

Chapter 11

Data Preparation

Tom Pollard, Franck Deroncourt, Samuel Finlayson
and Adrian Velasquez

Learning Objectives

- Become familiar with common categories of medical data.
- Appreciate the importance of collaboration between caregivers and data analysts.
- Learn common terminology associated with relational databases and plain text data files.
- Understand the key concepts of reproducible research.
- Get practical experience in querying a medical database.

11.1 Introduction

Data is at the core of all research, so robust data management practices are important if studies are to be carried out efficiently and reliably. The same can be said for the management of the software used to process and analyze data. Ensuring good practices are in place at the beginning of a study is likely to result in significant savings further down the line in terms of time and effort [1, 2].

While there are well-recognized benefits in tools and practices such as version control, testing frameworks, and reproducible workflows, there is still a way to go before these become widely adopted in the academic community. In this chapter we discuss some key issues to consider when working with medical data and highlight some approaches that can make studies collaborative and reproducible.

11.2 Part 1—Theoretical Concepts

11.2.1 Categories of Hospital Data

Data is routinely collected from several different sources within hospitals, and is generally optimized to support clinical activities and billing rather than research. Categories of data commonly found in practice are summarized in Table 11.1 and discussed below:

- Billing data generally consists of the codes that hospitals and caregivers use to file claims with their insurance providers. The two most common coding systems are the International Statistical Classification of Diseases and Related

Table 11.1 Overview of common categories of hospital data and common issues to consider during analysis

Category	Examples	Common issues to consider
Demographics	Age, gender, ethnicity, height, weight	Highly sensitive data requiring careful de-identification. Data quality in fields such as ethnicity may be poor
Laboratory	Creatinine, lactate, white blood cell count, microbiology results	Often no measure of sample quality. Methods and reagents used in tests may vary between units and across time
Radiographic images and associated reports	X-rays, computed tomography (CT) scans, echocardiograms	Protected health information, such as names, may be written on slides. Templates used to generate reports may influence content
Physiologic data	Vital signs, electrocardiography (ECG) waveforms, electroencephalography (EEG) waveforms	Data may be pre-processed by proprietary algorithms. Labels may be inaccurate (for example, “fingerstick glucose” measurements may be made with venous blood)
Medication	Prescriptions, dose, timing	May list medications that were ordered but not given. Time stamps may describe point of order not administration
Diagnosis and procedural codes	International Classification of Diseases (ICD) codes, Diagnosis Related Groups (DRG) codes, Current Procedural Terminology (CPT) codes	Often based on a retrospective review of notes and not intended to indicate a patient’s medical status. Subject to coder biases. Limited by suitability of codes
Caregiver and procedural notes	Admission notes, daily progress notes, discharge summaries, Operative reports	Typographical errors. Context is important (for example, diseases may appear in discussion of family history). Abbreviations and acronyms are common

Health Problems, commonly abbreviated the International Classification of Disease (ICD), which is maintained by the World Health Organization, and the Current Procedural Terminology (CPT) codes maintained by the American Medical Association. These hierarchical terminologies were designed to provide standardization for medical classification and reporting.

- Charted physiologic data, including information such as heart rate, blood pressure, and respiratory rate collected at the bedside. The frequency and breadth of monitoring is generally related to the level of care. Data is often archived at a lower rate than it is sampled (for example, every 5–10 min) using averaging algorithms which are frequently proprietary and undisclosed.
- Notes and reports, created to record patient progress, summaries a patient stay upon discharge, and provide findings from imaging studies such as x-rays and echocardiograms. While the fields are “free text”, notes are often created with the help of a templating system, meaning they may be partially structured.
- Images, such as those from x-rays, computerized axial tomography (CAT/CT) scans, echocardiograms, and magnetic resonance imaging.
- Medication and laboratory data. Orders for drugs and laboratory studies are entered by the caregiver into a physician order entry system, which are then fulfilled by laboratory or nursing staff. Depending on the system, some timestamps may refer to when the physician placed the order and others may refer to when the drug was administered or the lab results were reported. Some drugs may be administered days or weeks after first prescribed while some may not be administered at all.

