# A Practical Approach for Scoring Data Quality in Routinely Collected Health Data

by Kevin Garwood

In this article I outline a practical approach for scoring the quality of routinely collected health records as they may be used to support health studies. The approach grew out of data extraction activities I did while I was working as a database analyst using NHS Digital’s Hospital Episode Statistics (HES) records.

The ideas I present here were borne out of a practical operations interest rather than a research imperative: the need to minimise repetitive development work to extract data for multiple, similarly themed analysis projects.

The system uses the concepts of field-level, intra-record and inter-record data quality checks that are meant to be implemented in ways that would be agnostic with respects to the needs of any one health study. The quality checks would then be weighted first by yearly temporal trends in each check and then by a weighting of importance that would need to be decided by a research community.

The success of such a system would critically depend on having researchers invest in shared, re-usable data sets. It would require standardising aspects of data cleaning, filtering and de-duplication which would tend to preserve original data.

## Recognising Patterns in Development

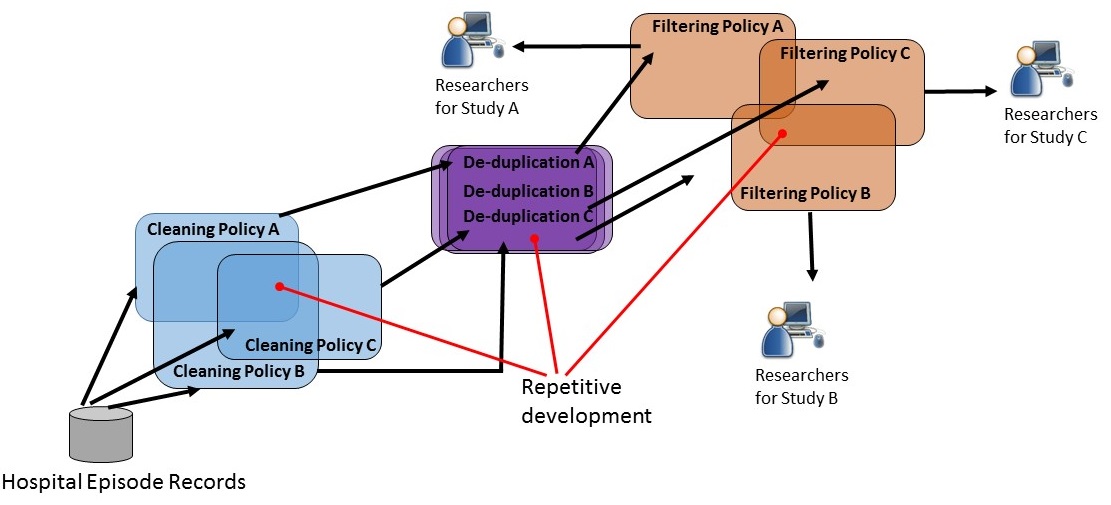
Much of my work in scientific computing could be characterised as identifying patterns of repetitive development that would motivate the creation of shared solutions in scarcely resourced IT environments. Key to the development of these technical solutions has been a frank recognition that when presented with common sources of data, scientists will want provisions that allow them to collaborate and compete, compare and contrast their results. Identifying the practical means whereby they can support these opposing needs represents something of an engineering instinct.

When I worked on developing model-driven software applications for bioinformatics groups at the University of Manchester, the instinct was about deciding which application features could be generically generated and which ones would be better supported through domain-specific plugins. The model-generated software tools I created combined generic and the specialised features that would allow labs to cooperate, yet differentiate themselves with respects to the needs of their own studies.

Some parallels can be drawn when making the transition from creating reusable software applications to creating reusable data sets. Once again, developing code requires an eye for distinguishing between generic and specialised needs of a research project. Whereas my work in Manchester focused on feature areas related to data capture, searching and data retrieval, at Imperial the feature areas were related to data cleaning, de-duplication and filtering. The scoring system I present here was developed as a way of reconciling the common tension I’ve observed in trying to accommodate both generic and specialised features.

## Assessing Generic and Specialised Qualities of Filtering, De-duplication and Data Cleaning

Most data extraction activities involve applying at least three kinds of policies in order to prepare a data set for analysis: data cleaning, de-duplication and filtering. The way and order in which these policies are applied can lead to repetitive ways of implementing similar protocols and create slightly different result sets that may support little comparability with one another. The areas of overlap shown in the following figure could represent duplicated effort that could otherwise be used to support more studies.



Bearing in mind an interest to reduce potentially repetitive development efforts, we will first describe each kind of policy and then decide whether they are best supported through generic or specialised protocol steps. We will explore filtering and de-duplication policies first, and use the discussion about data cleaning to help motivate having a data quality scoring system.

