Preface

April 1, 2020

1 Cleaning Data for Effective Data Science

> In order for something to become clean, something else must become dirty.–Imbesi's Law of the Conservation of Filth

1.1 Doing the other 80% of the work

It is something of a truism in data science, data analysis, or machine learning that most of the work needed to do your actual work lies in cleaning your data. The subtitle of this work alludes to a commonly assigned percentage. A keynote speaker I listened to at a data science conference a few years ago made a joke—perhaps one already widely repeated by the time he told it—about talking with a colleague of his. The colleague complained of data cleaning taking up half of her time, in response to which the speaker expressed astonishment that it could be so little as 50%.

Without worrying too much about assigning a precise percentage, in my experience working as a technologist and data scientist, I have found that the bulk of what I do is preparing my data for the statistical analyses, machine learning models, or nuanced visualizations that I would like to utilize it for. Although hopeful executives, or technical managers a bit removed from the daily work, tend to have an eternal optimism that the next set of data the organization acquires will be clean and easy to work with, I have yet to find that to be true in my concrete experience.

Certainly, some data is better and some is worse. But *all data is dirty*, at least within a very small margin of error in the tally. Even data sets that have been published, carefully studied, and that are widely distributed as canonical examples for statistics textbooks or software libraries, generally have a moderate number of data integrity problems. Even after our best pre-processing, a more attainable goal should be to make our data *less dirty*; making it *clean* remains unduly utopian in aspiration.

By all means we should distinguish *data quality* from *data utility*. These descriptions are roughly orthogonal to each other. Data can be dirty (up to a point) but still be enormously useful. Data can be (relatively) clean but have little purpose, or at least not be fit for purpose. Concerns about the choice of measurements to collect, or about possible selection bias, or other methodological or scientific questions are mostly outside the scope of this book. However, a fair number of techniques I present can *aid* in evaluating the utility of data, but there is often no mechanical method of remedying systemic issues. For example, statistics and other analyses may reveal—or at least strongly suggest—the unreliability of a certain data field. But the techniques in this book cannot generally automatically fix that unreliable data or collect better data.

1.2 Types of grime

There are roughly two families of problems we find in data sets. Not every problem neatly divides into these families, or at least it is not always evident which side something falls on without knowing the root cause. But in a general way we can think of structural problems in the formatting of data versus content problems in the actual values recorded. On the structural branch a format used to encode a data set might simply "put values in the wrong place" in one way or another. On the content side, the data format itself is correct, but implausible or wrong values have snuck in via flawed instruments, transcription errors, numeric overflows, or through other pitfalls of the recording process.

The several early chapters that discuss "data ingestion" are much more focused on structural problems in data sources, and less on numeric or content problems. It is not always cleanly possible to separate these issues, but as a question of emphasis it makes sense for the ingestion chapters to look at structural matters, and for later chapters on anomalies, data quality, feature engineering, value imputation, and model-based cleaning to direct attention to content issues.

In the case of structural problems, we almost always need manual remediation of the data. Exactly where the bytes that make up the data go wrong can vary enormously, and usually does not follow a pattern that lends itself to a single high level description. Often we have a somewhat easier time with the content problems, but at the same time they are more likely to be irremediable even with manual work. Consider this small comma-separated value (CSV) data source, describing a 6th grade class:

```
Student#,Last Name,First Name,Favorite Color,Age
1,Johnson,Mia,periwinkle,12
2,Lopez,Liam,blue,green,13
3,Lee,Isabella,,11
4,Fisher,Mason,gray,-1
5,Gupta,Olivia,9,102
6,,Robinson,,Sophia,,blue,,12
```

In a friendly way, we have a header line that indicates reasonable field names and provides a hint as to the meaning of each column. Programmatically, we may not wish to work with the punctuation marks and spaces inside some field names, but that is a matter of tool convenience that we can address with the APIs that data processing tools give us (perhaps by renaming them).

Let us think about each record in turn. Mia Johnson, student 1, seems to have a problem-free record. Her row has five values separated by four commas, and each data value meets our intuitive expectations about the data type and value domain. The problems start hereafter.

Liam Lopez has too many fields in his row. However, both columns 4 and 5 seem clearly to be in the lexicon of color names. Perhaps a duplicate entry occurred or the compound color "blue-green" was intended. Structurally the row has issues, but several plausible remediations suggest themselves.

Isabella Lee is perhaps no problem at all. One of her fields is empty, meaning no favorite color is available. But structurally, this row is perfectly fine for CSV format. We will need to use some domain or problem knowledge to decide how to handle the missing value.