11.2.2 Context and Collaboration

One of the greatest challenges of working with medical data is gaining knowledge of the context in which data is collected. For this reason we cannot emphasize enough the importance of collaboration between both hospital staff and research analysts. Some examples of common issues to consider when working with medical data are outlined in Table 11.1 and discussed below:

- Billing codes are not intended to document a patient’s medical status or treatment from a clinical perspective and so may not be reliable [3]. Coding practices may be influenced by issues such as financial compensation and associated paperwork, deliberately or otherwise.
- Timestamps may differ in meaning for different categories of data. For example, a timestamp may refer to the point when a measurement was made, when the measurement was entered into the system, when a sample was taken, or when results were returned by a laboratory.
- Abbreviations and misspelled words appear frequently in free text fields. The string “pad”, for example, may refer to either “peripheral artery disease” or to an

absorptive bed pad, or even a diaper pad. In addition, notes frequently mention diseases that are found in the patient's family history, but not necessarily the patient, so care must be taken when using simple text searches.

- Labels that describe concepts may not be accurate. For example, during preliminary investigations for an unpublished study to assess accuracy of fingertip glucose testing, it was discovered that caregivers would regularly take “fingerstick glucose” measurements using vascular blood where it was easily accessible, to avoid pricking the finger of a patient.

Each hospital brings its own biases to the data too. These biases may be tied to factors such as the patient populations served, the local practices of caregivers, or to the type of services provided. For example:

- Academic centers often see more complicated patients, and some hospitals may tend to serve patients of a specific ethnic background or socioeconomic status.
- Follow up visits may be less common at referral centers and so they may be less likely to detect long-term complications.
- Research centers may be more likely to place patients on experimental drugs not generally used in practice.

11.2.3 Quantitative and Qualitative Data

Data is often described as being either quantitative or qualitative. Quantitative data is data that can be measured, written down with numbers and manipulated numerically. Quantitative data can be discrete, taking only certain values (for example, the integers 1, 2, 3), or continuous, taking any value (for example, 1.23, 2.59). The number of times a patient is admitted to a hospital is discrete (a patient cannot be admitted 0.7 times), while a patient's weight is a continuous (a patient's weight could take any value within a range).

Qualitative data is information which cannot be expressed as a number and is often used interchangeably with the term “categorical” data. When there is not a natural ordering of the categories (for example, a patient's ethnicity), the data is called nominal. When the categories can be ordered, these are called ordinal variables (for example, severity of pain on a scale). Each of the possible values of a categorical variable is commonly referred to as a level.

11.2.4 Data Files and Databases

Data is typically made available through a database or as a file which may have been exported from a database. While there are many different kinds of databases and data files in use, relational databases and comma separated value (CSV) files are perhaps the most common.

Comma Separated Value (CSV) Files

Comma separated value (CSV) files are a plain text format used for storing data in a tabular, spreadsheet-style structure. While there is no hard and fast rule for structuring tabular data, it is usually considered good practice to include a header row, to list each variable in a separate column, and to list observations in rows [4].

As there is no official standard for the CSV format, the term is used somewhat loosely, which can often cause issues when seeking to load the data into a data analysis package. A general recommendation is to follow the definition for CSVs set out by the Internet Engineering Task Force in the RFC 4180 specification document [5]. Summarized briefly, RFC 4180 specifies that:

- files may optionally begin with a header row, with each field separated by a comma;
- Records should be listed in subsequent rows. Fields should be separated by commas, and each row should be terminated with a line break;
- fields that contain numbers may be optionally enclosed within double quotes;
- fields that contain text (“strings”) should be enclosed within double quotes;
- If a double quote appears inside a string of text then it must be escaped with a preceding double quote.

