WINE INFORMATICS: CLUSTERING AND ANALYSIS OF PROFESSIONAL WINE REVIEWS

by

Christopher Thomas Rhodes

A thesis presented to the Department of Computer Science
and the Graduate School of University of Central Arkansas in partial
fulfillment of the requirements for the degree of

Master of Science
in
Applied Computing

Conway, Arkansas
May 2015

TO THE OFFICE OF GRADUATE STUDIES:

The members of the Committee approve the thesis of Christopher Thomas Rhodes as presented on April 20, 2015.

_____
Dr. Bernard Chen, Committee Chairperson

_____
Dr. Chenyi Hu

_____
Dr. Shengli Sheng

_____
Dr. Yu Sun

PERMISSION

Title Wine Informatics: Clustering and Analysis of Professional Wine Reviews

Department Computer Science

Degree Master of Science

In presenting this thesis/dissertation in partial fulfillment of the requirements for graduate degree from the University of Central Arkansas, I agree that the Library of this University shall make it freely available for inspections. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my thesis/dissertation work, or, in the professor's absence, by the Chair of the Department or the Dean of the Graduate School. It is understood that due recognition shall be given to me and to the University of Central Arkansas in any scholarly use which may be made of any material in my thesis/dissertation.

_____

Christopher Rhodes

April 20, 2015

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my committee chair, Dr. Bernard Chen, who has been helping me grow and advance since my undergraduate years. Without his guidance and persistence to challenge myself, this thesis would not have been possible.

I would also like to thank my committee members, Dr. Chenyi Hu, Dr. Shengli Sheng, and Dr. Yu Sun, whose classes have helped push me to the state I am in today. The passion presented by these professors, and all others in the Computer Science department, has allowed an excellent environment for learning.

In addition, I would like to thank my parents, who have continually encouraged and supported me throughout all of my educational endeavors.

ABSTRACT

Even with the current state of technology, data growth is increasing so fast that without proper storage and analytical techniques, it is increasingly challenging to process and analyze large datasets. This applies to knowledge bases from all fields, but for the purpose of this paper, we will be discussing specifically the area of professional wine reviews in a new area we call Wine Informatics. In this area, we gathered over one thousand professional wine tasting reviews and manually extracted key attributes that we felt defined a wine. These attributes were categorized in three major ways: savory flavor attributes, physical characteristics, and overall subjective descriptors. The extraction process led to the creation of what we call a computational wine wheel, which is a wine attribute dictionary consisting of 899 categorized and normalized wine attributes, as well as a weight system to define a level of importance. We applied Hierarchical Clustering, BiMax Biclustering, and a proposed TriMax Triclustering algorithm onto various wine review datasets formed around the computational wine wheel. We found that all three clustering methods produced promising and cohesive results that can be used in various aspects of the wine industry, such as defined palate grouping and wine searching.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER 1: INTRODUCTION

There is an intrinsic notion that the computational power of today is essentially limitless, especially when we realize that today's cell phones have more computational power than all of NASA had when it sent two astronauts to the moon in 1969 [1]. We can only imagine what future computational power will be like given said power is supposed to double every eighteen months according to Moore's Law. Even with contemporary capabilities though, it would seem that we could process anything imaginable. However, with more computational power comes the ability to actually generate new and vastly-growing data every single day. So much data in fact that it is estimated we will have generated 40 zettabytes of data by the year 2020 [2]. With ever-growing sizes in raw data, we have problems not only parsing the data itself, but pulling out meaningful information from it as well. The latter part of that statement is the basis behind a generalized concept of the Data Science field taking over nearly every industry. At its core, data science is about extracting and being able to successfully apply meaningful knowledge from large datasets. However, the process of understanding and retrieving the knowledge cannot stem from a single point of investigation. That means that we cannot simply only perform classical data mining techniques on a dataset and expect the results to make sense. We also cannot have mathematicians do basic statistical analysis on the data, as the data could be dirty with respect to its knowledge domain. Lastly, we cannot just have an expert in the field attempt to look over the data, as it could be much too large for any single person to investigate. In our opinion, data science can be described accurately using the definition created by Drew Conway, an expert on large-scale methods for social and behavioral problems [3]. According to Conway, data science is the combination of programming and hacking skills, math and statistics knowledge, and substantial expertise in the knowledge domain being evaluated. It is very important to remember that all three of these sections need to be explored because if any one section is missing, the result is an inability to fully extrapolate knowledge in the data at hand. Conway has visually detailed his perspective of data science in the venn diagram shown in the below figure.

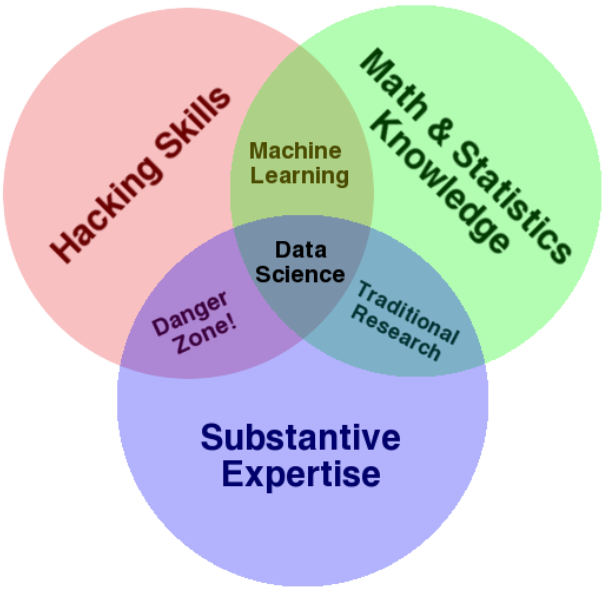

FIGURE 1.1. Data Science Venn Diagram

Exploring the diagram, we first see that the cross section of "Hacking Skills" and "Substantive Expertise" as the "Danger Zone!" This is because the individual knows how to structure data and form into ways easily digested and processed. However, the knowledge is not there to actually understand what the processed data is telling. An example of a linear regression model being applied to the data was used. In the "Danger Zone!" of the diagram for this example, the user might know enough to apply such a model, yet the results would be meaningless without the math and statistics knowledge to back it up. We can see there are two other cross sections, each where an overall topic is left out. This means a user might not have the "Hacking Skills" to produce certain algorithms or models, or the user might not have the "Substantive Expertise" in a topic to understand what the model or statistical output means according to the data domain itself.

For this paper, our hacking skills will explore the study of data mining, or data analysis of large datasets. This is really just a term to describe the actual process of exploring data and finding patterns or relationships within it. Examples of popular data mining techniques include clustering, classification, and association rules [4]. Clustering can generally be thought of as an unsupervised learning method that processes groups of objects into clusters that have high similarity between

one another. Clustering also operates under the assumption that objects in one cluster are dissimilar to objects in a different cluster. Overall most clustering algorithms are fairly open ended and require inputs from the user to try to guess the best way to join, parse, and evaluate the quality of produced clusters. Examples of popular clustering techniques include Hierarchical Clustering [5], Co-Clustering [6], and Density Based Scanning [7]. Classification on the other hand is generally a supervised learning approach. This means the user has a set of observations and attributes, such that each observation has one or more labels attached to it. The label is a classification designation that groups an observation into a specific category. This initial set of data can then be thought of as a training dataset. Using statistics on the training dataset, the user is able to form models, such as Decision Trees [8] or Support Vector Machines (SVM) [9], which can be applied on new testing observations. Testing observations are simply new observations that have an unknown classification label, but once they are sent through the trained model, a label can then be assigned with hopefully a high degree of confidence. Lastly, there is association rule learning, which is a method of finding implication patterns between items in a transactional database or information repository. A basic example would be finding that a consumer at a supermarket buying milk and eggs is probably also going to buy bread. This possible implication pattern is found by finding item set groups in the entire dataset that meet a certain support and confidence. Support is simply the number of transactions that contain both item X and item Y. Confidence is how strong the association between the two items is, and represents how often times item Y appears in a transaction that contains item X. By finding the support and confidence for all item sets, a user can filter out stronger associations by limiting records above a certain threshold for both measures. The most popular Association Rules technique is the Apriori method [10]. All three of these data mining techniques have many different methods for producing results, and when choosing a method to explore, it is up to the data itself and what the user hopes to achieve from that data. For this paper, we will only be focusing on the clustering tract of data mining, with a major focus on Agglomerative Hierarchical Clustering, Biclustering, and a newly proposed method called TriMax

TriClustering. These methodologies will be described in more detail in chapters 3, 4, and 5, respectively.

1.1: Wine Production

Our substantial expert knowledge for this paper will be the domain of wine and its flavor characteristics. Since we are implementing and analyzing the various clustering algorithms ourselves, we have fulfilled the top two portions of the data science venn diagram. To complete the full cross-section, we need to prove we have enough knowledge of the subject to make an accurate analysis of all results. We will be attempting to cluster wines based on data extracted from expert wine tasting reviews. Before we talk about what goes into tasting a wine, a little more background is needed. Wines are primarily made from fermented grapes and have been produced for thousands of years and all over the world. Grapes are favored as yeast is able to more easily convert the natural sugars into carbon dioxide and alcohol, without the need of other additives or catalysts [11]. Grapes also generally contain the right amount of acidity and tannins, which allow wines to maintain good balance and structure [12]. Once grapes begin to ripen, they are picked either by hand or by machine and are taken in for sorting and fermentation. To note a special difference, typically red wine grapes are fermented with their skins and white wine grapes are pressed to separate the juice and skin. This is actually where red wine gets its red hue as the juice extracts color and other properties from the skin itself [13]. At this point it is up to the winemaker to add natural or cultured yeast to help with the fermentation process. Once the fermentation process is complete and the wine has been pressed from any remaining skins or yeast, the wine is then stored in a cool place for anywhere from six months to three years. The wine is usually stored in wood barrels, and the type and size of these barrels can actually have a dramatic impact on how the wine eventually develops. To make sure the wine matures perfectly, the wine needs to receive as little oxygen as possible. Once a wine has been aged appropriately, the wine can then be filtered and finally bottled. Throughout this entire process, even before picking, a wine's future aroma and flavor compounds are seeping into the grapes from various sources. Natural compounds are formed

from the soil type and area of growth. As fermentation begins, chemical reactions are occurring between volatile and non-volatile compounds between the grapes and the yeast, and as the wine ages and matures, these reactions still happen, but at a much slower pace. While not unique to grapes, they have the more individual variety of possible aromas that can develop from these chemical compounds. The actual process entails these compounds combining with sugars to form odorless glycosides, and through the process of hydrolysis, they revert back to an aromatic form [14]. Apart from these naturally occurring processes, there are also external forces during the aging process that can inject other flavor compounds as well. The most notable is *vanillin* which seeps into the wines from the oak barrels they are sometimes stored in, which might give hints of vanilla to the taster. Since wine tasting can have so many unique flavors, the tasting experience is special as tasting a wine is really just smelling all the vaporized aroma compounds. Special cells called olfactory receptors, which are sensitive to different aromas, will send information via the olfactory bulb to the brain on how to interpret each aroma [15]. The variety of aromas in a wine can be staggering, and depending on the knowledge and sensitivity levels between individuals, two people can taste the same wine and report different aromas. The mind is a powerful agent as personal experience and bias of certain aromas can dramatically alter the perception of what an individual is tasting. There is not necessarily a wrong description of an aroma, but an inexperienced wine taster might not have the knowledge depth to accurately depict and describe everything. Consider also that different people have varying sensitivity levels and may not even recognize a certain aroma is even present. The next section will briefly discuss a simple review process and just how little bias it takes to alter individual perceptions.

1.2: Wine Tasting Reviews

Given the knowledge of the wine creation process and how varying aromas are developed within the wine, an actual tasting could then proceed. This process can be very delicate as a wine is examined not only for its tasting quality, but for physical appearance and physiochemical properties as well. A taster will usually evaluate the appearance of the wine, how it smells in the

glass before tasting, the different sensations once tasted, and finally how the wine finishes with its aftertaste. The taster will be looking for how complex the wine is, how much potential it has for aging for drinkability, and if there are any faults present. The experience required can be expansive as any given wine needs to be carefully assessed within comparable wine standards according to its price, region, varietal, and style. Also, if known, the actual wine production techniques will allow the taster to examine further characteristics. Should there be multiple wines being evaluated, there are a couple different types of tastings: vertical and horizontal. In the former, varying vintages, or years produced, are tasted from the same winery to evaluating different ages. The latter testing involves the same vintages from different wineries to help emphasize the differences in styles. However, professional wine tastings are held to a much higher standard are usually done with what is called a blind tasting. This is where the taster is not allowed to see the label of the wine or even the shape of the bottle. Oftentimes, the taster is also not disclosed the actual color of the wine that is consumed. Research has shown how powerful perception and bias is when there exists a strong expectancy based on preconceived notions of any aspect of a wine. A French researcher named Frédéric Brochet performed two experiments to show how vastly the bias can affect a tasting. The first experiment involved a mid-range Bordeaux wine to be split into two different bottles. One bottle was presented as a cheap table wine, and the other as a very high end specialty. The volunteers described the high-end bottle as "woody, complex, and round", while they noted the cheaper bottle as being "short, light, and faulty [16]." Brochet's second experiment involved a white wine being presented and eventually described as "fresh, dry, honeyed, [and] lively" from students studying wine. That same wine was then dyed red and presented again, but this time the students described the wine as "intense, spicy, supple, [and] deep [17]." The latter description is a usual depiction of red wines and is vastly different from the first inspection of the same white wine. Although some people like to use these results as evidence against wine tastings, Brochet's experiments merely highlight the need for a standardized blind test when performing any professional wine tasting.

To show an example of what might result from a professional blind tasting, below is an example wine tasting review for Wine Spectator's number one wine of 2014.

> *Dow's Vintage Port 2011*
> Powerful, refined and luscious, with a surplus of dark plum, kirsch and cassis flavors that are unctuous and long. Shows plenty of grip, presenting a long, full finish, filled with Asian spice and raspberry tart accents. Rich and chocolaty. One for the ages. Best from 2030 through 2060. –*Kim Marcus* [34][1]

It is important to again note that these attributes are specific to this taster's opinions and evaluation. A different taster might exclude or find differing attributes. However, given a fair amount of tasting experience, the expected differences between two reviewers should be subtle, especially when noting the strongest attributes.

This paper will present a methodology for extracting key attributes from wine reviews like the example shown above. We will detail the formation of a Computational Wine Dictionary, which will serve as a basis for future, automated extraction of attributes from wine reviews. Given the dictionary and a couple datasets of wine reviews, we will explore varying clustering techniques in an attempt to show that it is possible to group similar wines together using only the sensory attributes given in professional wine reviews. We believe our examination and subsequent evaluation of wine sensory information can form the base of new area called Wine Informatics. There is some existing research into evaluating and clustering wines, but its data is purely based on the actual chemistry of the wine, which are primarily numerical categories like alcohol, color intensity, and phenol amounts [18]. This paper aims to show that even based on technically subjective criteria, it is still possible to cluster wines using non-physiochemical data. We will examine our datasets and introduce the computational wine wheel in Chapter 2. Chapters 3 through 5 will discuss using these datasets as inputs to Hierarchical Clustering, Biclustering, and Triclustering, respectively. Finally we will conclude our paper with a summarization of results and give details on future work for this study and Wine Informatics as a whole.

[1] A Wine Spectator membership account may be required to view the tasting note.

CHAPTER 2: DATA

Diving deeper into the area of Wine Informatics can be a daunting task since the idea of looking at non-quantitative attributes of wine requires quite a bit of thought and preprocessing. To make it easier on our research and to form some consistency within the data, we thought it would be best to limit our initial data compilation to a single source. There are thousands of local and global wine reviewers, whether independent blogs or large publications. Some of the major publications include Wine Spectator [20], Wine Enthusiast [35], and Wine Advocate [36]. These three are arguably the most popular and they all use a derivation of the 50-100 point scale developed by Robert Parker, who runs The Wine Advocate publication and has had a tremendous impact on the wine industry. The point scale is used widely and is categorized in TABLE 2.1. The first column defines Parker's original scoring system, which has five 10 point ranges. Wine Spectator's derivation is detailed in the second column, which mostly uses 5 point ranges.

RP SCORE	WS SCORE	DESCRIPTION
96 – 100	95 – 100	Extraordinary/Classic wine
90 – 95	90 – 94	Outstanding; a wine of superior character and style
80 – 89	85 – 89	Very Good; various degrees of finesse
70 – 79	80 – 84	Average; little distinction, yet soundly made
60 – 69	75 – 79	Below Average; noticeable deficiencies
50 – 59	50 – 74	Poor or Undrinkable; not recommended

TABLE 2.1. Popular Point Scale for Wine Reviews [19]

With the 50 point rating system we a generally able to compare a wine across multiple sources. However, for creating our dataset, we decided to go with Wine Spectator only because of ease of review access and the comparable style of reviews even between different reviewers.

2.1: Wine Spectator

To start aggregating wine reviews, we decided to use Wine Spectator, which is a lifestyle magazine that focuses on wine and wine culture [20]. The magazine has been in production since 1976 and each issue can contain more than one thousand wine reviews. Luckily, the company has also since published hundreds of thousands of their reviews directly to their website for subscribers

to view. The reason Wine Spectator is a good fit for us is their strict wine tasting process, as well as the concise nature to their reviews and tasting notes. Wine Spectator prides themselves on evaluating using blind tastings, and sometimes double-blind, the latter meaning the reviewer has absolutely no information at all on the wine in the glass. Typically though, a single-blind methodology is chosen, in which the reviewer is given the vintage, appellation, and grape varietal, but the vineyard, producer, and wine price information is not disclosed. To quote their methodology instructions, "the goal is to arrive at the appropriate balance; enough information to contextualize the wine, but not so much information that "imaginary references" begin to distort judgment [21][2]." Once a tasting has concluded and the reviewer has noted their impressions, a review, or tasting notes, is published. As compared with other big name wine publications, the reviews from Wine Spectator are extremely concise without losing quality information concerning the wine itself. While other reviews might often to try to bring in life anecdotes or superfluous region information, Wine Spectator tends to only specify actual tasting notes. An example review is seen below for the top rated wine for 2014, as also shown in Chapter 1.

