Chapter 4 Data Preparation:

Dr. Ami Gates
drgatesdatascience@gmail.com

Introduction

Data is the beginning of information discovery. To retrieve information from data, the data must first be prepared, then explored, then analyzed and modeled, and then its information can be presented.

Data can be generated through observational studies or through experimentation. Data collection may be passive, such as utilizing an API (application programming interface) to gather data that is managed and maintained by others, or data collection may be active such that it is directly generated by observation or experimentation and then collection. For example, many applications such as Twitter or FaceBook, and many government agencies such as the CDC or the EPA, all produce, store, and share (in limited ways) a considerable amount of data. To collect such shared data, an API can be used. Alternatively, a research group might create and distribute a survey whose results would then be organized one or more into datasets. No matter how data is gathered, and no matter how clean it may appear, all data must be properly and thoroughly prepared before and for analysis. The first step in this preparation is generally and broadly referred to as "cleaning".

All data, no matter how it originated, will require thoughtful and thorough cleaning and preparation before analysis can begin. Cleaning and preparation are intertwined with exploration, visualization, transformation, normalization, and reformatting. Data cleaning and preparation, much like data science and analytics, are not a linear process. Instead, the process is iterative and circular. Hurrying through the cleaning process or using cleaning methods that are not well thought out and applicable can result in poor or incorrect results and subsequently misleading information and decision-making. In some cases, potentially incorrect information resulting from poor data preparation can be highly dangerous, such as if the information is for health and wellness.

It can be argued that the process of cleaning and preparing data for analysis is, and should be, the most time consuming, most important, and most critical first step in data science, data analytics, and data visualization. Relative time spent on data cleaning and preparation is often stated to be between 60% and 80% (Dasn and Johnson, 2003).

This chapter will discuss data formats, data cleaning steps and methods, data preparation, including ensuring correct data types, and data transformation and normalization. Both R and Python code examples will be included at the end. It is recommended that the reader practice with all examples illustrated in R and Python, as this will reinforce the concepts and offer improved retention and enhanced comprehension.

Data Types and Formats and Applications in Cleaning and Preparation

Gathering Data

There are many avenues and methods that can be utilized to gather data. In addition, there is a difference between generating data and using existing data. The two general methods for generating data are observational and experimental.

Observational methods might include interviews, questionnaires, direct observation, focus groups, surveys, phone calls, and case studies. These methods are not mutually exclusive and can blend and overlap. The key to observational data generation is that the individual (or group) collecting the data does not alter or affect or try to alter or affect the outcome or environment; rather they only observe and record it. Alternatively, experimental methods generate data by interacting with (rather than just observing) a selected and effected sample and recording the outcomes and results of the alternations.

For example, if the outside ambient air quality PM10 (particulate matter 10 micrometers or less in diameter) is measured at 100 global locations and recorded, this data generation method is observational. Alternatively, if a leaf-blower is run and the ambient air quality (PM10) within 10, 20, and 30 yards from the leaf-blower is measured, this is an experimental method because the air quality is being purposely affected by the leaf-blower.

Gathering data can be active, through observation or experiment, or passive through the use of data already generated or gathered by previous observational or experimental processes. Using data from the Web, via an API, or from a database, is passive. Generating data is active. In either case, the data must then be properly cleaned, prepared, and formatted.

Data Formats

The nature or format of collected data, whether actively or passively acquired, can range from standard record/table data to unstructured, raw data. There are many data formats, such as record data, transaction data, network data, image data, sound data, video data, sequential ordered data, tagged data (such as XML or HTML), JSON data, text data, and raw unformatted data. Where the data originated and how it originated can often affect its format. Genomic and protein amino acid data must be sequential and will rarely be in record format. Data collected using the Twitter API via JSON will be in a language-independent parsable format that is very similar to a Python dictionary type. Data directly scraped from the source code of a website will be in HTML format. Data downloaded from data-housing sites like Kaggle will often be in record format.

When cleaning and preparing data, the current format and the desired format must be considered. For example, data from Twitter as JSON may be required or desired as record data. Images may be collected as jpg (Joint Photographic Experts Group) format may need to be converted to a integer matrix.

In addition, programming languages maintain their own data structures (formats for data objects). Examples include lists, data frames, vectors, matrices, tuples, dictionaries, and so on. While each programming language has its own uniquenesses, many programming language data structures overlap in similarity to common data formats. For example, Twitter JSON data is very similar to Python dictionaries, and record data is very similar to data frames (in both R and Python pandas). Network data can be stored in adjacency matrices. Proteins (linear sequences of amino acids) can be stored as lists.

Different functions or methods within programming languages also accept or expect data to be in specific formats. For example, some functions expect data as a data frame, while other functions may expect a matrix, and still others might expect a list. While this requirement can seem minor, having data in a format that does not properly connect with methods being used can cost hours (if not days) of time. Understanding data formats, programming language structures, and the requirements of methods and functions will assist in the successful coding of data science models.

While datasets can each be in a format (such as record, transaction, etc.), the data within a dataset also has its own “type,” such as integer, float/decimal/numeric, character/string, factor/category, and so on. The “*type”* of the data (often each variable or column in record data has its own type) is also closely intertwined with the data format, the programming structure used to represent the data, and especially the data cleaning and preparation. It is not possible to clean, prepare, and format data without knowing the current data or variable types and the desired types.

Data Types: Qualitative and Quantitative Data and Levels of Measurement

From a statistical standpoint, a “data type” may be quantitative or qualitative and from there can be a member of one of the four levels of measurement: nominal, ordinal, interval, or ratio. Quantitative data, or data that represents numeric quantities, can also be discrete or continuous. For example, the number of children in a family is quantitative and discrete. A family can have 1 child or 2 children, but certainly not 3.4 children. Alternatively, the mean number of children that families in each US state has is quantitative and continuous. The average number of children per family in Utah is 2.32 and the average in New Hampshire is 1.73. (These values are estimates and are based on the United States 2017 Census Bureau data, https://www.census.gov/programs-surveys/acs/news/data-releases/2017/release.html)

Quantitative data can also be ratio or interval. Ratio data is data for which the value of zero has an absolute meaning of none. For example, with respect to the earth, a car that is parked in a driveway has a speed of zero. This is called a “true zero” is the is no speed. Alternatively, the measure of sea-level is interval data. Sea level is considered to be at measure zero on the interval scale. However, it is possible to go “below” sea level. As such, the zero for sea level is not a true zero. It does not imply that it is impossible to go lower.

Qualitative data (based on the word quality or characteristic) can be nominal or ordinal. Qualitative and nominal data might be labels such as gender, political alignment, or hair color. Qualitative and ordinal data are data for which there is an order or hierarchy within the data. For example, restaurant rankings from 1 to 5 is qualitative and ordinal.

Data Types Matter in Programming Languages

Programming languages maintain a similar but individualized definition for both simple and complex data types. For example, qualitative and nominal data, that is not categorical, would be considered *character* data in R and *string* data in Python. An example of this might be people’s names. Qualitative and nominal data that is categorical (such as gender, political group, or iris species) is considered a *factor* type in R and a *category* type in Python.

Understanding and paying close attention to data types, especially when using R and Python, is critical to data cleaning, preparation, modeling, and analysis. The use of incorrect data typing can generate errors and false outcomes. For example, R offers many supervised modeling methods, such as Naive Bayes, Decision Trees, and Support Vector Machines. These models must be trained with labeled data. The label column or variable within the dataset must be of type “factor.” If the label is accidentally set as a non-factor data type, the model may fail to work even though R may not generate an error. Think about why this is?

What is the difference between a character string (such as a person’s name) and a factor (or category – such as gender)? The answer is that character strings, such as the names of individuals, are each unique and are intended to be unique. Alternatively, gender names, such as “female” and “male” are not unique, but rather are part of a defined group (or factor). All “male” are part of one category and all “female” are part of the other. For R (or another programming language) to understand that all “male” are one label and all “female” are another label, the data type must be set to “factor” in R and “category in Python.

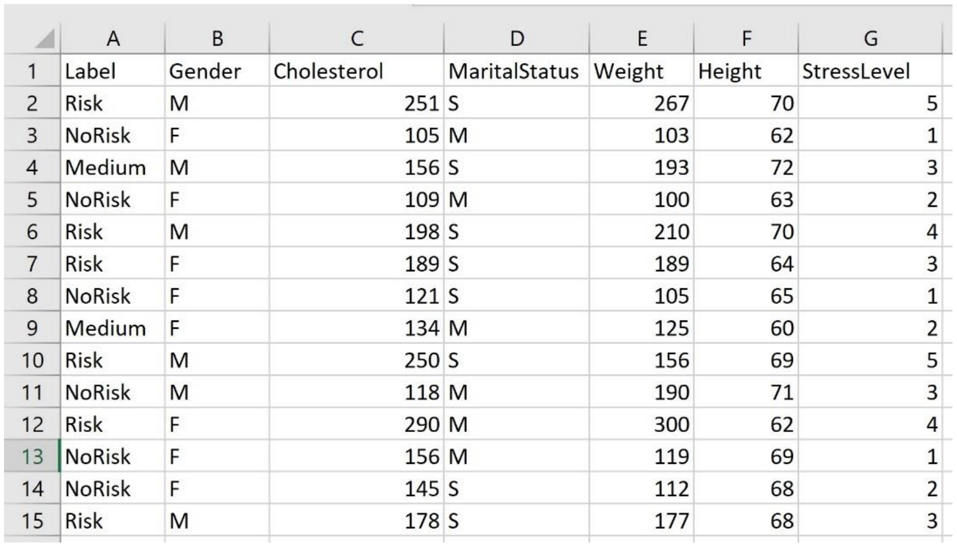
Several of the following sections and subsections will investigate common data formats more closely, and will offer examples of reading in data using R and Python. Following sections will also investigate methods for transitioning from one format to another, which is often necessary. The formats considered here will be record, transaction, network (also known as graph), image, text, web data (html and xml), and JSON. Once data formatting is reviewed, the next section will cover data cleaning and processing.

As an aside, there are hundreds of file and data formats that range from plain text files such as .txt, to portable graymap files (.pmg), to audio file formats (.wav or .mp3), to biological sequence data formats (FASTQ). There are also hundreds of complex data types and structures, and each is often dependent on the programming language, the venue, or the area of study. The following reviews some of the more common formats used in data science and analytics.

Record Data

Record data, also sometimes called tabulated or table data, is made of up rows and columns. Each row in a record dataset represents an individual observation, object, entity, vector, or instance in the dataset. It is important to recognize that different research areas utilize different words when describing rows. For example, mathematical applications may refer to rows as vectors. Social science research may refer to rows as individuals. In computer science, rows may be referred to as objects. Rows may or may not be named.

Each column in record data most often represents the data collected for a particular variable within the dataset. As is the case with rows, columns may also be referred to by different names including variables, fields, attributes, features, or dimensions. Columns are often named, and the name is generally assumed to be the variable name of the data in the corresponding column. A column or variable can be anything that varies from observation to observation, such as Age, Height, Weight, Hair Color, etc.



	A	B	C	D	E	F	G
1	Label	Gender	Cholesterol	MaritalStatus	Weight	Height	StressLevel
2	Risk	M	251	S	267	70	5
3	NoRisk	F	105	M	103	62	1
4	Medium	M	156	S	193	72	3
5	NoRisk	F	109	M	100	63	2
6	Risk	M	198	S	210	70	4
7	Risk	F	189	S	189	64	3
8	NoRisk	F	121	S	105	65	1
9	Medium	F	134	M	125	60	2
10	Risk	M	250	S	156	69	5
11	NoRisk	M	118	M	190	71	3
12	Risk	F	290	M	300	62	4
13	NoRisk	F	156	M	119	69	1
14	NoRisk	F	145	S	112	68	2
15	Risk	M	178	S	177	68	3

Figure 1: Example of Record Data

Labeled and Unlabeled Data

As with any subject or area, learning the new terminology is an important step. Often, words are repurposed with new meanings, which can be confusing, or new words are created that replicate already existing words. In any case, it is helpful to learn them all, and to become familiar with the vocabulary of data science. For example, the words, “label,” “name,” “target,” “response variable,” “dependent variable,” and “output” tend to all mean similar things. For the purposes of this book, the word “label” will be used to describe a feature or column in a dataset that categorizes a given row.

What is labeled data and what does it have to do with record data and with supervised and unsupervised learning? Record data may be labeled or unlabeled. A label can be any variable (column) that categorizes the data. Datasets can have more than one label, such as gender and political orientation. Datasets may also have no label at all. Labels can also be generated from data. For example, is a record dataset contains a variable called “Age” that is quantitative and continuous, it is an option to generate a new variable (feature) in the dataset that is qualitative and ordinal, such as AgeGroup. For example, one may choose to have three groups of ages, Under30, ThirtyTo60, and Over60. Once this new variable is generated, each row of the dataset will be a member of one of these three categories. In that case, AgeGroup may be used as a label. In short, labels may be added or removed from datasets. Future sections will discuss feature generation further.

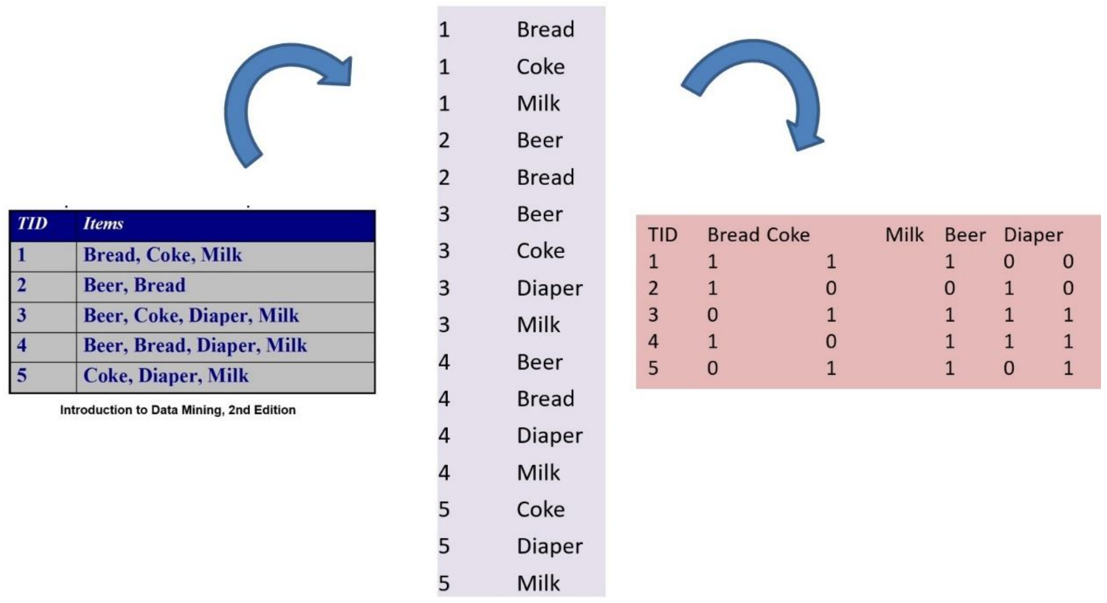
Labeled data is necessary when utilizing models that are supervised. Supervised learning, which will be discussed in detail in other chapters, includes methods that train and build models with known and labeled datasets, and then use those models to classify or predict. While learning methods are not the focus of this chapter, it is necessary to understand what labels are and how to clean and prepare data that is either labeled or unlabeled.

