Critical Care Project Report

Research Software Development Group

University College London

7th July 2016

Contents

1 Introduction

The critical care patient data will be the cornerstone of many future medical and health services research. However, from data to research, there is still a huge gap to cover. Data has to be converted to a query-able form and we also need to know more about the data quality before it is ready for use. The mission of our UCL research software group (RSDG) is to provide an open source software to bridge this gap and deliver the ready-to-use data alongside with the data manipulation tools to the researchers.

A pipeline which converts the XML datasets to the query-able RData format has been built. It also allows us to export the selected data from RData to CSV for Excel. In the pipeline data quality check will be conducted guiding by a business readable data quality check dictionary. The non-validated data will be either modified or removed according to the criterion. Subsequently, an auto-generated data quality report will be created to report the key fields missingness and data sanity by site/unit. By the end of the pipeline the researchers will be able to receive a R table which is query-able and validated. We also have demonstrated the ability of deriving variables, such as calculating SOFA score from physiology data and identifying sepsis. We believe the paper about this will be ready soon.

Beyond the functionalities, we are keen on the usability and sustainability of the software as we believe an robust, understandable and accessible software is the first step of concrete science. All of our development are published on GitHub which is visible by all the allowed users. We made the installation of our software as easy as possible. The code are self-explanatory and well documented, which provides a great accessibility to the users and future developer. Testings are conducted after any modification of code automatically on Travis CI system. We believe the software is a gateway of the data which may play a great part in the future success of the community.

2 The Data Process Pipeline

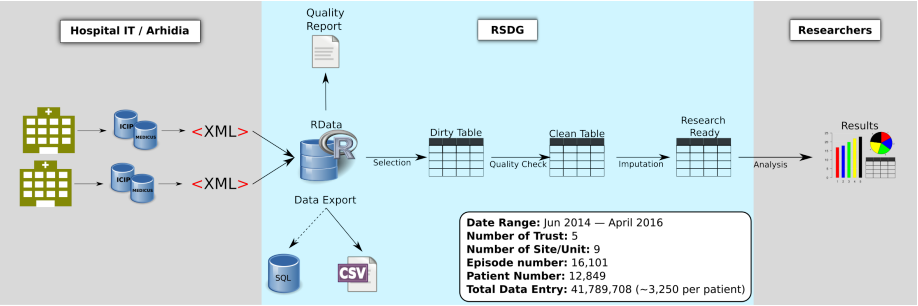

Figure 1: The Data Process Pipeline

2.1 The XML data and the Data Safe Haven

Data associated with more than 12,000 patients from 5 NHS trusts are available in XML format. The data were drawn from the ICIP and Medicuse database which contains demographic, drugs, laboratory, nursing and physiology data. The de-identifiable data are processed locally and the identifiable data is on the UCL Data Safe haven (IDSH). In order to create the pipeline of on IDSH, we ordered and configured a UNIX virtual environment. Due to the limit capacity of IDSH, it is still useful to keep a local de-identifiable copy for development purpose. Our pipeline is designed to be portable on multiple platforms. It can be executed on both local environments and IDSH with either de-identifiable or identifiable data. ## XML parser

The XML parser in R combines and restructures the XML files into a newly defined R data structure **ccRecord**, which significantly improved the clarity of the data by organising data under tables and removing the redundancy of the XML files. **ccRecord** is designed as a flexible, simplified, and query-able data structure for critical care data. Data in **ccRecord** format will be eventually stored in a RData file which is about 500 times smaller than that of the XML files. In addition the data provenance is recored by each episode, henceforth we will be able to tell which file does each episode data comes from and when it has been parsed. The selected data fields can be exported as a CSV file for the Excel users.

It is worth mentioning that a C++ equivalent XML parser, which has a much better performance comparing to the R parser, is developed but not deployed due to the time constraint of the project. To incorporate the C++ parser into the pipeline will be a potential future improvement.

2.2 Auto-generated quality report

Data can have defects in many different ways. Therefore a synthetic quality assessment which may allow us to report back to the source of the data can be extremely useful. An automatic generated data quality report which reflects to quality issues have been developed. Based on the report, we will be able to tell the major missingness issues and some data sanity issues and the basic information such as the duration, sites, number of episodes, and number of patient.

2.3 Data validation and cleaning

Data validation and cleaning functions have been incorporated in the ccdata package. Data sanity check is conducted in many aspects regarding to the intrinsic property of the individual fields. There are three major validation, the numeric range, the text category, and the missingness. The users are required to fill a yaml form to guide the validation check. We will be able to flag the level of the data sanity and opt out the data that do not make sense or inform the users according to the yaml dictionary. We

will again go through the yaml dictionary in the next session. After this stage, we are able to deliver a "cleaned" and query-able R table to the researchers.

3 The R package: ccdata

The main part of the pipeline can be performed by the functions in the R package ccdata. A package bundles all code, documentation and tests together, which makes the code sharing easy. ccdata is portable in almost all platforms where R environment is provided. It can be installed effortlessly on Windows or Unix based systems. Although it is necessary to have some further tidy-ups in the subsequent development cycle, in current stage, the main part of the R code is well documented and properly tested.

The data manipulation functions are all bundled in the

3.1 Selection and cleaning yaml configuration

Here is an example of the yaml data cleaning configuration of heart rate, in which three filters **nodata**, **range**, **missingness** are presented in the following data cleaning configuration for heart rate. The yaml form is straight forward and self-explanatory for non-programmers.

```
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

wrap up