VOC METABOLOMICS: MEDICINAL APPLICATIONS AND METHODOLOGICAL IMPROVEMENTS

by

Karl Anthony Navarro

A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Chemistry

Committee:

_____	Dr. Robin D Couch, Thesis Director
_____	Dr. Barney Bishop, Committee Member
_____	Dr. John Schreifels, Committee Member
_____	Dr. John Schreifels, Department Chair
_____	Dr. Donna M. Fox, Associate Dean, Office of Student Affairs & Special Programs, College of Science
_____	Dr. Peggy Agouris, Interim Dean, College of Science
Date: 4/25/14	Spring Semester 2014 George Mason University Fairfax, VA

VOC Metabolomics: Medicinal Applications and Methodological Improvements

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

by

Karl Anthony Navarro
Bachelor of Science
Christopher Newport University, 2010

Director: Robin D Couch, Professor
Department of Chemistry and Biochemistry

Spring Semester 2014
George Mason University
Fairfax, VA

DEDICATION

I dedicate this thesis work to my dog, Chewy.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my thesis director, Dr. Robin Couch, whom has provided much guidance in every aspect of my thesis.

I would also like to thank my committee members, Dr. Barney Bishop, and Dr. John Schreifels for providing invaluable help.

I would like to thank Cynthia Clubb, Trish Ike, Sara Pittman, Quyen DeRoule, Alison Hight, Emma Dixon, Andrew Pham, Tiffani Willis, David Derby, Colleen Cook, Phung Vo, Hien Le, and Taelim Kim for analyzing the numerous samples that the ALD project required.

Thanks go out to the members of the Couch Lab for providing many laughs, memories, and stories for years to come.

Finally, I would like to show gratitude toward my encouraging friends, and loving relatives who have shown me support throughout my years at GMU.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS AND SYMBOLS

Acetaldehyde Dehydrogenase ..... ALDH
Alanine Aminotransferase ..... ALT
Alcohol Dehydrogenase ..... ADH
Alcohol Liver Disease ..... ALD
Aspartate Aminotransferase ..... AST
Body Mass Index ..... BMI
Carboxen/polydimethylsiloxane ..... CAR/PDMS
Centers for Disease Control and Prevention ..... CDC
Cytochrome P450 ..... CYP2E1
Divenylbenzene/carboxen/polydimethylsiloxane ..... DVB/CAR/PDMS
Gamma-Glutamyltransferase ..... GGT
Gas Chromatography-Mass Spectrometry ..... GC-MS
Headspace-Solid Phase Microextraction ..... h-SPME
Inflammatory Bowel Disease ..... IBD
Interleukin-1 Receptor-associated Kinase ..... IRAK
Interleukin-6 ..... IL-6
Interleukin-8 ..... IL-8
Kyoto Encyclopedia of Genes and Genomes ..... KEGG
Lipopolysaccharide ..... LPS
LPS-Binding Protein ..... LBP
Multiple-Analyte Cocktail ..... MAC
Multiple-Extraction Device ..... MED
Myeloid Differentiation Factor 88 ..... MyD88
Noise Factor ..... N_f
Non-Alcoholic Fatty Liver Disease ..... NAFLD
Nuclear Factor kappa B ..... NFκB
Partial Least Squares-Discriminant Analysis ..... PLS-DA
Pentoxifylline ..... PTX
Phosphate Buffered Saline ..... PBS
Polyacrylate ..... PA
Polydimethylsiloxane/divenylbenzene ..... PDMS/DVB
Polyethylene Glycol ..... PEG
Practical Extraction and Report Language ..... PERL
Principal Component Analysis ..... PCA
Principal Components ..... PCs
Reactive Oxygen Species ..... ROS

Short Chain Fatty Acids.....SCFA
Simultaneous Multiple h-SPME .....SIMULTI h-SPME
Solid-Phase Extraction.....SPE
Solid-Phase Microextraction..... SPME
T helper ..... Th
Toll-Like Receptor 4..... TLR4
Trichuris suis ..... *T. suis*
Trichuris trichiura ..... *T. trichiura*
Tumor Necrosis Factor .....TNF-α
Tumor Necrosis Factor Receptor-associated Factor ..... TRAF
United States Department of Agriculture .....USDA
Variables Important in the Projection ..... VIP
Volatile Organic Compounds .....VOCs

ABSTRACT

VOC METABOLOMICS: MEDICINAL APPLICATIONS AND METHODOLOGICAL IMPROVEMENTS

Karl Anthony Navarro, M.S.