Mason Fisher is perhaps similar to Isabella. The recorded age of -1 makes no sense in the nature of "age" as a data field, at least as we usually understand it (but maybe the encoding intends something different). On the other hand, -1 is one of several placeholder values used very commonly

to represent missing data. We need to know our specific problem to know whether we can process the data with a missing age, but many times we can handle that. However, we still need to be careful not to treat the -1 as a plain value; for example, the mean, minimum, or standard deviation of ages might be thrown off by that.

Olivia Gupta starts to present a trickier problem. Structurally her row looks perfect. But '9' is probably not a string in our lexicon of color names. And under our understanding of the data concerning a 6th grade class, we don't expect 102 year old students to be in it. To solve this row, we really need to know more about the collection procedure and the intention of the data. Perhaps a separate mapping of numbers to colors exists somewhere. Perhaps an age of 12 was mistranscribed as 102; but also perhaps a 102 year old serves as a teaching assistant in this class and not only students are recorded.

Sophia Robinson returns us to what looks like an obvious structural error. The row, upon visual inspection, contains perfectly good and plausible values, but they are separated by duplicate commas. Somehow, persumably, a mechanical error resulted in the line being formatted wrongly. However, most high-level tools are likely to choke on the row in an uninformative way, and we will probably need to remediate the issue more manually.

We have a pretty good idea what to do with these six rows of data, and even re-entering them from scratch would not be difficult. If we had a million rows instead, the difficulty would grow greatly, and would require considerable effort before we arrived at usable data.

1.3 Nomenclature

In this book I will use the terms *feature*, *field*, *measurement*, *column*, and occasionally *variable* more-or-less interchangeably. Likewise, the terms *row*, *record*, *observation*, and *sample* are also near synonyms. *Tuple* is used for the same concept when discussing databases. In different academic or business fields, different ones of these terms are more prominent; and likewise different software tools choose among these.

Conceptually, most data can be thought of as a number of occasions on which we measure various attributes of a common underlying *thing*. In most tools, it is usally convenient to put these observations/samples each in a row; and correspondingly to store each of the measurements/features/fields pertaining to that thing in a column containing corresponding data for other comparable *things*.

Inasmuch as I vary my use of these roughly equivalent terms, it is simply to fit better with the domain under discussion and to make readers familiar with all the terms, which they are likely to encounter in various places for a similar intention. The choice among near synonyms is also guided by the predominant use within the particular tool, library, or programming community that is currently being discussed.

In many cases, a general concept has a strong overlap with the particular name a tool or library uses to implement or express that concept. Where relevant, I attempt to use the small typographic distinctions in the names to indicate focus. For example, I discuss *data frames* as a general paradigm for manipulating data, but refer to *DataFrame* when discussing Pandas or other libraries that use that spelling for the specific class used. Likewise, R's *data.frame* object is a specific implementation of the paradigm, and capitalization and punctuation will be adjusted for context. Similarly, in generically discussing a collection of associated data, I describe that as a *data set*; but when discussing the specific array-like object in HDF5, I use the spelling *dataset*.

1.4 Typography

As with most programming books, code literals will be set in a `fixed width` font, whether as excerpts inline or as blocks of code between paragraphs. Names of software libraries and tools will be shown in **boldface** on first mention, but generally in the default typeface as common nouns elsewhere. Where a name is used infrequently or special attention is drawn to its use in nearby discussion, the boldface may be repeated. *Italics* are used in places in the main text simply for emphasis of words or clauses in prose. Usually a term that is used in a special or distinctive sense within data science is italicized on first, and sometimes subsequent, use.

The names of software tools and libraries is a bit of a challenge to orthography. Capitalization, or lack thereof, is often used in a stylized way, and moreover sometimes these bits of sofware are rendered differently in different contexts. For example **Python** is a good proper name for a programming language, but the actual executable that launches a Python script is `python` in lower case. Such tools or libraries that will usually be typed in literal form at a command-line or as a name in code that will be set in fixed width.

Still other tools have both an informal and a literal name. For example **scikit-learn** is stylized in lowercase, but is not the actual imported name of the library, which is `sklearn`. Moreover, the informal name would look out of place when referring to subpackages such as `sklearn.preprocessing`. In general, the names of software libraries are actually pretty intuitive, but a short appendix lists the name variants used in slightly different contexts in this book.

1.5 Included code

In this book, I will primarily use Python and associated tools, such as **Pandas**, **sklearn.preprocessing**, and **scipy.stats**, to solve the data cleaning problems presented. **R**, and its **Tidyverse** tools, will often be shown as code alternatives. Some code samples will simply use **bash** and the many text/data command-line processing tools available. Examples from other programming languages are occasionally mentioned, where relevant.

All of the code in this book is released to the Public Domain, or as Creative Commons CC0 if your jurisdiction lacks a clear mechanism for placing content in the Public Domain. The repository https://bitbucket.org/davidmertz/cleaning-data contains the code directly printed in this book, and small modules or libraries supporting techniques demonstrated under the same terms. All of the data sets utilized are provided at the same location. Some data sets may have different license terms, but only ones with reasonably open terms for use and modification are utilized. Because data sets are often large, this book will only reproduce directly very small data sets; I will often show a few representative sections of larger data in the text.