The CSV format is popular largely because of its simplicity and versatility. CSV files can be edited with a text editor, loaded as a spreadsheet in packages such as Microsoft Excel, and imported and processed by most data analysis packages. Often CSV files are an intermediate data format used to hold data that has been extracted from a relational database in preparation for analysis. Figure 11.1 shows an annotated example of a CSV file formatted to the RFC 4180 specification.

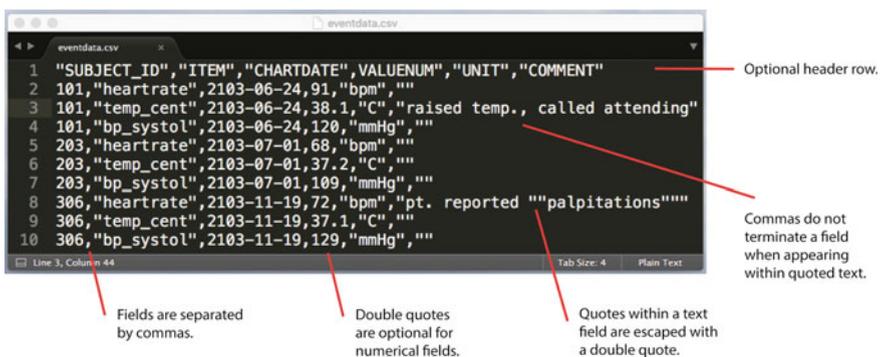


Fig. 11.1 Comma separated value (CSV) file formatted to the RFC 4180 specification

Relational Databases

There are several styles of database in use today, but probably the most widely implemented is the “relational database”. Relational databases can be thought of as a collection of tables which are linked together by shared keys. Organizing data across tables can help to maintain data integrity and enable faster analysis and more efficient storage.

The model that defines the structure and relationships of the tables is known as a “database schema”. Giving a simple example of a hospital database with four tables, it might comprise of: Table 1, a list of all patients; Table 2, a log of hospital admissions; Table 3, a list of vital sign measurements; Table 4, a dictionary of vital sign codes and associated labels. Figure 11.2 demonstrates how these tables can be linked with primary and foreign keys. Briefly, a primary key is a unique identifier within a table. For example, `subject_id` is the primary key in the `patients` table,