We’ll begin with **de-duplication policies**, which are used to identify duplicate records and to choose which ones to keep. I’ve covered this topic in another article, but to review, applying the policy begins by choosing all or a subset of fields which together can be used to assess whether two records are the same. Once members of duplicate record groups have been identified, some criteria for preserving the best duplicate needs to be applied. Typically, the policy involves keeping a random duplicate, keeping the first in an ordered list of duplicates, or keeping the duplicate having the most populated fields.

De-duplication will often be viewed as an activity that has the following characteristics: it requires little domain-specific decision making; it would be relevant to most studies that would use the same records; and it would be used mainly to identify whether an entire record represented an error. As I mentioned in a previous article, the best candidate fields for identifying duplicate records in data sets such as HES would be well-populated, mechanically generated field values that were not likely to be affected by data cleaning.

De-duplication policies are a specialised form of **filtering policies**, which are used to select the records that will be used in the analysis for some given study. Filters that do not relate to de-duplication are often used to preserve records which may be valid but are either not relevant or not good enough relative to the specific needs of a given study. For example, a maternity study may apply a filter to isolate delivery events from all other events recorded in a health data set. Another health study concerned with patient demographics may apply a filter that includes only records whose sex field is not empty and whose birth dates seem reasonable. In order to foster reproducibility of results and reuse amongst multiple studies, filtering policies should use flag fields and not actually delete records.

Often, de-duplication policies can be implemented as part of a generic protocol that services multiple projects. However, other filters may involve judgements that are more study-specific in nature and whose outcomes might differ when made by one domain expert or another. Often it is a good idea to defer most filtering policies as activities that should be done by individual researchers when they draw from a shared source of records.

**Data cleaning policies** involve altering the representation or meaning of field values within a record. Examples of altering representation include: changing sex values such as ‘M’, ‘male’ and ‘1’ to all be ‘1’; removing periods from ICD code values; and ensuring that all post codes can be rendered with the same number of characters. Examples of altering the meaning of field values include: imputing blank birth weights with typical values; mapping ten different ethnicity categories to five that may be used in some study; or changing deprecated postal codes to new ones.

Applying generic data cleaning policies tends to present a number of challenges, especially with routinely collected health data sets. First, the quality of records can vary by year, health provider, health outcome and record field. Second, cleaning rules that could be applied might be applied differently depending on the study and on the preferences of the scientists who run it. Third, although in theory a data extract can contain both cleaned and original versions of the same fields, in practice it is usually just the cleaned fields that appear in the data sets used by analysis. Preserving only the changed values can cause a loss of provenance which would make it difficult for peer researchers to determine whether they would have made the same decisions using the same original data set.

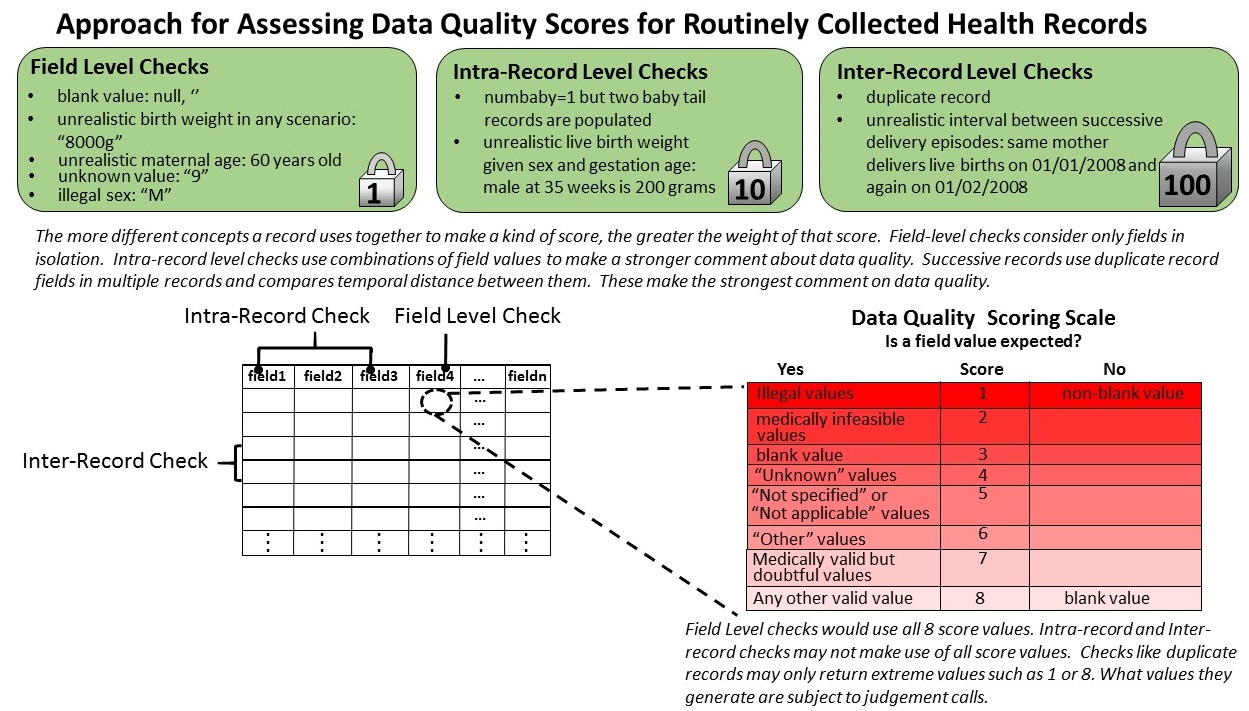
Some data cleaning actions could be implemented in a generic protocol. However, the potential variation in the way scientists could implement cleaning policies to cater for specific subsets of records and fields might prove complicated. It is tempting to push many filtering and data cleaning policies onto individual researchers, but doing so would detract from the value a reusable data set would have for a community. It is worth asking whether there is anything else we could do which would preserve original field values for all studies, yet would add value to analysis. Now it is appropriate to discuss creating a community-driven set of data quality checks that could help score records.

