Critical Care Project Report

Research Software Development Group

University College London

7th July 2016

Contents

1 Introduction

Digital patient data will be the cornerstone of much future medical and health services research, particularly in critical care. However, the data needs to be converted into a research-ready structure to enable easy scientific and statistical analysis. The data must be transformed into a queryable form to allow its properties and quality to be understood.

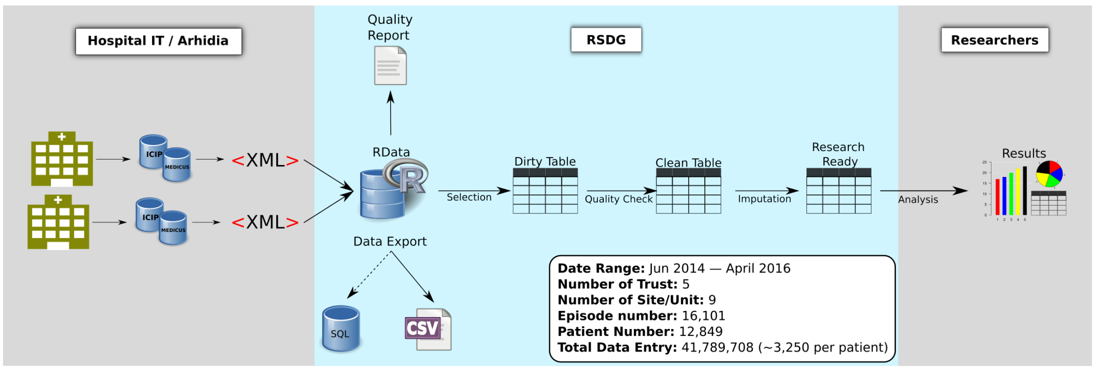
The UCL Research Software Development Group's objective on this project is to provide an open source software solution to bridge this gap and deliver ready-to-use data and data manipulation tools for researchers. We have built a pipeline which converts `XML` raw data to the queryable RData format. The resulting data can also be exported into a `CSV` format to allow it to be used outside R (*e.g.*, with Excel).

A "data quality check dictionary" is used in the pipeline to clean the data. Invalid or corrupted data is either modified or removed according to these criteria, which are presented in a "business readable" file format allowing researchers to design and verify the criteria for record inclusion. An auto-generated data quality report is produced to report on the rate of "missingness" of key fields, and on data sanity by site and unit. By the end of the pipeline researchers obtain an R table which is queryable and validated.

We also have demonstrated the ability of programmatically deriving data products from this pipeline, such as both automatic generation of a SOFA score and identification of sepsis. A publication about this process is in preparation.

We believe robust, understandable and accessible software is the first step of solid computationally-based research. Therefore, in addition to the functionalities described above, we have also focused on the usability and sustainability of the software produced. All of our development is constantly updated in the GitHub platform which is accessible to all the collaboration. The software is simple to install and configure. The code is self-explanatory and well documented empowering both current users and future developers. Full automated tests of code correctness are conducted after every modification of code, automatically, on our servers. Such robust software practices will permit the software to be a long-term gateway to understanding the data, and will play a critical role in the future success of the community.

2 The Data Process Pipeline



Date Range: Jun 2014 — April 2016
Number of Trust: 5
Number of Site/Unit: 9
Episode number: 16,101
Patient Number: 12,849
Total Data Entry: 41,789,708 (~3,250 per patient)

Figure 1: Data Process Pipeline with summary report of the data processed

2.1 The XML data and the Data Safe Haven

Data associated with more than 12,000 patients from five NHS trusts are available in `XML` format. The data were drawn from the ICIP and Medicuse databases which contains demographic, drugs, laboratory, nursing and physiology data. Both de-identified and identifiable data are processed, locally and on the UCL Data Safe Haven (IDHS) respectively. In order to create the pipeline on IDHS, UCL's standards-compliant solution for working with restricted and confidential data, we requested and configured an appopriate Linux virtual environment.

Due to the limited computational capacity of IDHS, it is still useful to keep a local de-identified copy of the data for development purposes. The pipeline is designed to be portable to multiple platforms, and will additionally be deployed to UCH servers in due course.