Transaction Data

A transaction can be thought of as an event or instance that generates a collection of related data. For example, when a person purchases a product online, that purchase is a transaction. That transaction contains information such as the person's name, address, phone number, purchased item(s), the date, the time, credit or bank card number, card type, pin, expected delivery dates, shipping address, and so on.

A similar and rather popularized example of transaction-type data is market-basket transaction data. Market basket transaction data generally includes a transaction ID (but does not need to) and all the items that a particular shopper (an instance) has placed into her or his basket or cart. The items in the basket do not retain or require any order.

There are several formats that can be used to store transaction data and it is important to notice that transaction data is not the same as record data. Figure 2 illustrates three basket formats: single format, basket format, and sparse item matrix format. All three formats represent the same exact data.



TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

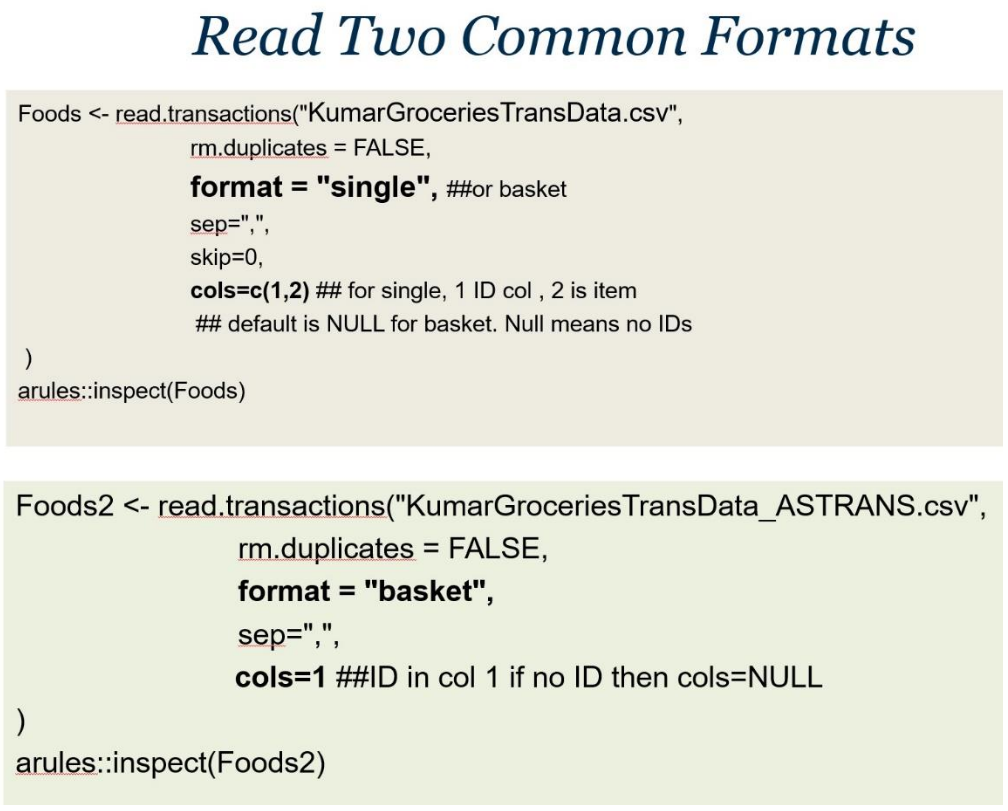
Introduction to Data Mining, 2nd Edition

1	Bread
1	Coke
1	Milk
2	Beer
2	Bread
3	Beer
3	Coke
3	Diaper
3	Milk
4	Beer
4	Bread
4	Diaper
4	Milk
5	Coke
5	Diaper
5	Milk

TID	Bread	Coke	Milk	Beer	Diaper
1	1	1	1	0	0
2	1	0	0	1	0
3	0	1	1	1	1
4	1	0	1	1	1
5	0	1	1	0	1

Figure 2: Transaction Data in Basket (left), Single (center), and Matrix (right) formats

Both R and Python can be used to read in transaction data. It is imperative to understand the format of the data, how to reformat the data, and what format the R or Python method expects. For example, in R is a library called "arules" and within that library is a method that allows R to read in transaction data. Within that method, there are format options – such as single or basket. The following examples will illustrate both.



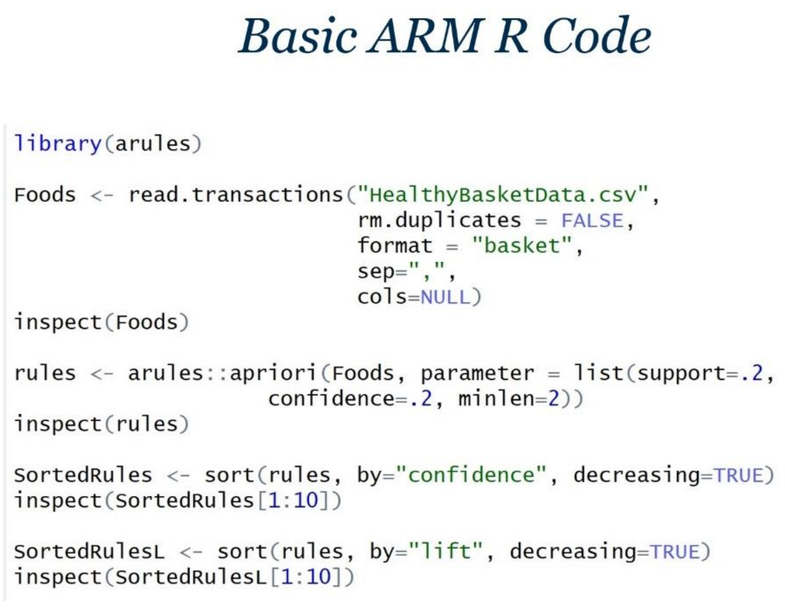
Read Two Common Formats

```
Foods <- read.transactions("KumarGroceriesTransData.csv",
                    rm.duplicates = FALSE,
                    format = "single", ##or basket
                    sep=",",
                    skip=0,
                    cols=c(1,2) ## for single, 1 ID col , 2 is item
                     ## default is NULL for basket. Null means no IDs
 )
arules::inspect(Foods)
```

```
Foods2 <- read.transactions("KumarGroceriesTransData_ASTRANS.csv",
                    rm.duplicates = FALSE,
                    format = "basket",
                    sep=",",
                   cols=1 ##ID in col 1 if no ID then cols=NULL
)
arules::inspect(Foods2)
```

Figure 3: Two options for using R to read in transaction – type data. "Single" format (top) and "Basket" format (bottom).

The following code example will illustrate how to use R to convert transaction data, in basket format, to "rules".



Basic ARM R Code

```
library(arules)

Foods <- read.transactions("HealthyBasketData.csv",
                           rm.duplicates = FALSE,
                           format = "basket",
                           sep=",",
                           cols=NULL)
inspect(Foods)

rules <- arules::apriori(Foods, parameter = list(support=.2,
                    confidence=.2, minlen=2))
inspect(rules)

SortedRules <- sort(rules, by="confidence", decreasing=TRUE)
inspect(SortedRules[1:10])

SortedRulesL <- sort(rules, by="lift", decreasing=TRUE)
inspect(SortedRulesL[1:10])
```

Figure 4: Using R code and transaction-formatted data to apply Association Rules Mining (ARM)

At this point, it should be evident that datasets can range widely in format. The format of the data must coincide with the method, model, or application that is intended to be applied to the data.

The next set of examples will illustrate how transaction data (in all three common formats) is read into Python.

Transaction Data in Python:

There are often multiple libraries and methods for performing similar tasks. Python offers pandas and the package called apyori for transaction data and association rule mining. The following example illustrates a method for reading transaction data into Python. The data is read into a dataframe using pandas. However, to perform association rule mining and to apply the apriori algorithm, it is necessary to convert the format of the data to a *list of lists*. Because each method, library, and programming language may be unique, it is critical to understand the different formats that data may be in and how to convert between them.

Example:

TransactionData2.csv dataset:

```
1,blueberries,soymilk
2,soymilk,quinoa,kale
3,salmon,kale,coffee,soymilk
4,blueberries,apples
5,salmon,kale
```

