedSequence** (7)

Attribute ReferenceOffset (Integer)
isA Sequence
mapsTo *exactly 1* ReferenceSequence
isOutputOf *some* SequenceAlignment

Figure 3. A subset of the full formalization of the SAM format to which actual data will be mapped either directly or through complex transformation.

attribute for stating the offset in which the alignment occurs against *exactly 1 ReferenceSequence* (the reference used in sequence alignment).

### B. Choice of Ontology

The snippet of the ontology in Figure 3 is based on the "EMBRACE Data And Methods" (EDAM) ontology [19]. EDAM was based on several different resources, including myGRID [20], and follows many principles of the Open Biological and Biomedicial Ontologies. EDAM attempts to represent formats used in bioinformatics, their data, the operations those data might be involved in, and associated topics. The EDAM ontology is impressive in its coverage and design. There is, to our knowledge, no more comprehensive ontology than EDAM that exists with concepts tying both bioinformatics formats to their data and use in associated operations.

We, however, do not find EDAM capable of fulfilling all the needs of our goal here, since we require more complete coverage of individual formats with more format-specific constraints. When trying to fit more specifics of SAM data with what is presented in EDAM, we found some concepts to be missing. This is most likely because EDAM was designed to represent workflows, which is at a higher level of granularity than the specifics of a data format's contents. These missing concepts include a quality sequence for SAM's QUAL field and a concept appropriate for mismatching character sequences such as CIGAR strings. Further, we wanted to be able to pair individual portions of an alignment line with one another, which required more specifics relating to the SAM file.

To be clear, in some ontologies, such functions may be represented as a concept, but it is in name and relationships only; they are function signatures, but do not contain function internals. The myGrid ontology, for example, contains the concept of the Needleman-Wunsch sequence alignment algorithm. The EDAMS ontology has the more abstract *SequenceAlignment* concept. While these concepts may contain relationships such as output, input, etc., they lack a conceptualized view of the algorithm — or process — they are meant to define. In other words, there is no implementation-agnostic function internals to pair with real-world instances.

In Figure 3, we show our plans for adding such functional internals to ontological definitions. This "FunctionInternal" attribute (found in the *SequenceAlignment* concept) of type XML literal can be represented with MathML [21]. It is with this language that the implementation-agnostic function internals can be described.

### C. Functions and Extensibility

Consider Figure 4, a commutative diagram [22] connecting the worlds of data and ontology via functions. Functions of the first kind are like the dashed arrows in that figure. They take one or more data elements and bridge them to a concept in the ontology [23]. Our second kind of functions are like $F(x)$ and $F'(x)$. They represent some idealized computation of some domain concepts from others, reflecting data elements that are computed from other data elements in an equivalent manner.

For example, let *Data* map to a pair of read and reference, *Ontology Term 1* to a pair of sequences, *Results* to an alignment with an offset, and *Ontology Term 2* to an aligned sequence. Let $F(x)$ be the Needlemann-Wunsch algorithm and $F'(x)$ be a particular implementation of that algorithm. It is easy to see the power of this framework; for example, it would provide the ability to use more complex alignment algorithms that take into account the probabilistic nature of the sequences under consideration.

One synergistic consequence is extensibility. One can take existing data and compare against the results of newer alignment algorithms (e.g., BLAST and FASTA, or algorithms being researched).

## V. Investigation of Reproducibility of Results

We asked the question: can the final result of the dry lab process, the MAPQ value, be reproduced by a consumer of a SAM data file?

 ISBN: 978-1-61208-415-2 

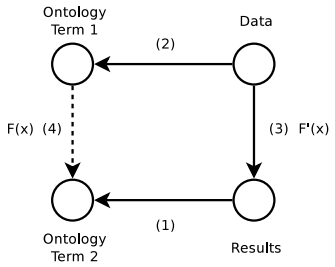


Figure 4. Commutative diagram for data and ontology

### A. Computing Alignment Score

As mentioned earlier, the SAM file includes a MAPQ field which contains a number indicating what confidence one may have in the alignment inferred by the dry lab process. The value is given by (2), where *P* is the probability of error of the stated alignment of read $z$ at offset $u$ in genome $x$:

$$MAPQ = -10\log_{10}(P) \tag{2}$$

$P$ is given by

$$P = 1 - \frac{p(z|x,u)}{\sum_v p(z|x,v)} \tag{3}$$

where $p(z|x,u)$ is the probability that read $z$ maps to reference $x$ at offset $u$ [24]; in the denominator, the summation is over all such offsets.