## Developing an Approach for Scoring Routinely Collected Health Records

Scoring all records and record fields carries the advantage of preserving the state of original data for scrutiny by study scientists and peer reviewers, all while assessing its quality for the benefit of research communities. A data set that comes with corresponding scoring data produces a reusable data asset that fosters reproducibility in a way that is not biased by the aims of any one study. A scoring system may also be used to influence the identification of trends amongst variables in machine learning algorithms.

Scoring records and fields also carries the disadvantage of producing a great amount of data about each record and could easily double the size of an original data set. On balance however, the effort needed to create scoring data would likely only have to be done once for each year of records that have accumulated. Once made, the scores would be preserved for the benefit of future studies.

We can begin discussion of a scoring system by introducing three types of data quality checks, and mapping their possible values to a general-purpose scoring scale of data quality states. Then we can apply two types of weightings for various checks: yearly average values of each check and a weighting of importance that would be determined by a research community. The system I’ll describe is shown below. It uses maternity delivery events in the examples.



## Developing Field Level, Intra-Record Level and Inter-Record Data Quality Checks

**Field level** checks evaluate the quality of a field value irrespective of what other field values may be in the same record. For example, in a delivery event record, a baby’s birth weight of 8000 grams is unreasonable, regardless of whether the mother’s maternal age has been specified correctly. In health data sets, field level checks can identify whether a field is blank, or populated with a value that is illegal, blank, medically infeasible or reasonable.

**Intra-field checks** use combinations of values from multiple fields within the same record. Maternity delivery events provide good examples. Although it is possible that a baby can have a mass of 400 grams at birth, it is unreasonable that a live birth having a gestational age of 40 weeks would weigh that little. This kind of check would involve domain knowledge, but could be applied regardless of whatever maternity health outcome was being considered.

Intra-field checks could also indicate whether a given field should be expected to be blank or not. For example, if the number-of-babies field shows “2” in a delivery record field, we would expect that fields for “baby 3” should be left blank. If they are non-blank, then we could provide a penalty rather than a reward for having certain fields appear populated.

**Inter-record checks** derive a score based on comparing one record with others in the data set. For example, an inter-record check could score a record highly if it is unique and low if it is part of a group of duplicates. Another kind of inter-record check could provide a score based on the temporal interval for two successive health events for the same person and the same kind of health outcome. For example, it would be invalid for a mother to have two single live birth delivery events a week apart. It might be possible but unlikely if successive births were separated by 23 weeks and reasonable for any greater interval. Temporal intervals could also be evaluated for scenarios like organ transplants or serious operations.