George Mason University, 2014

Thesis Director: Dr. Robin D Couch

Metabolomics, the science of extracting and analyzing small molecule metabolites from a given sample matrix, has become the method of choice for investigations ranging from environmental to human-health related issues. Solid-phase microextraction (SPME) fibers can be used to extract metabolites as volatile organic compounds (VOCs) from the sample headspace. VOCs were analyzed via gas chromatography-mass spectrometry (GC-MS) and were identified using Automated Mass Spectral Deconvolution Identification System (AMDIS) software and the National Institutes of Standards and Technology (NIST) database. This methodology was applied to differentiate alcoholic patients from healthy controls in an attempt to develop a non-invasive indication of alcoholism. Fecal samples were collected from patients after home passage (defecation into a specimen bag) and via endoscopy. These samples were used to justify endoscopy collected fecal samples, establish the extraction duration (18 hours or 20 minutes) better

suited fecal VOC investigations, and discover if these metabolomics methods can be used to indicate alcoholism. Based on the results seen on the principal component analysis (PCA) plots, endoscopy collected samples are justified since home and endoscopy samples are separate. It was determined that fecal VOC extractions should be completed at 18 hours because the analytes identified to be significant in the 20 minute extractions were found to be insignificant at 18 hours, based on a fold change analysis. Since the longer SPME fiber exposure time is completed at equilibrium, there is less variability due to operator use and the variability between cohorts can be properly deduced. A diagnosis for alcoholism can be completed by analyzing home collected fecal samples based on our methodology. This can be said because the two cohorts, alcoholic and healthy, segregate on PCA and partial least squares-discriminant analysis (PLS-DA) plots, due to the differences in their VOC identities and abundances. Another study using this fecal metabolomics approach, compared samples from pigs infected with the whipworm, *Trichuris suis* (*T. suis*), to healthy controls. Disparities in the metabolites extracted from the two cohorts signify differences between the two intestinal metabolomes. Results from this study can be used to optimize treatments of irritable bowel diseases using parasites. To facilitate sample throughput, we have developed a Multiple-Extraction Device (MED) (Patent #20120264227). Coupled with a simultaneous multiple (SIMULTI) h-SPME fiber technique, this device is used to extract VOCs from a sample headspace, expediting sample analysis while maintaining strict analytical performance.

THESIS OVERVIEW

This thesis, divided into three sections, is a metabolomics study of volatile organic compounds (VOCs) from various samples. To extract these analytes, a headspace-solid phase microextraction (h-SPME) technique was used. Analysis was then completed using gas chromatography-mass spectrometry (GC-MS). All of these techniques will be discussed in the Applications section of this thesis, which is further divided into two parts. The first of which concerns alcohol consumption and the effect on fecal VOCs, while the second describes tracking VOC changes in pig feces that have been exposed to a whipworm. Although the VOC extraction methods using SPME fibers provide reliable results, there are a few limitations. These will be addressed and resolved in the Improvements portion of this thesis.

PART I - APPLICATIONS: FECAL VOCS AND ALCOHOL LIVER DISEASE

Introduction

Metabolomics is a science concerned with the collection and analysis of a comprehensive set of small molecule metabolites isolated from a sample of interest. Furthermore, these metabolites are characterized and quantified to ascertain specific information from a given metabolome. Metabolomics investigations are often designed to be targeted, focusing on specific analytes, or global, detecting all analytes in a sample. A common goal of a metabolomics investigation is the differential comparison of conditional state, such as healthy vs. disease, aerobic vs. anaerobic, and differences in food production. The typical progression of a metabolomics pipeline is seen in Figure 1.[1]

Microbiomes, from a specific area of the body, are a complex network of organisms whose genetic information can be used to distinguish healthy from disease. These microbiota help produce metabolites that pertain to a metabolome representing a unique body part.[2] Analysis of the metabolome, which consists of amino acids, sugars, organic acids, bases, or vitamins that are found in a cell, organ, or organism, has multiple applications in health related areas. These areas include diabetes, osteoarthritis, genetic disease diagnosis, oral health, among others.[3–6] In contrast to the genome and proteome, the metabolome is not directly coded into the genome. Metabolites are products of cellular and physiological metabolism from various biochemical pathways that are essential to survival.[6] Additionally, analysis of the metabolome, can be used to track the

differences between two or more cohorts. This is done to ascertain an understanding of the underlying metabolic pathways for biological mechanisms. Moreover, these metabolites could possibly be used as biomarkers, which are indicators of normal biological processes, to screen for disease in a patient.[6,7] For example, investigations have used oral rinse to study metabolites produced from fungal microbiome (mycobiome). These results could be used for therapies in the prevention and treatment of oral ailments.[5] Other microbiomes that have been studied are extracted from skin, and urogenital parts of the body.[2]