> *Dow's Vintage Port 2011* (#1/10 top wines of 2014)
> **Powerful**, **refined** and **luscious**, with a surplus of **dark plum**, **kirsch** and **cassis** flavors that are **unctuous** and **long**. Shows plenty of **grip**, presenting a **long**, **full finish**, filled with **Asian spice** and **raspberry tart** accents. **Rich** and **chocolaty**. One for the ages. Best from 2030 through 2060. *–Kim Marcus*

In this version of the example review, we have bolded what we might consider to be key attributes to the review itself, and these attributes range from actual savory properties, such as "chocolate" and "Asian spice", to subjective properties, such as "powerful" and "refined." Our goal is to extract enough key attributes from these professional wine reviews so that we can form a solid foundation of a wine attribute dictionary for the area of Wine Informatics. While we are planning on mining reviews solely from this review source, we do have a bit of diverseness in that Wine Spectator has many editors that perform tastings, so there is no single point of bias giving out all tasting notes.

[2] A Wine Spectator membership account may be required to view the full letter.

2.2: Original Wine Aroma Wheel

The tasting notes given in a review are very important as they describe the heart and soul of a wine. Even without knowing the producer or varietal, a well-described review can adequately sway a potential consumer into a purchase. Our idea is to build a Savory Wine Dictionary where common, yet important attributes can be stored and referenced as needed. Luckily, this idea was already introduced in 1980 by a sensory chemist and retired professor named Ann C. Nobel [22]. She created what she called the Wine Aroma Wheel and a representation of it can be seen below.

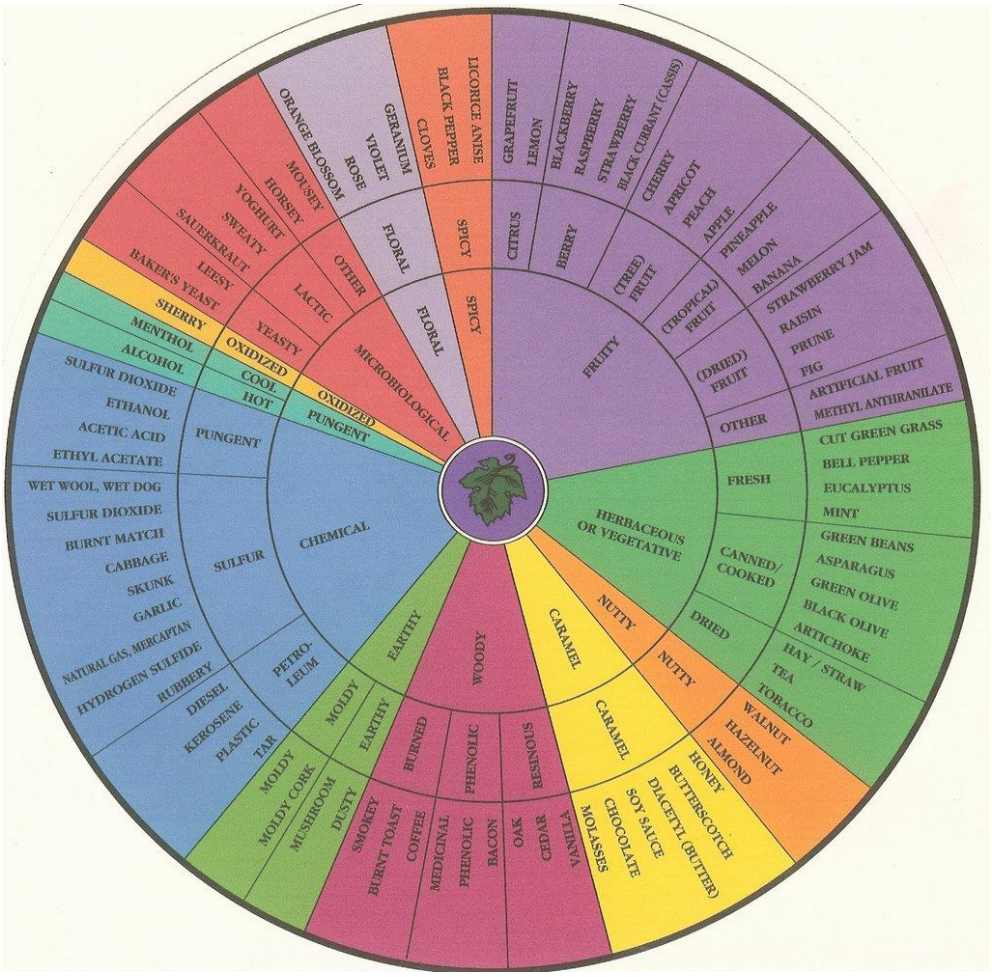

FIGURE 2.1. Wine Aroma Wheel

The wheel is composed of twelve categories of overall wine aromas someone might experience when tasting a wine. The idea for the wheel was to help people describe tastes or aromas that might be hard to formulate without having given previous impressions. While Nobel's wine aroma wheel

is a good start, we did not believe it would be enough for us in its original form. Without being overly specific there are times when certain distinct flavor attributes are not unique enough to encapsulate all flavors. An example of this would be the FRUITY -> (TREE) FRUIT -> APPLE attribute. As we will show later with our expansion attributes, things like APPLE and GREEN APPLE are unique enough to warrant a distinction in the (TREE) FRUIT subcategory. However, we cannot add flavors arbitrarily as it might not reflect actual flavors and aromas found in real world wines. We need a wine aroma wheel that, while expansive, is accurate and can be used easily by ourselves and others for automated processing of raw wine reviews. As we will discuss in the next sections, there quickly becomes a point where manually extracting flavor properties from wines becomes incredibly difficult and time consuming. Conversely, without a properly checked base of initial, accurate descriptions and properties, any automated attempt could be futile and possibly miss something important.

2.3: Computational Wine Aroma Wheel for 100 Wines

By expanding the wine aroma wheel, we hope to form what we call a computational wine wheel. To form this dataset, we initially extracted all reviews from Wine Spectator's Top 100 Wines of 2011 [37]. The idea here was that all of Wine Spectator's Top 100 lists contain only wines that have a review score of 90 or higher. By only picking those wines considered outstanding or classic, we will be gathering savory attributes that most wines should have and descriptive attributes that all wines hope to achieve during a tasting. The extraction process for these reviews was purely manual as we handpicked key attributes as well as noted secondary information about the wine. In total we gathered the following information: name, vintage, review, varietal, regional information, and price. However, it is worth noting that for our processing purposes the review is the single most important piece of information for a wine. For the review and attributes themselves, there were a few types of attributes we are concerned with. Besides actual biological flavor attributes, we also tried to include anything corresponding to a wine's physical structure, including things like acidity, body, structure, weight, tannins, and finish. These are properties of wine that a

taster will physically taste or feel, such as how acidic the wine tastes or how well the wine coats the tongue. Lastly, we also decided to keep generic, subjective terminology that may or may not be the same between two different tasters. For example, one taster may find a wine "vivid" and "beautiful" while another taster may make no mention. Originally we thought about generalizing words into their derived connotations, such as "grand" rating higher than a word like "fine." However, since we are extracting the top rated wines, the level connotative differences would be subtle and hard to differentiate. By that, we mean the tiny difference in positive connotations between two words may not be worthwhile in investigating. Instead, we opted to keep as many subjective descriptors that we could find, as we found that many reviews still share many positive descriptions and generally, different tasters are generally referring to the same aspects when using a word such as "vivid." We do not believe the context would be too different between separate tasters using the same word to describe a wine.

Showing the previous example review again, we want to highlight how we would extract the review's key attributes into the three mentioned categories: savory, body, and descriptive.

> *Dow's Vintage Port 2011* (#1/10 top wines of 2014)
> **Powerful**, **refined** and **luscious**, with a surplus of **dark plum**, **kirsch** and **cassis** flavors that are **unctuous** and **long**. Shows plenty of **grip**, presenting a **long**, **full finish**, filled with **Asian spice** and **raspberry tart** accents. **Rich** and **chocolaty**. One for the ages. Best from 2030 through 2060. –*Kim Marcus*

For this review, **red** words indicate specific flavors and aromas that could possibly be found on Nobel's wine aroma wheel. **Orange** words indicate traits corresponding to the physical wine itself like its body and finish. That is, how the wine feels physically to a taster. Lastly, **blue** words indicate subjective adjectives used by the taste to describe the overall wine. Should a word or phrase not exist in the original wine aroma wheel, we would add it. Also, if a word or phrase does not fit into any previous categories or subcategories, we would create one for it. This methodology generally worked well, but one thing we found while extracting properties from the 100 wines from 2011 was that there was slight contextual overlap between different reviews. That is, there would

be two different reviews using slightly different words to express the same tasting notes. A simple example would be one review using the word "distinctive" and another review saying a wine was "very distinct." The human thought process would naturally assume these two differences are the same thing, but computationally, we might miss the connection. For this reason, we added a fourth level to the wine aroma wheel that we like to call a normalized attribute name. This portion of the wheel would represent a base, or normalized, word to encompass a variety of word usages. This is extremely important not only for differences in word tense or suffixes, but especially the verbiage used when describing biological elements like fruits and their descriptions. An example of this would be the taster either using the phrase "lemon peel" or "lemon rind", both of which refer to the outer layer of the lemon. Another example would be being too verbose when describing a specific flavor, such as "cocoa", "cocoa powder", and "cocoa-filled". Certain phrases like this generally refer to the same thing. However, there are times when the phrases make them distinct enough to become unique attributes. A good example of this would be "blueberry", "blueberry fig", and "blueberry jam." Even though all three are components of the same fruit, the taste and consistency of each item convey different connotations and perceptions.

In TABLE 2.2, we show a count summarization of the computational wine wheel as formed from the top 100 wines for 2011. We manually extracted 547 total specific attributes across 12 specific categories. After normalizing all attributes if possible, we were able to reduce the total attributes from 547 to 376 unique attributes. Even with the reduced size, most people would consider this number of attributes to be a very high number of possible dimensions. With that in mind, appropriate processing methods will need to be chosen in order to handle this situation.

CATEGORY	SUBCATEGORY	COUNT	DISTINCT NORMALIZED COUNT
CARAMEL	CARAMEL	9	7
CHEMICAL	PETROLEUM	3	1
EARTHY	EARTHY	18	2
FLORAL	FLORAL	15	15
FRUITY	BERRY	18	15
FRUITY	CITRUS	11	11
FRUITY	DRIED FRUIT	21	21
FRUITY	FRUIT	5	4
FRUITY	OTHER	7	7
FRUITY	TREE FRUIT	12	9
FRUITY	TROPICAL FRUIT	15	11
HERBS/VEGETABLES	CANNED/COOKED	7	7
HERBS/VEGETABLES	DRIED	6	6
HERBS/VEGETABLES	FRESH	15	12
MEAT	MEAT	1	1
MICROBIOLOGICAL	LACTIC	3	2
MICROBIOLOGICAL	YEASTY	3	3
NUTTY	NUTTY	3	3
OVERALL	ACIDITY	14	3
OVERALL	BODY	17	10
OVERALL	FINISH	50	6
OVERALL	FLAVOR/DESCRIPTORS	217	179
OVERALL	STRUCTURE	9	2
OVERALL	TANNINS	24	3
SPICY	SPICE	26	21
WOODY	BURNED	11	8
WOODY	PHENOLIC	1	1
WOODY	RESINOUS	6	6
TOTAL COUNTS		**547**	**376**

TABLE 2.2. Computational Wine Wheel for 100 Wines Aggregate Counts

For our computational wine wheel, we added a couple new overall categories that the original wine aroma wheel did not have, such as MEAT. The most important category added is the OVERALL category, which represents the set of subcategories describing the body of the wine and the subjective descriptors. The fourth column represents the new level discussed, as it shows the distinct number of unique attributes for any given subcategory. As we expected, the FLAVOR/DESCRIPTORS subcategory represents the most possible attributes, and this actually

becomes a small problem when trying to find accurate similarity between two wine reviews. We questioned if two wines are closely related if they intersect fully on savory flavor attributes, but nearly none with subjective word descriptors. We believe they should relate more closely in that case, but as the Hierarchical Clustering section will discuss in more detail, we need to apply a weight to every normalized attribute to shift the similarity between reviews. The table below shows the numerical weight value assigned to the different categories.

Attribute Type	**Weight**
Biological Flavors and Aromas	3
Physical Wine Characteristics	2
Important Subjective Descriptors	2
All Other Subjective Descriptors	1

TABLE 2.3. Weight Values for the Computational Wine Wheel

Actual biological flavors like “APPLE”, “CHERRY”, and “SPICE” are given the most weight as we felt that these aromas might be the most important when comparing two wines for the basic features. Physical wine characteristics, such as “LONG FINISH” and “DENSE,” are given a middle tier weight as they are certainly important, but these attributes might vary more wildly. Lastly, subjective words such as “BEAUTIFUL” and “VIVID” are given the least weight since they are the most common types of attributes and it does not accurately compare two wines on their own. It is also important to mention the third row of the table labeled “Important Subjective Descriptors,” as we felt there were some subjective word choices that were more important than others. They were few and far between, but some examples include “POWER”, “RICH”, and “SAVORY”. These are words or phrases that cannot be used as stand-alone descriptors of the overall wine, but seemingly transcend their meaning into other categories and possibly are also intended to include the context of the physical tasting properties. We felt two wines sharing these special adjectives might actually reflect more similarity. The attributes that fall into this category were few and chosen at our discretion. After forming this initial computational wine wheel, we show what our 100 wine dataset looks like before and after applying the weights.

WINE NAME	APPLE	TANNINS_HIGH	VIVID	…	*Attribute N*
Wine 1	1	0	0	…	…
Wine 2	0	1	1	…	…
Wine 3	1	0	1	…	…
…	…	…	…	…	…
Wine M	…	…	…	…	…

TABLE 2.4. Example Dataset with Non-Weighted Values

TABLE 2.4 shows our multi-dimensional dataset as essentially a binary set of attributes for every wine. A wine either has an attribute or it does not. Since there are 376 possible normalized attributes though, it is possible the review starts to lean more heavily towards the number of subjective descriptors than actual flavors and aromas which we consider to be the most important. When dealing with comparing sets, this could skew any similarity calculations. To combat this problem we apply the weights mentioned in TABLE 2.3 to form a new dataset that now looks like the one presented in TABLE 2.5.

WINE NAME	APPLE	TANNINS_HIGH	VIVID	…	*Attribute N*
Wine 1	**3**	0	0	…	…
Wine 2	0	**2**	**1**	…	…
Wine 3	**3**	0	**1**	…	…
…	…	…	…	…	…
Wine M	…	…	…	…	…

TABLE 2.5. Example Dataset with Weighted Values

We still have binary sets of attributes per wine, but any distance methodology using sets can be altered to take the weighting into account. We will explain our methodology for detecting and handling weighted similarity in the Hierarchical Clustering chapter.

2.4: Computational Wine Wheel for 999 Wines

After forming the computational wine wheel on the 2011 wines, we figured that 100 wines might not be a large enough sample size, so we performed the same attribute extraction on 999 additional wine reviews from Wine Spectator. These wines were the Top 100 wines from the years 2003 to 2010, and 2012 to 2013 [37]. One wine review was not able to be retrieved. We used the computational wine wheel for 2011's data as a basis to help filter out previously known attributes,

and then had our group of students manually examine any new attributes we could find. The below table represents the final results of the 999 wine reviews. We ended up with 13 distinct categories and a total of 31 distinct subcategories. From all wines mined, we found a total of 1,748 specific wine attributes, and of those attributes we were able to finalize 889 distinct normalized attributes.

CATEGORY	SUBCATEGORY	COUNT	DISTINCT NORMALIZED COUNT
CARAMEL	CARAMEL	67	37
CHEMICAL	PETROLEUM	5	2
CHEMICAL	SULFUR	2	2
EARTHY	EARTHY	68	30
FLORAL	FLORAL	60	35
FRUITY	BERRY	54	26
FRUITY	CITRUS	37	22
FRUITY	DRIED FRUIT	61	54
FRUITY	FRUIT	24	8
FRUITY	OTHER	8	8
FRUITY	TREE FRUIT	40	31
FRUITY	TROPICAL FRUIT	47	25
HERBS/VEGETABLES	CANNED/COOKED	10	9
HERBS/VEGETABLES	DRIED	24	20
HERBS/VEGETABLES	FRESH	38	26
MEAT	MEAT	25	13
MICROBIOLOGICAL	LACTIC	11	3
MICROBIOLOGICAL	OTHER	9	3
MICROBIOLOGICAL	YEASTY	4	4
NUTTY	NUTTY	21	15
OVERALL	ACIDITY	33	3
OVERALL	BODY	43	22
OVERALL	FINISH	175	5
OVERALL	FLAVOR/DESCRIPTORS	611	404
OVERALL	STRUCTURE	38	2
OVERALL	TANNINS	79	4
PUNGENT	HOT	2	2
SPICY	SPICE	83	39
WOODY	BURNED	43	25
WOODY	PHENOLIC	2	1
WOODY	RESINOUS	24	9
TOTAL COUNTS		**1748**	**889**

TABLE 2.6. Computational Wine Wheel for 999 Wines Aggregate Counts

Compared to the 2011 counts in TABLE 2.2, this data shows large increases in almost every category and subcategory. This is important to highlight just how important it was to update our original dataset. When doing a direct comparison of both computational wine wheels, we find that there are 992 distinct normalized attributes across both. Of those 992, 103 attributes are unique to the 100-wine dataset, 616 attributes are unique to the 999-wine dataset, and 273 attributes are shared by both. Between the three subsets of attributes, we can analyze the impact by adding in the additional 10 years of wine data to on overall wine attribute dictionary. TABLE 2.7 shows the counts by weight for each of the three subsets between the two data sets.