Although code shown is freely available, the purpose of this book is not learning to use the particular tools used for illustration, but to understand the underlying purpose of data quality. The concepts presented should be applicable in any programming language used for data processing and machine learning. I hope it will be easy to adapt the techniques I show to your own favorite collection of tools and programming languages.

1.6 Running the book

This book is itself written using Jupyter notebooks. This manner of creation allows for (almost) all the code within the book to be actively run before publication. The repository given above provides

instructions and configuration files for creating a similar working environment. Code samples shown will usually be accompanied by the actual output of running them. For example, Python code:

[1]:
```
from src.intro_students import data, cleaned
print(data)
cleaned
```

```
Student#,Last Name,First Name,Favorite Color,Age
1,Johnson,Mia,periwinkle,12
2,Lopez,Liam,blue,green,13
3,Lee,Isabella,,11
4,Fisher,Mason,gray,-1
5,Gupta,Olivia,9,102
6,,Robinson,,Sophia,,blue,,12
```

[1]:

Student_No	Last_Name	First_Name	Favorite_Color	Age
1	Johnson	Mia	periwinkle	12.0
2	Lopez	Liam	blue-green	13.0
3	Lee	Isabella	<missing>	11.0
4	Fisher	Mason	gray	NaN
5	Gupta	Olivia	sepia	NaN
6	Robinson	Sophia	blue	12.0

Likewise in this configuration, I can run R code equally well. At times the code samples will show data being transferred between the R and Python kernels.

[2]:
```
%load_ext rpy2.ipython
```

[3]:
```
%%R -i cleaned
library('tibble')
tibble(First=cleaned$First_Name,
       Last=cleaned$Last_Name,
       Age=cleaned$Age)
```

```
# A tibble: 6 x 3
  First    Last       Age
  <chr>    <chr>
<dbl>
1 Mia      Johnson     12
2 Liam     Lopez       13
3 Isabella Lee         11
4 Mason    Fisher     NaN
5 Olivia   Gupta      NaN
6 Sophia   Robinson    12
```

Command-line tools will also be shown within code cells, for example:

[4]:
```
!sed s/,,/,/g data/students.csv \
  | cut -f2,3 -d, \
  | tail -n +2 \
  | tr , ' ' \
  | sort
```

```
Fisher Mason
Gupta Olivia
Johnson Mia
Lee Isabella
Lopez Liam
Robinson Sophia
```

1.7 Using this book

> Slovenliness is no part of data science… cleanliness is indeed next to godliness.–c.f. John Wesley

This book is intended to be suitable for use either by self-directed reader or in more structured academic, training, or certification courses. Each chapter is accompanied by exercises at the bottom that ask readers or students to complete tasks related to what they just learned in the preceding material. Appendices will contain additional discussion of some exercises, but will avoid presenting explicit solutions for mere copy-paste.

Instructors are encouraged to contact the author if they wish to plan course material around this book. Under a consulting arrangement, I am happy to provide solution code, suggestions on use of the exercises and other content, and so on.

The datasets and supporting materials for this book are available at the repository described above, and will be needed to engage fully with some of the more open ended problems presented. These extra materials will allow more interactive use of the book, and accompanying materials, than reading only would allow. However, sufficient explanation to understand the content based on the written material only will also be provided in the text.

Throughout this book I am *strongly opinionated* about a number of technical questions. I do not believe it will be difficult to distinguish my opinions from the *mere facts* I also present. I have worked in this area for a good number of years, and I hope to share with readers the conclusions I have reached. Of course, even book authors are fallible beings, and if you decide to disagree with claims I make, I hope and wish that you will gain great benefit both from what you learn anew and what you are able to reformulate in strengthening your own opinions and conclusions.

1.8 Exercises

The toy tabular data on students given as an example is available at:

> http://gnosis.cx/cleaning/students.csv

For this exercise, create a cleaned up version of the data following the assumptions illustrated in the code samples shown. Use your favorite programming language and tools, but the goal has these elements:

- Consistent doubled commas should be read as a single delimeter.

- Missing data in the *Favorite Color* field should be substituted with the string `<missing>`.
- Student ages should be between 9 and 14, and all other values are considered missing data.
- Some colors are numerically coded, but should be dealiased. The mapping is:

Number	Color	Number	Color
1	beige	6	alabaster
2	eggshell	7	sandcastle
3	seafoam	8	chartreuse
4	mint	9	sepia
5	cream	10	lemon

Using the small test data set is a good way to test your code. But try also manually adding more rows with similar, or different, problems in them, and see how well your code produces a reasonable result.