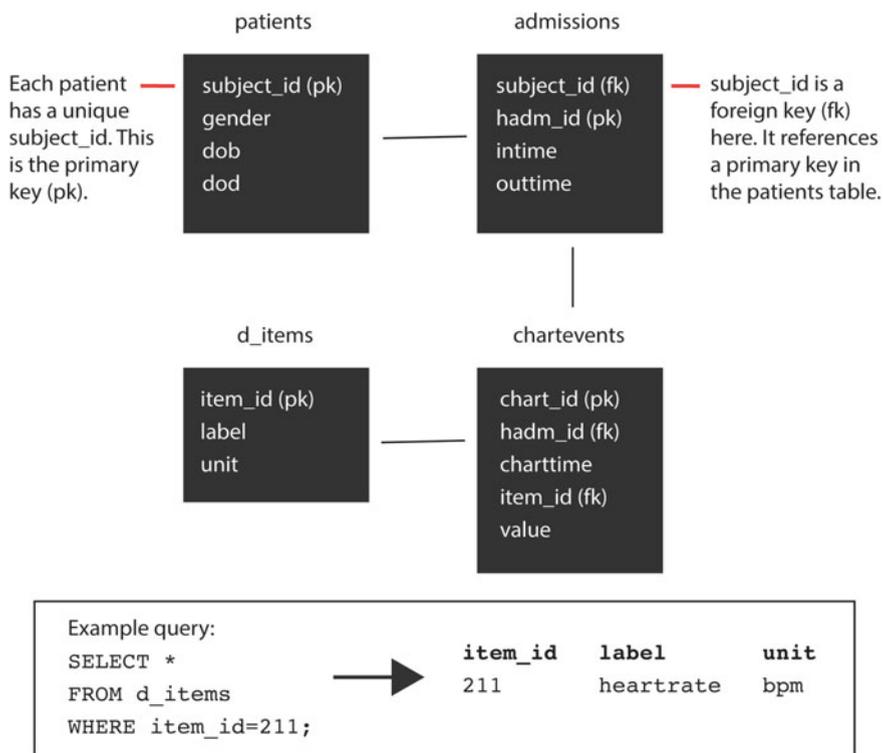


Fig. 11.2 Relational databases consist of multiple data tables linked by primary and foreign keys. The `patients` table lists unique patients. The `admissions` table lists unique hospital admissions. The `chartevents` table lists charted events such as heart rate measurements. The `d_items` table is a dictionary that lists `item_ids` and associated labels, as shown in the example query. *pk* is primary key. *fk* is foreign key

because each patient is listed only once. A foreign key in one table points to a primary key in another table. For example, `subject_id` in the `admissions` table is a foreign key, because it references the primary key in the `patients` table.

Extracting data from a database is known as “querying” the database. The programming language commonly used to create a query is known as “Structured Query Language” or SQL. While the syntax of SQL is straightforward, queries are at times challenging to construct as a result of the conceptual reasoning required to join data across multiple tables.

There are many different relational database systems in regular use. Some of these systems such as Oracle Database and Microsoft SQL Server are proprietary and may have licensing costs. Other systems such as PostgreSQL and MySQL are open source and free to install. The general principle behind the databases is the same, but it is helpful to be aware that programming syntax varies slightly between systems.

11.2.5 Reproducibility

Alongside a publishing system that emphasizes interpretation of results over detailed methodology, researchers are under pressure to deliver regular “high-impact” papers in order to sustain their careers. This environment may be a contributor to the widely reported “reproducibility crisis” in science today [6, 7].

Our response should be to ensure that studies are, as far as possible, reproducible. By making data and code accessible, we can more easily detect and fix inevitable errors, help each other to learn from our methods, and promote better quality research.

When practicing reproducible research, the source data should not be modified. Editing the raw data destroys the chain of reproducibility. Instead, code is used to process the data so that all of the steps that take an analysis from source to outcome can be reproduced.

Code and data should be well documented and the terms of reuse should be made clear. It is typical to provide a plain text “README” file that gives an introduction to the analysis package, along with a “LICENSE” file describing the terms of reuse. Tools such as Jupyter Notebook, Sweave, and Knitr can be used to interweave code and text to produce clearly documented, reproducible studies, and are becoming increasingly popular in the research community (Fig. 11.3).

Version control systems such as Git can be used to track the changes made to code over time and are also becoming an increasingly popular tool for researchers [8]. When working with a version control system, a commit log provides a record of changes to code by contributor, providing transparency in the development process and acting as a useful tool for uncovering and fixing bugs.