Weight	100 wines – UNIQUE	BOTH – SHARED	999 wines – UNIQUE
3	43	126	319
2	4	27	14
1	56	120	283
TOTAL	**103**	**273**	**616**

TABLE 2.7. Differences between 100 and 999 Wine Computational Wine Wheels

The most important aspect is that we only found 43 unique highly-weighted attributes in the 100-wine dataset, compared to the 319 unique highly-weighted attributes found in the 999-wine dataset. On average, that means the additional 10 years' worth of wines added almost 32 new, unique attributes. That is a strong indication that a 100 sample wine size is not enough to deliver accurate results should a review be automatically matched to that version of the computational wine wheel, as many key attributes would most likely be missed. The same applies to the lowest weighted attributes, which saw a comparison of 56 versus 283. For biological taste indicators and subjective descriptions, we can only assume that continuing to add wines would grow these sections. However, we noticed that the original 100 wines were actually able to encompass a majority of the mid-weighted attributes, which mostly contain descriptions of the physical body of the wine. This is because the few key components of a wines body, such as weight and finish, are concepts that do not change over time, as it would take a radical change of what wine is to alter these subcategories.

As we expanded our computational wine wheel, we figured we also needed a way to decide if the cluster results provided any significant quality. To do this, we decided against evaluating against the sensory attributes themselves, and instead decided to bring in extra data not contained within the wine review. We call this extra information non-savory attributes and it consists of an individual wine's type, varietal, country, and world. Type indicates generally if a wine is considered red, white, or blended, which means multi-varietal. Varietal is the specific grape type(s) used in the wine. We also pull information on the country of origin, which also strongly correlated to the category of new or old world. The world designation of a wine is just a general categorization of wines based on whether or not they were produced in traditional wine making countries or not. For example, most European countries are considered Old World, whereas the United States is generally considered New World. For all 999 wines in this dataset, we retrieved values for all 4 non-savory attributes, and have listed their names and percentages in the four tables below.

Varietal (Grape)	Count	Percent
BLEND (RED)	277	27.73%
PINOT NOIR	87	8.71%
CHARDONNAY	72	7.21%
CABERNET SAUVIGNON	67	6.71%
SYRAH	58	5.81%
SHIRAZ	53	5.31%
RIESLING	45	4.50%
SANGIOVESE	43	4.30%
SAUVIGNON BLANC	33	3.30%
MALBEC	31	3.10%
NEBBIOLO	30	3%
TEMPRANILLO	23	2.30%
BLEND (SPARKLING)	22	2.20%
BLEND (DESSERT)	21	2.10%
ZINFANDEL	20	2%
MERLOT	17	1.70%
BLEND (WHITE)	12	1.20%
Only showing top 17 results		

TABLE 2.8. Percentage of Wine Varietals from 999 Wines

Country	Count	Percent
USA	296	29.63%
FRANCE	224	22.42%
ITALY	145	14.51%
AUSTRALIA	79	7.91%
SPAIN	70	7.01%
ARGENTINA	32	3.20%
PORTUGAL	29	2.90%
CHILE	29	2.90%
NEW ZEALAND	25	2.50%
GERMANY	25	2.50%
SOUTH AFRICA	20	2%
AUSTRIA	13	1.30%
HUNGARY	5	0.50%
GREECE	5	0.50%
CANADA	1	0.10%
ISRAEL	1	0.10%

TABLE 2.9. Percentage of Wine Country Origins from 999 Wines

Type	Count	Percent
RED	737	73.77%
WHITE	214	21.42%
SPARKLING	24	2.40%
DESSERT	24	2.40%

TABLE 2.10. Percentage of Wine Types from 999 Wines

World	Count	Percent
OLD	517	51.75%
NEW	482	48.25%

TABLE 2.11. Percentage of Wine World Category from 999 Wines

2.5: Computational Wine Wheel over Time

Our third and final dataset encompasses 50 Cabernet Sauvignon wines from the Napa Valley region in California. For every wine in this set, we retrieved its review for every year from 2006 to 2010. The wines were picked out in a first come, first serve order from the Wine Spectator

repository as long as they met the above criteria. Because of this, there are some wines that share the same producer, but each wine has a distinct designation and is technically a different wine production. For this dataset it is best to imagine it as a three dimensional cube of reviews, where the height, width, and depth are the wine name, attributes, and vintage, respectively. This dataset is special as there was nothing manual about attribute extraction. We used the computational wine wheel for 999 wines and scripted the output of only matched attributes. The result of this was 50 wines with 259 attributes across 5 years. There are actually two purposes to this dataset. One reason is to attempt to cluster a dataset that was matched automatically to the computational wine wheel. The second reason is to move away from Hierarchical Clustering and to attempt a couple algorithms for subspace clustering: BiClustering and TriClustering. We will discuss these further in Chapter 4 and Chapter 5, respectively.

CHAPTER 3: HIERARHICAL CLUSTERING

This chapter will give an overview on what clustering is and how we plan to use it in this paper. We will discuss the basics of clustering, as well as an in depth look at agglomerative hierarchical clustering. This includes clustering properties such as distance measurements and linkage types. Lastly, we will apply our hierarchical clustering knowledge on our Wine datasets discussed in Chapter 2.3 and 2.4.

3.1: Clustering Introduction

Clustering is generally considered an unsupervised learning and analysis tool. It is an open-ended process that is open to many different interpretations and techniques. Generally though, clustering is mostly thought of as a way of grouping objects or observations into intra-similar clusters. A generated cluster is oftentimes considered dissimilar to all other clusters formed. A given clustering algorithm is generally characterized by the model it tries to form. For example, there are connectivity models that build clusters based on distance connectivity. Clustering models like this generally can results in any number of clusters. Centroid models, such as K-Means, generally try to fit data into a predefined number of clusters via a mean vector. Distribution models create clusters based on statistical distributions of the data itself. Density models form clusters by finding dense regions of data. This is especially useful if dense data form irregular shapes or patterns. Subspace models, such as BiClustering, are special in that they create clusters using both observations and relevant attributes at the same time. We will discuss more on BiClustering in Chapter 4. Apart from specific clustering models, we can also characterize how an observation is defined to a cluster. For example, we can ride under the assumption that each observation either belongs to a cluster, or it does not, which is called hard clustering. Soft clustering, or fuzzy clustering, otherwise allows an observation to partially exist in any cluster to a certain degree of possibility. Clustering can also get into stricter rules that can specify that an observation can belong to one, and only one, cluster. With so much diversity in the algorithm details, it is not hard to guess that the results might be even more complicated. Generally there is no such thing as training data

with clustering, so the results are unpredictable and there is no one way to analyze the output clusters. It is up to the user to fully understand the data that is being clustered and to reprocess as needed to make sure the desired clustering algorithm has valid information. It is also important to make sure the distance or similarity measurements are chosen wisely as this can wildly affect the outcome. The following sections will deal with diving in depth with hierarchical clustering, and how we chose to apply it to our computational wine wheels.

3.2: Agglomerative Hierarchical Clustering

For this chapter, we have specifically chosen to work with agglomerative hierarchical clustering, which is a bottom-up, connectivity-based clustering approach. To help explain how agglomerative hierarchical clustering works, we have detailed the algorithm in FIGURE 3.1 [23]. The figure features the algorithm's basic pseudocode with an explanation following.

Agglomerative Hierarchical Clustering Algorithm

$AgglomerativeClustering(D = \{x_i\}_{i=1}^{n}, k)$:

1 $C = \{C_i = \{x_i\} \mid x_i \in D\}$
2 $\Delta = \{\delta(x_i, x_j) : x_i, x_j \in D\}$
3 **while** $|C| > k$ **do**
4 $\quad$ *Find the closest pair of clusters* $C_i, C_j \in C$
5 $\quad$ $C_{ij} = C_i \cup C_j$
6 $\quad$ $C = \{C - C_i - C_j\} \cup C_{ij}$
7 $\quad$ *Update distance matrix* Δ *to reflect new clustering*
8 $C \rightarrow$ *dendrogram analysis*

FIGURE 3.1. Agglomerative Hierarchical Clustering Pseudocode

To preface the algorithm, it should be noted that agglomerative clustering starts with all observations in the data being their own initial cluster. We start with a dataset *D*, which consists of a set of *n* number of observations. The second argument can also be an optional *k*, which is simply the threshold number of clusters to reach before the algorithm terminates. Typically, an agglomerative clustering algorithm will keep on clustering until there is only a single cluster left, so *k* is usually set to 1. The first step is to create a set of initial clusters *C* that corresponds to all

initial observations in our dataset. The line of thinking is that every initial observation starts out as its own cluster. We now need to create a distance matrix Δ. This matrix is an n x n table representing the distance of all clusters against each other. The distance measurement chosen depends on the data at hand, but typically a very simple approach would be to use Euclidean Distance. It should also be noted that the entire $n \; x \; n$ matrix does not need to be calculated as the values of the upper right triangle are going to mirror the values in the lower left triangle. The diagonal will consist of zero-distance values and should be ignored as it represents clusters against themselves. With the set of current clusters and the corresponding distance matrix, the main loop of the algorithm can begin. While the size of C, or the total number of current clusters, is greater than the threshold limit assigned to k, we first find the minimum value in our distance matrix. This value is the distance value of the two most-similar, current clusters. We then form a new cluster C_{ij} and set its two-cluster set of child nodes to point to our most similar clusters, C_i and C_j. We then need to update our cluster set C by removing C_i and C_j from the set, and pushing in the new parent cluster, C_{ij}. Lastly, since our cluster set C has changed, the distance matrix Δ will need to be recalculated so all remaining clusters can get a distance value to the newly introduced cluster. This cycle is repeated until the size of the cluster set reaches the threshold limit and at that point, the algorithm will terminate. Typically, the best way to represent a cluster is an object with a left and right child object of the same type. What results for classical hierarchical clustering is a binary tree where all initial observations are represented by the leaf nodes, and every node above the leaves are the clustering operations performed in the *while* loop in FIGURE 3.1. Also, the binary tree produced is special because the length of a cluster's stem represents the distance it takes to represent all children belonging to that cluster. This allows someone to get an accurate, visual interpretation of the relative similarity between all data set observations. FIGURE 3.2 shows a basic example of a small dataset in its graphical form (right) as well as the resulting binary tree, or dendrogram (left) [24].

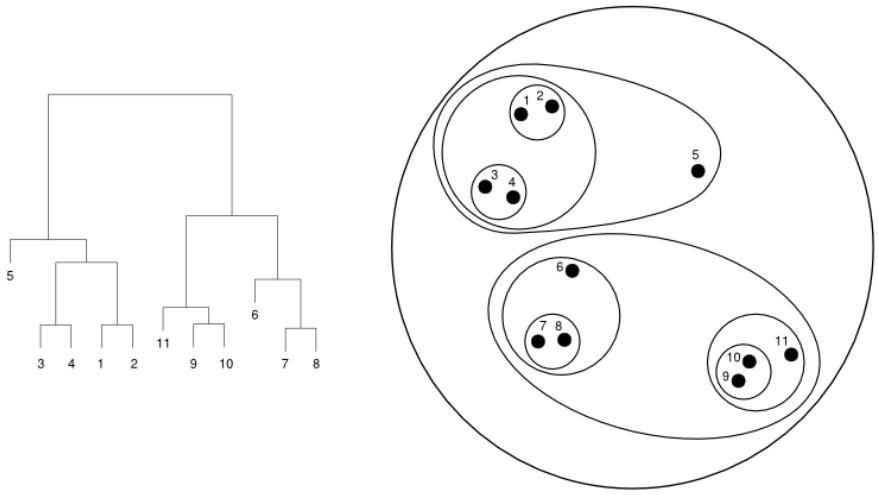

FIGURE 3.2. Example Dendrogram and Data Set for Hierarchical Clustering

For example, we can take a look at the two clusters ((9,10),11) and ((7,8),6). Both images show how observation 11 is much closer to cluster (9,10) than observation 6 is to cluster (7,8). A user could then infer that that cluster ((9,10),11) is a much more cohesive structure. The hierarchical algorithm itself is relatively simple, but there are two major components that should be taken into a lot of consideration before blindly clustering data. Those components are the distance measurement and the clustering linkage type, which we will discuss in the following two sections.

3.3: Distance Measurement

The distance value between two clusters, which can also be thought of as a similarity value, can generally be thought of as a representation of their location or their properties. A representation of location can be compared to data in a Euclidean space, which has a notion of an average between any two points. However, as datasets increase their number of columns, or dimensions, a growing problem occurs called the Curse of Dimensionality [25]. As the volume increases in a dataset, the points within that set become sparse. For statistical accuracy, this means that more and more data is needed to get accurate results. For our dataset specifically, as the number of dimensions grow, the distance between any two points start to become the same, and because of this reason it might be necessary to find a distance measure that works on sets of properties rather than a location. Examples of such distances include Hamming Distance, Cosign Distance, and Jaccard's Distance.

There is no single best measure for a dataset, but depending on the types of values present and the number of dimensions there are generally choices that are better than others. Once a distance measurement has been chosen, any two single points can then be compared.

When evaluating our Wine Wheel dataset, we noticed it had two prominent features. One was that any dataset we formed would most likely have hundreds of dimensions. The other was that our datasets will be binary in nature. That is, a wine either has an attribute, or it does not. For both of these reasons we decided to stay away from location-derived distance measures. After careful consideration, we chose to use the Jaccard's Coefficient measure, which is a similarity formula for set comparisons [26]. When comparing two sets, the measure can be defined simply as the size of the intersection divided by the size of the union.

$$JaccardsCoef(X,Y) = \frac{|\,X \cap Y\,|}{|\,X \cup Y|} \qquad \begin{array}{l} if\ |\,X \cup Y\,| = 0, then\ JaccardsCoef(X,Y) = 0 \\ else\ 0 \leq JaccardsCoef(X,Y) \leq 1 \end{array}$$

EQUATION 3.1. Jaccard's Coefficient Measurement (Similarity)

The coefficient produces a number between 0 and 1, which can be thought of as a percentage of similarity between two sets. Should two sets contain no points of intersection, the ratio will be zero over the size of the union. Conversely, if two sets contain a complete union, then the ratio will be same for both the intersection and the union, resulting in one hundred percent similarity. A complementary value is the Jaccard's Distance, which can be defined by the following formula.

$$JaccardsDist(X,Y) = 1 - JaccardsCoef(X,Y) = \frac{|\,X \cup Y| - |\,X \cap Y\,|}{|\,X \cup Y|}$$

EQUATION 3.2. Jaccard's Distance Measurement

Simply put, if two sets result in a 0.8 Jaccard's Coefficient similarity value, then the two sets can also be considered 0.2 dissimilar via the Jaccard's Distance measurement. Either measure can be used in clustering, but the user needs to remember to look for the maximum values for Jaccard's Coefficient or the minimum values for Jaccard's Distance.

3.4: Weighted Distance Measurement for Wines

In Chapter 2.3 we proposed turning possible values in our dataset from 0 and 1 to any value in the range 0 through 3. We wanted to specify that certain attributes belonging to a wine were more important than others. To do this we still use the same Jaccard's formulas presented in the previous section. However, we use an alternate, but equivalent definition that will allow us to eventually account for weights. Both the original coefficient and the distance formulas can be redefined as the following [27].

$$Jaccard's\ Coefficient = \frac{P}{P+Q+R} \qquad Jaccard's\ Distance = \frac{Q+R}{P+Q+R}$$

$$P = Number\ of\ variables\ positive\ in\ both\ sets$$
$$Q = Number\ of\ variables\ positive\ in\ Q, but\ not\ R$$
$$R = Number\ of\ variables\ positive\ in\ R, but\ not\ Q$$

EQUATION 3.3. Jaccard's Non-Weighted with Distinct Measurements

These formulas are equivalent to EQUATION 3.1 and 3.2, but programmatically we can now avoid using set notation and allow modification of the specified variables *P*, *Q*, and *R*. Using just values of 1 or 0, we would originally increment these variables as we compared attributes between two wines. Now that we have weighted values, when incrementing the variables, we can increase the value higher for strongly-weighted attributes. This idea is simply faking the set comparison by inflating the values for the intersection and the union size. We are proposing this idea as the overall wine data contains a disproportional amount of non-biological descriptive attributes, or subjective adjectives described during the tastings. While these are nice, we consider actual biological or wine body descriptions as taking priority over the more subjective descriptions. In the example below, we will show how the weights can possibly significantly change the similarity between two wine sets. Suppose we have the following subset of data with four wines and the following four attributes: BLUEBERRY, CHERRY, CHEWY TANNINS, and BEAUTY. We can examine the first wine against the second and third, which alternate the sharing of the CHERRY and BEAUTY attributes.

	BLUEBERRY	CHERRY	CHEWY TANNINS	BEAUTY
Wine1	0	1	1	1
Wine2	0	0	0	1
Wine3	0	1	0	0
Wine4	1	0	0	0

TABLE 3.1. Example Non-Weighted Dataset for Jaccard's Similarity

If we find the Jaccard's Coefficient for Wine1 against Wine2, we get 1/3 since P=1, Q=2, and R=0. The Jaccard's Coefficient for Wine1 against Wine3 is also 1/3 since P=1, Q=2, and R=0. This makes since as the size of the intersection between both comparisons is one attribute, and the size of the union between both comparisons is three total attributes. Wine1 is thusly a third similar to Wine2 and Wine3 as it shares only one out of three attributes between them. However, for our research, these results are little misleading in that we feel the shared attribute of CHERRY between Wine 1 and Wine3 is a much stronger bond than the shared attribute of BEAUTY between Wine1 and Wine3. While BEAUTY describes a wine nicely, different reviewers could use a variety of nice words to describe the wines. Also, since our research in this section is dealing with all Top 100 wines, we would expect nothing less than many words with positive connotations. Therefore, we like to try to compare wines against the characters which hopefully should not be as subjective. Now we can walk through the same example, but with their weighted values as shown in the table below.