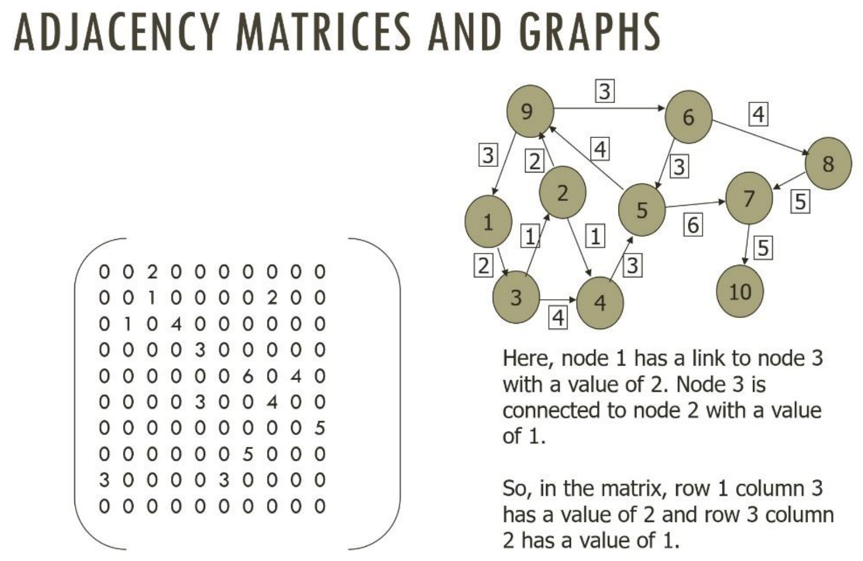
#Python Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from apyori import apriori
# from command line or terminal
# pip install apyori
filename="TransactionData2.csv"
dataset=pd.read_csv(filename, header=None)
dataset = dataset.replace(np.nan, '', regex=True)
## i are num rows, j num cols
records = []
NumRows,NumCols = dataset.shape
## Format the dataframe as a list of lists for the apriori method
for i in range(0, NumRows):
  records.append(
        [str(dataset.values[i,j])
        #Do not include column 0
        for j in range(1, NumCols-1)]
        )
print(records)
rules = apriori(records, min_support = 0.01,
            min_confidence = 1, min_lift = 0,
            min_length = 3)
results = list(rules)
print(results)
```

In summary, transaction data can be in several formats and the required format depends on the programming language, the package, and the method being used.

Graph/Network Data

A graph or network is a collection (or set) of edges (also called links or relationships) and a collection (or set) of vertices (also called nodes). Network data does not generally fit the common record format. The nature or required format of network data is dependent on the programming language, package, method, and application. For example, if using NetworkD3 with R expects one format and igraph expects another. In all cases, network data describes nodes and edges, along with additional attributes.



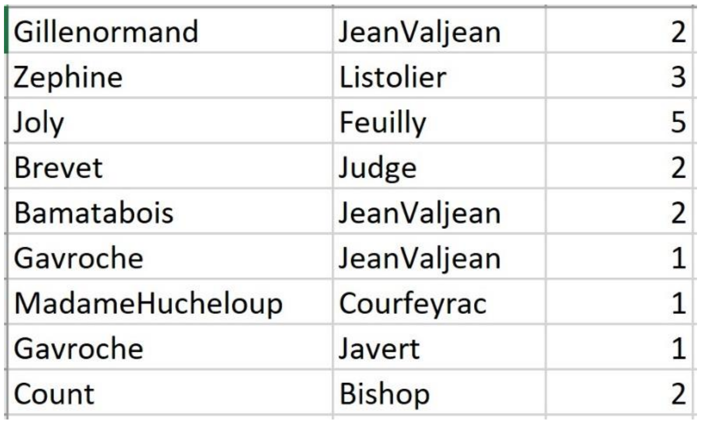
ADJACENCY MATRICES AND GRAPHS

$$\begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Here, node 1 has a link to node 3 with a value of 2. Node 3 is connected to node 2 with a value of 1.

So, in the matrix, row 1 column 3 has a value of 2 and row 3 column 2 has a value of 1.

Figure 5: Network (graph) data as an adjacency matrix and the equivalent graph.



Gillenormand	JeanValjean	2
Zephine	Listolier	3
Joly	Feuilly	5
Brevet	Judge	2
Bamatabois	JeanValjean	2
Gavroche	JeanValjean	1
MadameHucheloup	Courfeyrac	1
Gavroche	Javert	1
Count	Bishop	2

Figure 6: A node and edge list. This dataset is formatted to create a network (graph). For example, the node Gillenormand has an edge to the node Jean Valjean with a weight of 2.

Image Data

Images, whether color, grayscale, or black and white, can be represented as “flattened” vectors of numbers. Each number in the vector represents the value of a pixel in the image. A collection of image column (or row) vectors can also be grouped together to form a matrix of images. For example, each column of the matrix might represent one full image vector and each row would then represent a specific pixel location. The inverse is also common, where each row might represent an image and each column a pixel location. In both cases, this data format is not standard record data, even though it does contain rows and columns.

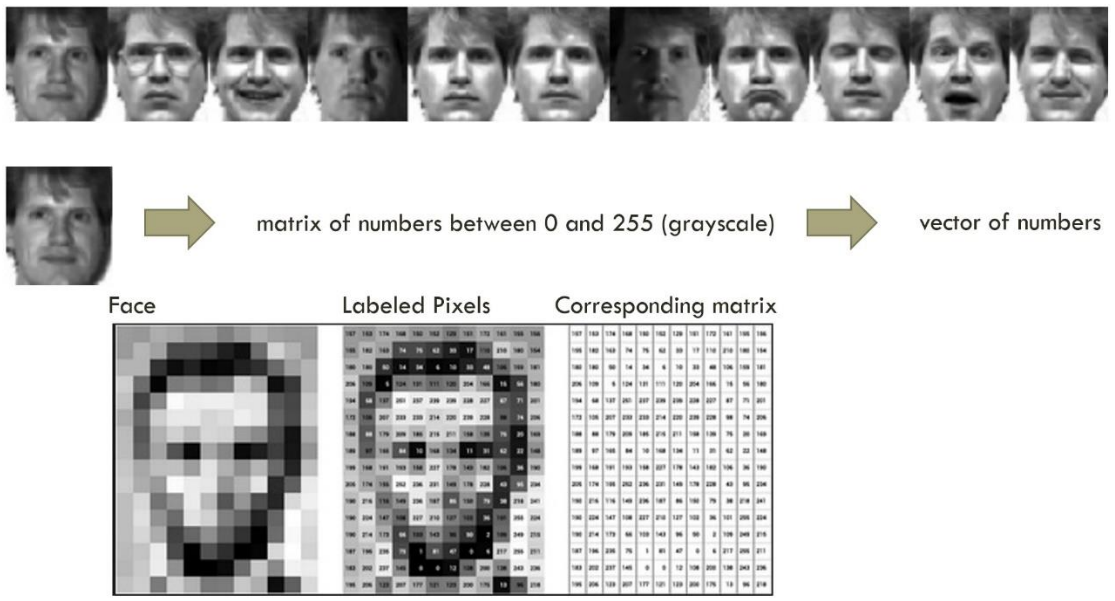
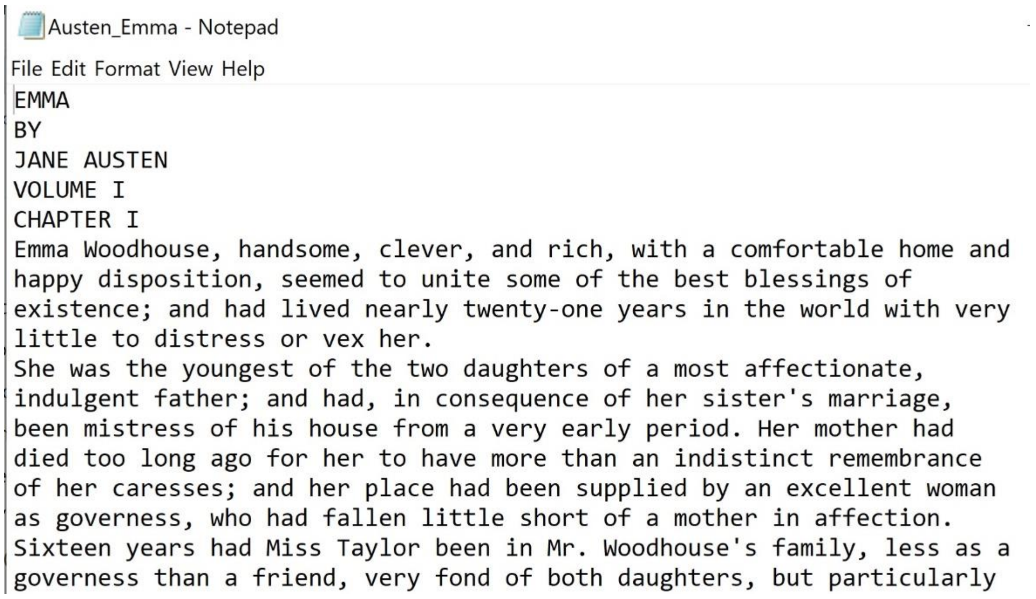


Figure 7: A conversion from an image to a vector; shows a collection of image vectors in one matrix, with each column representing an image.

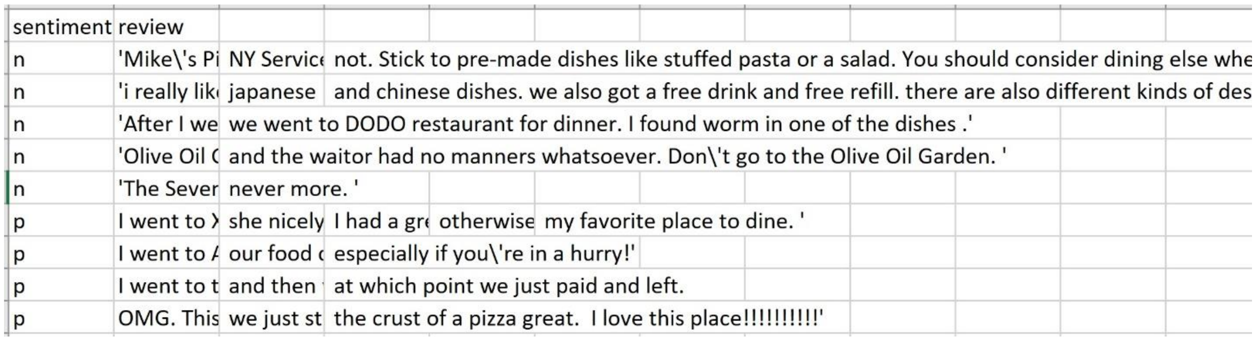
Text Data

Text data can be in several formats, including a .txt, .doc, or .docx text file, HTML, XML, and JSON. Text files can have very little or no structure and can be a collection of words along with numbers, punctuation, and other symbols. HTML, XML, and JSON have some structure, which may need to be cleaned and processed to fit a selected programming language method.

Text data can be converted to a record format. There are several options for converting text (a collection of words and symbols) into rows and columns. There are also several programming language packages, such as *NLTK* or sklearn (CountVectorizer) in Python and *tm* or *tidytext* in R that will assist in the processing and evaluation of text data. The following figure illustrates two examples of text data.



```
EMMA
BY
JANE AUSTEN
VOLUME I
CHAPTER I
Emma Woodhouse, handsome, clever, and rich, with a comfortable home and
happy disposition, seemed to unite some of the best blessings of
existence; and had lived nearly twenty-one years in the world with very
little to distress or vex her.
She was the youngest of the two daughters of a most affectionate,
indulgent father; and had, in consequence of her sister's marriage,
been mistress of his house from a very early period. Her mother had
died too long ago for her to have more than an indistinct remembrance
of her caresses; and her place had been supplied by an excellent woman
as governess, who had fallen little short of a mother in affection.
Sixteen years had Miss Taylor been in Mr. Woodhouse's family, less as a
governess than a friend, very fond of both daughters, but particularly
```



sentiment	review				
n	'Mike\'s Pi	NY Service	not. Stick to pre-made dishes like stuffed pasta or a salad. You should consider dining else whe		
n	'i really lik	japanese	and chinese dishes. we also got a free drink and free refill. there are also different kinds of des		
n	'After I we	we went to DODO restaurant for dinner. I found worm in one of the dishes .'			
n	'Olive Oil (	and the waiter had no manners whatsoever. Don\'t go to the Olive Oil Garden. '			
n	'The Sever	never more. '			
p	I went to X	she nicely	I had a gre	otherwise	my favorite place to dine. '
p	I went to A	our food c	especially if you\'re in a hurry!'		
p	I went to t	and then	at which point we just paid and left.		
p	OMG. This	we just st	the crust of a pizza great. I love this place!!!!!!!!!!'		

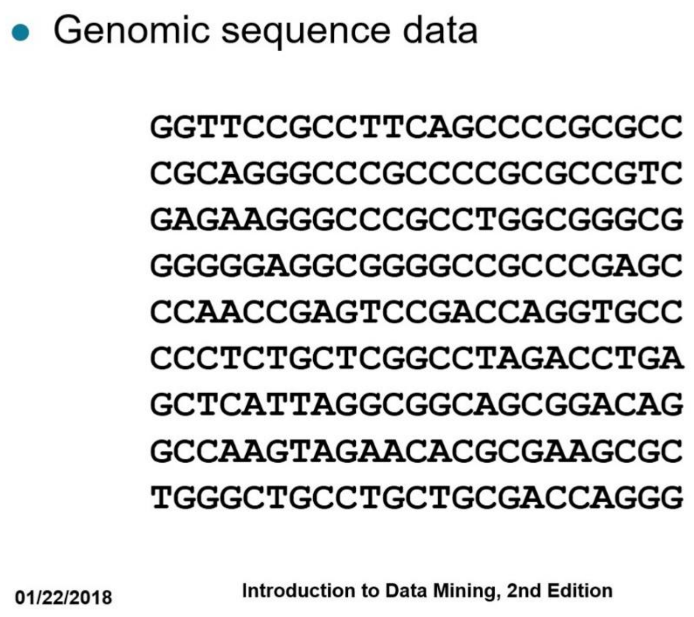
Figure 8: Examples of "text" data. The top is a .txt file that contains a novel (of text). The bottom is a .csv file that contains restaurant reviews and sentiment labels.

Sequential Data
https://www.ncbi.nlm.nih.gov/pubmed/27008007

Sequential data is unique in that the order of the data elements contains information about the data. In most cases, if the order is lost, some or all of the information will be lost. Therefore, the order must be preserved. A sequence of amino acids, in a specific order, may define a specific protein.

A sequence of nucleic acids may define and RNA or DNA sequence. There are several special file formats that hold sequential data, such as FASTA and BAM. It is possible to convert sequential data into record data by using indices. Each column in the dataset represents an index or location in the sequence and each row represents an individual sequence.

Figure 9 illustrates sequential data, where the order is critical and cannot be changed or lost.



- Genomic sequence data

```
GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG
```

01/22/2018 Introduction to Data Mining, 2nd Edition

Figure 9: (Borrowed from Introduction to Data mining, 2nd Ed. Kumar, et. Al.
Illustrates an example of sequential data.

The R programming language offers options such as read.fasta for reading in sequential data. Python also offers several options for managing biological data, including BioPython (http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc11).

The following is an example of FASTA data format:

```
>gi|2978501|gb|AAC06133.1| vacuolar ATPase proteolipid subunit [Giardia intestinalis]
MSSIDSPVAVEKCPAGASFWSMLGQVVAVVFSSIGAAYGTAKAGSGLGV
AGLINPAPVTKLTLPVIMAGILSIYGLITSLLINSRVRSYTNGMPLYVS
YAHFGAGLCCGLAALAAGLAIGVSGSAAVKAVAKQPSLFVVMLIVLIFS
EALALYGLIIALILSTKSADSNFCVNNVNQ
```

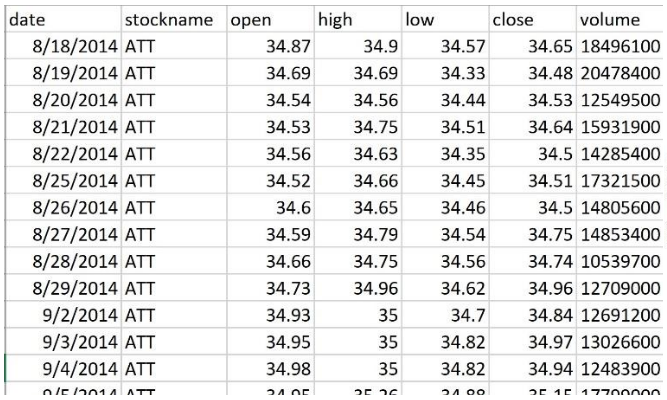
This book will not dive into bioinformatics. However, all topics covered in this book can be applied to biological and medical analytics and informatics.

Temporal Data:

In some cases, data is gathered based on known and specific dates and/or times. When data is labeled with time (a temporal label) then the progression, change, and characteristics over time can be part of the formatting and analysis.

Not all data has a temporal label. For example, Figure 1 at the start of this chapter is record data with no information or label that offers dates, times, or progression through time. Alternatively, other types of data, such as the price of a stock collected as 12 noon ET each day for one year will have a date (and time) associated with it.

This enables temporal analysis – an investigation of changes and information through time. Figure 10 illustrates temporal data.



date	stockname	open	high	low	close	volume
8/18/2014	ATT	34.87	34.9	34.57	34.65	18496100
8/19/2014	ATT	34.69	34.69	34.33	34.48	20478400
8/20/2014	ATT	34.54	34.56	34.44	34.53	12549500
8/21/2014	ATT	34.53	34.75	34.51	34.64	15931900
8/22/2014	ATT	34.56	34.63	34.35	34.5	14285400
8/25/2014	ATT	34.52	34.66	34.45	34.51	17321500
8/26/2014	ATT	34.6	34.65	34.46	34.5	14805600
8/27/2014	ATT	34.59	34.79	34.54	34.75	14853400
8/28/2014	ATT	34.66	34.75	34.56	34.74	10539700
8/29/2014	ATT	34.73	34.96	34.62	34.96	12709000
9/2/2014	ATT	34.93	35	34.7	34.84	12691200
9/3/2014	ATT	34.95	35	34.82	34.97	13026600
9/4/2014	ATT	34.98	35	34.82	34.94	12483900

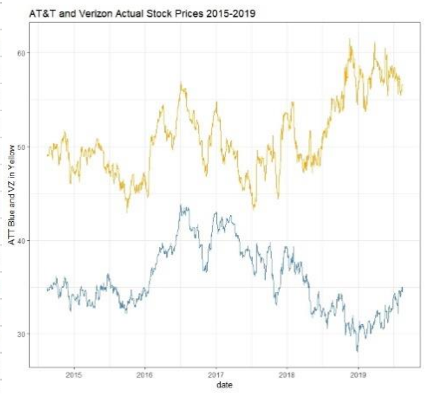


Figure 10: Temporal data that describes the stock AT&T price over time using dates.

Summary

Data science and analytics begins with a collection of data. Such data must be formatting, cleaned, and prepared for analysis. Understanding different data formats and how to transform between them is critical when utilizing various programming models and methods.

PART 2

Data Cleaning and Preparation

Introduction

Data cleaning is a cyclic and iterative set of processes that includes exploring, visualizing, updating, formatting, transforming, normalizing, discretization, and correcting data. Data preparation often also includes steps such as the identification and management of outliers, as well as the generation of new features. These processes are repeated until the data is prepared for analysis, or *clean*.

Data can range from relatively organized (and so more straightforward to clean), to absolutely disorganized and raw. Data can also be stored in multiple datasets and with multiple (and differing) formats, or it can be contained in one csv (comma separated values) file. It is not uncommon for data to have been collected by different and often disconnected organizations. In addition to being unclean and disorganized, data may also be inconsistent. For example, a variable name may be used in two datasets, but may not be used in the same way.

When cleaning data, the methods of the data collection must also be considered. For example, if survey data was collected by different organizations and without direct consideration for subsequent data analysis, the data may be inconsistent, disorganized, and incomplete. This is not uncommon. While the following sections will discuss and describe common and basic data cleaning and preparation methods, it is important to keep in mind that this is the beginning of data cleaning and is not an exhaustive list of actions. Each dataset collection is unique and so will require personalized cleaning and preparation. Data cleaning is never a “black box” and should be done mindfully.

The idea of cleaning data can sometimes carry with it notions of manipulation, alteration, and even foul-play. For example, removing a data value can alter the meaning of the dataset and so the information that it conveys. For this reason, the process of cleaning should be clearly documented, fully transparent, measured with quantitative and qualitative analysis, and done with great care. Values in data should never be purposely removed in order to alter the meaning or information contained in the dataset.

Data cleaning and preparation can be broken down into stages. However, in many cases, stages are often repeated. Therefore, it is best to think of cleaning stages as suggestions, rather than ordered rules. Each dataset is unique and will need to be cleaned and prepared carefully.

During a first stage, basic data cleaning generally involves the discovery and management of missing values, incorrect values, values with incorrect formatting or inconsistencies, and duplications. During a second stage, data conversion and transformation can take place, which may include evaluating and correcting data types, such as converting labels to factors or categories, or converting date and time data to appropriate formats. This stage can also involve data transformation, such as normalization, discretization, and data reduction. During a third stage, outliers and other noise issues are evaluated and managed. A fourth stage of cleaning and preparation may include new feature generation, as well as evaluating and ensuring expected consistency within and between dataset collections.

The following sections will consider each stage of data cleaning and preparation, and will offer examples and illustrations in R and Python. Measures and quantitative cleanliness evaluations will also be discussed in the last section. It should be noted that while these stages seem to move forward in a linear fashion, they do not do so in practice. In practice, cleaning and preparation, as well as exploration and visualization, iterate and jump around until the entire dataset is prepared for analysis.

The following discussion of data cleaning and preparation will assume that the data is in a standard record format, where columns represent variables and rows represent observations. Examples will also assume one single dataset and will not engage in discussion for multiple disparate datasets that form a collection.

An Observational Example: Thinking About Cleaning

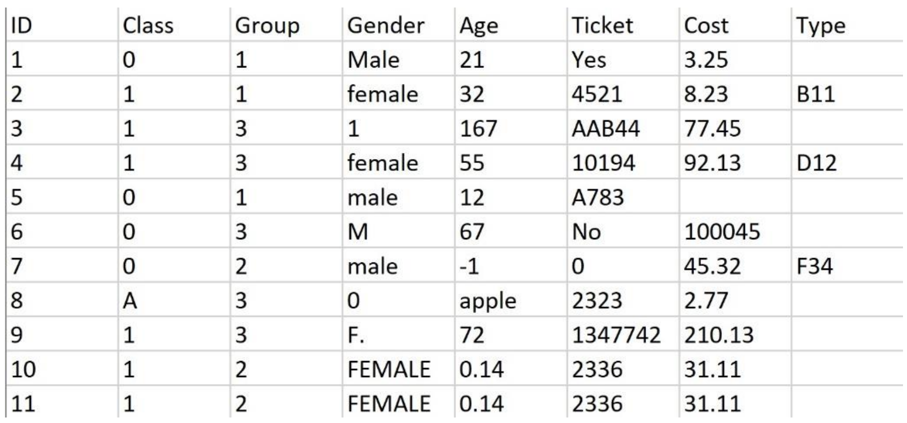
During data cleaning, each variable (column) of the dataset is investigated. For this discussion, it will be assumed that there are a reasonable number of columns/variables in the dataset (so less than 30). For other cases, such as in text mining were there may be 1000s of columns, some of the techniques may need to be altered. Because there are an infinite number of possible datasets, the following will illustrate basic cleaning for a standard record-style dataset. As an additional note, prior to initializing the cleaning and preparation process, it is helpful, though not always possible, to have an understanding of what each column or variable represents and the information that it contains.

Data cleaning manages missing values, incorrect values, values with incorrect formatting or inconsistencies, and duplications. Consider the following record dataset in Figure 11. What is immediately visually notable? In other words, what can be seen by looking at this dataset?

Is visually inspecting data a good first step?

What are the limitations of visual inspection?

Look at the record dataset in Figure 11 synthesized from the popular Titanic Dataset from Kaggle.com. What do you see?



ID	Class	Group	Gender	Age	Ticket	Cost	Type
1	0	1	Male	21	Yes	3.25	
2	1	1	female	32	4521	8.23	B11
3	1	3	1	167	AAB44	77.45	
4	1	3	female	55	10194	92.13	D12
5	0	1	male	12	A783		
6	0	3	M	67	No	100045	
7	0	2	male	-1	0	45.32	F34
8	A	3	0	apple	2323	2.77	
9	1	3	F.	72	1347742	210.13	
10	1	2	FEMALE	0.14	2336	31.11	
11	1	2	FEMALE	0.14	2336	31.11	

Figure11: Small sample of uncleaned, record dataset extracted and updated from the Titanic Data set on Kaggle.

Visually inspecting data is always a good idea. While smaller datasets can be visually inspected with relative ease, larger datasets and "big" data may not lend themselves well to visual inspection. A visual inspection, when possible and feasible, is a great starting point for cleaning. Visual inspections can generate initial observations and ideas about areas in the data that need attention. However, visual inspection must not be relied upon and is only an initial starting point. Programming languages such as R or Python will be used to evaluate, visualization, explore, prepare, and clean the data.

Using Figure 11 above, perform a visual inspection of all the variables to begin to identify potential issues within the dataset. What do you see?

An organized method for cleaning data is to begin by considering each variable individually. In the Figure 11 above, start with the variable called ***ID***. Visual observation suggests that the variable ID offers little or no information about the data, and is little more than a row number. The ID variable is qualitative and nominal. It is not a factor or category. Do you agree?

Because no values are missing from the ***ID*** column, and all values appear to be correct and consistent, that column is considered to be clean. How might this be determined using R or Python rather than inspection? Even though the variable, ID, can be determined to be clean, it may be unnecessary and so may be eliminated from the dataset to reduce its dimensionality without losing critical information. In addition, if the ID column is not removed, care should be taken not to accidentally include the ID column when modeling the data. For example, if the ID column were inadvertently included in the building of a decision tree model, that model would be incorrect and poor. Keep in mind that programming languages will not differentiate between useful data and irrelevant data.

The next variable in Figure 11 is ***Class***. Without direct knowledge about the dataset, one can observe that most of the values in the Class attribute are 0 or 1. This may suggest that Class is a label of some kind. This may also suggest that the value "A" in row 9 may be an incorrect value. However, without knowing more about the dataset, there is no way to know this for certain. Let's assume for the purposes of this example that it is known that Class is a label and that its values are 0 or 1. This means that "A" is an incorrect value, and must be managed. How might R or Python be used to determine if values are incorrect? What are some limitations?

The next variable in Figure 11 is ***Group***. This also appears to be categorical in nature and so represents another label. Observation suggests that the levels are 1, 2, and 3. It does not appear that any of these values are incorrect or missing. How might R or Python be used to determine all the possible values of "Group" without using observation?

Gender is the next variable. It is immediately evident that the data in the Gender column has several issues. Upon closer visual inspection, the key issue with the Gender column is inconsistency. In other words, some of the values are inconsistent. While it is not always safe to assume, suppose it was known that 0 is male and 1 is female. If so, the data can be corrected. However, to correct the Gender data, one format must be selected and all data must be updated to conform to that one format. For example, the format may be M for male and F for female. In that case, options such as Male, 0, and MALE must all be converted to "M." This same format can be applied to the female labels as well. This is an example of inconsistent, but not incorrect data. However, suppose one of the entries under Gender is "Sam". In this case, this cannot be corrected given that there is no way to know if Sam was intended to be male or female.

The ***Age*** variable also needs to be cleaned. However, unlike the Gender column, the Age column has errors and possibly outliers. Within the Age column, the values 167, -1, "apple," and 0.14 are all items of concern. It is very likely that 167 is an outlier (and an error). Outliers and their management will be discussed shortly as they are not simple to identify and there are many options for identification and remediation. The values -1, "apple," and 0.14 are likely errors. However, it is possible that .14 is a new born baby age. Without more knowledge about the dataset, it may not be feasible to be certain. How should these issue be corrected? Can they be corrected? How might R or Python be used to find such errors?

The ***Ticket*** column is an interesting case. It clearly contains inconsistencies and errors. However, the more important question might be whether it contains any useful information for the analysis of the data. The Ticket data is not categorical and is qualitative and nominal. In other words, it contains character strings. It appears that some of the values are noting whether or not a person has a ticket and others are offering the ticket ID. Assume that in this case, it is known that Yes or some character string implies that the person has a ticket and that No, 0, and blank means that they do not. This is a strong assumption that in reality may not be able to be made. However, for this example, assume that this information is available. In this case, an interesting option is to transform this data into Yes or No. This at least offers some consistent information. Another option might be to eliminate the column. When a column or variable contains either many missing values, values that are inconsistent, or little or no information, it may not be necessary to retain the column.

As an important aside, it is best to clean data with versioning. This means that the original raw dataset is retained, as are all updates made to the data as the cleaning process progresses.

Next, the ***Cost*** feature (variable) in the dataset has some missing values and possible outliers. What can be done about missing values in this case? While "100045" appears to be an outlier based on the other values, it is not clear that 210.13 is an outlier. Why?

The ***Type*** column is the last feature in the dataset. It contains more missing values than present values and does not appear to be categorical or useful information. Here again, it is likely that this column will be eliminated.

At this point, each column has been initially and individually considered. Next, consider the consistency within each row, and also between columns. For example, suppose it is known that the Group column is related to the Cost column. Suppose it is known that Group 3 tickets must cost $50 or more. This information causes further inconsistencies in the data. For example, in row 9, the person is in Group 3, but the Cost is $2.77. The greater the knowledge about the data, the better the cleaning.

Within this first stage, duplicate values can also be identified. It can be noted that rows 10 and 11 are identical in every way except for their ID. This may be an example of a duplication, but it also may not. It is not always simple to determine if two rows are duplicates (one is an exact repeat of the other and both represent the same observation), or if both rows happen to contain the same data. Before a potentially duplicated row is removed, it must be confirmed that it is in fact a true duplicate. Not all duplicates are true duplicates. For example, within transaction data, two transactions may contain identical items. However, they do not represent the same transaction. In the case being evaluated here, rows 10 and 11 are true duplicates as the ticket number is the same in both cases. To identify duplicates, a primary key (a variable that must be unique) can be used. Examples might be social security number, ID, etc. Caution should always be taken in the identification of true duplicate rows.