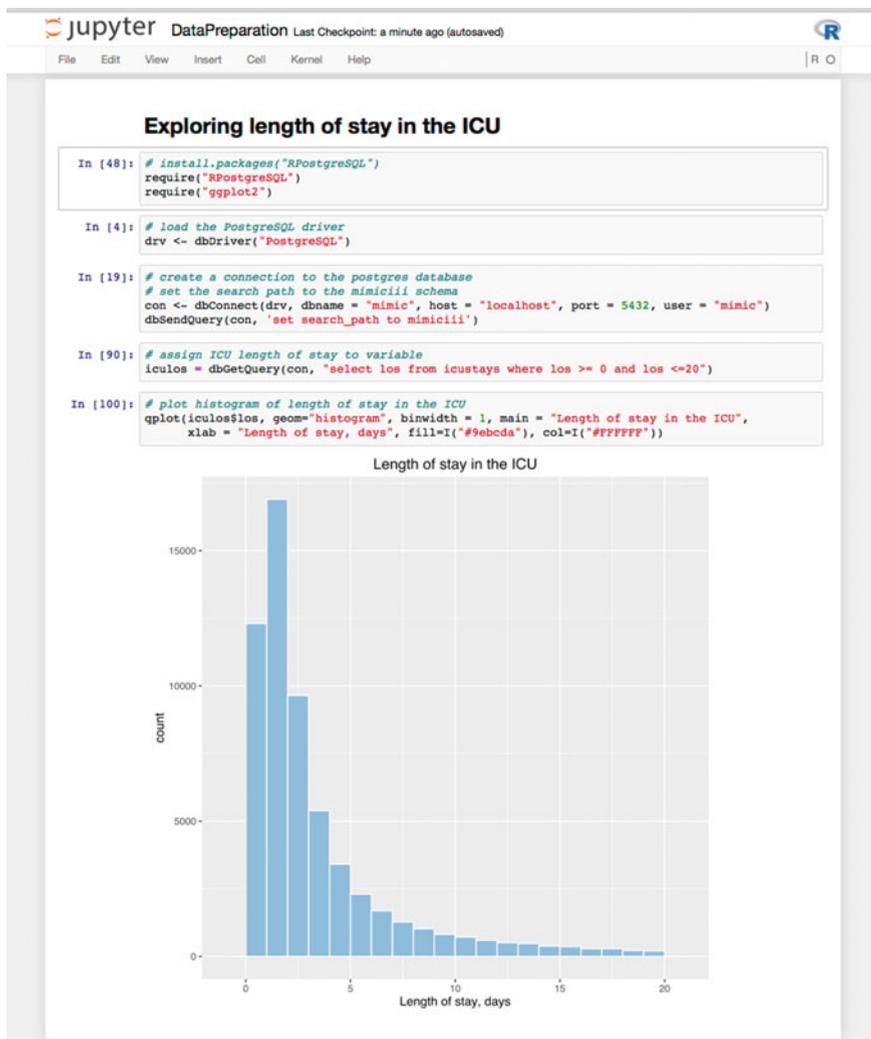


Fig. 11.3 Jupyter Notebooks enable documentation and code to be combined into a reproducible analysis. In this example, the length of ICU stay is loaded from the MIMIC-III (v1.3) database and plotted as a histogram [11]

Collaboration is also facilitated by version control systems. Git provides powerful functionality that facilitates distribution of code and allows multiple people to work together in synchrony. Integration with Git hosting services such as Github provide a simple mechanism for backing up content, helping to reduce the risk of data loss, and also provide tools for tracking issues and tasks [8, 9].

11.3 Part 2—Practical Examples of Data Preparation

11.3.1 MIMIC Tables

In order to carry out the study on the effect of indwelling arterial catheters as described in the previous chapter, we use the following tables in the MIMIC-III clinical database:

- The `chartevents` table, the largest table in the database. It contains all data charted by the bedside critical care system, including physiological measurements such as heart rate and blood pressure, as well as the settings used by the indwelling arterial catheters.
- The `patients` table, which contains the demographic details of each patient admitted to an intensive care unit, such as gender, date of birth, and date of death.
- The `icustays` table, which contains administrative details relating to stays in the ICU, such as the admission time, discharge time, and type of care unit.

Before continuing with the following exercises, we recommend familiarizing yourself with the MIMIC documentation and in particular the table descriptions, which are available on the MIMIC website [10].

11.3.2 SQL Basics

An SQL query has the following format:

```
SELECT [columns]
FROM [table_name]
WHERE [conditions];
```

The result returned by the query is a list of rows. The following query lists the unique patient identifiers (`subject_ids`) of all female patients:

```
SELECT subject_id
FROM patients
WHERE gender = 'F';

-- returns:
subject_id
-----
        654
        655
        656
        ...
```

We often need to specify more than one condition. For instance, the following query lists the `subject_ids` whose first or last care unit was a coronary care unit (CCU):

```
SELECT subject_id
FROM icustays
WHERE first_careunit = 'CCU' OR last_careunit = 'CCU';

-- returns:
  subject_id
-----
         109
         109
         111
         ...
```