The *best match* gives the offset $u$ that results in the smallest $P$ as per (3). This is the location offset with respect to the reference where the best alignment match can be found. Equation (3) implies an exhaustive computation (albeit there are positions that can be skipped over).

However, there are more modern, less time-consuming alignment methods based on heuristics. For example, the following approximation is faster [24].

$$\begin{aligned} MAPQ = min[q_2 - q_1 - 4.343\ log(n_2), \\ 4 + (3 - k')(\bar{q} - 14) - 4.343\ log(p_1)\ (3 - k', 28)] \end{aligned} \tag{4}$$

where $q_2$ is the sum of the quality scores corresponding to the bases that have mismatches for the best alignment score and $q_1$ the corresponding sum for the second-best alignment. $k'$ is the minimum number of mismatches on some 28 base-pair *seed* (which are locations to index against the reference with the read; larger seeds result in lower sensitivity [25]) and $\bar{q}$ is the average of base qualities in that seed [24].

Clearly then, the MAPQ computed depends on the equation used; more generally, if context-sensitive cases are considered, on the algorithm used. This means that different programs used in alignment of the same experimental sequence against the same reference may result in different values of MAPQ. Without knowledge of the equation or algorithm used, one cannot reproduce the MAPQ element presented in the SAM file. Worse, comparing MAPQ values of multiple SAM files may well be invalid. (Adding a pointer to a program used for alignment is possible, but such additions would be in optional fields with no guarantee of inclusion in all published data files.)

For reproducibility, we suggest that the function (or algorithm) used in the dry lab process for MAPQ be included in the SAM data file.

This will allow users not only to verify the correctness of the MAPQ reported in the file but also use their own alignment algorithm to compare and publish the results obtained. This could lead to publications providing insight into various aspects of experiments such as innovative approximations and the effect of errors in the underlying wet lab process.

## VI. Conclusion and Future Work

Examining a popular file format in bioinformatics, we asked two questions relating reproducibility and extensibility of data. What we found was that neither reproducibility nor alternative conclusion testing was possible. We identified the cause as the absence of functions for arriving at these conclusions.

Such functions are essential extensions to whatever current metadata may be included with the data, since these functions expose the details of how the data was created in the first place. When such functions exist, one can start to meaningfully compare different data sets that did not employ the same function and also to compare the computed results (e.g., MAPQ) in a data set with the corresponding results that one would obtain using an alternative function or algorithm.

We have also explored the choice of what metadata to add and how to add it. We have shown that this choice need not be guided by a particular pre-defined purpose, but from the contents of the data itself. When data is formalized, ambiguity and confusion about how different parts of data relate to one another are decreased. Necessary metadata becomes apparent as the portions of data are formalized and their component parts are established.

This has culminated in the idea of the formalization of data as a marriage between an ontology and a set of functions that describe both data and its underlying processes. Such a formalization creates an accessible point from which data can be understood, reproduced, and have its functionality extended.

While we have explored an example from bioinformatics and presented a subset of the constructions required for a formalization, we have tried to show that the generality of the approach goes beyond bioinformatics and embraces any data file containing computed elements. Such data files are pervasive; consider for example, a medical report such as a blood test that contains diagnostic elements that are based on complex analysis.

Of course, our work is part of a larger solution. For example, the decision of how best to represent such additional metadata is left as further research.

## Acknowledgment

We would like to thank anonymous reviewers whose comments have helped improve this paper. This work is supported by a grant from The United States Department of Homeland Security 2011-ST-062-000051. We gratefully acknowledge travel support from the Institute for Complex Additive Systems Analysis (ICASA) at New Mexico Tech enabling the presentation of this paper.

 ISBN: 978-1-61208-415-2 

## REFERENCES


[1] V. Stodden, "Reproducible research: Addressing the need for data and code sharing in computational science," Computing in Science & Engineering, vol. 12, no. 5, 2010, pp. 8–12.

[2] R. LeVeque, I. Mitchell, and V. Stodden, "Reproducible research for scientific computing: Tools and strategies for changing the culture," Computing in Science and Engineering, vol. 14, no. 4, 2012, p. 13.