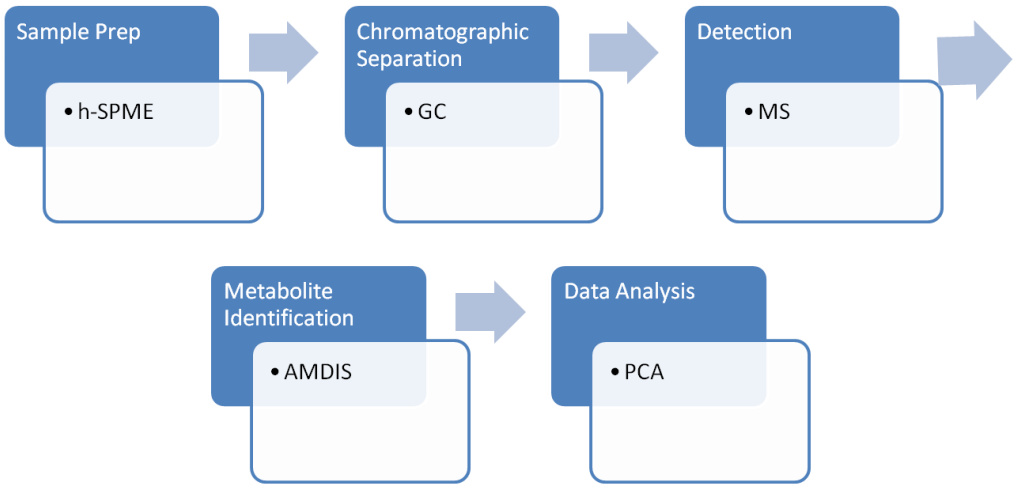


Figure 1. Metabolomics Pipeline. Examples of each phase are presented in the flow chart. Metabolites are extracted from samples using h-SPME technique. The analytes are then desorbed onto a GC column to be separated. Detection of the VOCs is executed on an MS, while identifying the metabolites is performed with the AMDIS software. Once all the metabolites and their corresponding abundances from all samples are identified, PCA can be performed to detect differences between the two cohorts.

When isolating the analytes from the sample matrix, the type of metabolite needs to be considered. Due to differences in vapor pressure, volatile organic compounds (VOCs) are found in the gaseous phase, while the non-volatile compounds remain in the sample matrix, at room temperature.[8] These differences between non-volatile compounds and VOCs require unique extraction techniques.[8,9] Recently, VOC metabolomics have become widely used in multiple research areas as a result of the advances in technology to extract and identify metabolites.[4] For example, areas concerning human health related issues,[10,11] environmental issues,[12,13] and the food sciences[4,14,15] have targeted VOCs as their analytes of choice. This is due to a recent development in extraction technique (h-SPME) as this methodology eliminates organic solvent use, preconcentrates the analytes in one step, and is inexpensive.[16]

VOCs can be collected using various complex devices such as spirometers and Tedlar bags or trapping the analytes. Likewise, the analysis methods of VOCs are numerous. These include Proton Transfer Reaction-MS, Selected-Ion Flow-Tube-MS, Ion-Mobility Spectrometry, laser spectroscopy, colorimetric sensor array, gold nanoparticles, and or Gas Chromatography-Time of Flight-MS.[17–20] In this thesis, extraction and analysis methods of VOCs were headspace-solid phase microextraction (h-SPME) coupled with gas chromatography-mass spectrometry (GC-MS).

SPME is the extraction technology developed by Janusz Pawliszyn at the University of Waterloo in the early 1990s. Its predecessor, solid phase extraction (SPE), was an upgrade from liquid-liquid extraction since analyte isolation times were decreased. Even with the reduced analyte extraction times, limitations in the analysis of a

sample were numerous when SPE was utilized. These limitations included high blank values, analytes binding to the plastic in the SPE cartridges, and product to product SPE variation.[13] SPME technology uses various polymers that are chemically fused to silica fibers. This eliminated derivatization of samples with the use of solvents and extraction times drastically diminished.[13] Originally, this equipment was used for environmental purposes; however, the applications have expanded to human health and food science related studies among others.[4,11–15]

VOCs can be collected through h-SPME where analytes, which are found in the sample headspace, bind to the fibers. Differences in polymer coatings on the fibers will isolate a variety of VOCs based on the polarity and functionality of the fiber and analyte combination. Once the VOCs have been extracted from the sample, they are desorbed in a heated inlet port of a gas chromatograph and separated in the column. Analytes are then detected by either flame ionization or mass spectrometry.[8,15,21–23] When extracted from different biological samples, VOCs can be used to examine the microbiome of that respective sample. Moreover, VOCs extracted from human fecal samples have been used to diagnose various disorders including liver disease.[24] We have shown that the use of multiple SPME fibers with differing chemistries increase the diversity of the metabolites that are extracted from a sample.[8] Therefore, this study will employ multiple SPME fibers with different polymer chemistries to accumulate the greatest number of metabolites.

Being one of the largest organs in the human body, the liver acts as a multi-faceted tool as it stores important nutrients, metabolizes proteins, and also serves as a

natural filter for toxins in the body.[25,26] After any toxin, including alcohol, is introduced into the body, the liver will metabolize it. Since the liver is the primary organ in metabolizing alcohol, it is more prone to alcohol-related injuries or diseases. Additionally, the liver can be damaged indirectly though intestinal permeability due to alcohol consumption.[27] Some experts believe that mixing as few as three alcoholic beverages and over-the-counter drugs such as acetaminophen can cause such illnesses.[25]

According to the Centers for Disease Control and Prevention (CDC) 52% of Americans age 18 and older regularly consume alcohol, which equates to at least 12 alcoholic beverages per year.[28] Furthermore, about 8.1 million Americans suffer from alcoholism, which is a physical addiction to alcohol.[29,30] These two statistics alone show that alcohol-related liver injuries affect millions in the United States.