	BLUEBERRY	CHERRY	CHEWY TANNINS	BEAUTY
Wine1	0	3	2	1
Wine2	0	0	0	1
Wine3	0	3	0	0

TABLE 3.2. Example Weighted Dataset for Jaccard's Similarity

For the weighted attributes, we slightly change the definition of *P*, *Q*, and *R*.

$$P = Sum\ of\ weight\ values\ of\ variables\ positive\ in\ both\ sets$$
$$Q = Sum\ of\ weight\ values\ of\ variables\ positive\ in\ Q, but\ not\ R$$
$$R = Sum\ of\ weight\ values\ of\ variables\ positive\ in\ R, but\ not\ Q$$

EQUATION 3.4. Jaccard's Weighted with Distinct Measurements

Using the new definition, we can make the same comparisons again to see how they might change. Calculating Wine1 against Wine2 gets us a new coefficient of 1/6 since P=1, Q=5, and R=0. Compared to the unweighted coefficient of 1/3, this makes a lot more sense as Wine1 contains two important attributes that Wine2 does not have. Also, the only shared attribute is a least important attribute. Calculating Wine1 against Wine3 gets us a new coefficient of 1/2 since P=3, Q=3, and R=0. Compared to the unweighted coefficient of 1/3, this also makes sense as while Wine1 and Wine3 only share a single attribute, it is a highly weighted one. The strongly-weight attribute is enough to lean the comparison from 33% to 50%. We believe that by using the weighted Jaccard's measurement, our clustering results will allow us to gain more accurate clusters.

3.5: Cluster Linkage

Clustering quickly presents a problem though in that clusters quickly encapsulate more than a single point, which begs the question of how to compare objects where each represents multiple observations. Multi-observation cluster comparison occurs via a pre-chosen linkage type. FIGURE 3.3 shows the three most common linkage types.

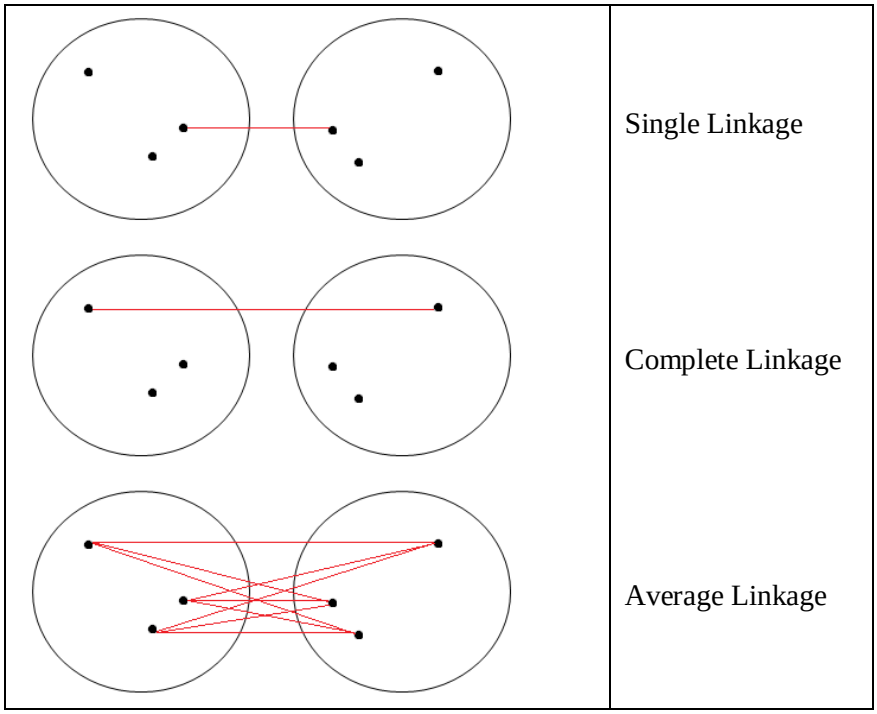

FIGURE 3.3. Clustering Linkage Types

The first linkage type is called Single Linkage, which defines the distance between two clusters as the minimum distance between a point in Cluster X and a point in Cluster Y. The name derives from the observation that if only the minimum distance was found between points in two clusters, then only a single link between the clusters would exist. All other point combinations would fall outside the minimum distance. Single Linkage can be represented by the following formula.

$$\delta(X_i, Y_j) = \min\{\delta(x, y) \mid x \in X_i \, , y \in Y_j\}$$

EQUATION 3.5. Single Link Clustering

The second linkage type is called Complete Linkage, which can be thought of as the opposite of Single Linkage. Complete Linkage defines the distance between two clusters as the maximum distance between a point in Cluster X and a point in Cluster Y. Should a linkage be made using the maximum distance between two points, all other combinations of points would have distances that fall under that value. That is, a complete linkage would be achieved by linking every point combination. Complete Linkage can be represented by the following formula.

$$\delta(X_i, Y_j) = \max\{\delta(x, y) \mid x \in X_i \, , y \in Y_j\}$$

EQUATION 3.6. Complete Link Clustering

The final linkage type presented here is the Average Linkage type. It is represented by the average distance between all possible point combinations between two clusters. Average Linkage can be represented by the following formula.

$$\delta(X_i, Y_j) = \frac{\sum_{x \in X_i} \sum_{y \in Y_j} \delta(x, y)}{|X_i| \cdot |Y_j|}$$

EQUATION 3.7. Average Link Clustering

Another option that is useful is to take a centroid approach to new clusters. When a new cluster is formed, the average attribute values for all observations can be found which converts a multi-observation cluster into a theoretical single point entity. With only a single, averaged location, there is no need for a linkage type as all clusters one be a one-to-one point comparison. However,

with this approach, the location of a cluster can shift as more and more points are clustered. Observations that might be considered noise can have more of a dramatic affect when clustered. Sometimes it might be important to test with all linkage types and examine the results independently.

3.6: Agglomerative Hierarchical Clustering Example

To make sure the reader has an appropriate understanding of the previous sections, we will introduce an example dataset that we will perform agglomerative hierarchical clustering on. This section will show step by step on how to start with many clusters and bring them down to one. This example will also use non-weighted values. An initial dataset is shown below with its starting, empty dendrogram

	cherry	blueberry	Plum	spice
wine1	1	0	1	0
wine2	0	0	1	1
wine3	1	0	1	0
wine4	0	0	1	1
wine5	1	1	1	0

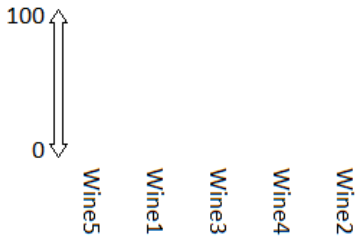

FIGURE 3.4.1. Example Hierarchical Clustering – Initial Dataset

We have 5 example wines with 4 example attributes. The values of 1 indicate that the wine contains the given attribute. The first thing that needs to be done is to form the initial distance matrix, which is just the pairwise distance between all wines. For this example, we used standard Jaccard's Distance since the only possible values are 0 or 1. The initial distance matrix is shown in FIGURE 3.4.2.

	W1	W2	W3	W4	W5
W1	0	0.67	0	0.67	0.33
W2		0	0.67	0	0.75
W3			0	0.67	0.33
W4				0	0.75
W5					0

Current Clusters
Wine2
Wine4
Wine5
(wine1,wine3):0

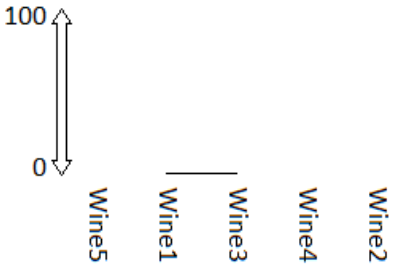

FIGURE 3.4.2. Example Hierarchical Clustering – Step 1

In this example, we see there are actually two different pairs of wines that have the smallest distance: [wine1,wine3] and [wine2,wine4]. In this scenario, we can just pick one arbitrarily. We will turn wine1 and wine3 into the cluster (wine1,wine3) and append its distance to the cluster. Next we will reform the distance matrix by removing the rows for wine1 and wine3, and then adding a new row to represent the total cluster. For future runs, there will be an increasing chance that comparing clusters contain more than a single observation, so for those cases, this example will be using the single linkage clustering method. Below are the remaining clustering steps.

	W2	W4	W5	(W1,W3)
W2	0	0	0.75	0.67
W4		0	0.75	0.67
W5			0	0.33
(W1,W3)				0

Current Clusters
Wine5
(wine1,wine3):0
(wine2,wine4):0

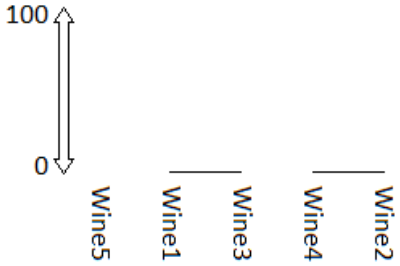

FIGURE 3.4.3. Example Hierarchical Clustering – Step 2

In the FIGURE 3.4.3, we find that wine2 and wine4 have the smallest distance so we form another cluster. At this point we only have 3 total clusters remaining.

	W5	(W1,W3)	(W2,W4)
W5	0	0.33	0.75
(W1,W3)		0	0.67
(W2,W4)			0

Current Clusters
(wine2,wine4):0
(wine5,(wine1,wine3):0):0.33

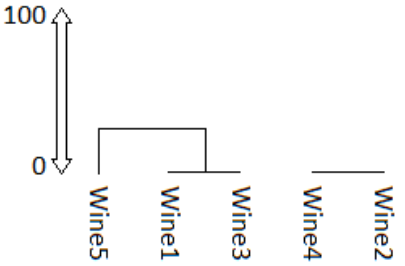

FIGURE 3.4.4. Example Hierarchical Clustering – Step 3

In the FIGURE 3.4.4, we see that wine5 and the cluster (wine1,wine3) have the smallest distance. We cluster both together to form (wine,(wine1,wine3):0):0.33.

	(W2,W4)	(W5,(W1,W3))
(W2,W4)	0	0.67
(W5,(W1,W3))		0

Current Cluster
((wine2,wine4):0,(wine5,(wine1,wine3):0):0.33):0.67

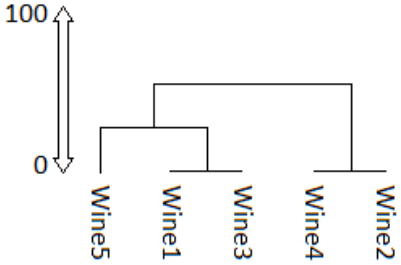

FIGURE 3.4.5. Example Hierarchical Clustering – Step 4

In the FIGURE 3.4.5, we see that there are only two clusters left, so there is really not a need to find the distance, but for clarity we show the value. The final newick string is presented here, which is just a textual representation of the dendrogram shown on the right. We can see that by applying a horizontal cut point around the 0.34 distance mark, we can split the dendrogram into two major clusters: (wine5,(wine1,wine3)) and (wine2,wine4).

3.7: Hierarchical Clustering on 100 Wines

We took the dataset presented in Chapter 2.3, which consists of the Top 100 wines for 2011 according to Wine Spectator. The attributes pulled manually from these wines helped form the initial computational wine wheel, and we wanted to see if just these 100 wines were enough to

allow accurate clustering. For the 100 wines in this experiment, we have 376 total unique attributes. We applied agglomerative hierarchical clustering on the dataset, using weighted Jaccard's Distance as a measurement of dissimilarity as well as single linkage when needing to cluster multi-observation clusters. To initially observe our output, we chose to use the Hierarchical Clustering Explorer (HCE) tool [28]. This tool allowed us to insert our initial distance matrix since Jaccard's Distance was not offered natively. Once the data was processed we are given a movable, horizontal cut point in which to decide which clusters to view on screen. FIGURE 3.5 and 3.6 below show the complete final dendrogram as well as a subset we might consider important [38].

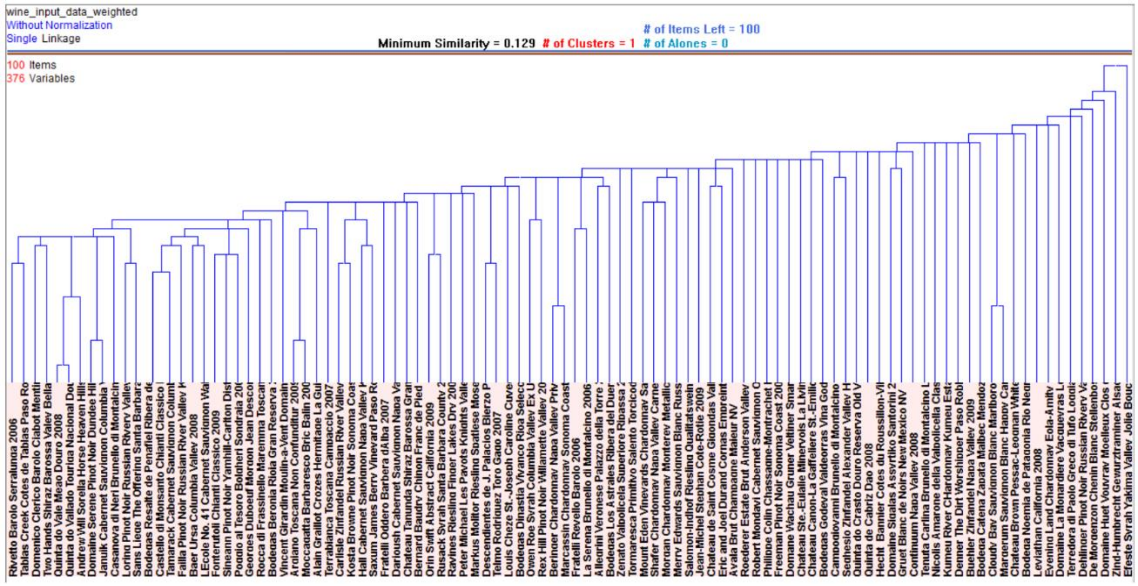

FIGURE 3.5. HCE Results (Full) of 100 Wines of 2011

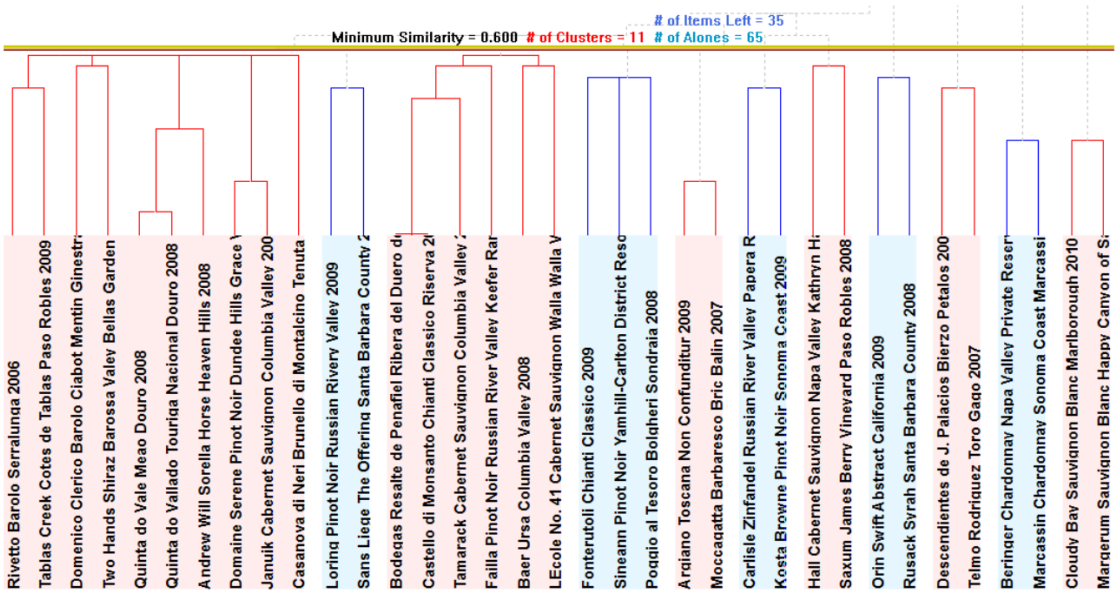

FIGURE 3.6. HCE Results (Subset) of 100 Wines of 2011

The similarity measure in both figures are slightly misleading though and we want the readers to be aware that HCE shows similarity relative to the two closest points in the dataset. Wine13 and Wine14 from the left (Bodegas Resalte and Castello di Monsanto) are the two closets initial wines to be clustered. Since the cut bar only goes from 0% (top) to 100% (bottom) similarity, it might be inferred that two distance between those two wines is 0 and that there similarity is 100%. In reality, their Jaccard's distance is 0.447368. That means if the tree is examined from the bottom to the top, then every cluster's similarity is 100% to 0% relative to the initial minimum Jaccard's distance of 0.447368.

FIGURE 3.6 allows us to see the clusters available when the cut point is moved to only showing wines with at least 60% similarity compared to the closest two wines. Also, for this example we also do not consider any wine left alone to be considered a valid cluster. This is merely to test out the waters with the initial wine dataset, so we chose this point merely because it visually offered a larger number of clusters containing at least two observations. Our goal is to try to prove that the clusters are coherent, and to do that we can examine the shared attributes that make up each of the eleven presented clusters.