[3] R. Stevens, A. Robinson, and C. Goble, "MyGrid: personalised bioinformatics on the information grid," Bioinformatics, vol. 19, no. suppl 1, 2003, pp. i302–i304.

[4] C. Pancerella et al., "Metadata in the collaboratory for multi-scale chemical science," in International Conference on Dublin Core and Metadata Applications, 2003, pp. pp–121.

[5] T. Oinn et al., "Taverna: a tool for the composition and enactment of bioinformatics workflows," Bioinformatics, vol. 20, no. 17, 2004, pp. 3045–3054.

[6] J. Frew and R. Bose, "Earth System Science Workbench: A data management infrastructure for earth science products," in Scientific and Statistical Database Management, 2001. SSDBM 2001. Proceedings. Thirteenth International Conference on. IEEE, 2001, pp. 180–189.

[7] D. Wood, M. Lanthaler, and R. Cyganiak, "RDF 1.1 concepts and abstract syntax," W3C, W3C Recommendation, February 2014, [Retrieved May, 2015]. [Online]. Available: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[8] P. Patel-Schneider and B. Motik, "OWL 2 Web Ontology Language mapping to RDF graphs (second edition)," W3C, W3C Recommendation, December 2012, [Retrieved May, 2015]. [Online]. Available: http://www.w3.org/TR/2012/REC-owl2-mapping-to-rdf-20121211/

[9] S. Needleman and C. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," Journal of Molecular Biology, vol. 48, no. 3, 1970, pp. 443–453.

[10] X. Yu et al., "How do alignment programs perform on sequencing data with varying qualities and from repetitive regions?" BioData Mining, vol. 5, no. 1, 2012, p. 6.

[11] O. Gotoh, "Multiple sequence alignment: algorithms and applications," Advances in Biophysics, vol. 36, 1999, pp. 159–206.

[12] W. Pearson and D. Lipman, "Improved tools for biological sequence comparison," Proceedings of the National Academy of Sciences, vol. 85, no. 8, 1988, pp. 2444–2448.

[13] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, "Basic local alignment search tool," Journal of Molecular Biology, vol. 215, no. 3, 1990, pp. 403–410.

[14] "Sequence alignment map format specification," May 2013, [Retrieved May, 2015]. [Online]. Available: http://samtools.sourceforge.net/SAMv1.pdf

[15] H. Li et al., "The Sequence Alignment Map format and SAMtools," Bioinformatics, vol. 25, no. 16, 2009, pp. 2078–2079.

[16] E. Cerami, "SAMtools: Primer," [Retrieved May, 2015]. [Online]. Available: http://biobits.org/samtools_primer.html

[17] A. Coghlan, "Sequence Databases," [Retrieved May, 2015]. [Online]. Available: http://a-little-book-of-r-for-bioinformatics.readthedocs.org/en/latest/src/chapter3.html

[18] B. Hochhut et al., "Escherichia coli 536, complete genome," [Retrieved May 2015]. [Online]. Available: http://www.ncbi.nlm.nih.gov/nuccore/NC_008253

[19] J. Ison et al., "EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats," Bioinformatics, vol. 29, no. 10, 2013, pp. 1325–1332.

[20] J. Zhao et al., "Using semantic web technologies for representing e-science provenance," in The Semantic Web–ISWC 2004. Springer, 2004, pp. 92–106.

[21] R. Miner, "The importance of MathML to mathematics communication," Notices of the AMS, vol. 52, no. 5, 2005, pp. 532–538.

[22] J. Adamék, H. Herrlich, and G. Strecker, Abstract and Concrete Categories. John Wiley, 1990.

[23] J. McCloud and S. Mazumdar, "Translation of Various Bioinformatics Source Formats for High-Level Querying," in 2013 Linked Data in Practice Workshop (LDPW2013), 2014, pp. 39–53.

[24] H. Li, J. Ruan, and R. Durbin, "Mapping short DNA sequencing reads and calling variants using mapping quality scores," Genome Research, vol. 18, no. 11, 2008, pp. 1851–1858.

[25] J. Buhler, "Efficient large-scale sequence comparison by locality-sensitive hashing," Bioinformatics, vol. 17, no. 5, 2001, pp. 419–428.


 ISBN: 978-1-61208-415-2 