It is speculated that people in the obese weight range have a different gut microbiome in comparison to persons in the healthy weight range.[24] Because of the differences in intestinal microbiota in the two cohorts, different metabolites would be produced. Therefore, tracking differences between healthy and obese cohorts using the VOC metabolites from fecal samples can be used to determine microbiome shifts in a non-alcoholic fatty liver disease (NAFLD).[24] However, in this study,[24] VOCs were only extracted for 20 minutes using one SPME fiber.[24,31] Figure 2 displays an extraction profile of identified metabolites from human fecal samples over time.[8] The maximum number of metabolites is not reached until 300 minutes, where equilibrium between the SPME fiber, headspace and sample matrix, is established. The extraction profile shows that the slope is the greatest at the 20 minute point, indicating that the number of

metabolites could fluctuate depending on the accuracy of the user in regards to the completion of the extraction time. As previously stated, we have determined that using multiple SPME fibers with different polymer coatings should be used to obtain a full suite of metabolites from the sample.[8] It is possible that many metabolites may not have been extracted from the sample due to a non-expansive SPME fiber chemistry in this study.[24] Therefore, the results concerning VOCs from human feces should be taken with caution because of the fluctuating metabolite numbers and the lack of diversity in SPME fibers. This thesis, however, will provide insight as to why one should be hesitant about results gathered from fecal VOCs after a 20 minute SPME fiber extraction.

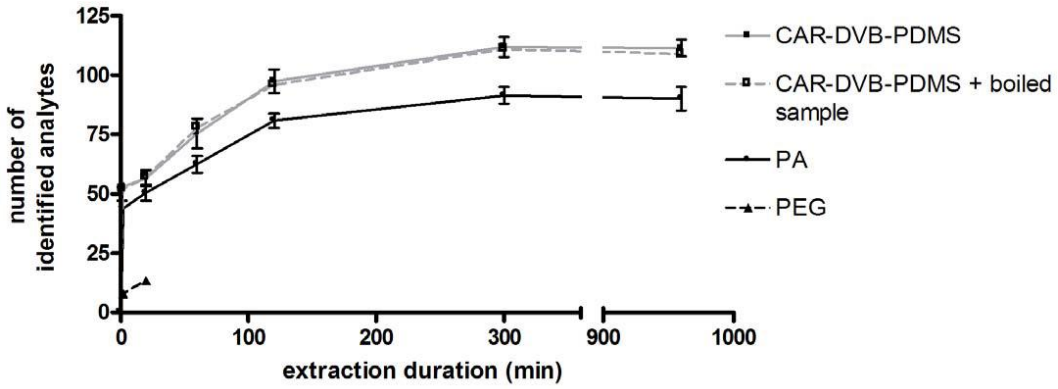


Figure 2. Number of metabolites and extraction duration using SPME. Three SPME fibers were used to extract VOCs from fecal samples. This was done to determine when VOC equilibrium is established between the phases of the sample, sample headspace, and SPME fiber.

Another form of liver disease, ALD, is a result of alcohol abuse over numerous years.[32] The progression of ALD is the development of steatosis, alcoholic hepatitis, and cirrhosis.[25,32] Heavy drinking, over the course of several days, can lead to steatosis, or a

"fatty" liver. Steatosis is the first phase of ALD and is also the most diagnosed, affecting approximately 90% of people who consume alcohol.[33] During this stage, an ample amount of fat builds up in the liver, preventing proper function of the organ. The easiest way to reduce this fat build up and reverse the condition is to abstain from alcohol consumption. Medications are available to aid people with abstaining from alcohol.[25,34]

Alcoholic hepatitis, the second stage of ALD, is caused from continued heavy drinking after the development of steatosis. In this potentially fatal stage, the liver becomes inflamed. Nausea, lack of appetite, vomiting, fever, abdominal pain and tenderness, jaundice, and mental confusion are just some symptoms that can arise from this form of hepatitis.[35] At this stage, nearly 70% of the affected individuals will develop cirrhosis; however, a full recovery may be possible if early symptoms are recognized and the individual abstains from alcohol.[25]

The ultimate stage of ALD is the cirrhosis of the liver, seen in approximately 30% of alcohol abusers. This form of the disease occurs when the healthy liver cells are replaced by scar tissue which reduces the blood flow and further inhibits the function of the organ.[25,36,37] In addition to the symptoms seen in alcoholic hepatitis, cirrhosis also leads to: skin changes, abnormal bleeding, brain and nervous system changes such as mood changes, pain and/or numbness in the legs.[25,34–36]