Cluster #	# of Wines	Common Attributes found in >50% of a cluster's wines
1	10	Plum(10), Mineral(7), Long Finish(7), Tennins_Medium(6)
2	2	Floral(2), Blackberry(2), Berry(2), Mineral(2), Firm(2)
3	6	Blackberry(6), Long_finish(5), Spice(4)
4	3	Spice(3), Raspberry(3), Tannins_medium(2), Black Cherry(2), Mineral(2), Smooth(2). Harmony(2), Rich(), Long Finish(2)
5	2	Tannins_Medium(2), Acidity_High(2), Violet(2), Black Currant(2)
6	2	Ripe(2), Tannins_High(2), Mineral(2), Complex(2), Well-Structured(2), Rasberry(2)
7	2	Toasty Wood(2), Spice(2), Black Licorice(2), FullBodied(2), Pure(2), Finesse(2), Mineral(2)
8	2	Pepper(2), Spice(2), Complex(2), Full-Bodied(2), Sage(2)
9	2	Dense(2), Herbs(2), Mineral(2), Red(2), Smoke(2)
10	2	Spice(2), Fig(2), Finesse(2), Rich(2), Delicacy(2), Melon(2), Layers(2)
11	2	Peach(2), Mineral(2), Mango(2), Tangerine(2), Smooth(2)

TABLE 3.3. Cluster Attributes for Subset Cluster

TABLE 3.3 shows all 11 clusters in this subsection, as well as the total number of wines and the most common attributes for each cluster. There are various ways to actually use these results and to confirm they make sense. One good thing for this kind of result is that it is good for consumers looking for specific features that they most enjoy. For example, if "Plum", "Mineral", and "Long Finish" are the search criteria, then Cluster #1 could be offered as a selection. Another curious observation is that although the wines in a given structure generally share similar attributes, they can actually have wide price ranges. Cluster #1 actually ranges in price from $30 (Januik Cabernet Sauvignon Columbia Valley 2008 and Tablas Creek Cotes de Tablas Paso Robles 2009) to $125 (Domaine Serene Pinot Noir Dundee Hills Grace Vineyard 2008). If a consumer cannot afford a higher priced wine, then notable substitutions can be made that offer a similar palette.

Other than the sensory attributes and price, we can also look into the geographical attributes of the clusters. We might assume that wines with the same region or varietal to have similar properties, and we can actually see that in these results. For all 11 clusters in this subsection, every cluster is either all red wines or all white wines. Also, many of these clusters share similar country and region information as well. For example, Cluster #2,6,7,8,10 are all from California. Cluster #5 contains only Italian wines. Also, Cluster #9 are specifically from Castilla y Leon, Spain.

Lastly, there are clusters that also seem to have captured the same varietal. For instance, Cluster #10 are all Chardonnay wines and Cluster #11 are all Sauvignon Blanc wines [38].

We are confident in these results and decided to explore more on examining individual clusters by their wine type, varietal, and geographic origin on a larger array of wines. We will discuss these results in the next section.

3.8: Hierarchical Clustering on 999 Wines

In the previous section, we tested a wine dataset built only on 100 wines. We used a third-party application to view our clustering results to see if they made sense and we believe we were able to find meaningful clusters. However, we did not believe that 100 wines was enough to conclusively build our computational wine wheel, so we decided to vastly increase our sample size from 1 year of Top 100 wines to 10 additional years of Top 100 wines. It should be noted though that we were unable to retrieve one of the wine reviews, so we only ended up finding 999 additional wines rather than the full 1000. For this section we built an agglomerative hierarchical clustering application ourselves as we wanted the ability for automated custom clustering analysis. Using the non-savory dataset presented in Chapter 2.4, we want to introduce a model of automated analysis specifically designed for our wine dataset. Below are 3 formulas we have designed to give a score to an individual cluster, give a score to an individual cut point, and to introduce the concept of a quality cluster.

$$ClusterScore_C = \left(\prod_{i=1}^{M} maxPercent(A_i)\right) W_C$$

C = Cluster
M = Number of Non-Savory Attributes
A = Set of Non-Savory Attribute
W_C = Total number of wines in Cluster

EQUATION 3.8. Cluster Score for a Wine Cluster

$$CutPointScore_{CP} = \sum_{i=1}^{N} C_i$$

CP = Cut Point
N = Total Number of Clusters in Cut Point
C = Set of Clusters' Scores

EQUATION 3.9. Cut Point Score for a Wine Dendrogram

$$QualityScore_C = \frac{\left(\sum_{i=1}^{M} maxPercent(A_i)\right)}{M}$$

C = Cluster
M = Number of Non-Savory Attributes
A = Set of Non-Savory Attribute

EQUATION 3.10. Quality Score for a Wine Cluster

The first component is the Cluster Score, which attempts to define a cluster primarily on the most common value in each non-savory attribute. We find the product of the percent distribution for each of those most-common values. We then multiply that result by the total number of wines in the cluster and the result will be the score corresponding to that cluster. We can walk through an example given by the two tables below to show just how the score is derived.

Wine	Type	Varietal	World	Country
W1	Red	Pinot Noir	New	USA
W2	Red	Pinot Noir	New	USA
W3	Red	Pinot Noir	New	USA
W4	Red	Nebbiolo	Old	Italy
MAX	100%	75%	75%	75%

TABLE 3.4. Non-Savory Cluster Example 1

Wine	Type	Varietal	World	Country
W1	White	Chardonnay	New	USA
W2	White	Riesling	New	USA
MAX	100%	50%	100%	100%

TABLE 3.5. Non-Savory Cluster Example 2

In both tables, the final row represents the percent distribution of the most common value in that column. In TABLE 3.4, we can derive a cluster score by finding the product of those percentages as well as multiplying that result against the four wines present. We would get a cluster score of (1.0 * 0.75 * 0.75 * 0.75) * 4 = 1.6875. Using the same method on TABLE 3.5, we would get a cluster score of (1.0 * 0.5 * 1.0 * 1.0) * 2 = 1. There are two different entities adding weight to this score: the consistency of attribute values and the number of wines. In our example, even though TABLE 3.5 is off by a single value in a single attribute, TABLE 3.4's score is higher because it contains additional identical wines that pushes its weight higher. However, if TABLE

3.5's wines were identical, it would have produced a score of 2, which would have been better as TABLE 3.4's fourth wine introduced enough diversity to keep it score lower.

Now that we have introduced a scoring system for individual clusters we want to be able to score a cut point, which is the main way a user might analyze a dendrogram. Since we define a cluster as having at least two wines, the Cut Point Score is simply the summation of all cluster scores at that cut point for clusters with at least two wines.

Lastly we want to introduce a final analysis mechanism in the form of a Quality Cluster Score, which is a user-specified threshold designed to define what a good cluster is. The quality score can be thought of as a percentage between 0% and 100%, and is derived by taking the sum of the max attribute percentages for a cluster and dividing that number by the total number of non-savory attributes. Looking back at TABLE 3.5, we get a quality score of (1.0 + 0.5 + 1.0 + 1.0) / 4 = 0.875, or 87.5%. This means that a user can set a threshold of 75% for good clusters, so the cluster in TABLE 3.5 would pass. The quality score does not have to be used, but it gives alternative measuring capabilities.

We then performed the hierarchical clustering on the 999 savory dataset discussed in Chapter 2.4. Since showing and manually analyzing a dendrogram of 999 observations is difficult, we wanted our system to automatically try and apply the Cut Point Scores and Quality Scores described above. FIGURE 3.7 and FIGURE 3.8 are graphical representations of the entire dendrogram across all used Jaccard's Coefficient percentages. It might be worth noting that the actual algorithm uses Jaccard's Distance to cluster, but the results are showing the Coefficient, which represents the similarity rather than the dissimilarity.

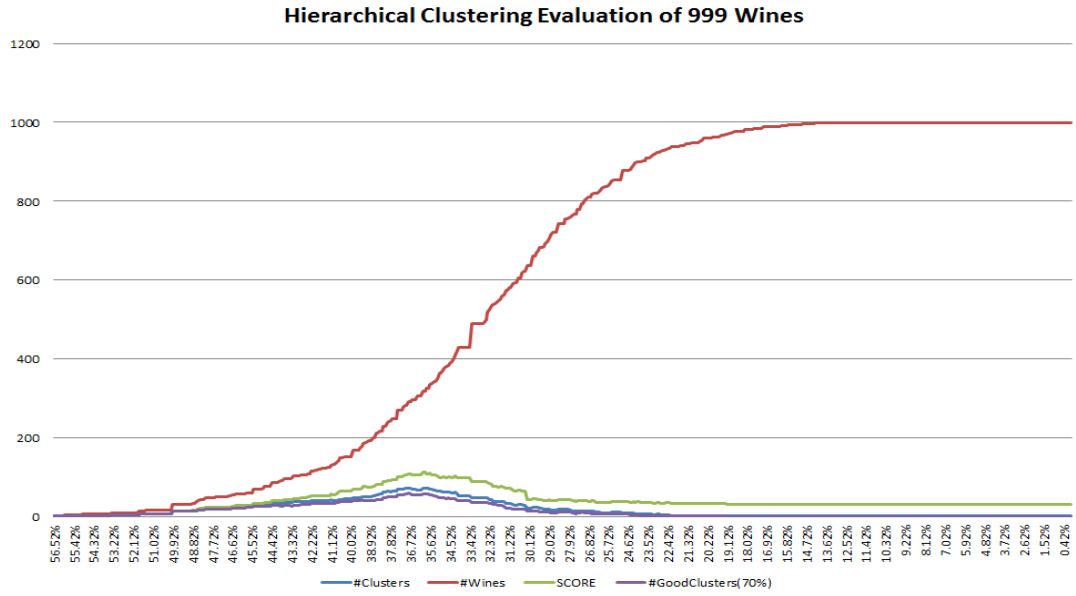

FIGURE 3.7. Scores across Entire Dendrogram of 999 Wines

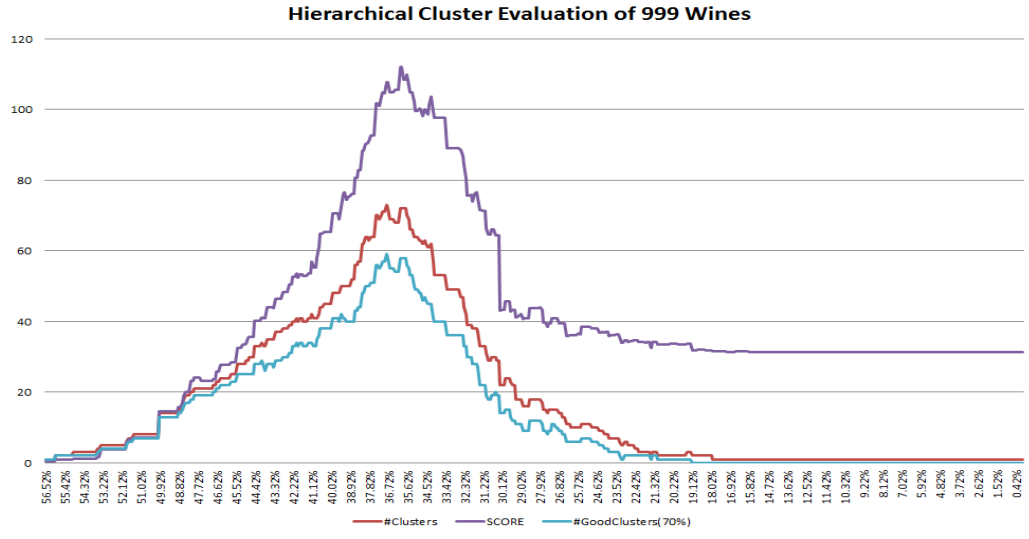

FIGURE 3.8. Scores across Entire Dendrogram of 999 Wines (without Wine Amounts)

Both figures are the same dataset, but FIGURE 3.7 adds in the total number of wines at the cut point to show how much of the data is being clustered at any given time. FIGURE 3.8 was created to exclude this number so better visual quality can be given to the more important scores. The x-axis in both figures represents all possible ranges of similarity percentages between 100% and 0%. As the graph indicates, the closest two wines were about 56.52% similar, and as the similarity percentage decreases, we chart the number of clusters, the similarity cut-point score, and the number of quality clusters at that cutpoint. For the quality score, it should be noted that we applied a quality threshold of 70% for these results. While it may seem to be the case, it should be noted that the cut point with the highest number of clusters does not always reflect the cut point with the highest number of total clusters. Generally, with our data set though, there does seem to be a major correlation between increased cut point score and increasing number of clusters. This is not always the case though as we will examine the cut point with the highest cut point score, and it actually does not contain the most clusters, but just barely. We hope that this occurrence highlights the possibility that quality clusters can sometimes outweigh quantity of clusters, which is what users should be after. The cut point with the best score was valued at 112.063 and was cut at 35.92% of the dendrogram. This means that any of the 72 clusters here will be at least that amount or higher in similarity. This cut point contains just about 300 out of the 999 possible wines. Since hierarchical clustering analysis can be highly subjective, we hope to show that the clusters present at this cut point not only show a high similarity in sensory attributes from the wine wheel, but high similarity in non-sensory attributes as well. This can show high correlation between clustering on sensory evaluations how that alone has the capability of grouping wines are type, varietal (grape type), country of origin, and world type. In FIGURE 3.9, we examine the output of a cluster at this cut point.

Cluster #66 (Similarity = 36.1702)

1) BOOKER SYRAH PASO ROBLES FRACTURE 2010
2) DIERBERG PINOT NOIR SANTA MARIA VALLEY 2005
3) KOSTA BROWNE PINOT NOIR RUSSIAN RIVER VALLEY 2004
4) LORING PINOT NOIR STA. RITA HILLS CLOS PEPE VINEYARD 2005

		Sensory Attributes	#Wines
TYPE	RED => 100%	Earthy	4
VARIETAL	PINOT NOIR => 75%	Rich	4
	SYRAH => 25%	Berry	4
		Blackberry	3
COUNTRY	USA => 100%	Raspberry	3
		Loam	3
WORLD	NEW => 100%		

FIGURE 3.9. Example Red Wine Cluster from Best Cut Point

The cluster contains four wines, which are all new-world, red wines made in the United States. The only subtle difference is one of the wines was made with Syrah grapes versus the majority Pinot Noir. Out of a max cluster score of 4, we find that this cluster receives a score of (1.0 * .75 * 1.0 * 1.0) * 4 = 3.0. This might seem like a large deviation from the highest possible score, but that is only because the cluster contains a smaller number of total wines. For such a small cluster though, the score is nice, but we like it mostly because of its quality score, which is much closer to the max. This cluster receives a quality score of (1.0 + 0.75 + 1.0 + 1.0) / 4 = 0.9375, or 93.75%. While these scores are only representative of the non-savory attributes, we have outlined the majority savory attributes that the wines were actually clustered on to give an idea of what the cluster's sensory attributes contain. We see that these four wines are centered on rich earthy and berry flavors. Reviewers seemed to have specifically pulled out blackberry and raspberry notes, as well as loam, which indicates overall pleasant earthy flavors. This cluster is an excellent example of a quality cluster, but we can actually find better. FIGURE 3.10 describes a perfect cluster, at least when referencing the non-savory scores.

Cluster #11 (Similarity = 36.3636)

1) AUBERT CHARDONNAY SONOMA COAST RITCHIE VINEYARD 2008
2) BYRON CHARDONNAY SANTA MARIA VALLEY 2005
3) STAGLIN CHARDONNAY RUTHERFORD 2002
4) LEWIS CHARDONAY RUSSIAN RIVER VALLEY 2007
5) SBRAGIA FAMILY CHARDONNAY NAPA VALLEY GAMBLE RANCH VINEYARD 2004

		Sensory Attributes	#Wines
		Citrus	5
TYPE	WHITE => 100%	Pear	4
		Smoke	4
VARIETAL	CHARDONNAY => 100%	Toasty Wood	4
		Rich	4
COUNTRY	USA => 100%	Complex	3
		Concentrated	3
WORLD	NEW => 100%	Layer	3
		Fig	3
		Dimension	3

FIGURE 3.10. Example White Wine Cluster from Best Cut Point

All five wines in this cluster are new-world Chardonnays from the United States. Since the max percentage for all non-savory attributes is 100%, then the cluster receives a cluster score of 5 and a quality score of 100%. While this cluster does not contain a differing varietal, this type of cluster allows us to further examine the savory attributes present between the wines. From this example, we are able to make some assumptions that CITRUS, PEAR, SMOKE, TOASTY WOOD are probably more likely to only belong to white wines, and in this case specifically, the Chardonnay varietal. RICH was also described in 80% of the wines in this cluster, but it is good to remember this is lower-weighted attribute. However, even lower weighted savory attributes can have some good cluster descriptive capabilities. In this cluster, we see that a majority of these wines were described as COMPLEX, with LAYER[S] and DIMENSION[S]. A wine consumer might be more interested in a wine from this cluster should he want a chardonnay that has multitudes of overlapping flavors and body style while tasting.

We have shown that based on our computational wine wheel, hierarchical clustering of the dataset results in viable clusters that can have quality groupings in both savory and non-savory attributes. It is good to remember that our dataset has a very large number of dimensions, so the

average number of attributes per wine is small. This means that a majority of the clusters are going to be small, but most likely very similar, such as in FIGURE 3.9 and FIGURE 3.10. Our abilities to find these clusters shows promise, and that the computational wine wheel and review parsing techniques should be refined and continued.

CHAPTER 4: BICLUSTERING

The classical clustering algorithms, such as hierarchical clustering and k-means clustering, are usually very good places to start when attempting to explore data. However, they are flawed in a sense as both algorithms are attempting to detect patterns in observations across all given attributes of a dataset. Sometimes it might be more important to find patterns that consist of a subset of attributes, as is primarily the case with locating patterns within gene expression data. For biological gene data, a subspace clustering mechanism was needed, and while the algorithm idea was created in the 1970s, it was not popularized until 2000 when Cheng and Church proposed a variance-based biclustering method and successfully applied it to the field [29]. Their paper is still considered one of the most influential aspects to field of gene expression clustering.