Since a patient may have been in several ICUs, the same patient ID sometimes appears several times in the result of the previous query. To return only distinct rows, use the `DISTINCT` keyword:

```
SELECT DISTINCT subject_id
FROM icustays
WHERE first_careunit = 'CCU' OR last_careunit = 'CCU';

-- returns:
  subject_id
-----
       25949
       6158
       27223
       ...
```

To count how many patients there are in the `icustays` table, combine `DISTINCT` with the `COUNT` keyword. As you can see, if there is no condition, we simply don't use the keyword `WHERE`:

```
SELECT COUNT(DISTINCT subject_id)
FROM icustays;

-- returns:
  count
-----
  46476
```

Taking a similar approach, we can count how many patients went through the CCU using the query:

```
SELECT COUNT(DISTINCT subject_id)
FROM icustays
WHERE first_careunit = 'CCU' OR last_careunit = 'CCU';

-- returns:
count
-----
  7314
```

The operator `*` is used to display all columns. The following query displays the entire `icustays` table:

```
SELECT *
FROM icustays;

-- returns
subject_id | hadm_id | icustay_id | ...
          109 | 139061 | 257358 | ...
          109 | 172335 | 262652 | ...
          109 | 126055 | 236124 | ...
...

```

The results can be sorted based on one or several columns with `ORDER BY`. To add a comment in a SQL query, use:

```
SELECT subject_id, hadm_id, icustay_id
FROM icustays
ORDER BY subject_id ASC; -- ASC sorts by ascending number

-- returns:
subject_id | hadm_id | icustay_id
-----+-----+-----
          2 | 163353 | 243653
          3 | 145834 | 211552
          4 | 185777 | 294638
...

```

11.3.3 Joins

Often we need information coming from multiple tables. This can be achieved using SQL joins. There are several types of join, including **INNER JOIN**, **OUTER JOIN**, **LEFT JOIN**, and **RIGHT JOIN**. It is important to understand the difference between these joins because their usage can significantly impact query results. Detailed guidance on joins is widely available on the web, so we will not go into further details here. We will however provide an example of an **INNER JOIN** which selects all rows where the joined key appears in both tables.

Using the **INNER JOIN** keyword, let's count how many adult patients went through the coronary care unit. To know whether a patient is an adult, we need to use the **dob** (date of birth) attribute from the **patients** table. We can use the **INNER JOIN** to indicate that two or more tables should be combined based on a common attribute, which in our case is **subject_id**:

```
-- INNER JOIN will only return rows where subject_id
-- appears in the patients table and the icustays table
SELECT p.subject_id
FROM patients p
INNER JOIN icustays i
ON p.subject_id = i.subject_id
WHERE (i.first_careunit = 'CCU' OR i.last_careunit = 'CCU')
      AND (i.intime - p.dob) >= INTERVAL '18' year
ORDER BY subject_id ASC;

-- returns:
subject_id
-----
          13
          18
          21
          ...
```

Note that:

- we assign an alias to a table to avoid writing its full name throughout the query. In our 0 given the alias 'p'.
- in the **SELECT** clause, we wrote **p.subject_id** instead of simply **subject_id** since both the **patients** and **icustays** tables contain the attribute **subject_id**. If we don't specify from which table **subject_id** comes from, we would get a "column ambiguously defined" error.
- to identify whether a patient is an adult, we look for differences between **intime** and **dob** of 18 years or greater using the **INTERVAL** keyword.

11.3.4 Ranking Across Rows Using a Window Function

We now focus on the case study. One of the first steps is identifying the first ICU admission for each patient. To do so, we can use the `RANK ()` function to order rows sequentially by `intime`. Using the `PARTITION BY` expression allows us to perform the ranking across `subject_id` windows:

```
SELECT subject_id, icustay_id, intime,
       RANK() OVER (PARTITION BY subject_id ORDER BY intime asc)
FROM icustays;
```

-- returns:

subject_id	icustay_id	intime	rank
6	228232	2175-05-30 21:30:54	1
7	278444	2121-05-23 15:35:29	1
7	236754	2121-05-25 03:26:01	2
...			