The percentages show that cirrhosis is a prevalent disease and is the 12th leading cause of death in the United States, although the morality rates vary among the different age groups. Commonly, the rates are lower in youth, but they increase when middle age

is reached. In fact, cirrhosis is the 4th leading cause of death in people from the ages of 45-54.[25]

There are many factors besides solely alcohol consumption that can lead to ALD. Genetics, ethnic or racial background, gender, age, socioeconomic status, and a history of family drinking disorders can all be contributing causes to ALD. For example, women are at a higher risk of developing cirrhosis. Experts believe that this is due to the mechanism in which the body metabolizes the alcohol. A woman's stomach may have less alcohol dehydrogenase (ADH), the critical enzyme that breaks down the alcohol when it first enters the body. With the decrease in ADH, the alcohol is metabolized at a slower rate and can become more toxic as it reaches the liver.[25,33] This direct injury to the liver is one of two ways the organ harmed.

Alcohol reaches the liver by travelling through the gastrointestinal (GI) tract. This thermo stable and nutrient rich stretch of the body approximates to 8 meters (25 feet) and consists of different bacteria, fungi, and archaea, known as flora.[26] Approximately 10^{14} flora, belonging to 500 different bacterial species, reside in the GI tract, demonstrating the diverse inhabitants of this organ.[38] Furthermore, anaerobic bacteria outnumber aerobic bacteria by a ratio of 100-1000:1 in this region of the body.[39] When a human consumes food or drink, it is metabolized by the intestinal flora. The body uses these metabolites, produced by the flora, in processes that sustain life. If a person consumes an alcoholic beverage, it is sent to the liver and metabolized by way of the blood flow from the GI tract.[38]

As discussed earlier, when the alcohol enters the body, damage can be done one of two ways, indirect and direct. The first method, indirect injury, occurs when an increased amount of endotoxin flows through the bloodstream to the liver, due to intestinal leakage caused by alcohol consumption.[27,32] Endotoxin is a lipopolysaccharide that is released into the intestine from Gram-negative bacteria cell walls when they are lysed. In the liver, the endotoxin reacts with Kupffer cells, which are hepatic macrophages. Macrophages are immune cells and will act to eliminate foreign objects. Kupffer cells specifically remove bacteria and unfamiliar proteins from the blood. As seen in Figure 3, when these cells are activated by the endotoxin, they release inflammatory cytokines, a regulatory protein that the host cell uses to combat infection. Products include TNF-α and the discharge of oxygen-free radicals, both of which cause damage to the liver.[27,40–43]

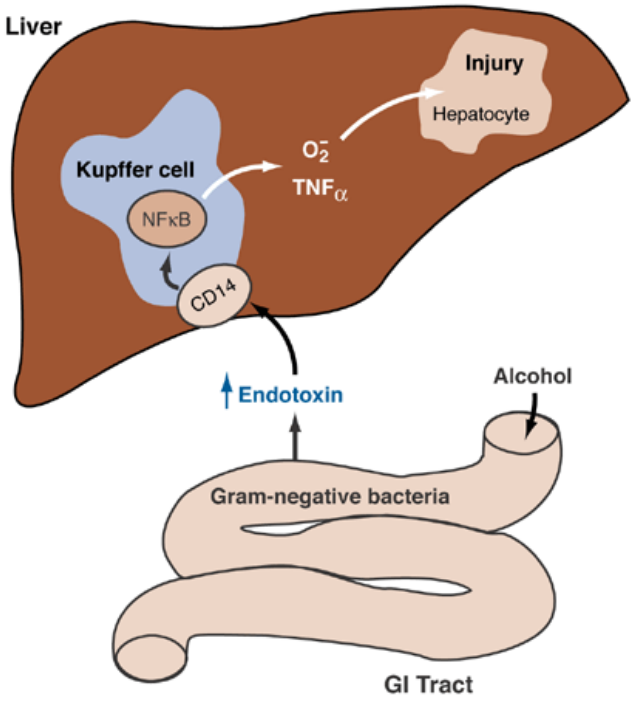


Figure 3. Diagram of Endotoxin and Kupffer cell interaction.[40] This demonstrates the flow of endotoxin leaking from the stomach to the reaction with the Kupffer cell. The cell releases TNF and damages the liver.

Endotoxin activates the Kupffer cells by binding to CD14, a receptor protein, with the help of a lipopolysaccharide (LPS)-binding protein (LBP).[44] The interaction initiates multiple biochemical reactions activating the Kupffer cell. CD14 is only on the Kupffer cell exterior and does not extend to the innards of the cell. Therefore, the endotoxin must react with Toll-like receptor 4 (TLR4), a coreceptor, to transfer the signal inside. When the signal reaches the TLR4, it is between the proteins myeloid differentiation factor 88 (MyD88), tumor necrosis factor receptor-associated factor (TRAF), and the enzyme interleukin-1 receptor-associated kinase (IRAK). The signal is then sent from the proteins to the nuclear factor kappa B (NFκB), a regulatory molecule, which is attached to the

inhibitory molecule IκBα. The NFκB molecule is inactive until it is released by the IκBα, when the signal reaches the complex. The released, activated NFκB produces superoxide through the NADPH oxidase complex and cytokines. This entire process is displayed in Figure 4.[40,42,43]