2.2 XML parser

The `XML` parser in R combines and restructures the `XML` files into a newly defined R data structure `ccRecord`, which significantly improves the clarity of the data by organising it into tables and removing significant redundancy in the `XML` files. `ccRecord` is designed as a flexible, simplified, and queryable data structure for critical care measurements. Data in `ccRecord` format is eventually stored in a RData file which is about 500 times smaller than the original `XML` files.

Data provenance is recorded for each episode, thus we are able to tell from which file each episode comes and when it has been parsed. Selected data fields can be exported into a `CSV` file for Excel users or to be used with any other analysis tool or programming language.

We have on going work to develop an equivalent parser in `C++`. The prototype has a much better performance comparing to the `R` version, but has not yet been deployed into the current pipeline. The incorporation of the `C++` parser into the pipeline will reduce from Xs to Ys the time to parse all the files.

2.3 Auto-generated quality report

Data may have imperfections in various ways. Therefore an automated quality assessment that allows us to report on problems with the source data is extremely useful. A data quality report which reflects the quality issues has been developed and deployed. Based on the report, we are able to discover major missing and corrupted data together with basic information such as the duration, sites, number of episodes, and number of patients.

2.4 Data validation and cleaning

Data validation and cleaning functions are included in the `ccdata` package. A "data sanity check" is conducted considering many aspects of various data fields. Three major checks are carried out: numeric range, text category, and missingness. Researchers using the tool complete a simple form (in `yaml` format) to guide the validation check. Consequently, data is flagged,excluding the data that do not make sense. After this stage, we are able to deliver a "cleaned" and queryable R table to the researchers. Next section discusses in more detail the `yaml` dictionary.

3 The R package: `ccdata`

The `ccdata` R package is the centralised toolset we have developed for piping and data manipulation. The package bundles not only all the R and C++ code but also its documentation and tests making for easier code sharing. The `ccdata` package is portable to all platforms where a R environment is available.

It can be installed effortlessly on Windows or UNIX based systems. Although continuing code improvement and tidying will be a part of our ongoing development cycle, the main part of the R code in the current stage is well documented and properly tested.

In order to prepare the first scientific publication, many data manipulation processes were needed. Instead of making one-off scripts for these, as we encountered each problem, we incorporated each data manipulation process as re-usable functions in the `ccdata` package. Frequently used functions such as detecting unique patients, determining ward spells, and imputation of missing data can be called from the package, reducing therefore the duplication of work in the future. The package will continue to grow as the research goes on.

3.1 Data Selection and cleaning in the `yaml` configuration

To be able to use and program with the R language and `ccdata` package confers great flexibility in data analysis. However, it is important that the pipeline can also be invoked by users without any programming knowledge.

The users can run the pipeline by only filling in a `yaml` configuration file. The form is straight forward and, we hope, self-explanatory for non-programmers. The user selects data fields for the cleaning process obtaining a single CSV file at the end of the pipeline which can be opened in MS Excel.

Below is an example of the configuration file for managing data on heart rate. Three filters `nodata`, `range`, `missingness` are presented in the following data selection and cleaning configuration.

```
NIHR_HIC_ICU_0108:
  shortName: hrate
  dataItem: Heart rate
  distribution: normal
  decimal_places: 0

  # filter1: do not use the episode where hrate cannot be found.
  nodata:
     apply: drop_episode

  # filter2: mark all the values based on reference range (traffic colour)
  # remove entries where the range check is not fullfilled.
  range:
      labels:
          red: (0, 300)
          amber: (0, 170)
          green: (50, 150)
      apply: drop_entry

  # filter3: compute the item missing rate on given cadences; in this case, we
    compute the daily (red) and hourly (amber) missing rate, and only accpet
    episodes of which hourly missing rate (amber) is lower than 30%.
  missingness:
      labels:
          red: 24
          amber: 1
      accept_2d:
          amber: 70
  apply: drop_episode
```

4 Summary

The UCL Research Software Development group has been involved in the NHIC critical care data project since January 2016. Over the last six months we have contributed to preparation of a research paper with the critical care team while developing re-usable and sustainable software tools for the data pipeline and data manipulation. In the next phase of this project, we will focus on the linkage of the data to external sources. We will continue supporting researchers by providing sustainable software tools.