11.3.5 Making Queries More Manageable Using WITH

To keep SQL queries reasonably short and simple, we can use the `WITH` keyword. `WITH` allows us to break a large query into smaller, more manageable chunks. The following query creates a temporary table called “rankedstays” that lists the order of stays for each patient. We then select only the rows in this table where the rank is equal to one (i.e. the first stay) and the patient is aged 18 years or greater:

```
WITH rankedstays AS (
  SELECT subject_id, icustay_id, intime,
         RANK() OVER (PARTITION BY subject_id ORDER BY intime asc)
  FROM icustays
)
```

```
SELECT r.subject_id, r.icustay_id, r.intime, r.rank
FROM rankedstays r
INNER JOIN patients p
ON r.subject_id = p.subject_id
WHERE r.rank = 1
AND (r.intime - p.dob) >= INTERVAL '18' year;
```

-- returns:

subject_id	icustay_id	intime	rank
3	211552	2101-10-20 19:10:11	1
4	294638	2191-03-16 00:29:31	1
6	228232	2175-05-30 21:30:54	1
...			

Open Access This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, duplication, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, a link is provided to the Creative Commons license and any changes made are indicated.

The images or other third party material in this chapter are included in the work's Creative Commons license, unless indicated otherwise in the credit line; if such material is not included in the work's Creative Commons license and the respective action is not permitted by statutory regulation, users will need to obtain permission from the license holder to duplicate, adapt or reproduce the material.

References

1. Wilson G, Aruliah DA, Brown CT, Chue Hong NP, Davis M, Guy RT et al (2014) Best practices for scientific computing. *PLoS Biol* 12(1):e1001745. doi:10.1371/journal.pbio.1001745. <http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001745>
2. Editorial (2012) Must try harder. *Nature* 483(509). doi:10.1038/483509a. <http://www.nature.com/nature/journal/v483/n7391/full/483509a.html>
3. Misset B, Nakache D, Vesin A, Darmon M, Garrouste-Orgeas M, Mourvillier B et al (2008) Reliability of diagnostic coding in intensive care patients. *Crit Care* 12(4):R95. <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2575581/>
4. Wickham H (2014) Tidy data. *J Stat Softw* 59(10):1–23. doi:10.18637/jss.v059.i10. <https://www.jstatsoft.org/article/view/v059i10>
5. Sustainability of Digital Formats Planning for Library of Congress Collections. Accessed: 24 Feb 2016. CSV, Comma Separated Values (RFC 4180). <http://www.digitalpreservation.gov/formats/fdd/fdd000323.shtml>
6. Editorial (2013) Unreliable research: trouble at the Lab. *Economist*. <http://www.economist.com/news/briefing/21588057-scientists-think-science-self-correcting-alarming-degree-it-not-trouble>
7. Goodman A, Pepe A, Blocker AW, Borgman CL, Cranmer K, Crosas M, et al (2014) Ten simple rules for the care and feeding of scientific data. *PLoS Comput Biol* 10(4):e1003542. doi:10.1371/journal.pcbi.1003542. <http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003542>
8. Karthik R (2013) Git can facilitate greater reproducibility and increased transparency in science. *Source Code Biol Med* 28; 8(1):7. doi:10.1186/1751-0473-8-7. <http://scfbm.biomedcentral.com/articles/10.1186/1751-0473-8-7>
9. GitHub. <https://github.com>. Accessed 24 Feb 2016
10. MIMIC website. <http://mimic.physionet.org>. Accessed 24 Feb 2016
11. MIMIC Code Repository. <https://github.com/MIT-LCP/mimic-code>. Accessed 24 Feb 2016