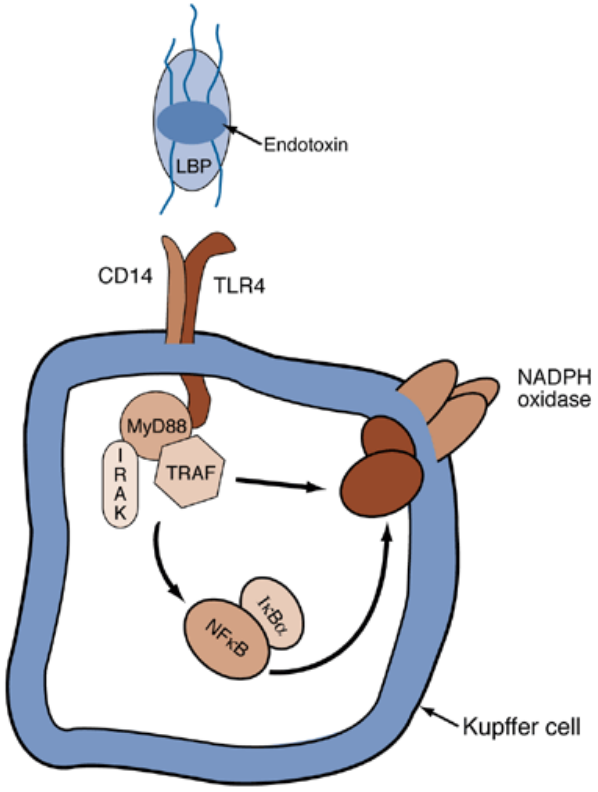


Figure 4. Depiction of endotoxin activating the Kupffer cell.[40] The endotoxin binds to the Kupffer cell via the CD14 protein with the help of LBP. The signal is transferred from the coreceptor TLR4 to the proteins MyD88 and TRAF, which activates NFκB. This generates superoxides and cytokines which damage the liver.

The second process of an alcohol induced liver injury is a direct method. This is further divided into oxidative and non-oxidative injuries. ADH and acetaldehyde

dehydrogenase (ALDH) are the two primary enzymes in this oxidative pathway. First, ADH converts alcohol into acetaldehyde, which is then oxidized into acetic acid by ALDH.[33,45] Acetic acid is excreted by the body through the urine, saliva, expired air, and sweat.[45] The byproduct of this process, acetaldehyde, is considered by many to be the key toxin in alcohol-related liver injuries as it causes cell damage, inflammation, and extracellular matrix remodeling and fibrogenesis. Moreover, acetaldehyde covalently bonds to proteins and DNA to form harmful compounds that also deter the function of the liver. Although most of the converted acetic acid is disposed by the body, residual acetate can be further converted to acetyl-coenzyme A. This enhances histone acetylation and can lead to liver inflammation because it regulates synthesis of macrophage inflammatory cytokines. These include interleukin-6 (IL-6), IL-8, and TNF-α. Metabolism of alcohol also occurs through the enzyme cytochrome P450 (CYP2E1). The liver is damaged by this metabolism because CYP2E1 converts alcohol to acetaldehyde and the reactive oxygen species (ROS), hydrogen peroxide and superoxide ions. ROS activate redox-sensitive transcription factor, NFκB, causing inflammation. The second type of direct injury to the liver occurs during non-oxidative metabolism of alcohol, performed by catalase. This peroxisomal enzyme produces fatty acid ethyl ester, a compound that causes alcoholic steatosis.[33]

Many ALD symptoms can subside if it is detected in the early stages and the patient alters his or her lifestyle and diet. In other cases, the symptoms will not be diminished, and the damage to the liver is irreversible. Smoking cigarettes and obesity worsen the condition and decrease the chances for a recovery from ALD. Smoking

cigarettes often coincides with drinking alcohol, which can accelerate liver disease in smokers.[34] A person with an obese body mass index (BMI) has a risk factor that lends to the development of ALD. On the other hand, malnutrition is often associated with ALD, especially those with diets lacking specific nutrients, zinc and folate, for example.[25,34] Therefore, reducing the amount of cigarettes a person smokes, lowering a patient's BMI, and increasing nutritional supplements will retard the development of ALD.[25,34] Currently, these are the only methods to treat a patient with ALD since there are no medicines that are FDA-approved for this ailment. There are, however, "off-label" drugs, such as pentoxifylline (PTX) and corticosteroids that are theorized to treat the disease.[25,34] PTX is believed to inhibit tumor necrosis factor (TNF-α) synthesis, a pro-inflammatory cytokine that damages the liver.[27,40,46] Corticosteroids may decrease the immune response and pro-inflammatory cytokine response in the liver.[34]