The overall idea of biclustering is fairly straight forward. Given an m x n matrix, a biclustering algorithm tries to find subsections of rows that have similar behavioral patterns over a subsection of columns. A bicluster is equivalent to a biclique in a corresponding bipartite graph. This essentially means that all of a bicluster's rows, or observations, are all connected to every column, or attribute, presented in the bicluster. For this statement to be true, then a bicluster is generally thought of as having constant values throughout the cluster, constant values across either all rows or all columns, or coherent values of some kind [30]. Generally speaking, coherent values refer to a pattern where values in the bicluster could be anything from additive, multiplicative, or have some other kind of special mathematical relationship.

For our computational wine wheel, the idea of a bicluster should be explored as it presents the opportunity to find subspaces in our data where subsections of columns define a cluster instead of all attributes contained from that cluster's wines. We introduced the idea of strongly and weakly influential attributes with our weight system. However, perhaps biclustering can transcend the need for *important* attributes, and simply identify the true patterns between wines. This is because we do not take distance between wines into effect, so we have no need for the weighted attributes. The following sections in this chapter will introduce the BiMax algorithm, show a running example,

and finally will discuss the algorithm's effect on wine data generated using our computational wine wheel as presented in Chapter 2.5.

4.1: BiMax BiClustering

The BiMax BiClustering algorithm was a reference method developed by Prelic et al. for baseline comparison of biclustering algorithms in general [31]. The process is fairly simple in that it searches for biclusters that consist entirely of 1s in a binary matrix. This is perfect for datasets generated with the computational wine wheel in mind because a wine fits the binary requisite; a wine either has an attribute or it does not. With this in mind, our goal is to use the BiMax algorithm to find all inclusion-maximal biclusters of wines and attributes. This means a bicluster cannot be fully contained within another bicluster. We hope to show that resulting clusters show accurate perceptions of grouped wines and attributes that make sense. This section will introduce the methodology behind the BiMax algorithm.

To introduce mathematically what a bicluster is, we can look at its definition. In terms of our computational wine wheel, a bicluster (W, A) corresponds to a subset of wines $W \subseteq \{1, \dots, n\}$ that jointly share a subset of attributes $A \subseteq \{1, \dots, m\}$. The pair $(W, A) \in 2^{\{1,\dots,n\}} \, x \, 2^{\{1,\dots,m\}}$ is considered inclusion maximal if and only if:

(1) $\forall\, i \in W, j \in A : e_{ij} = 1$
(2) $\nexists\, (W', A')$ with (a) meets criteria (1) and (b) $W \subseteq W' \wedge A \subseteq A' \wedge (W', A') \neq (W, A)$

EQUATION 4.1. BiMax BiCluster Definition

Criteria (1) states that given a possible bicluster, every possible value must be a 1. Criteria (2) is the inclusion-maximal stipulation that says a bicluster (W,A) is considered inclusion-maximal as long as there does not exist another bicluster (W',A') in which both the wines and attributes of (W,A) are true subsets of (W',A'). Also, there is no sense of duplication so (W,A) and (W',A') cannot be fully equal as well. It is worth noting that the basis of being a subset is determined by testing the wines and attributes independently of each other, as if the wines in W are not considered a subset of the wines in W', then there's no need to compare the attributes between the clusters.

Since we have defined what a bicluster is, we can dive into the actual processing of an input data matrix. The BiMax algorithm is a divide and conquer approach that recursively divides an input matrix into what can be considered three different matrices, one of which contains only values of 0 and can be thrown out. The remaining two possibly-overlapping matrices are then each recursively processed until a bicluster is found. The figure below shows how the initial data is reorganized to find the sub-matrices U and V, represented by the blue and gold rectangles, respectively.

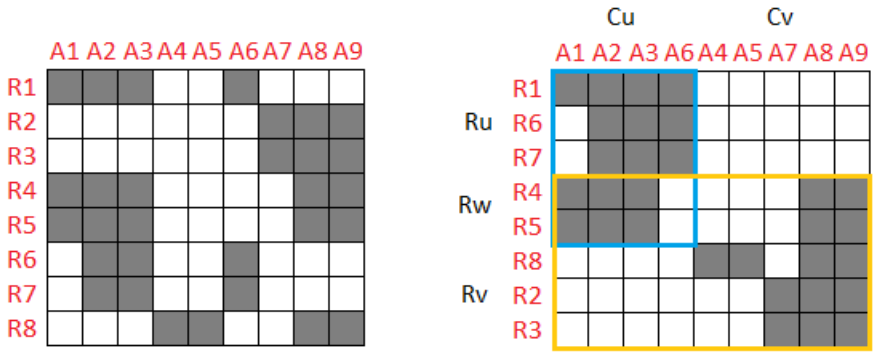

FIGURE 4.1. BiMax BiClustering Generic Step Before and After

In FIGURE 4.1, the left matrix represents a sample input matrix with eight rows and nine columns. The right matrix represents the end result of this initial conquer step. Since we can tell right away that the entire matrix is not entirely filled with 1 values, our first step is to find a row that has both 1s and 0s. Row R1 works perfectly for this so this will be our chosen row. The first step is to then shift the columns of the matrix around so that all 1-valued columns for row R1 are shifted all the way to the left. For this example, only column A6 needs to be moved to right after column A3. Once this shift is finished, we can think of the matrix as now having two vertical sections called *Cu* and *Cv* as depicted in FIGURE 4.1. Column section *Cu* represents all columns with values of 1 as shared by our chosen row. Column section *Cv* conversely represents all columns not attributed to our chosen row. With the columns correctly arranged, we now want to rearrange the matrix so that the rows are separated into three succinct row sections: *Ru*, *Rw*, and *Rv*. Row section *Ru* contains all rows where only columns in *Cu* contain a value of 1. In our example, that subsection

of rows are R1, R6, and R7. Row section *Rw* contains all rows where there are columns in both *Cu* and *Cv* with a value of 1. These rows include R4 and R5. Row section *Rv* is then all rows where only columns in *Cv* contain a value of 1. These rows include R8, R2, and R3. Once we know row and column order to fulfill these tasks, we can form our newly arranged matrix and divide it into two sub-matrices. The blue outline in FIGURE 4.1 corresponds to sub-matrix $U = (Ru \cup Rw, Cu)$, and the gold outline corresponds to the sub-matrix $V = (Rw \cup Rv, Cu \cup Cv)$. Should an area exist constrained by (*Ru*,*Cv*), then it will be filled with only 0 values and will subsequently be ignored. Matrices U and V are then recursively called and the above process is repeated until matrices are found where all values are 1. At that point, assuming the resulting matrix is inclusion-maximal, we consider it a maximal bicluster for the input data matrix.

There is one small caveat about making sure a bicluster is inclusion-maximal. Should the sub-matrices U and V contain any shared rows (*Rw*), it is possible that when a bicluster is found it may actually not be maximal. To make sure it is, each time the sub-matrix V is processed, we can pass along a "callstack" of columns that need to be checked at each level of the recursive calls. For each recursive function of V, we add a row to the callstack containing all columns in *Cv* at that current level. This callstack can keep growing depending on how many levels it takes to find a possible bicluster. Once a bicluster is found, it can be considered inclusion maximal if at least one column in the bicluster exists on each level of the callstack for the current recursive chain. This will guarantee that any bicluster found somewhere in V does not already exist as a superset in U. The next section will detail the processing and the callstack in more detail.

4.2: BiMax BiClustering Example

Before we dive into analyzing our wine data, we will run through a quick example using the same example data set used in Chapter 3.6 for Hierarchical Clustering. For the BiClustering example, we will define some perquisites that a bicluster must have at least two rows and at least two columns or we will reject it. The dataset in FIGURE 4.2.1 shows our initial matrix, which is composed of 5 rows (wine1, wine2, wine3, wine4, wine5) and 4 columns (cherry, blueberry, plum,

spice). For our running example, we will abbreviate the rows to (W1,W2,W3,W4,W5) and the columns to (C,B,P,S).

	cherry	blueberry	plum	spice
wine1	1	0	1	0
wine2	0	0	1	1
wine3	1	0	1	0
wine4	0	0	1	1
wine5	1	1	1	0

FIGURE 4.2.1. BiClustering Example – Initial Data

For the first iteration of operations in this example we will show the resulting table for each step. Subsequent illustrations will only show the beginning data and the ending result for each step. FIGURE 4.2.2 shows the initial data and the column and row operations we perform on it.

	C	B	P	S
W1	1	0	1	0
W2	0	0	1	1
W3	1	0	1	0
W4	0	0	1	1
W5	1	1	1	0

INITIAL DATA SET

	C	P	B	S
W1	1	1	0	0
W2	0	1	0	1
W3	1	1	0	0
W4	0	1	0	1
W5	1	1	1	0

COLUMN OPERATIONS

	C	P	B	S
W1	1	1	0	0
W3	1	1	0	0
W2	0	1	0	1
W4	0	1	0	1
W5	1	1	1	0

ROW OPERATIONS

	C	P	B	S
W1	1	1	0	0
W3	1	1	0	0
W2	0	1	0	1
W4	0	1	0	1
W5	1	1	1	0

FINAL MATRIX

FIGURE 4.2.2. BiClustering Example – Step 1

The first operation to perform on the matrix is to determine whether it is a potential bicluster or not. That is, we determine if every value in the matrix is 1. This matrix contains mixed values, so we need to transform it into a matrix that can be divided via the BiMax method. For this we need to first find a row that contains both values of 1 and 0. In this example, we can choose the very first row (W1). Once we have found our designated row, we need to perform the appropriate column moves so that Row W1 contains all of its columns with values of 1 on the left. This matrix is fairly simple so the only column operation is to move Column P directly after Column C. The result should look like the matrix shown in the COLUMN OPERATIONS section of FIGURE 4.2.2. The next operations are to rearrange the rows so that they form the *Ru*, *Rw*, and *Rv* row subsections. Note that we define *Cu* as being the set of columns that only Row W1 has 1-values in and we define *Cv* as the set of columns that Row W1 does not have 1-values in. With this in mind, we find *Ru* by shifting all rows to the top of the matrix that only have values of 1 in columns only found in *Cu*. In our example, we move Row W3 directly after W1 as it is the only other row that has 1-values in *Cu*. Next we need to find *Rw* by placing all rows that have 1-values in both *Cu* and *Cv* after the *Ru* rows. We see that all the remaining rows (W2, W4, W5) all contain 1-values in both *Cu* and *Cv* so there is nothing left to move since their current spot is fine. What should be left at this point are all rows that only have 1-values in *Cv*, but our example contain no such rows. The result of the row operations can be seen in the ROW OPERATIONS section of FIGURE 4.2.2. At this point, we have concluded the column and row operations and now only need to divide the matrix into two sub-matrices. As shown in the FINAL MATRIX section of FIGURE 4.2.2, the first Sub-matrix U is shown by the yellow-highlighted rows and column. In this example, it is composed of all five rows and the columns C and P. The second Sub-matrix V is shown by the rows and columns with the blue background. This sub-matrix is composed of three rows (W2,W4,W5) and all columns. We need to recursively apply the same column, row, and divide operations on each of these sub-matrices until we are left with a matrix with all 1-values, which will correspond to a potential bicluster. We will continue this example with the illustrations below.

In FIGURE 4.2.3 and FIGURE 4.2.4, the left column will represent the total recursive calling command for the current chain. The middle column will show a before and after image of the current matrix data. Lastly, the final column will show the callstack (Z) at that point.

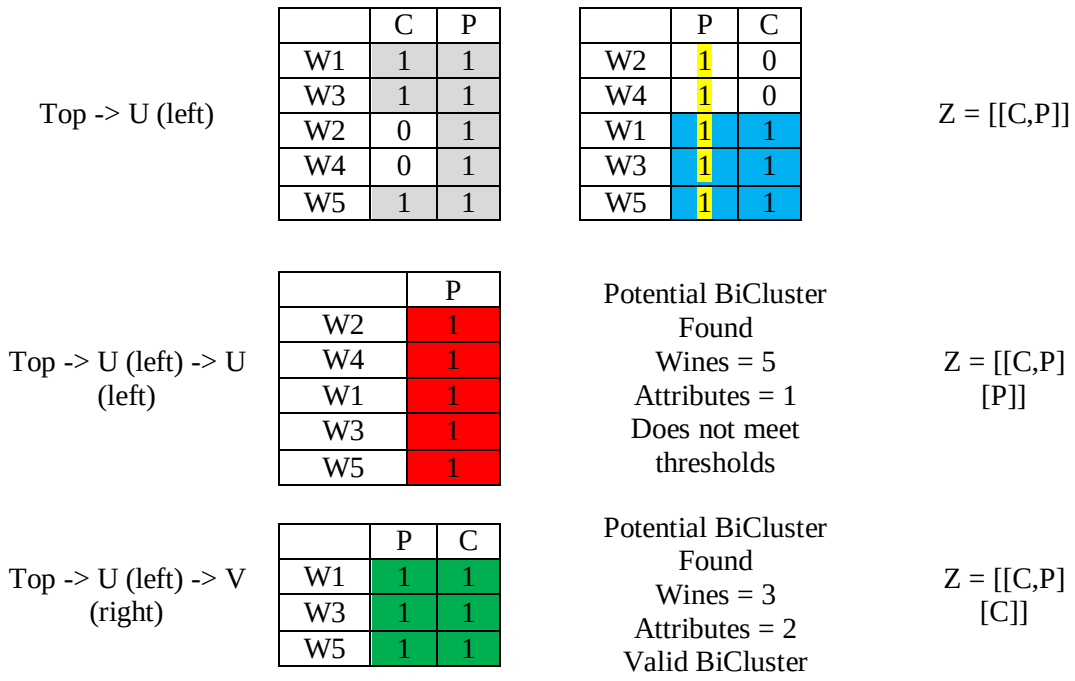

FIGURE 4.2.3. BiClustering Example – Step 2 (initial,left)

In FIGURE 4.2.3, we show the recursive results of processing the Sub-matrix U from the initial, top level. We also introduce the first addition of columns to the callstack, which is designed to check and throw out biclusters that are not maximal. For every recursive call on Sub-matrix U, we also add of U's columns as a list to the callstack. This is actually not necessary for U, but we want to drive home the point of the callstack. For every recursive call on Sub-matrix V, we add only the columns in the current matrix's *Cv* as a list to the callstack. When a potential bicluster is found, it must have at least one column present in every level of the callstack, or it is not considered maximal. We will see an example of this later. For the processing of the Top Level -> Sub-matrix U, we see that it gets divided into two possible biclusters. Both biclusters pass the callstack check by having columns in all levels of their respective callstacks. However, only the Top->U->V sub-matrix passes as the Top->U->U submatrix does not meet the minimum attribute threshold of 2. With

that, the entire top level U sub-matrix has been explored, so we will now the recursive function returns up in order to now process the topmost V submatrix as shown below.

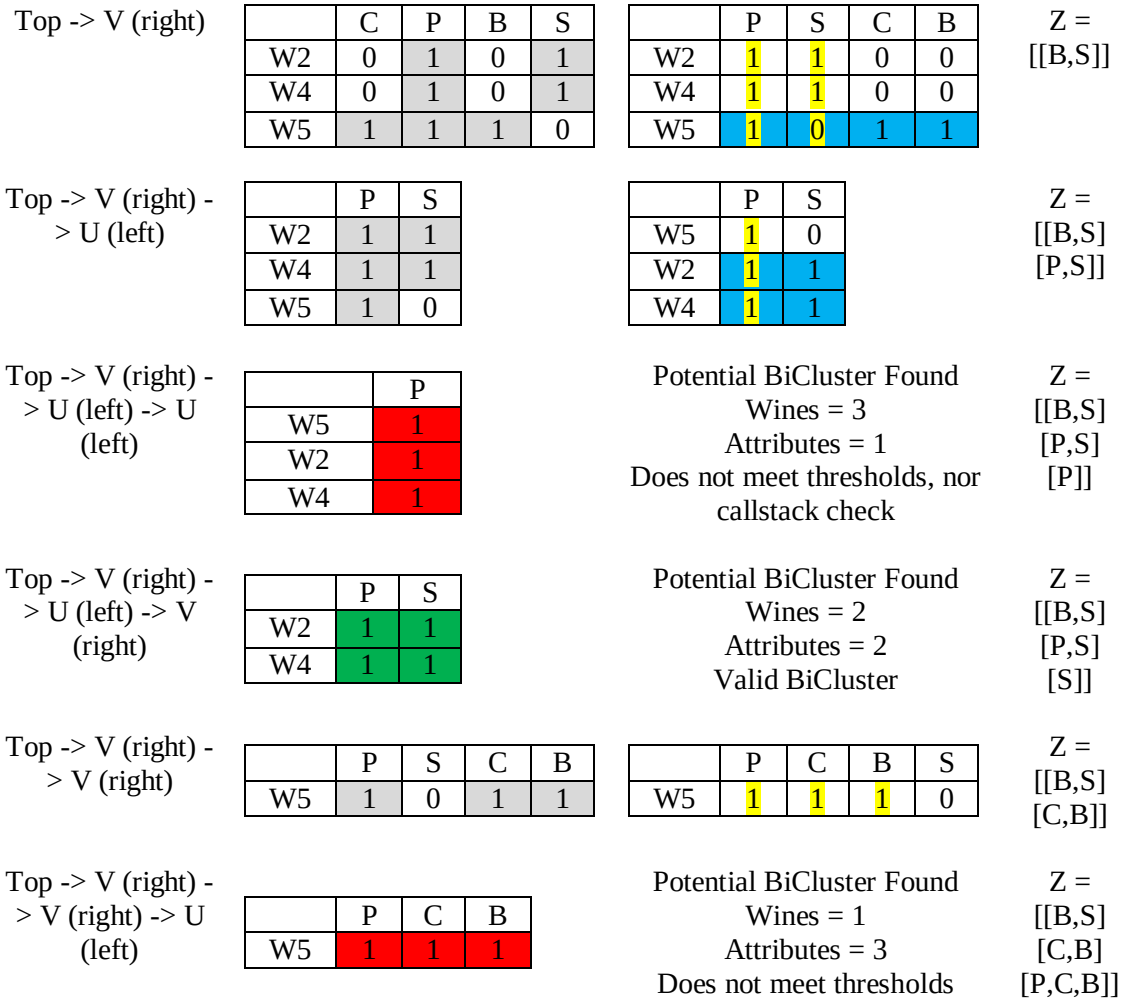

FIGURE 4.2.4. BiClustering Example – Step 3 (initial,right)

In FIGURE 4.2.4, we show the processing of Sub-matrix V from the initial, top level. This sub-matrix eventually finds three potential biclusters, but only the Top->V->U->V bicluster passes both the callstack and the minimum thresholds. The Top->V->V->U bicsluter fails the minimum threshold. It is the Top->V->U->U bicluster that is worth noting, as even though it does not meet the minimum attribute threshold, it is special in that it also does not meet the callstack check. It passes all levels of the callstack except for the very top level, which corresponds to the initial data matrix. We can see in Top->V that the slice (W2,W4,W5)x(P) is actually a subset of the Top->U-

>U bicluster. Even though it would have failed threshold requirements anyway, it is important to understand that we are able to detect non-maximal biclusters the moment they come up thanks to the callstack. This example resulted in five possible biclusters being found, yet one failed the callstack check, and only two of them actually met our threshold requirements. Taking a look at FIGURE 4.2.5, we can look back on the original data and try to visual identify where these two valid biclusters were.