Using the dataset in Figure 11, the task of thinking about how and where to clean the data was made easier because all data values are immediately visible. Being able to see all of the data enables not only the identification of elements that need cleaning, but also the nature of the issues, such as whether they are inconsistent, incorrect, absent, outliers, etc. However, when cleaning actual data, it is rarely the case that all the data can be observed. Rather, R or Python code must be written to explore each column so as to determine if there are issues that need attention. Code must be used to discover issues, to determine the nature of the issues, and to resolve the issues. As a practice exercise, write code in both R and Python to clean this dataset. Think about what can be done if viewing the data was not an option.

The Stages of Cleaning Data

Missing Values

A missing value is a single attribute (variable) observation or value that is not available. For example, if a dataset has a variable called Age and one of the Age values is missing for one of the rows, this is a missing value. A missing value may result from a lack of data during collection, such as a survey question left blank, or may result from errors in rendering the data, such as data-entry issues. Missing values can have a significant impact on the quality of a dataset and on the information that it contains. If a given column in a dataset contains too many missing values, it may not be a usable variable for analysis. Similarly, if a row contains too many missing values, it may not be a viable observation. Missing values may also be scattered throughout a dataset and may not be more or less frequent in any particular rows or columns.

Generally, finding missing values is straightforward. However, a much more complicated question is how to correct them. If a row (an observation) in a dataset contains a missing value, the entire row can be eliminated. While this immediately solves the problem of the missing value, it is not always an ideal solution. First, removing rows of data will reduce the size of the dataset. In some cases, especially when the amount of data is limited, this is not the best choice. Next, removing rows that contain missing values will change the information contained in the dataset. The greater the number of rows removed, the more the dataset loses its initial integrity. This is especially true if the rows being removed are otherwise critical to the information contained in the data. For example, suppose a variable in the dataset is “Religion”. Next, suppose several rows (observations) have a blank (missing data) under the Religion column. Finally, suppose that because of this, some religions are over-represented while others are under-represented. This may cause an imbalance in the data and correspondingly in the information that it generates. In some cases, missing values can create a significant issue. In other cases, and when they are fewer and more evenly distributed, they can be managed and the data can be saved and cleaned.

Data cleaning should always be careful, mindful, documented, and fully transparent. If data cannot be fairly and ethically prepared for analysis, it should not be used to generate information.

Assuming that missing values have been identified, there are several methods that allow for their mitigation. The above has reviewed the simple but often unwise method of removing a row that contains one of more missing values.

Another option is to remove an entire column for which a large percentage of the column is missing. For example, recall the dataset discussed in Figure 11 above. In this dataset, the “Type” variable (and data column) has the majority of its values missing. In addition, and very importantly, the data in the Type column does not contain relevant information with respect to the dataset. Therefore, rather than removing all rows that are missing the “Type” value, an alternative is to remove the “Type” column. This will result in creating several complete rows (rows for which there is no missing data), while both reducing the dimensionality of the dataset and eliminating a variable that is of no value.

In some cases, when removal is not a preferred option, an alternative is replacement. The missing value can be replaced by an estimate of what the value is expected to be. In some cases, estimating a missing value may be an acceptable option. For example, suppose a dataset is missing a value under Gender. However, suppose this same dataset has another variable that offers information that can assist in determining the missing gender value, such as "Number of Pregnancies." If this value is greater than 0, the gender is female and the missing value can be replaced. While this is an ideal example, it is also an unlikely example. More often, the missing value must be estimated in another and perhaps less definitive way.

Another replacement option is estimating the missing value based on the other values in the column. If the data column that contains the missing value is quantitative, the missing value can be estimated using the mean, median, or mode of the entire column in which it resides. For example, suppose the dataset of interest is about applicants to a graduate university program, and suppose the dataset has a variable for AGE. If there is a missing age value in the AGE column (and so in a row of the data), but the row cannot be deleted, the mean or median of the AGE column may be used to replace the missing age value. If the variance of the AGE column is relatively small, and the mean and median of the AGE column data are very similar, this means that most ages are close to the mean and that the data in the column is relatively balanced. In this case, the missing age value can be safely replaced with the mean age. Alternatively, if the variance is small, but the mean is much larger (or smaller) than the median, this means that the data is skewed but not greatly variable and so the median can be used to replace the missing age. Within this framework there are many variations depending on the nature of the data.

While the replacement method may be appropriate and safe in some cases, it may not be in others. For example, suppose the dataset being cleaned is the well-known Titanic dataset that offers data per survivors and non-survivors of the Titanic ship disaster. For those who are familiar with the Titanic, it is known that both age and gender played a significant role in survival. As such, a missing age value in the Titanic dataset cannot be replaced by the mean or median age. Why not?

Replacing or removing values should always be done with care and on a case by case basis. Cleaning rules should not be viewed as blind steps to take, but rather as options for creating a clean and reliable dataset.

Incorrect Values

Incorrect values are often more challenging to locate within a dataset than missing values. Unlike missing values, which tend to adhere to a given format, such as blank or NA, incorrect values can be anything. Therefore, incorrect values must first be discovered. This is generally done with smart coding in R, Python, or another programming language.

Discovering incorrect data values can range in difficulty depending on the nature of the data and prior knowledge of the data. For example, suppose a data column is qualitative (character or string data) and the appropriate values are known, such as state names or gender or a possible set of 12 colors. In the case where all possible values are known, each value in the column can be compared (via R or Python) to the appropriate set of possible and known values to determine if any are incorrect. This is a lucky and easy scenario.

However, if the appropriate values are not known or are infinite or continuous, then determining whether a value is absolutely incorrect may be much more challenging or even impossible. Incorrect values are different from inconsistent values, and inconsistent value cleaning will be discussed in the next subsection. For clarity, an incorrect value is a value that is wrong and cannot be altered into correctness. Alternatively, an inconsistent value is a value that is correct, but in an incorrect format, such as Fla rather than FL for Florida.

If the data column is quantitative (numeric), then incorrect values may be determined if possible ranges are known in advance. Measures such as the max, min, range, mean, or median can be applied to evaluate the correctness of the data. Each measure may reveal information about the data and may offer hints as to whether one or more values are incorrect. For example, consider a dataset that contains the attribute column of AGE. Next, consider methods for cleaning the AGE column of data. It is known and assumed that ages can only range from 0 to 120. Therefore, is the measure of minimum exposes a value of -1, this suggests an error that must be corrected. Similarly, if the maximum returns a value of 183, this also suggests an issue.

Other measures will also suggest possible concerns. For example, suppose the mean is 89, and the median is 25. Such measures suggest that there are incorrect values and possible outliers. Measures of center and variation offer insight into the data and can help to support the cleaning process.

The examples offered above are easy in that knowledge is available about the nature of the data as well as its possible and allowable values. However, this is not always the case. For example, suppose Hair Color is a variable in a dataset that is being cleaned. However, there is no list of possible hair colors. What can be done? The first option might be to create a table (using R or Python) that offers the frequencies of all the hair colors. Note this is a good example of when the data type of Hair Color must be factor in R or category in Python. Otherwise, it will not feasible to create a table. Making Hair Color a factor ensures that all "brown" is categorized together, as is all black, gray, blond, etc.

By creating a table (a visual tool), all hair colors will be listed along with their corresponding frequencies. By observing the table (and the list of all unique hair colors which can also be generated using R code), the cleanliness of the Hair Color column can be better evaluated. Suppose the hair color frequency table reveals a color called "banana" with a frequency of one. Now what? Is banana a real hair color or an incorrect value. Should it be removed or is it valid? The real answer is that there is no way to really know unless there is further and accessible information about the data. Assuming that there is no information, should the banana hair color be cleaned away or retained. This is a judgement call. Does retaining the data harm the analysis? Does removing the data risk the loss of a new style of hair that is just beginning to emerge. This is a real issue with no absolute answer.

When searching for incorrect values, whether quantitative or qualitative, several methods can and should be employed such as the use of statistical measure and visual tools.

Tables and other visualization options such as box plots, scatter plots, histograms, stem plots, density plots, etc. are also very helpful in "looking at" and exploring data. Data visualization should be integrated into data cleaning and may also be used to compare pre-cleaned and post-cleaned data.

Consider the following example. Recall the dataset in Figure 11 above. Notice that the Age variable represents quantitative data and has several incorrect values such as -1 and apple. Interestingly, when this dataset is read into R or Python, the Age variable an data will be read in as type char or string (not numeric). This is because of the value "apple" in row 9. Many programming languages have defaults and data type hierarchies. In this case, the "apple" will cause the data typing to assume that the column is non-numeric. However, this assumption is not correct.

As discussed in detail in previous sections, viewing and correcting the data types is a critical step in cleaning. The observation that the Age variable is mistyped as non-numeric is a hint. The hint is that one or more values in the Age column are either incorrect or in the wrong format. A table may be the best option for observing the possible issues. Measures such as minimum, maximum, and mean will not function (will generate errors) until "apple" is discovered and removed, and the data type is updated to a numeric type.

Other visualizations, such as boxplots or histograms can also be used to not only view the nature of the values, but also the distribution of the values. For example, suppose a column of 10,000 quantitative values is being cleaned. Next, suppose all values are confirmed to be numeric. Then, suppose a histogram of the column data is created and reveals that most of the data appears to fit a uniform distribution. However, a few data values do not fit within the observed distribution. In this case, the non-compliant data points can be further investigated. It is possible that these odd data points may be outliers, may be incorrect, or may simply be members of a second but correct distribution within the data.

The discovery of values that may (or may not) be incorrect is a process that should include observation of the data, visualizations, statistical measures, and creative value checking. In some cases, and especially if all possible correct values are not known, it may be difficult to identify all incorrect values. In such cases, further methods such as distribution evaluation, clustering, or binning may be employed. These will be discussed later in the chapter and in the book.

Once incorrect values are discovered and confirmed to be incorrect (and not just unusual), the process for cleaning them is largely identical to the process discussed above for cleaning missing data. The options include correction or removal, with similar pros and cons associated with both. Recall that removal may be a solution if the number of rows removed is small in comparison to the sample data size. Otherwise, removal can significantly affect the quality of and information in the data. Values can be corrected for the sake of retaining the entire row and avoiding loss of data. However, correction will alter the information contained in dataset and should be determined with caution. Correction of numeric values includes replacing the value with a measure such as the column mean, median, or mode. This topic is discussed in great detail for missing values and so the reader is instructed to review that area.

Tagging Incorrect or Missing Values

It is not uncommon to consider the option of replacing all incorrect values with a predetermined "tag" value, such as 0, -1, or 999. The justification of this method is that it allows for the rest of the data in the corresponding row to be retained, and it also preserves the information that the value is incorrect. For example, consider a quantitative variable that describes that number of college degrees a person has. In this case, values may range from 0 to perhaps 6, such that a person with a double BA, an MS, and a PhD may note their number as 4. In this case, using 0 for incorrect values will create not only incorrect information (because 0 means no degrees in this case), but it will alter the mean and the variance of the data. In other words, using 0 is a very bad idea here.

Rather than using a "0" to replace missing values, can values such as -1 or 999 be used? The answer is that it depends. Because -1 and 999 are not possible values, they will allow for the retention of the remaining row data, and will allow a viewer to understand that the value is incorrect. However, the use of a numeric tag, such as -1 or 999 will disable the direct use of any numeric measures on the column, such as mean or variance. That being said, smart programming can still utilize column measures by purposefully not including the "special" value in any calculations. The use of -1 or 999 for incorrect values can be thought of as labeling or tagging a value as incorrect while still retaining the rest of the row of data. This method should be used with great caution, and tags should be chosen with care so that there is no chance that they can be mistaken for real values.