To monitor the liver function when the organ is suspected to be damaged, physicians can test a patient's blood count, liver function, or perform a biopsy of the liver. Liver function tests are based on the levels of three liver enzymes found in the blood: gamma-glutamyltransferase (GGT), aspartate aminotransferase (AST), and alanine aminotransferase (ALT). GGT transports amino acids or peptides through outer membranes into liver cells.[47] Primarily found in liver and heart cells, AST catalyzes the reaction that produces glutamate from aspartate and α-ketoglutarate.[48] ALT aids in the production of glutamate and pyruvate from alanine and α-ketoglutarate.[49] These enzyme levels are tabulated because they are released in the bloodstream by the liver when the organ is injured.[50–52] A patient is inclined to have ALD if the AST levels are doubled that

of ALT. Heightened GGT levels are also indicative of the illness.[25,35] However, these tests are often used to monitor vitals of other organs as well, including the kidneys and pancreas.[53] Therefore, results of these enzyme levels may be a false positive for ALD.

Currently, liver biopsies are the only means of diagnosing a patient with ALD. This procedure is invasive, painful, and expensive. The purpose of this study is to determine a non-invasive diagnostic approach for indicating alcoholism. The goal of this research is to perform a metabolomics investigation (via h-SPME and GC-MS) to determine specific biomarkers that may lead to ALD. The results of the home vs. endoscopy study have been published in PLoS ONE.[10]

Materials and Methods: ALD Metabolomics Investigation Protocol

The following steps were followed to analyze fecal VOCs from healthy and alcoholic patients.

Acquiring Fecal Samples

Fecal samples were obtained from each patient via two methods. The first method involved the patient defecating into a plastic specimen bag, sealing it, and taking it to our collaborators at Rush University. It was then frozen in liquid nitrogen and sent, overnight, in dry ice for analysis. GasPacks were added to the bags to create anaerobic environments before the samples were shipped, limiting the oxygen supply used by aerobic bacteria for metabolic processes. This was an attempt to stabilize the number of metabolites in the sample. These were considered the "home" collected samples. The second way in which samples were obtained from patients was through an endoscopy, performed at Rush University. Collected samples were placed in small vials, immediately frozen in liquid

nitrogen, and shipped overnight for analysis. These samples were known as the "endoscopy" collected samples.

The biggest difference between the two collection methods is the type of metabolism the bacteria undergo. The home collected samples were exposed to oxygen; therefore, the metabolites extracted from these samples will be products of the bacteria that have undergone aerobic respiration. On the other hand, the endoscopy collected samples were exposed to small amounts of oxygen. Thus, the metabolites found in these fecal samples are products of anaerobic metabolism.

Dispensing Fecal Samples

While still frozen, approximately 0.2 g of feces was added to vials in a sterile biosafety cabinet environment. These samples were snap frozen in liquid nitrogen and stored at -80°C until they were ready to be analyzed.

Solid-phase Microextraction Fibers

SPME fibers were used to extract VOCs from the fecal samples. Figure 5 demonstrates the operation of the fiber. A 1-2 cm long fused silica fiber, coated with a polymer, is housed inside of a needle. This needle also acts as a method to pierce the septum, which covers and protects the sample.

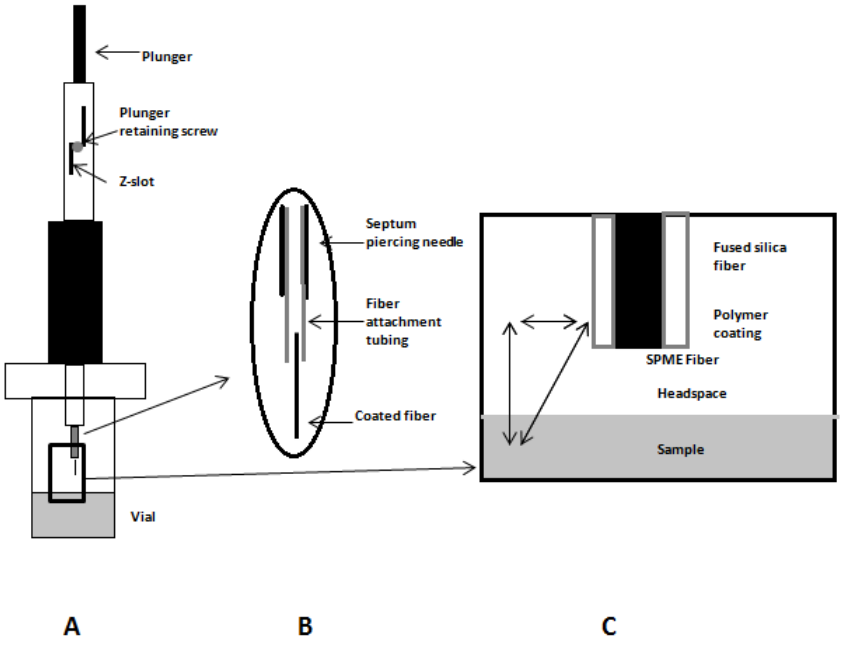


Figure 5. SPME Fiber Diagram. A) SPME fiber being exposed to sample headspace. B) Enlarged view of polymer coated fiber, and needle. C) Partitioning of VOCs between sample matrix, headspace, and SPME fiber.