	P	C
W1	1	1
W3	1	1
W5	1	1

	P	S
W2	1	1
W4	1	1

	cherry	blueberry	plum	spice
wine1	1	0	1	0
wine2	0	0	1	1
wine3	1	0	1	0
wine4	0	0	1	1
wine5	1	1	1	0

FIGURE 4.2.5. BiClustering Example - Results

Although they did not meet the threshold requirements, the two biclusters that did not make it are colored with red text. In case the colors are not available, those two rejected clusters are [(wine1, wine2, wine3, wine4, wine4) x (plum)] and [(wine5) x (cherry, blueberry, plum)].

4.3: BiClustering 50 Wines

This section will describe our biclustering and analysis process of 1 of the 5 sets of 50 wines as described in Chapter 2.5. We chose to only look into one vintage as Chapter 5 will focus on combining all five vintages together. For the 50 wines in the 2010 vintage, we implemented and applied the BiMax biclustering algorithm exactly as presented in this chapter. The overall bicluster summarization is described in FIGURE 4.3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	21	TOTAL
1	0	0	0	0	0	1	1	1	7	3	8	6	3	6	4	3	2	2	1	1	49
2	1	31	64	108	50	18	10	0	1	0	0	0	0	0	0	0	0	0	0	0	283
3	7	50	74	38	9	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	179
4	9	39	33	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	97
5	4	26	23	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54
6	2	16	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23
7	7	6	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20
8	3	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
9	1	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
10	1	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
11	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
12	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
14	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
15	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
16	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
17	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
18	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
19	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
21	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
22	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
27	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
28	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
TOTAL	48	193	207	163	59	20	11	1	8	3	8	6	3	6	4	3	2	2	1	1	749

FIGURE 4.3. Summarization of Biclusters of 50 Wines (2010)

This figure represents the total number of maximal biclusters found for the 2010 50-wine dataset. The table values represent the total number of biclusters that share a specific number of wines (vertical axis) versus a specific number of savory attributes (horizontal axis). In the table, there are darkened, rectangular borders that are meant to be a visual reference to show all biclusters where the minimum number of rows equals the minimum number of columns. For example, in this vintage there are no clusters that have at least five wines and at least five attributes. For a dataset with 50 wines and 259 possible attributes, this may seem like a low combination, but it makes sense as the number of total possible attributes in any given wine is fairly small. For this reason, we assume that any biclusters with many wines and attributes must have a stronger correlation to each other than biclusters where the total wines or total attributes are extremely lopsided, such as those biclusters with greater than 10 wines, but only 1 shared attribute. To check the quality of our

biclusters, we have to manually inspect example biclusters as we currently have no automated quality functions as we had developed for hierarchical clustering chapter.

We will first attempt to explore those biclusters that fall into the category of at least 4 wines and 4 attributes. In total, there were 17 total biclusters that fell into this group, which represents some of the most robust biclusters from this vintage, region, and varietal. We will examine a couple biclusters as well as some common themes below.

BiCluster #5/17 (Min Wines = 4, Min Attributes = 4) Cabernett Sauvignon – Napa Valley (2010)	
Wine Producer – Designation (4)	**Shared Attributes (4)**
CHAPPELLET SIGNATURE BERINGER PRIVATE RESERVE ARAUJO EISELE VINEYARD CAVUS STAGS LEAP DISTRICT	BLACK LICORICE RICH DARK BERRY DENSE

FIGURE 4.4. BiCluster from 50 Wines – Example 1 (Strong)

BiCluster #9/17 (Min Wines = 4, Min Attributes = 4) Cabernett Sauvignon – Napa Valley (2010)	
Wine Producer – Designation (5)	**Shared Attributes (4)**
BEAULIEU VINEYARD GEORGES DE LATOUR PRIVATE RESERVE DIAMOND CREEK VOLCANIC HILL DALLA VALLE MAYA BARNETT SPRING MOUNTAIN DISTRICT RATTLESNAKE HILL DAVID ARTHUR ELEVATION 1147	GREAT TANNINS_LOW FINISH FLAVORS

FIGURE 4.5. BiCluster from 50 Wines – Example 2 (Weak)

FIGURE 4.4 and FIGURE 4.5 show 2 of the 17 possible biclusters that we can examine from this group. The bicluster in FIGURE 4.4 shows four wines along with the four attributes shared among them. Bringing in the concept of weighted attributes from Chapter 2, we see important attributes, such as BLACK LICORICE and DARK BERRY. These are distinctive flavors that a taster might be accustomed to when sampling a Cabernet Sauvignon. This bicluster was also described as RICH and DENSE as well. FIGURE 4.5 shows a slightly bigger bicluster as it contains five total wines and four attributes. The biggest difference though is that this bicluster does not contain any important attributes that we might have considered in the Hierarchical Clustering section. We built

in no concept of weight or distance into the biclustering, so all attributes are treated equally. Looking at this bicluster, it may be of decent size on both ends, but the actual attribute selection is fairly weak and generic, and probably would not help a consumer when searching for potential wines. However, there are still advantages as the biclusters allow us to categorize wines with true subsets of attributes. Unlike hierarchical clustering, which would present groups of wines using all attributes among them, biclustering allows us to show many different, but smaller, groupings of the same wines across varying attribute patterns. This would give potential for consumers to select small flavor profiles and expect higher quality results since the biclusters might have filtered out unneeded attributes. To reinforce this idea, we can explore common themes found among biclusters. It makes sense that by biclustering only cabernet sauvignon wines from the Napa Valley region that there should be significant overlap among attributes between wines. Additionally, because of the sheer number of total attributes and by comparison, the low amount of attributes per wine, it is reasonable to see a very large amount of resulting biclusters, many of which share a very small subset of attributes. Among the 17 biclusters found in this group, 10 biclusters contain EARTHY or LOAM attributes, and another 6 biclusters are focused around the DENSE attribute. Should a flavor palette be used to search for (EARTHY or LOAM), then we could present the user with a set of wine groups that might alter against the following specific attributes: CEDAR, DARK BERRY, and BLACK LICORICE.

When examining biclusters for our wine dataset, we do not have to try to find clusters that are maximal in terms of both number of wines and attributes. We can also look for interesting wine and attribute combinations in those clusters that have either low number of wines and high number of attributes, or those with high number of wines and low number of attributes. The former suggests a smaller subset of wines stayed consistent across a majority of attributes across vintages, while the later suggests a larger subset of wines that might share a small pool of distinctive attributes. The figures below show examples from both types of biclusters found in group of biclusters that contain at least three wines and at least three attributes. This group contained a total of 208 biclusters.

BiCluster #98/208 (Min Wines = 3, Min Attributes = 3) Cabernet Sauvignon – Napa Valley (2010)	
Wine Producer – Designation (3)	**Shared Attributes (6)**
EHLERS ESTATE ST. HELENA 1886 BEAULIEU VINEYARD TAPESTRY RESERVE BERINGER PRIVATE RESERVE	EARTHY FLAVORS LOAM FINISH DENSE TRACTION

FIGURE 4.6. BiCluster from 50 Wines – Example 3 (Low Wines, High Attributes)

BiCluster #8/208 (Min Wines = 3, Min Attributes = 3) Cabernet Sauvignon – Napa Valley (2010)	
Wine Producer – Designation (7)	**Shared Attributes (3)**
BURLY CAKEBREAD BOND PLURIBUS ALPHA OMEGA DIAMOND CREEK RED ROCK TERRACE FORMAN COLGIN IX ESTATE	CEDAR LOAM EARTHY

FIGURE 4.7. BiCluster from 50 Wines – Example 4 (High Wines, Low Attributes)

FIGURE 4.6 shows a bicluster with three wines and six attributes. While there are many attributes shared between the three wines, the only important attributes are EARTHY and LOAM. However, if a consumer is only interested in these two savory flavors, then this and other similar biclusters allow the user to search via subjective descriptors. In this case, we present a set of DENSE wines that keep TRACTION through a FLAVOR[ABLE] FINISH. Alternatively, FIGURE 4.7 shows a bicluster with seven wines, but only three attributes. In this example, we continue the theme of EARTHY and LOAM flavors, but explore adding in additional highly-weighted attributes, such as CEDAR. If a consumer is only interested in these three savory attributes, then this bicluster offers a wide array of wines to try without getting into individual wine descriptors.

The beauty of biclustering is that many of the biclusters share subsets of both wines and attributes, but all are still maximal when taking the entire contents of the bicluster into account. We have presented examples that show how limiting or expanding wine attributes can change the biclusters that might come up in a potential attribute profile search. Depending on the search

criteria used, biclusters can either be generic with many wines, or specific with fewer wines. However, this is one issue with biclustering a single vintage of wines in that we only see patterns across a given year. Since wines are produced yearly, it would be nice to try to find these patterns across many years, which should help identify consistent and dominant wine attributes in a region. We try to solve this problem using a triclustering method as presented in the next chapter.

CHAPTER 5: TRICLUSTERING

Just as with biclustering, triclustering is becoming a popular method to explore gene expression microarray data. At its core, triclustering can actually be thought of as an extension to biclustering. Instead of working with two dimensional matrices, triclustering focuses on finding behavioral patterns between row and columns along a time series. For gene microarray analysis, the time series can be usually thought of as the same genes and sample attributes along different experiments, with the results being genes that share expression profiles under different scenarios. By adding the third dimension to the data, triclustering can be thought of as reinforcing the biclusters that could have been found on a single time slice. Should biclusters exist beyond their single time scope, their inferences hold greater weight. There has already been work done in the gene expression field for triclustering as detailed by Zhao et al. [32] and Bhar et al [33]. Their approaches assume that the data is a bit more complex than true binary choices like our wine data. Also, Zhao at al. propose a weighted, directed range multigraph to find biclusters within a given time slice. Then those biclusters are searched among each other to find the maximal versions of each that share multiple time slices. We want to post our exploration into a technique that borrows ideas from the BiMax algorithm in order to extend it into the three dimensional space. This chapter will discuss our look into a novel TriMax TriClustering reference algorithm, which should act as an extension to the BiMax BiClustering algorithm discussed in Chapter 4. To our knowledge, there is no preexisting work that attempts to find triclusters using our method, nor is there a reference algorithm for triclustering that emulates the role BiMax performs for biclustering.

5.1: TriMax TriClustering

Just as with BiMax BiClustering, Trimax Triclustering should be considered a reference algorithm in that it attempts to cluster on the most basic level and makes no assumptions of differing values in the data. That means it expects all values to either be zero or non-zero, so completely binary in nature. For our specific dataset, we will assume all data values are either 1 or 0. We will start with the definition, which should look very similar to the BiMax definition. We consider a

tricluster (W, A, T) to correspond to a subset of wines $W \subseteq \{1, \ldots, n\}$ that jointly share a subset of wine attributes $A \subseteq \{1, \ldots, m\}$ across a subset of time slices $T \subseteq \{1, \ldots, o\}$. The tuple $(W, A, T) \in 2^{\{1,\ldots,n\}} x\, 2^{\{1,\ldots,m\}} x\, 2^{\{1,\ldots,o\}}$ is considered inclusion maximal if and only if it meets the following two criteria.

(1) $\forall\, i \in W, j \in A, k \in T : e_{ijk} = 1$
(2) $\nexists\, (W', A', T')$ with (a) meets criteria (1) and (b) $W \subseteq W' \wedge A \subseteq A' \wedge T \subseteq T' \wedge (W', A', T') \neq (W, A, T)$

EQUATION 5.1 TriMax TriCluster Definition

Criteria (1) states that given a possible tricluster, every possible value must be a 1 across all rows, columns, and time slices. Criteria (2) is the inclusion-maximal stipulation that says a tricluster A is considered inclusion-maximal as long as there does not exist another tricluster B in which the grouping of wines, attributes, and time slices of A are a subset of B. If a tricluster A is found, there also cannot be a tricluster B, such that A = B. Now that we have defined a tricluster, we can discussed the algorithm to find them. However, there should be two points noted before we discuss the algorithm. (1) Our proposed algorithm uses the BiMax algorithm, so a good understanding of the algorithm, as we discussed in Chapter 4, is necessary to proceed. (2) We believe our program is able to find all triclusters, but unlike BiMax which knows at runtime whichs biclusters to ignore thanks to its column callstack, TriMax has to filter out duplicate or subset triclusters after finding all possible triclusters. We will examine an example dataset that shows how duplicates arise, but first we will run through the algorithm itself. The pseudocode is presented in FIGURE 5.1.

TriMax TriClustering Reference Algorithm

$ID = D = \{\{w_i\}_{i=1}^{a}\}_{i=1}^{t}$

$TriList = \{\emptyset\}$

TriMaxTriClust$(D, mWAT, vT)$:

1 **for any** $t \in D_T$ **where** $t \in vT$ **and all** $D_{W,A,t} = 1$

2 **return**

3 **for** $t \in D$ **do**:

4 **if**$(t \in vT$ **or** $(all\ values\ in\ t = 1\ or\ 0))$:

5 **if**$(all\ values\ in\ t = 0)$:

6 $append(vT, t)$

7 **continue to next time slice**

8 $B =$ ***BiMaxBiClust***$(t, mWAT)$

9 **for** $b \in B$ **do**:

10 $newD' = \{b\}_{t=1}^{ID_T}$

11 ***TriMaxTriClust***$(D', mWAT, vT)$

12 $append(vT, t)$

13 **if**$(len(D_W, D_A, D_T) \geq \{mWAT\})$:

14 $append(TriList, D_{W,A,(D_T - vT)})$: **unless** $D_T - vT = \{\emptyset\}$

15 **return**

16

17 $\forall\ t \in TriList$: *remove duplicates/subsets*

FIGURE 5.1. Proposed TriMax TriClustering Reference Algorithm Pseudocode

The pseudocode for our proposed TriMax TriClustering algorithm is shown in FIGURE 5.1. As a base concept, we want to take biclusters found in each time slice and see if they can extend across any and all other time slices. To accomplish this, we start with our dataset D and process each of D's *t* time slices iteratively. For a given bicluster *b* that is found in a given time slice *t*, we form a new dataset *D'*, which consists of the rows and columns of *b*, along every time slices of the input data. That new dataset is then recursively processed using the same methodology until the resulting dataset D' consists only of values of 1. Naively, we can consider a completely 1-valued dataset as a tricluster if it passes the minimum row, column, and time slice amounts set in *mWAT*. Since our process does not have any callstacks like the BiMax algorithm, TriMax will natively introduce duplicate triclusters or triclusters that are subsets, or non-maximal. To combat part of this problem,

we introduce a visited array *vT*, which is populated with the index of a time slice once that time slice's recursive processing has finished. This allows any tricluster found to be ignored if it includes a time slice within *vT* at any recursive level. If this occurs, ideally it means that the tricluster has already been found previously. However, this only attempts to filter out triclusters between given time slices. It does not work on duplicate or non-maximal subsets formed from partially overlapping biclusters originating from the same time slice. FIGURE 5.2 shows an example of duplication issues caused by overlapping biclusters in a given time slice T1.

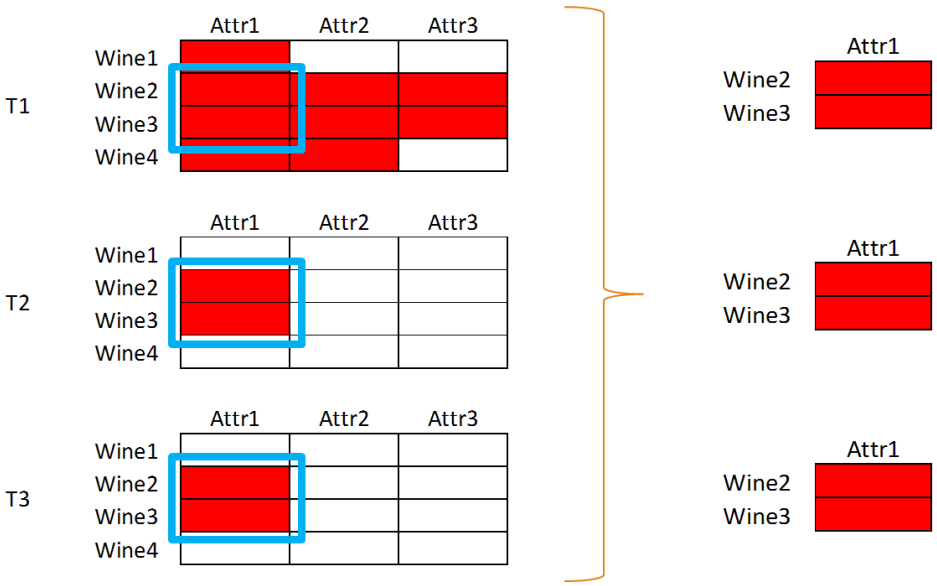

FIGURE 5.2. Tricluster Found from Multiple Intra-Timeslice Biclusters

In FIGURE 5.2, we can say we would process time slice T1 first by expanding the three biclusters found within it: {W1,W2,W3,W4}x{A1}, {W2,W3}x{A1,A2,A3}, and {W2,W3,W4}x{A1,A2}. As shown by the blue squares, all three biclusters share the following subset of rows and columns: {W2,W3}x{A1}. By expanding all three, the same tricluster, as presented on the right of FIGURE 5.2, will be found three times, and thusly will have to be filtered down to one instance afterwards. Even with the slight timing inefficiency here in the post processing, we believe this method will still find all maximal triclusters between all time slices given in a three dimensional data set. The next section will detail the results when applying triclustering to our multi-vintage 50-wine dataset.