In general, incorrect values are common in datasets and must be discovered and managed. Management will depend on the nature of the data, the information contained in the data, the size of the dataset, and the data type (such as qualitative or quantitative). Methods for discovery can involve observing the data, visualizing the data, and using a programming language to measure and determine the validity of each data value. Once incorrect values are identified, they can be managed by row removal, value correction, or value tagging. Each method has benefits and drawbacks, and the final decision will depend on the research goals and the nature of the data.

Inconsistent Values

A value is inconsistent if it does not have the expected format, but is informationally correct. For example, the state of Florida can be represented as FL, Fla, Florida, or FLORIDA. All such alternatives have the same meaning and can be updated to all fit one determined format, such as FL.

Discovering inconsistent values is similar to the process of discovering incorrect values. For example, if a dataset has a column called State, then each value can be compared to the expected and desired format, such as FL, CA, etc. All values that do not fit the expected format can be visualized via a table or graphic. By observing values that do not match the expected format, programming code can be written to detect and correct the inconsistent values.

It is more common for qualitative data to contain discoverable and correctable inconsistencies than quantitative data. In the case of quantitative or numeric data, it may not be feasible to determine if a value is inconsistent or incorrect. For example, suppose a variable is Car Speed with continuous values between 0 and 250 miles per hour. Therefore, values might be 23.56, 47.7, 89, and so on. Suppose a value in the column is 7.8. This value is feasible, though unlikely. However, it might be a typo and might really be 78. Unfortunately, there would be no method for knowing.

With qualitative data, it can sometimes be easier to take a very educated and supportable guess at what a value should have been. For example, suppose the variable Hair Color has a color “blondy”. It is fair to assume that this was intended to be “blond”.

Finding inconsistencies, and determining what their correct representation should be, can be a challenging and time consuming process. In some cases, it may not be possible, and in others, it may be acceptable to make educated assumptions.

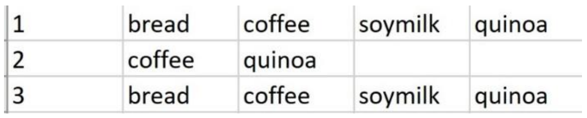
Duplicate Values

Datasets can sometimes have duplicated data. These duplicates may be true duplicates - errors caused by the same data being repeated by accident, or may not be true duplicates. For example, in record data, if all attribute values for two or more rows are identical, it is possible that these rows are not unique and are the same observation accidentally repeated. If this is the case, the duplicate observations should be eliminated.

However, not all duplicates are identical observations. It is possible, for example, for two observations to have the same data values, but not be true duplicates. If a dataset contains the attributes: Firstname, Lastname, Age, and CollegeDegree, it is not impossible for two or more rows to be identical. Ideally, datasets should have at least one primary key variable. A primary key is a variable that cannot be the same for any two observations. Examples are ID Numbers, College IDs, Driver License Numbers, Social Security numbers, or any other unique identifier. Unfortunately, not all datasets are created with primary keys.

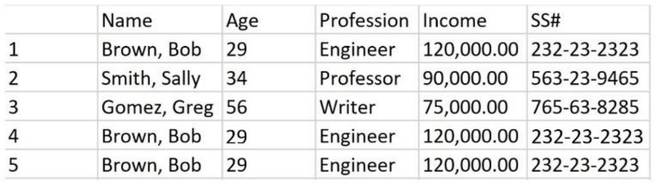
In addition, data in formats other than record format, may have identical rows that are not the same observation. For example, it is possible for two rows in transaction data to appear to be identical, but are in fact not the same transaction. It is not uncommon or impossible for two shoppers to purchase identical items for example. Similarly, image data stored in a matrix or dataframe will very likely have identical rows. These are certainly not duplicates.

When identifying and managing duplicates, a first step is to understand the format and nature of the data. Next, if the data is in record format and two or more rows appear to be identical, it is prudent to consider the probability of such duplicates. If true duplicates are discovered, they can be removed.



1	bread	coffee	soymilk	quinoa
2	coffee	quinoa		
3	bread	coffee	soymilk	quinoa

(a)



	Name	Age	Profession	Income	SS#
1	Brown, Bob	29	Engineer	120,000.00	232-23-2323
2	Smith, Sally	34	Professor	90,000.00	563-23-9465
3	Gomez, Greg	56	Writer	75,000.00	765-63-8285
4	Brown, Bob	29	Engineer	120,000.00	232-23-2323
5	Brown, Bob	29	Engineer	120,000.00	232-23-2323

(b)

Figure 12: (a) illustrates transaction data for which two rows appear to be identical, but are not true duplicates. In this case, no rows should be eliminated. (b) illustrates record data for which two rows (4 and 5) are clear duplicates. In this case, one of the rows should be removed.

If duplicates are discovered and confirmed, they can be eliminated so that the information contained in the dataset is not skewed or exaggerated by repeated identical observations. Duplicate removal should never be perfunctory and should be considered with caution.

In summary, the initial steps of data cleaning include the use of observation, visualization, statistical measures, and programming to discover and manage missing values, incorrect values, inconsistent values, and duplicate values. The format and nature, as well as the type of the data, should be considered and corrected as needed. Once these initial cleaning steps are complete, the next set of steps in data cleaning can begin. These include outlier identification and management, as well as data conversion, normalization, and transformation.

Stage 2: Data Conversion and Transformation

Data Conversion

Once the initial stage of cleaning is completed so that missing, incorrect, inconsistent, and duplicate values have been properly discovered and managed, the next stage of preparation may include data conversion and transformation. While the terms "conversion" and "transformation" are often used interchangeably, and while it can be said that data conversion is the transformation of data from one form to another, the two terms can be thought of as describing the extent of the effect of the change.

Data conversion can be thought of as converting data values, columns, or rows to more appropriate formats or types. For example, the variable "Gender" might be read in as a character string, but converted to a factor or categorical type. This concept was introduced at the start of this chapter and will be revisited here. Changing data types is a direct conversion that does not change the values, but rather changes the type, format, or structure. Similarly, data might contain date or time values that may have been read in as characters or strings and must be converted to more appropriate date-type formats; either for ease of use or as required input for a given programming language.

Data Transformation

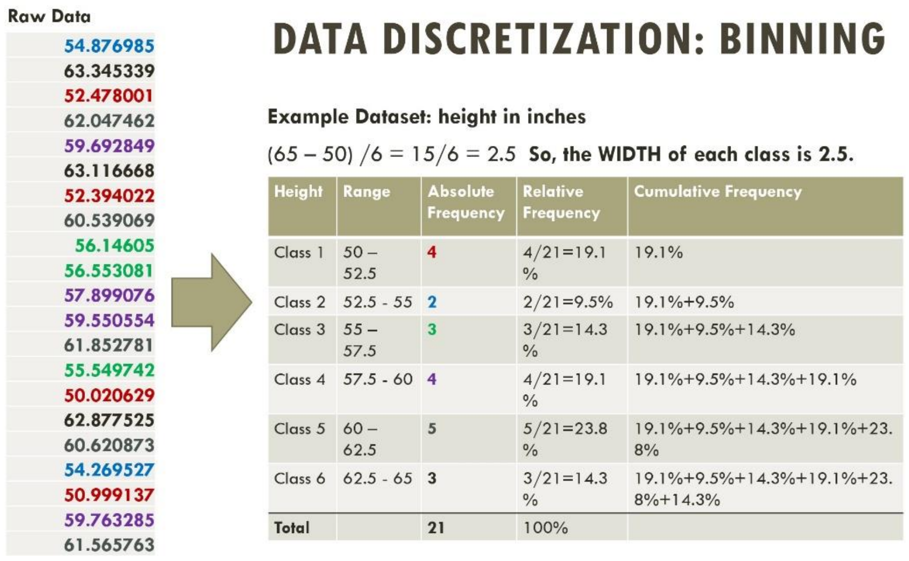
Data transformation involves transforming or changing the expression of the data. Examples of data transformation include discretization (such as binning), applying a function, such as the log or square root, or normalizing data. In some cases, transforming data changes not only its innate format, but also its specific information. To see the difference, consider an "Age" variable. Age is quantitative and continuous, and is assumed to range in value from 0 to 120. Age can be represented using decimal values, for a more precise representation, or by integer values. If the Age data is transformed via discretization (binning) and categorized into 3 groups, it becomes ordinal and categorical, and loses some of its original information.

For example, placing numeric values into groups or categories (or bins) is called discretization or binning. One such binning option for the Age variable (with numeric data) is to create three Age Groups: “Under 21,” “21 - 49,” and “50 and Over.” Next, all age values that fit into “Under 21” are updated to reflect “Under 21” as a label or level. Therefore, the label replaces the numeric value such that the age value of 20 would be transformed into “Under 21.” This transformation changes the nature of the data and the information it contains. If “20” is converted to “Under 21,” then the precise value of “20” is lost.

Because transformation, unlike basic format or type conversion, can alter the data information, it is often best to retain the original data and to create a new column (a new feature) in which to place the newly transformed data. Creating a new column in a dataset is called “feature engineering or generation. Which will be discussed in a following section. In addition to discretizing data, data can be altered (transformed) by applying a function to the data, such as a log function, a quadratic function, a normalizing function, a sigmoidal function, and so on. While applying a function with a defined inverse will not irreparably change the data, it can alter its behavior and appearance, especially in models and visualizations. As with discretization, it is often a best practice to use feature generation (create a new column) to house functionally transformed data.

Regression, or “smoothing,” can also be considered a transformation. Like discretization (which is a type of smoothing), smoothing will alter the information contained in the data. Linear regression smoothing uses a minimization function, such as the sum of squares difference to calculate a “best-fit” or regression line that most closely estimates all data points in the columns of interest. The result is a linear equation that offers an estimate of the original data.

Figure 13 illustrates smoothing using discretization which is also called binning.



Raw Data
54.876985
63.345339
52.478001
62.047462
59.692849
63.116668
52.394022
60.539069
56.14605
56.553081
57.899076
59.550554
61.852781
55.549742
50.020629
62.877525
60.620873
54.269527
50.999137
59.763285
61.565763

DATA DISCRETIZATION: BINNING

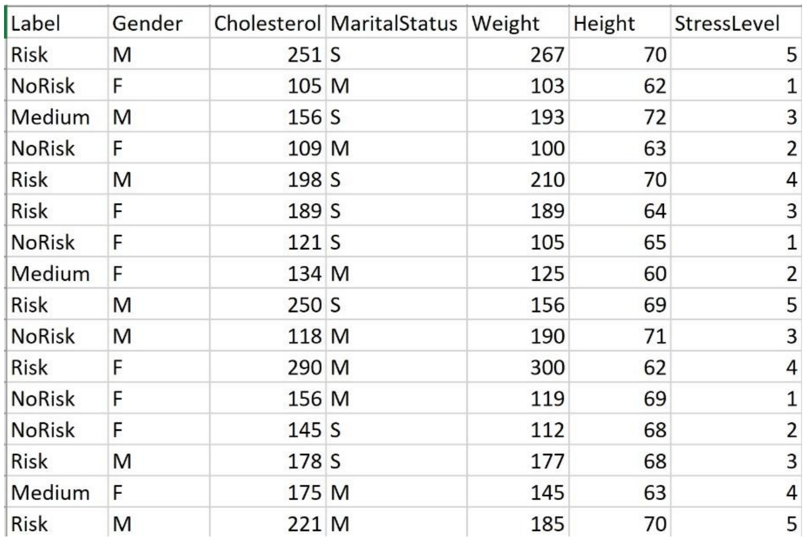
Example Dataset: height in inches

(65 – 50) /6 = 15/6 = 2.5 **So, the WIDTH of each class is 2.5.**

Height	Range	Absolute Frequency	Relative Frequency	Cumulative Frequency
Class 1	50 – 52.5	4	4/21=19.1 %	19.1%
Class 2	52.5 - 55	2	2/21=9.5%	19.1%+9.5%
Class 3	55 – 57.5	3	3/21=14.3 %	19.1%+9.5%+14.3%
Class 4	57.5 - 60	4	4/21=19.1 %	19.1%+9.5%+14.3%+19.1%
Class 5	60 – 62.5	5	5/21=23.8 %	19.1%+9.5%+14.3%+19.1%+23.8%
Class 6	62.5 - 65	3	3/21=14.3 %	19.1%+9.5%+14.3%+19.1%+23.8%+14.3%
Total		**21**	100%	

Figure 13: Using discretization to smooth continuous data into discrete categories.

Data can be transformed in many ways and for different purposes. As with all of data science, the goals, analysis, and models determine how and if data is to be transformed. The following will show three transformation examples. The starting dataset is called HeartRisk.csv and it is pretend data used only to illustrate data transformations. Figure 14 shows the dataset.



Label	Gender	Cholesterol	MaritalStatus	Weight	Height	StressLevel
Risk	M	251	S	267	70	5
NoRisk	F	105	M	103	62	1
Medium	M	156	S	193	72	3
NoRisk	F	109	M	100	63	2
Risk	M	198	S	210	70	4
Risk	F	189	S	189	64	3
NoRisk	F	121	S	105	65	1
Medium	F	134	M	125	60	2
Risk	M	250	S	156	69	5
NoRisk	M	118	M	190	71	3
Risk	F	290	M	300	62	4
NoRisk	F	156	M	119	69	1
NoRisk	F	145	S	112	68	2
Risk	M	178	S	177	68	3
Medium	F	175	M	145	63	4
Risk	M	221	M	185	70	5

Figure 14: Pretend record dataset that represents labeled data with both qualitative and quantitative data types.

Example 1: Normalizing Data with min–max

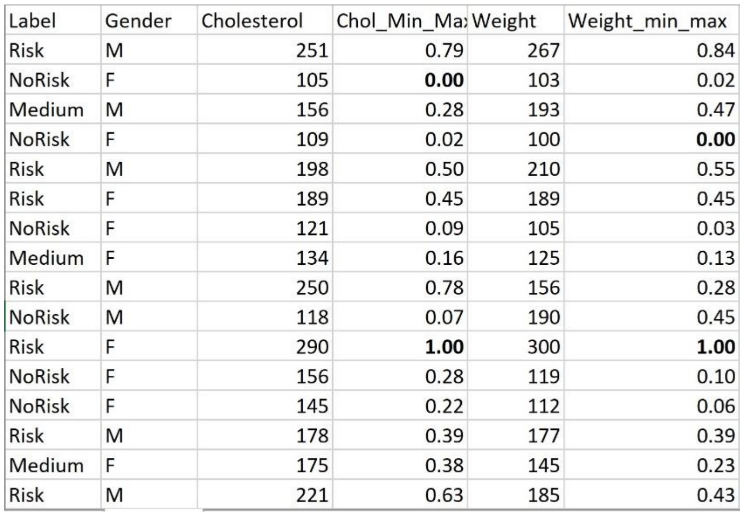
The min-max normalization method transforms all values so that each value occurs between 0 and 1.

For example, for Cholesterol, the min value the dataset in Figure 11 is 105, and the max is 290.

The min – max formula is:

Transformed value x = (x – min) / (max – min)

Figure 12 illustrates this transformation normalization for Cholesterol and for Weight.



Label	Gender	Cholesterol	Chol_Min_Ma	Weight	Weight_min_max
Risk	M	251	0.79	267	0.84
NoRisk	F	105	**0.00**	103	0.02
Medium	M	156	0.28	193	0.47
NoRisk	F	109	0.02	100	**0.00**
Risk	M	198	0.50	210	0.55
Risk	F	189	0.45	189	0.45
NoRisk	F	121	0.09	105	0.03
Medium	F	134	0.16	125	0.13
Risk	M	250	0.78	156	0.28
NoRisk	M	118	0.07	190	0.45
Risk	F	290	**1.00**	300	**1.00**
NoRisk	F	156	0.28	119	0.10
NoRisk	F	145	0.22	112	0.06
Risk	M	178	0.39	177	0.39
Medium	F	175	0.38	145	0.23
Risk	M	221	0.63	185	0.43

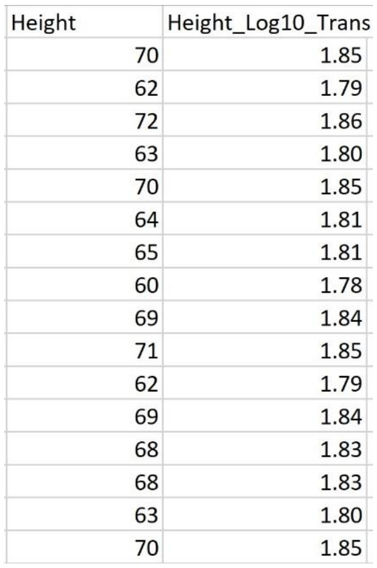
Figure 12: Cholesterol and Weight are normalized using min-max.

In Figure 12, notice that the minimum or smallest value for Cholesterol is 105. Using min-max will transform this to "0". Similarly, min-max will transform the largest value to 1. Also notice that after the normalization, both Cholesterol and Weight can be directly compared as they are now presented on the same scale (relative to the values in each variable column).

Example 2:

There are many other options for data normalization, such as transforming using the z-values or, if working with text frequencies and documents, tf-idf.

Transforming data can also include applying the log to all values. This will "contract" all of the values onto a scale that might be easier to visualize. Figure 13 illustrates a log base 10 transformation of the variable Height. Notice that the log transformation makes the values much smaller and closer together. Similar methods can be used to transform data, such as squaring or cubing all of the values. This would have the opposite effect as the log transform and so would move the numbers farther apart and make them larger. Why would one want to do that? One reason is visualization using numbers to choose the colors. When the numbers are "more different" from each other, then the colors will be easier to tell apart.



Height	Height_Log10_Trans
70	1.85
62	1.79
72	1.86
63	1.80
70	1.85
64	1.81
65	1.81
60	1.78
69	1.84
71	1.85
62	1.79
69	1.84
68	1.83
68	1.83
63	1.80
70	1.85

Figure 13: The result of the log base 10 transformation of the data under the Height column.

Stage 3: Noise and Outliers

Noise in data is generally considered irrelevant, erroneous, or meaningless data values. The concept of noise overlaps with the idea of incorrect values, as well as with outliers. An incorrect value can be considered noise and the management of incorrect values was discussed above.

An outlier, while noisy and perhaps incorrect, is in its own data –cleaning category. While a confirmed outlier may be considered noisy and incorrect, not all noisy or incorrect values are outliers. Specifically, an outlier is a value that is significantly "far" or "different" from the other values and from what is expected. For example, consider again the Age variable. The value of "-1" is incorrect, but not an outlier. The value of 125 is likely incorrect, but also not an outlier. However, a value of 13456 is an outlier. An outlier value is significantly different (non-similar) to expected values. The concept of significantly different requires a measure of similarity.

Because outliers and incorrect values are often detected and managed separately and individually, the idea of "noise" often takes on its own meaning. Noise in data, unlike an incorrect value or outlier, may more often occur in many values. Noise in data can be highly dependent on the nature of the data and the methods by which it was gathered. Noise might be introduced into data implicitly due to variation in sensors or tools. For example, if a measurement instrument has an innate inaccuracy, this inaccuracy will be passed on to the data it generates. In some cases, the cause of the noise is consistent and can be measured, identified, and perhaps removed or cleaned from the data systematically.
As a simplified example, suppose a scale is imprecise with a Gaussian (normal) distribution. If this distribution is identified, the error can be singled out and removed from the data. However, not all noise is consistent nor does it always adhere to a pattern.