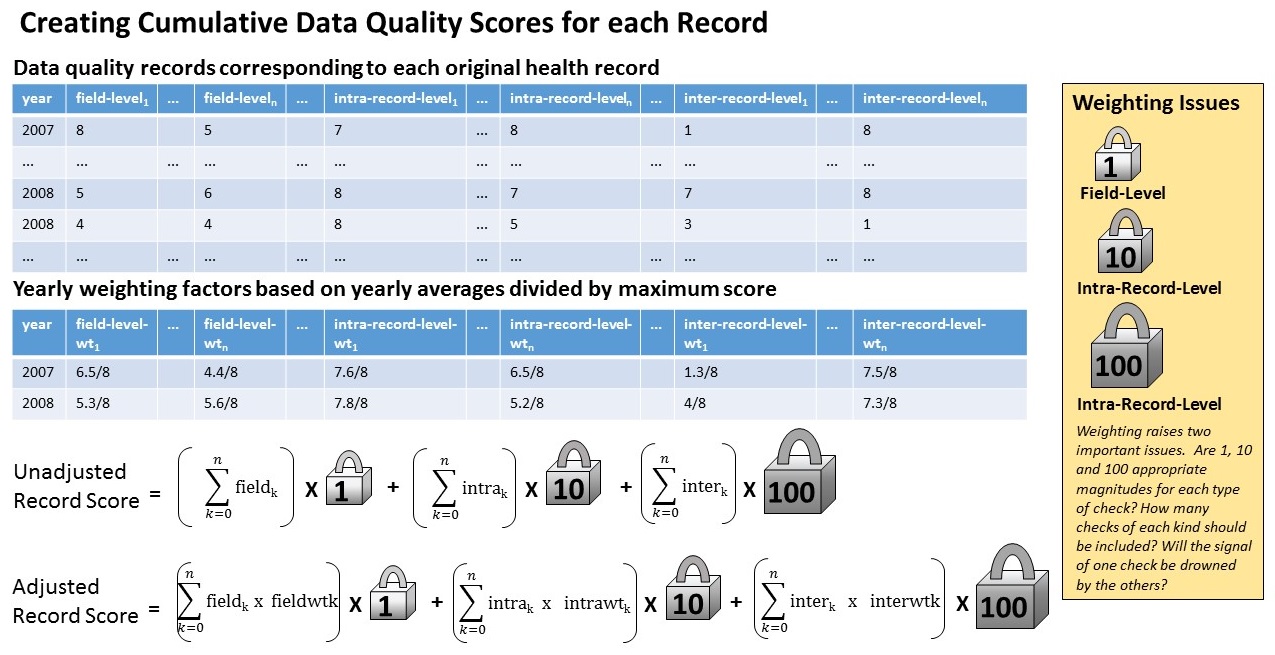
## Developing a Scoring Scale

I’ve developed an eight point scoring scale that can be influenced by whether we expect certain field values to be populated or not. As we’ll later discuss, the scale has weaknesses and warrants further discussion. But in general, in cases where we expect a field value to be populated, illegal values provide the worst values because they indicate something is wrong with data collection. Medically infeasible values provide more information but again, they would tend to indicate a serious data recording problem. Blank values provide no data but they may also exhibit an absence of illegal values. “Unknown” doesn’t provide data but begins to establish that data collection was made with some kind of intent. “Not specified” still provides no usable field value but the conviction of intent seems stronger. “Other” implies an active data collection action that resulted in a value that wasn’t accommodated by the schema. Medically valid but doubtful values provide meaningful data but may end up being considered outliers. Anything else seems like a good value. If the field is expected to be blank, then we only care whether it is or isn’t.

The values of all checks are expressed in one of these 8 values, often depending on which is applicable. For example, evaluating the combination of birth weight and gestation age for a delivery record would not likely result in a 7 being used. Inter-record checks that used temporal intervals between events could use 1, 2, 7, or 8 values but others may not be appropriate.

## Weighting Data Quality Scores

Each kind of score would provide useful data quality information on it its own, but they when combined they could provide an overall score for a record. As the collection of data quality check values increases, so too does the likelihood that the influence of some checks might be drowned out by all the others. The following figure shows two ways of applying weights to scores.



One way to weight the data quality check scores is to multiply each one by a factor based on whether it is a field level, intra-record or inter-record check. In the example, it is far more significant if a record is a duplicate than it is for an individual field value to have a high level of quality. Another way to weight checks for a given record is to multiply each check score by the average score out of 8, measured across all records for the same year.

## Illustrating the Scoring System through a Code Example

I’ve created a code example that demonstrates how the scoring system would work. It is released under the GNU LGPL v.30 open source license. Run the script through PostgreSQL and if any of the code seems unclear, please ask me about it.

## The Need for Further Refinement of the Scoring System

My scoring system should raise several points of debate. The first is whether ranking of data quality scenarios in the scoring scale is reasonable. The second is whether it makes sense to use only “1” and “8” for cases where we expect a field value to be blank. The third is whether it is reasonable to express field and record-level checks using one scale. The fourth is whether the orders of magnitude of “1”, “10” and “100” are appropriate to use for weights.

Beyond the mechanics of creating a scoring system, there is the broader issue of whether research communities would want to come together to develop their own scoring system and use it in their studies. It is also worth considering whether scientists would exclude records from analysis that had low total data quality scores, even if some of the composite checks had no bearing on any one study. For example, if you’re doing a maternity study that only cares about the mother’s age and ethnicity, will you exclude records if the baby’s method of delivery has not been recorded correctly?

The goal of presenting this piece is not so to indicate that the scoring system I’ve described is the best one to use. Rather, it’s the idea that any kind of community-developed scoring system could be developed at all to assess the quality of routinely collected health data that are used to support research studies. For research groups that are compelled to try and do more with less, investing in a shared reusable data set to service families of research projects may prove appealing.

## About Me

I’m a software developer and data scientist working in London. See my web site for more of my background.