5.2: TriClustering 50 Wines across 5 Years

To test out the TriMax algorithm, we used the dataset presented in Chapter 2.5, except this time we used the full 5 years' worth of vintages. We applied the TriMax algorithm presented in the previous section, and in total we found 23,225 possible triclusters. Since we knew a large percentage of these would actually be duplicates or non-maximal, we performed the pairwise subset comparison and pulled out a total of 7,296 superset triclusters. Of all the triclusters found, 6,357 of them only exist in a single time slice. These can actually be thought of biclusters in a three dimensional space that only existed individually. We found 735 triclusters that spanned 2 time slices. We found 166 triclusters that spanned 3 time slices, and 31 triclusters that spanned 4 time slices. Lastly, we found 7 triclusters that spanned all 5 time slices. FIGURE 5.3.1 through FIGURES 5.3.5 summarize the findings just as we did in FIGURE 4.3 for biclusters.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	TOTAL
1	0	0	0	0	0	3	1	10	23	17	20	28	24	26	21	30	11	12	10	2	8	2	1	249
2	2	82	306	440	418	280	179	67	27	4	1	1	0	0	0	0	0	0	0	0	0	0	0	1807
3	24	241	520	464	279	98	31	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1660
4	29	224	395	241	96	21	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1007
5	17	185	219	128	27	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	580
6	15	124	137	53	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	334
7	33	81	80	12	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	210
8	17	84	41	10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	153
9	14	43	27	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	88
10	12	38	12	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	64
11	5	32	5	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	44
12	11	15	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35
13	6	13	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21
14	10	10	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22
15	3	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13
16	11	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15
17	6	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
18	6	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
19	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
20	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
21	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
22	3	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
23	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
25	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
26	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
27	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
28	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
30	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
33	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
36	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
38	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
TOTAL	251	1198	1756	1356	830	406	212	80	50	21	21	29	24	26	21	30	11	12	10	2	8	2	1	6357

FIGURE 5.3.1. TriMax Results on 50 Wines in 1 Time Slice

	1	2	3	4	5	6	7	8	9	TOTAL
1	7	33	77	78	40	21	7	3	1	267
2	28	119	61	3	0	0	0	0	0	211
3	36	61	5	0	0	0	0	0	0	102
4	19	24	0	0	0	0	0	0	0	43
5	22	15	0	0	0	0	0	0	0	37
6	12	4	0	0	0	0	0	0	0	16
7	10	1	0	0	0	0	0	0	0	11
8	7	1	0	0	0	0	0	0	0	8
9	7	2	0	0	0	0	0	0	0	9
10	4	0	0	0	0	0	0	0	0	4
11	6	0	0	0	0	0	0	0	0	6
12	3	0	0	0	0	0	0	0	0	3
13	5	0	0	0	0	0	0	0	0	5
15	1	0	0	0	0	0	0	0	0	1
16	2	0	0	0	0	0	0	0	0	2
17	3	0	0	0	0	0	0	0	0	3
18	1	0	0	0	0	0	0	0	0	1
19	2	0	0	0	0	0	0	0	0	2
20	1	0	0	0	0	0	0	0	0	1
21	1	0	0	0	0	0	0	0	0	1
24	1	0	0	0	0	0	0	0	0	1
25	1	0	0	0	0	0	0	0	0	1
TOTAL	179	260	143	81	40	21	7	3	1	735

FIGURE 5.3.2. TriMax Results on 50 Wines in 2 Time Slices

	1	2	3	4	TOTAL
1	9	33	26	9	77
2	23	13	0	0	36
3	15	5	0	0	20
4	7	1	0	0	8
5	5	0	0	0	5
6	4	0	0	0	4
7	4	0	0	0	4
8	1	0	0	0	1
10	2	0	0	0	2
11	2	0	0	0	2
12	2	0	0	0	2
13	2	0	0	0	2
14	1	0	0	0	1
16	1	0	0	0	1
19	1	0	0	0	1
TOTAL	79	52	26	9	166

FIGURE 5.3.3. TriMax Results on 50 Wines in 3 Time Slices

	1	2	3	TOTAL
1	4	8	1	13
2	4	3	0	7
3	2	0	0	2
4	4	0	0	4
5	1	0	0	1
9	1	0	0	1
10	1	0	0	1
11	1	0	0	1
12	1	0	0	1
TOTAL	19	11	1	31

FIGURE 5.3.4. TriMax Results on 50 Wines in 4 Time Slices

	1	2	TOTAL
1	2	2	4
2	1	0	1
3	1	0	1
8	1	0	1
TOTAL	5	2	7

FIGURE 5.3.5. TriMax Results on 50 Wines in 5 Time Slices

We will now explore some of the triclusters that we found above. To show a tricluster that managed to expand across all time slices, we can look at a tricluster in FIGURE 5.3.5 that has eight wines and one attribute.

Tricluster #102 Cabernet Sauvignon – Napa Valley		
Wine Producer – Designation (8)	**Shared Attributes (1)**	**Vintages (5)**
BARNETT SPRING MOUNTAIN DISTRICT RATTLESNAKE HILL BERINGER PRIVATE RESERVE BOND MELBURY BOND QUELLA BOND ST. EDEN BOND VECINA DIAMOND CREEK GRAVELLY MEADOW DIAMOND CREEK VOLCANIC HILL	GREAT	2010 2009 2008 2007 2006

FIGURE 5.4. TriCluster from 50 Wines – Example 1

While FIGURE 5.4 only has a single, least-weighted attribute, the triclustering aspect lets us know that all eight of these wines are considered GREAT for five years in a row. We can also see that

four of the wines share the same producer, so it is probable that any other wine produced by BOND would also probably be considered great. Apart from groups of many wines, we can also look out how a single wine changes between vintages as seen in the figure below.

Tricluster #3456,3457,3458 Cabernet Sauvignon – Napa Valley		
Wine Producer – Designation (1)	**Shared Attributes (4)**	**Vintages (3)**
CASA PIENA	BLACKBERRY CONCENTRATED GREAT TIGHT FINISH (2007) MINERAL (2007) CURRANT (2006) FOCUSED (2006) TANNINS_HIGH (2006)	2008 2007 2006

FIGURE 5.5. Three TriClusters from 50 Wines – Example 2

FIGURE 5.5 shows us a single wine that has four attributes across three vintages. The distinctive BLACKBERRY taste is found across all three years, as well as a TIGHT, CONCENTRATED, and GREAT tasting. Using 2008 as a base year, we can add in 2007 and gain the FINISH and MINERAL attributes. Alternatively, we can add in 2006 and gain FOCUSED and TANNINS_HIGH. These three triclusters allow us to research a specific wine and see how the attributes change over time. Anything can affect a given vintage from bad weather to slight changes in harvesting and fermentation processing. Unless something major changes though, we can probably use the entire range of possible attributes to forecast what this specific wine might entail in the future. Lastly, we will show an example with at least two items in every category.

Tricluster #1155 Cabernet Sauvignon – Napa Valley		
Wine Producer – Designation (2)	**Shared Attributes (2)**	**Vintages (4)**
CASA PIENA DANCING HARES	BLACKBERRY GREAT	2009 2008 2007 2006

FIGURE 5.6. Tricluster across 4 Years – Example 3

FIGURE 5.6 shows a tricluster that has two wines containing two attributes across four consecutive vintages. The main purpose of this figure is just to provide an example of triclusters with many vintages that are not limited to just one wine or wine attribute. While both wines and attributes are still fairly small, this just further provides opportunity for specialized searching and classification.

Since the dataset used for this chapter contained only a specific varietal from a specific region, we were able to get highly defined cluster results. We believe that triclustering data from a variety of types and sources should produce interesting results and it will be worth exploring those datasets in the future. We would have tried for wines contained within our computational wine wheel, but we could not be guaranteed to find reviews on a large set of wines for a specific span of years, so we chose a narrow approach by choosing only one type of wine from a single region.

CONCLUSION

This paper has introduced the data analysis area of Wine Informatics, which comprises of the gathering and analysis of specialized data sets consisting of key attributes extracted from professional wine tasting reviews. Within the Wine Informatics field itself, we have also introduced the concept of a computational wine wheel, which corresponds to a specific set of wines and the key attributes belonging to those wines. For our specific methods of analysis, we chose to focus on the following three clustering methods: Single Linkage Agglomerative Hierarchical Clustering, BiMax Biclustering, and a proposed TriMax Triclustering. Using all three of these methods, we have extracted clusters of varying structures in an attempt to show that the data inside the computational wine wheel can present highly correlated sets of wines and attributes that not only make sense, but can possibly be used by consumers, wine enthusiasts, wine suppliers, and specialized wine websites. For hierarchical clustering specifically, we introduce a set of cluster quality metrics that can help automatically derive quality scores when using large sets of wines. Instead of deriving the quality on the attributes used in the clustering, we used set of non-savory attributes that do not affect the clustering process at all. We show that we are able to successfully group types of wines by type, varietal, country, and world type based only on the review attributes of the tasting. We also tackle a dataset of wines of the same varietal and region (Cabernet Sauvignon and Napa Valley) in order to see how subspace biclustering resulted. We were able to pull out cohesive clusters that highlight subsets of wine and attribute combinations. These clusters show promise in allowing palette searching for similar wines. Lastly, we applied the same dataset used in biclustering, except across multiple years, in a proposed triclustering method. We used this show that biclusters found in a given vintage might actually extend across time, meaning we found wines that stayed consistent across vintages. This could strengthen the chances that a future vintage of a wine might share similar properties. We showed that it is indeed possible to cluster similar wines using just the review tasting notes, and that it will be worth continuing to work with this data in order to make it more useful for many aspects of the wine industry.

FUTURE WORKS

This paper presents several major opportunities to expand on the work presented. The first path involves expanding the computational wine wheel dictionary by looking into automated ways of extracting new wine attributes from wine reviews. Also, further review of the dataset is welcomed as this represents the effort of only a few individuals. Another avenue is taking our dataset and using clustering algorithms that this paper did not explore. Given the nature of the dataset, we feel that hierarchical clustering, biclustering, and triclustering were great entry points, but a wider range of application would help to further reinforce the data. Lastly, we introduced a new reference subspace clustering algorithm called TriMax TriClustering. We believe the direction the algorithm takes in one that can result in a complete list of maximal triclusters, just as BiMax does for maximal biclusters. However, the algorithm still results in some duplicated or non-maximal triclusters that require post-processing to filter out. While the tricluster results are still valid, the algorithm methodology will need further review in order to ensure efficient and accurate results after the first pass of the data. We would like to see TriMax truly be the successor to BiMax in all aspects.

REFERENCES

[1] Kaku, Michio. *Physics of the Future: How Science Will Shape Human Density and Our Daily Lives by the Year 2100*. Doubleday, 2011. Print.

[2] Gantz, John. Reinsel, David. "THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East", IDC. December 2012. Available from EMC, accessed on 21 January 2014. <http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf>.

[3] Conway, Drew. "The Data Science Venn Diagram." September 30, 2010. Available from Drew Conway, accessed on 21 January 2015. <http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram>.

[4] Han, Jiawei, and Micheline Kamber. *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann. pp. 227-459, 2001.

[5] Rokach, Lior, and Oded Maimon. "Clustering methods." Data mining and knowledge discovery handbook. Springer US. pp. 321-352, 2005.

[6] Govaert, Gerard. Nadif, Mohammed. *Co-clustering: models, algorithms and applications*. London: Wily-ISTE. 2013.

[7] Ester, Martin. Kriegel, Hans-Peter. Sander, Jörg. Xu, Xiaowei. *A density-based algorithm for discovering clusters in large spatial databases with noise*. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231. 1996.

[8] Quinlan, J. R. "Simplifying decision trees". *International Journal of Man-Machine Studies.* Volume 27, Issue 3: pp 221-234. 1987.

[9] Kecman, Vojislav. *Learning and Soft Computing - Support Vector Machines, Neural Networks, Fuzzy Logic Systems*. The MIT Press. Cambridge, MA, 2001.

[10] Agrawal, Rakesh. Srikant, Ramakrishnan. *Fast algorithms for mining association rules in large databases*. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pp 487-499. Santiago, Chile, 1994.

[11] Johnson, H. *Vintage: The Story of Wine*. Simon & Schuster. pp. 11–6, 1989.

[12] Colby, Chris. "Why Grapes?" Beer and Wine Journal. 11 July, 2013. Web. 1 February, 2015. <http://beerandwinejournal.com/why-grapes/>.

[13] Goode, Jamie. "How Wine Is Made: An Illustrated Guide to the Winemaking Process." Wineanorak. August, 2011. Web. 1 February, 2015. <http://www.wineanorak.com/howwineismade.htm>.

[14] Robinson, J. *The Oxford Companion to Wine*. Third Edition. Oxford University Press. pp 273-274, 2006.

[15] MacNeil, K. *The Wine Bible.* Workman Publishing. pp 100-104, 2001.

[16] Brochet, Frédéric. *Chemical Object Representation in the Field of Consciousness*. 2001.

[17] Morrot, Gil. Brochet, Frédéric. Dubourdieu, Denis. *The Color of Odors*. Academic Press. 2001.

[18] Sherman, Michael (n.d.). *K-means clustering of wine chemistry data: Creating and evaluating a predictive model and an analysis of the data*. Web. 29 March 2015. < http://www.michaelwsherman.com/ projects/winecluster/report.pdf>.

[19] Parker, Robert. "The Wine Advocate Rating System". Available from The Wine Advocate, accessed on 29 March 2015. < https://www.erobertparker.com/info/legend.asp>.

[20] Wine Spectator. N.p., n.d. Web. 29 March 2015. <http://winespectator.com>.

[21] Shanken, Marvin R. Matthews, Thomas. N.d. "Why We Taste Blind". Available from Wine Spectator, accessed on 29 March 2015. < http://images.winespectator.com/wso/pdf/ WShowwetasteLTR.pdf>.

[22] Nobel, Ann C. N.d. "Wine Aroma Wheel". Web. 29 March 2015. <http://winearomawheel.com/>.

[23] Zaki, Mohammed J. Meira, Jr, Wagner. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press. pp 367-372. 2014. Print.

[24] Musaloiu-E, Razvan. N.d. "Linux Kernal 2.6.29 + tux3". Web. 30 March 2015. <http:// www.cs.jhu.edu/~razvanm/fs-expedition/tux3.html>.

[25] Bellman, Richard E. Rand Corporation. *Dynamic programming*. Princeton University Press. 1957.

[26] Levandowsky, Michael. Winter, David. "Distance between sets". *Nature*. Volume 234 (5): pp 34–35, 5 November 1971.

[27] Teknomo, Kardi. "Jaccard's Coefficient". Web. 30 March 2015. < http://people.revoledu.com/kardi/tutorial/ Similarity/Jaccard.html>

[28] Seo, J., Shneiderman, B. "Interactively Exploring Hierarchical Clustering Results", *IEEE Computer*, Volume 35, Number 7, pp. 80-86, July 2002.

[29] Cheng,Y. and Church,G.M. "Biclustering of expression data". ISMB 2000 proceedings, pp 93–103, 2000.

[30] Madeira SC, Oliveira AL (2004). "Biclustering Algorithms for Biological Data Analysis: A Survey". *IEEE Transactions on Computational Biology and Bioinformatics* **1** (1): 24–45. 2004.

[31] Prelic, A. Bleuler, S. Zimmermann, P. Wille, A. Bühlmann, P. Gruissem, W., Hennig, Lars. Thiele, L. Zitzler, E. "A systematic comparison and evaluation of biclustering methods for gene expression data". *Bioinformatics*, 22(9), pp 1122-1129. 2006.

[32] Zhao, L. Zaki, M J. "TRICLUSTER: An Effective Algorithm for Mining Coherent Clusters in 3D Microarray Data". *SIGMOND05*. 2005.

[33] Bhar, A. Haubrock, M. Mukhopadhyay, A. Wingender, E. "Application of a Novel Triclustering Method (delta-TRIMAX) to Mine 3D Gene Expression Data of Breast Cancer Cells". *GCB2013*. 2013.

[34] Wine Spectator. N,p.,n.d. "Tasting Note - Dow's Vintage Port 2011". Web. 1 April 2015. <http://www.winespectator.com/wine/ detail/source/search/note_id/351764>.

[35] Wine Enthusiast. N.p., n.d. Web. 29 March 2015. <http://wineenthusiast.com>.

[36] Wine Advocate. N.p., n.d. Web. 29 March 2015. <http://wineadvocate.com>.

[37] Wine Spectator. N.p., n.d. "Top 100 List". Web. 29 March 2015. <http://top100.winespectator.com/lists/>.

[38] Bernard Chen, Christopher Rhodes, Aaron Crawford, Lorri Hambuchen "Wineinformatics: Applying Data Mining on Wine Sensory Reviews Processed by the Computational Wine Wheel", 2014 IEEE Internaltioanl Workshop on Domain Driven Data Mining (DDDM 2014), proceeding pp. 142-149.