Outliers are incorrect and noisy values that are far from the expected data. Outliers are different from other types of noise or incorrect values in that they are considered to be "far" from what is expected. The most challenging question to answer when identifying outliers is what constitutes the idea of a data value being "far" or "significantly different". This question can be broken down into two questions.

What defines far?

Does "far" necessarily imply that the value is incorrect?

One of the most famous outlier removal blunders occurred when NASA's Nimbus-7 satellite "removed" values that were in fact correct but were mistaken as outliers. These so-called outliers were thus ignored and resulted in the continued and unnoticed (so unchecked) depletion of the ozone layer. Nimbus-7 was programmed to delete all values that were "sudden, large drops in ozone concentration." When the Nimbus-7 was re-run without the outlier-drop encoded, it was discovered that the ozone hole could be seen as far back as 1976. This error was not discovered until 1985, by Farman, Gardiner, and Shanklin (Nature, May 1985).

Outliers can have a significant effect on data analysis and the information generated from gathered data. A data value that seems unusual may or may not be a true outlier and may or may not be incorrect. In some cases, values that seem like outliers are simply new and important data that should be retained and considered. However, there are also cases where outliers are incorrect. When prior information is well known about the distribution of the data, such as adult, female height, it is easier to identify an outlier. For example, a female adult height value of 120 inches can be identified as an outlier with considerable certainty. The identification of true outliers becomes much more challenging when the distribution of the data is unknown, unfamiliar, or chaotic.

There is no universal definition for an outlier. Common definitions include an observation that differs so much from other values that it raises suspicion (Hawkins, 1980), or an observation that appears to be very inconsistent with the other data (Barnett and Lewis, 1994). In both of these definitions, the notion of an outlier is that it is rare (so very few or just one data point), and very different (far) from all the other data points. However, even the idea of "different" is based on a measure of similarity or distance, and different distance measures can offer different results.

Information about the data and the nature of the data can, in some cases, can offer support for outlier detection. For example, if the data is known to be normally distributed, an outlier may be defined as any value that is more than 3 or 4 standard deviations from the mean. Alternatively, suppose the data is uniformly distributed, such as buying patterns for a given stock that are generally consistent year after year. If a spike is observed on a single day in the year and that spike is 5 or 10 times the normal pattern, this may also arouse suspicion. However, suspicion does not imply an outlier. Suspicion should draw attention to the data value and should engage further research and evaluation.

Another example may occur in medical research, such as in a new pharmaceutical trial. In such a case, if 99% of the participants show positive results, but 1% fall very ill, the 1% is not an outlier, but rather an indication of the need for further research. Recall that 1% of 7 billion people is 70 million people.

If an outlier is identified with relative certainty, its management is very similar to that of incorrect values. The row can be removed or the value can be replaced with a measure such as the column mean or median. There are no set rules, and each dataset, application, and data value should be carefully considered. Several outlier detection methods include statistical, distance-based, k-nearest neighbor (kNN) based, clustering-based, and an approach based on the local outlier factor (LOF) of an object.

Outlier Detection Methods

An outlier is a data value or vector that is notably dissimilar from the other values or vectors in the dataset. This definition is ambiguous and in some cases, unusual values are not true outliers. The idea of different, far, dissimilarly, unusual, or unexpected, are all subjective ideas. To identify an outlier, a threshold for dissimilar must be set.

Many factors affect outlier detection, including the dimensionality of the data being investigated. For example, consider a standard record-style dataset made up of rows and columns. Assume that this example dataset has 100 rows and 10 columns, and also assume that each of the 10 columns represents a numeric variable, such as Age, Height, Weight, etc. In such a case, one might search for outliers within each column individually. By investigating each column, the use of visualization is feasible, given that one column of data is one dimension. Alternatively, one might investigate all vectors (rows) in the dataset. Each vector or row in this example is a 10-dimensional point in 10-D space. One could then ask if any of the rows were significantly dissimilar from the other rows in the dataset. In this case, visualization would not be feasible without the use of reduction (such as PCA).

The dimensionality (or number of columns) involved in the outlier detection will directly affect which outlier detection method to select. While a histogram or boxplot may reveal distinct values for a single column (1D) of data, and a scatterplot may be interesting to investigate outliers for two-dimensional data, higher dimensional data may require other methods such as distance, density, or cluster based. The following subsections will discuss the more common methods of discovering outlying values and vectors.

Statistics-based Outlier Detection Methods

Statistics-based outlier detection utilizes presumptions about known distributions and related expectations. For example, if a column of data values is suspected to be normally distributed, then all values outside of a given range (such as mean +/- 3 std dev) can be further investigated as potential outliers. This is a reasonable threshold for Gaussian (normal) data because it is known that values farther than 3 standard deviations from the mean are highly improbable. Of course, the threshold of "3 deviations from the mean" is arbitrary and subjective. One might choose a threshold of 2.5 deviation or 4 deviations instead.

More generally, if the distribution of the data is known, selecting a threshold for "far or unusual" can be done with greater confidence. The assumption is that outliers must deviate strongly from the expected distribution. Statistical measures such as mean, standard deviation, range, min, and max can all be employed in the analysis of data and in the detection of unusual data values. The use of statistical tests and visualization quickly break down as the dimensionality of the data increases. Alternative methods include distance-based, density-based, and cluster-based.

Distance-Based Outlier Detection Methods

Distance-based outlier detection can measure the distance between all pairs or groups of points or can measure the distance between each point and a measure of center (such as mean or median). If data is normally distributed, then each point may be compared with the mean and its distance in standard deviations may be used as a distance measure. It may be determined that an outlier must be 3 or 4 deviations from the mean. This determination is not set and can be decided by the researchers and the nature of the data and goals. Another way of thinking about a data value being “far” is that its “locality” is sparse (Aggrawal, 2013). In other words, it is not near to other points. When using distance-based detection, a measure of distance must be selected. Common options are Euclidean, Manhattan, Mahalanobis, and Cosine Similarity. However, there are many measures of distance and the choice often depends on the nature of the data. If a given point has a distance from all other points that exceeds a selected threshold, then that point may be considered as a possible outlier. This raises the immediate question of how the threshold is determined.

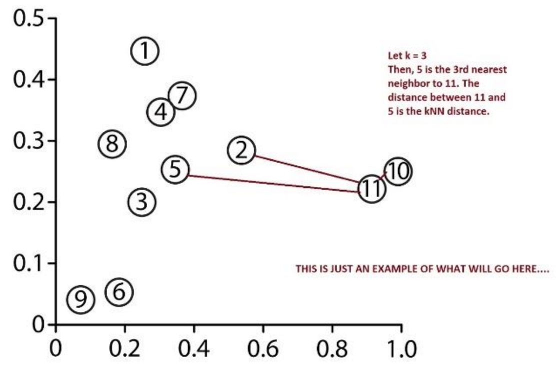
Knorr and Ng were to first to formalize a distance-based outlier detection method. The method states that a data value in a dataset is an outlier if its distance to all other values in the dataset exceed a pre-selected threshold. This requires that every point-to-point distance is measured and compared with the threshold. The immediate drawback to distance-based methods is selecting the threshold. However, evaluations on the distances between all pairs of points can be used to make educated guesses. Another challenge presented by distance-based outlier detection is the ability to rank or compare outliers. A possible solution to these issues is the use of k-Nearest Neighbor (kNN).

K-Nearest Neighbor Outlier Detection

The k-Nearest Neighbor (kNN) method uses distance to determine “neighbors” or closer points. For example, suppose k is 3. Then, the distance measure for a data point is set as the distance between that data point that the 3rd closest point to it. The following example illustrates this measure.

kNN Example:

- The kth-NN score $q_{k\text{thNN}}(x) := d^k(x; X)$
  - $d^k(x; X)$ is the distance between x and its kth-NN in X

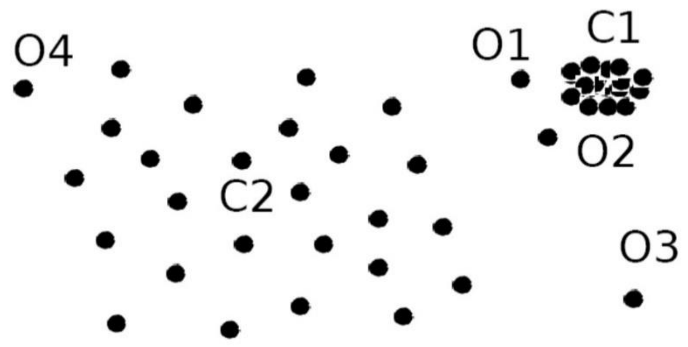


If using kNN to determine distance, if the distance between the data point and its kth neighbor exceeds a selected threshold, then the point may be considered an outlier. A key issue with the kNN approach occurs when data points within the dataset are part of different distributions and so have a different level of point density. In other words, some collections of points may be generally closer together than other collections of points. To mitigate this issue, the density-based local outlier factor (LOF) method may be employed (Breunig et al. SIGKDD 2000).

Density-Based Outlier Detection and LOF

Before investigating the LOF method, it is relevant to distinguish between local and global outliers. A global outlier is a data value that is considered significantly or suspiciously far or distant from all other data values. A local outlier is considered to be different or distant from its local neighbors. This idea takes density into account. Consider the following illustration in Figure XX. Here, the relative distances between data values in each cluster is different. Therefore, point o1 is a local outlier to cluster c1, as is o3. However, interestingly, o1, o2, o3, and o4, may not be outliers to cluster c2 (depending on the threshold and method selected).

Example:



The LOF method uses the notion of a "reachability" distance. The LOF blends together the standard distance measure with the kNN measure to create a more robust measure. For example, the reachability distance from point p1 to point p2 is the maximum of the distance between p1 and p2 and the distance between p1 its kth nearest neighbor. This reachability distance can be calculated for all points in the dataset. The distance measure used may be Euclidean, Manhattan, or any other selected distance measure.

Cluster-Based Outlier Detection Methods

Clustering is the process of discovering collections or groups of data points (rows or vectors) that are similar or close to each other and dissimilar or far from other points or clusters. Clustering can be partition-based, density-based, or hierarchical. Not all clustering methods place all points into clusters, and a natural by-product of clustering may be the discovery of points that do not fit well into any cluster or that disrupt a cluster. Such points may be potential outliers. While clustering and outlier detection are distinct goals, they are clearly coupled.

There are several clustering methods that lend themselves to outlier detection. For example, k-means clustering can be used to identify outliers. As a first step, the data are placed into k clusters. Next, the distance from each data point to the centroid of the cluster in which that point was placed is measured. Points with the largest distances from their cluster centroids are further evaluated. As with other clustering methods, the selected distance measure may affect the results, and the value of k (in k means) and the threshold for labeling points as "far" from their cluster centroid are all subjective. Figure 14 illustrates clusters and potential outliers.

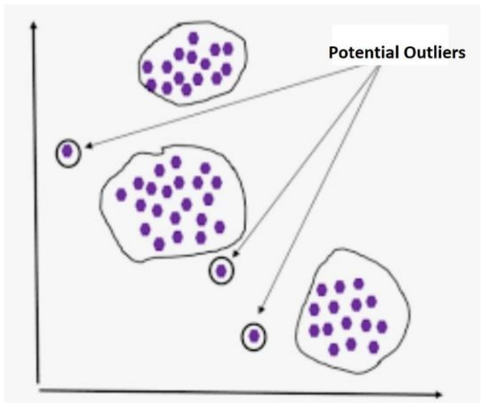


Figure 14: Clustering and potential outliers.

High-Dimensional Outlier Detection and ABOD

Angles and measures such as cosine similarity can be more stable when datasets have a high degree of dimensionality. The "Angle-Based Outlier Degree" (ABOD) approach utilizes angles and specifically a measure of cosine similarity to determine if a given data point (vector) is a possible outlier.

The ABOD method considers each vector (row or object) in a dataset. The vector is considered to be an outlier if most of the other vectors in the dataset are located in similar directions. Alternatively, a vector is not an outlier if many other data vectors are located in various directions with respect to that point. Figure 15 below illustrates this interesting method. Like all other outlier discovery methods, this method is subjective. One must determine what is meant by "many" and "similar."

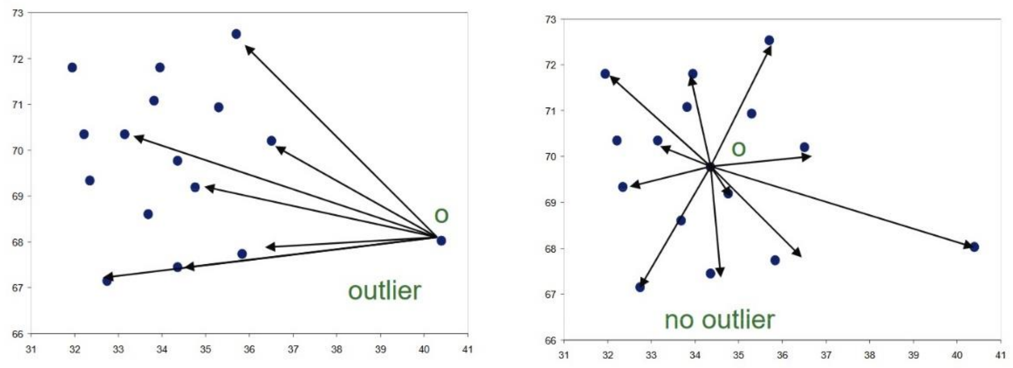


Figure 15: Illustration of the "Angle-Based Outlier Degree" (ABOD) approach to finding outliers.

The ABOD method uses angles between data points (vectors or rows) to discover potential outliers. The range or spectrum of all of the angles can be visualized for both threshold generation and discovery. For each data point, the variance of all angles made with that point is calculated. A higher variance implies that the point is not an outlier. This corresponds to the image on the right above in Figure XX. If the variance is small, this implies a possible outlier and corresponds to the image on the left in Figure XX above.

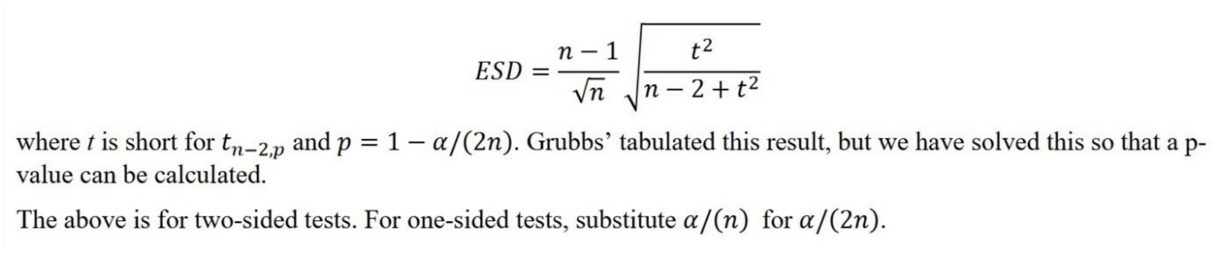
Grubbs' Outlier Test

Frank Grubbs (author of the Grubbs' Outlier Test, 1950, 1969, etc.) notes that an outlier may be a correct but extreme value that can exist within the distribution of the data (so not a true outlier) or it may be the result of a deviation from the experimental procedure or an error in calculation or data recording (a true outlier).

The Grubbs procedure (1950) looks at the distance from a datapoint to the dataset sample mean. The goal is to determine if the point is an outlier. Grubbs defined a measure called the extreme studentize deviate (ESD).

ESD = max i = 1...n |xi - sample mean | / s , where the dataset has n data points, xi is one of the data points, the sample mean is calculated by summing all data points and dividing by the number of points in the dataset, and s is the sample standard deviation.

The significance level of the ESD is then determined using the following formula:



$$ESD = \frac{n-1}{\sqrt{n}} \sqrt{\frac{t^2}{n-2+t^2}}$$

where t is short for $t_{n-2,p}$ and $p = 1 - \alpha/(2n)$. Grubbs' tabulated this result, but we have solved this so that a p-value can be calculated.

The above is for two-sided tests. For one-sided tests, substitute $\alpha/(n)$ for $\alpha/(2n)$.

The Grubbs Test is intended to investigate a single data point. To investigate multiple outliers in a similar way, the Rosner (2011) method is suggested.

In summary, noise and outliers can affect the information contained in a dataset. There are many methods for the identification and management of noise and outliers. The nature and dimensionality of the data should assist in selecting an appropriate method. In all cases, outlier removal must be done with care and transparency so as not to distort or corrupt the true information contained in the data.

Stage 4: Harmonization and Feature Generation

Data Harmonization

It is often the case that multiple datasets are utilized in data analysis. In addition, such multiple datasets may have arisen from different sources and may have different formats and naming conventions. Harmonizing data involves blending and combining multiple datasets together such that all information is properly retained and all datasets come together to form a single source of consistent data. Data harmonization enables a connected and cohesive view and improves the robustness of predictive and descriptive analysis.

Data cleaning and processing may be done both before and after data harmonization. The result of data harmonization is that all data related to a great process or enterprise is blended, follows the same format, and is cleaned and prepared for analysis.

Harmonizing multiple datasets can be a time-consuming process. In many cases, variable naming conventions, formats, data types, and data structures will not match. Inconsistencies between different datasets, even containing identical variables, are likely to occur. A simple example is the representation of "Gender." Different datasets may use different variable names for Gender, such as “Sex” or “Sexual Orientation.” Some datasets may restrict gender to two options, while others may permit all of the defined gender categories. Finally, some datasets may use 0 for male and 1 for female, while others may use “M” and “F.” All such inconsistencies must be discovered and addressed while harmonizing data.

Consistency Within Datasets

Once a dataset is cleaned and pre-processed, a next step is to assure consistencies, expectations and dependencies within the dataset. For example, suppose a dataset has two columns: Age and Cost. Next suppose the Cost depends on the Age, such that Age < 50 requires Cost < 100 and Age >= 50 requires Cost < 45. In this case, the dataset may be fully cleaned, but still incorrect. Each Age and Cost must be checked for correctness per the particular requirements of the dataset.

In general, knowledge about the dependencies and expectations of the variables in a dataset may assist with further cleaning of the data.

Feature Generation

Feature generation (also known as feature engineering) is the process of creating new variables (or features) using the current variables in the dataset. New features, through the utilization of domain knowledge, may be created through discretization, the application of a function, or the alteration of a format.

Feature generation has the potential to improve both supervised and unsupervised techniques and can offer greater insight into the data. For example, suppose a dataset contains the variable DateOfBirth. This is important information, but will create a challenge when performing analysis. However, the DateOfBirth variable may be used to generate a new feature (column) called Age, which is created using a function that subtracts the date of birth from the current date. While DateOfBirth is a “date” type and is not quantitative, Age is a numeric data type that can be used more broadly in analyses. This type of feature generation, in which a new column is created using direct information from an existing column, is known as “derived.” The new feature, Age, was derived from an existing feature, DateOfBirth.

Similarly, variables in a dataset may be quantitative and continuous. While this format may be excellent for some models and methods, it may not be ideal for others. An option is to create a new feature that bins or discretizes the data into classes or categories. Categorical data may then be used to train supervised machine learning models. For example, student grades may be quantitative and continuous, such as 90.34% or 78.1%. However, such data may be binned into groups or categories called: A, A-, B+, B, B-, etc. By creating a new “Letter Grade” feature in the dataset, further predictive and descriptive analyses may be performed.

Measuring Data Cleanliness and Quality

Data Quality

In a very general sense. data quality refers to the ability of data to accurately, reliably, and sustainably serve its purpose. Inaccurate decisions made from “bad” data are inconvenient and exceptionally costly. According to Gartner research, “the average financial impact of poor data quality on organizations is $9.7 million per year.” (https://www.gartner.com/smarterwithgartner/how-to-create-a-business-case-for-data-quality-improvement/, 2018)

For those who use data to make decisions, the quality or benefit of the data depends on data completeness, correctness, and timeliness. For example, incomplete data might be data that does not fairly represent the population. Incomplete data may be unbalanced (such as 95% female and 5% male observations), or not large enough to represent all cases. Data with many errors, or incorrect data, may not only poorly represent the population, but may falsely represent it. Incorrect data can result in incorrect decisions and conclusions. Finally, timeliness refers to the availability of the data when connected decisions are to be made.

Within the three core dimensions of completeness, correctness, and timeliness, are several further factors to consider. The following table illustrates several quality considerations within the three levels of completeness, correctness, and timeliness, as well as brief definitions.

Correctness	
Correctness: Error-Free	How correct and reliable is the data.
Correctness: Believability	How reliable and true is the data.
Correctness: Consistent and Concise Representation (Formatting)	Is the data consistent and are all values within variables represents both concisely and consistently?
Correctness: Unbiased, Non-prejudice, Balanced.	The data fairly, evenly, and without prejudice or skew represents the population.
Completeness	
Completeness: Missing Values	The number of missing values in the data and also relative to the size of the data.
Completeness: Relevancy	The extent to which the data is useful for the purposes required.
Completeness: Understandability	Is the meaning of the data and what is represents both clear and easy to understand.
Timeliness	
Timeliness: Accessibility	The data is available to be used in time to make decisions.
Timeliness: Currentness	The data is up-to-date and still relevant and representative.

Table 1: Data Cleanliness Questions for Decision Science

Data, as well as related models, methods, measures, and applications are unique and situation-dependent. Therefore, the consideration of data quality must also be both situation-dependent as well as continuous. Each situation can benefit from an appropriate set of quality metrics that are re-applied often to both measure and maintain overall and on-going data quality. Quality assurance can be considered an effort of developing both objective and subjective measures based on essential and particular elements, goals, and applications of the data.

Measures of Cleanliness: Quality Metrics

Data can come in all states of uncleanness and can be placed (or can originate) in a wide variety of formats. For example, if data is collected from a website using the website source, the data will contain text, HTML, and other common webpage content. Alternatively, if data is generated from an image, it will appear as a matrix of numeric values. As discussed in this chapter, data types and formats direct affect data cleaning methods and choices. Consequently, data cleaning methods and choices will affect data cleaning metrics. The following explanations and descriptions will focus on record data (rows and columns) such that each column represents a feature (also known as a variable or attribute), and each row represents an observation (also known as a vector or instance). While cleaning record data, or any other format of data is partially unique to the nature and goals of the data, there are some common steps. These steps and their metrics will be addressed here.

Missing Value Metrics:

Missing values are values in a record-formatted dataset that are not available and so are blank or are labeled with an NA or an NaN. When considering the number of missing values within a dataset, both missing values in columns and missing values in rows can be viewed.
Record data columns represent data collected to represent a given variable. For example, a dataset may have a column called "Age" such that the data in that column are numeric values that represent the age of each observation or row. It is also possible for a variable (a column name) to be a word, such as "hike". This is common when working with text-based data that can been vectorized (converted into record format with the words as variables).

When investigating missing values, each column of a record-format dataset can be considered, as can each row. The number of values missing from a column, from a row, or from the entire dataset, can be counted. From there a percentage can be calculated. Figure XXX illustrates a set of metrics that measures missing values.

ID	AGE	HEIGHT
NA	23	64.3
2739	NA	66.4
2992	25	62.3
NA	55	62.3
4486	27	59.6
4488	NA	NA
4878	25	59.8
NA	22	63.3
NA	41	67.9
4835	32	61.4

General Formulas:

Percentage of Missing Values Per Column:
Percentage of Missing Values in ColumnX =
Number of Missing Values in ColumnX / Total Number of Values in ColumnX

Examples:
Percentage of Missing Values in the ID column = 4/10 = 40%
Percentage of Missing Values in AGE column = 2/ 10 = 20%
Percentage of Missing Values in the HEIGHT column = 1/10 = 10%

Percentage of Rows with Missing Values:
Percentage of Rows with Missing Values=
Number of Rows that contain at least one missing value / Total Number of rows
Example:
Percentage of Rows with Missing Values = 6/10 = 60%

Percentage of Missing Values Per Total Values:
Percentage of Missing Values Per Total Values =
Total Number of Missing Values / Total Number of Values

Example:
Percentage of Missing Values Per Total Values = 7/30 = 23.3%

Figure 16: Different methods for measuring the number of different values in a dataset.

Figure 16 above illustrates different methods for measuring the number of different values in a dataset. The example dataset above contains three variables, ID, AGE, and HEIGHT. The metrics in Figure 16 suggest several things. Because the number of rows with missing values is 60%, it may be an unwise choice to simply remove rows that contain missing values. If all rows with missing values were removed, 60% of the data would be lost. This would not only affect the size of the dataset, but also the integrity. For example, removing rows may alter the mean and median of both the AGE and HEIGHT columns, thereby altering the original information contained in the dataset. Similarly, removing 60% of the rows may affect measures of variation and other statistical measures (such as the minimum, the maximum, the range, the mode, etc.) In other words, removing rows, "changes" the data. Altering data alters the information contained in the data, the results generated from the data, and subsequently decisions made from those results.

Notice that the ID column, while perhaps an important label to retain for reference, does not contain "information". For this reason, it is feasible to consider removing the ID column and then recalculating all of the metrics. Figure 17 illustrates this step.

AGE	HEIGHT
23	64.3
NA	66.4
25	62.3
55	62.3
27	59.6
NA	NA
25	59.8
22	63.3
41	67.9
32	61.4

Percentage of Missing Values Per Column:
Examples:
Percentage of Missing Values in AGE column = 2/ 10 = 20%
Percentage of Missing Values in the HEIGHT column = 1/10 = 10%

Percentage of Rows with Missing Values:
Example:
Percentage of Rows with Missing Values = 2/10 = 20%

Percentage of Missing Values Per Total Values:
Example:
Percentage of Missing Values Per Total Values = 3/20 = 15%

Figure 17

Figure 17 above shows that removing (and saving for reference) the ID column improves the missing values metrics; in some cases, significantly. For example, the percentage of rows with missing values reduced from 60% to 20%. Metrics that measure and compare missing values rates can be both helpful in making cleaning decisions and can offer transparency in cleaning.

In addition to measuring percentages of missing values, further metrics may be applied to access the effects of correcting missing values. For any row that contains one or more missing values, the row can be eliminated or the missing values can be corrected. In either case, several measures (including measures of center and variation) will be altered. Metrics can be used to determine the extent of the alternation. Figure 18 below will illustrate several before and after measures.

	AGE_original	AGE_replace_mean	AGE_replace_median
	23	23	23
	NA	**31.25**	**26**
	25	25	25
	55	55	55
	27	27	27
	NA	**31.25**	**26**
	25	25	25
	22	22	22
	41	41	41

	32	32	32
Mean	31.25	31.25	30.2
Variance	113.69	90.95	95.36

Figure 18:

Here, the mean of the original AGE data is 31.25 and the median is 26. Replacing NA/missing values with the mean alters the variance, but not the mean. It reduces the variance. Replacing NA/missing values with the median reduces the variance and the mean (in this case).

Figure 18 above illustrates that replacing missing values with measures of center, such as the mean or the median can affect the overall measures of center and variation of the column. In the example above, the original AGE column contains two missing/NA values. The column to the right shows the result if these two missing values are replaced by the column mean of 31.25. The last column shows the result if the two missing values are replaced by the column median.

Recall that metrics do not offer a judgement about whether updates to a given dataset are acceptable or prudent. Rather, they offer measures that illustrate how updates have altered the information within the dataset. Whether such alterations are acceptable must be determined on a case-by-case basis with further knowledge of the dataset, the goals, and the decisions that will be made based on the data.

The following Table 2 offers a review of several metrics that can be used when managing missing values.

Percentage of Missing Values Per Column	#NA in Col / Total Values in Col	**Percentage of Missing Values per Dataset**	# of missing values in dataset / # values in dataset
Percentage of Rows in a dataset with one or more missing values.	# Rows with one or more NA / Total # of rows	**Before and After measures for center and variation one missing values are updated or removed.**	1) Column mean before and after replacing missing values with median. 2) Column variance before and after replacing missing values with mean or median.

Table 2

Incorrect Value Metrics

Because incorrect values are corrected in much the same way that missing values are, the same basic metric apply in both cases. The key difference between incorrect versus missing values is that missing values are much easier to locate. If an incorrect values goes undetected, it can affect the quality of the dataset and the results.

Outlier Management Metrics

Several methods for detecting outliers were discussed in previous sections. This section will focus only on before and after metrics. Outliers can be detected only within a given column, such as a age that is 456. Outliers can also be detected between rows. For example, if a clustering method is used to cluster all “points” (rows) in a numeric dataset, and a few of the rows do not fit well into any of the clusters, these rows can be considered to be outliers.

Before and After Metrics: Removing Outliers from a Column

Once a column is determined to have one or more outliers, the common method of cleaning is to remove these outliers. Removing values that are considered to be outliers can affect several measures, such as the mean, median, variance, minimum, maximum, and range. In addition, removing one or more outliers may affect more complex functional measures such as linear regression equations.

To measure all such changes, both before and after measures of at least the column mean, median, variance, range, minimum, and maximum should be determined. From there, percentage differences between the before and after measures may be calculated.