Once the septum has been pierced, the SPME fiber can be exposed to the sample by plunging the fiber and locking it in place using the z-slot. Analytes can then adsorb or be absorbed by the fiber. The polyacrylate (PA) fiber acts as an absorbent, meaning the VOCs are extracted by partitioning and analytes do not compete for binding sites. On the other hand, the divenylbenzene/carboxen/polydimethylsiloxane (DVB/CAR/PDMS) and the carboxen/polydimethylsiloxane (CAR/PDMS) fiber coatings are adsorbents. This means that VOCs chemically react with the fiber or they are trapped on the fiber. Analytes have to compete for different binding sites.[13,54,55]

Analytes travel through multiple phases until an equilibrium is reached. The three-phase equilibrium consists of the SPME fiber, the headspace and sample matrix. As

previously stated, VOCs bind to the fiber based on the "like dissolves like" theory. Hence, polar analytes will bind to fibers that are coated in polar polymers whereas non-polar analytes will bind to fibers coated in non-polar polymers. Table 1 displays the polarities of each of the SPME fibers.[55]

Table 1. SPME Fibers with their corresponding polarities. Bipolar fibers are coated both polar and non-polar polymers.

SPME Fiber	Polarity
65 µm DVB-PDMS	bipolar
85 µm PA	polar
75 µm CAR-PDMS	bipolar
85 µm CAR-PDMS	bipolar
100 µm PDMS	non-polar
7 µm PDMS	non-polar
60 µm PEG	polar
50/30 µm CAR-DVB-PDMS	bipolar

Headspace Solid-phase Microextraction Procedure

The SPME fibers were preconditioned in a GC inlet based on the specifications provided by the manufacturer. Preconditioning durations and temperatures are shown in Table 2. Samples were first heated to 60°C for 30 minutes. The entire SPME fiber assembly was manually positioned above the feces to extract the VOCs for 18 hours. Fibers that were used in this experiment were PA 85 µm, DVB/CAR/PDMS 50/30 µm, and CAR/PDMS 75 µm. The SPME fiber was then inserted into a heated GC inlet port and the metabolites were desorbed onto the column. Since the stationary phase was non-

polar, the polar metabolites eluted first, while non-polar metabolites eluted towards the end of the analysis.

Table 2. GC Inlet temperatures and precondition times per fiber type.[8]

SPME Fiber	Temperature (°C)	Duration (min)
65 µm DVB-PDMS	250	30
85 µm PA	280	60
75 µm CAR-PDMS	300	60
85 µm CAR-PDMS	280	60
100 µm PDMS	250	30
7 µm PDMS	320	60
60 µm PEG	240	30
50/30 µm CAR-DVB-PDMS	270	60

Instruments

Samples were analyzed using an Agilent 7890A GC and 5975C inert XL mass selective detector (MSD) with triple axis detector (Agilent, Palo Alto, CA). The GC-MS was equipped with a DB5-MS capillary column (Agilent), 30 m in length, 0.25 mm ID, and 0.25 mm film thickness, and a 0.75 mm ID SPME injection port liner operated in splitless mode at varying inlet temperatures, dependent upon the SPME fiber used. In splitless mode, all of the analytes proceed to the GC column since the purge vent remains closed during analyte desorption. This option was chosen to increase sensitivity and to allow the greatest amount of analytes to be separated and identified. A split injection is the other setting where some of the sample continues onto the column while the remnants

go to waste. If the sample is too concentrated to be loaded onto the column, this type of injection should be used as a method of dilution.[56]

GC-MS Conditions

For the GC-MS, helium carrier gas was set to 1.17 mL/min flow rate and the GC oven was held at an initial temperature of 35°C for 1 min, ramped to 80°C at 3°C/min, then to 120°C at 10°C /min, and finally to 260°C at 40°C /min. The final temperature of 260°C was held for 1.5 min. The total run time for the analysis method was 25.0 min. The Agilent 5975C MSD was scanned from 30 to 550 amu at a rate of 2.81 scans/s.

AMDIS and NIST Analyses

Compounds were identified using the National Institute of Standards and Technology (NIST, Washington, DC) Automated Mass Spectral Deconvolution and Identification System (AMDIS, ver 2.69) software and mass spectral library (NIST08). The AMDIS analysis of chromatograms can be divided into four steps: 1) noise analysis, 2) component perception, 3) spectral "deconvolution," and 4) compound ID. The initial step determines the noise that could detract from correctly identifying a peak in the chromatogram. For the selected segment to be considered as noise, the number of times the signal crosses over the mean value must be greater than half the number of points in a given segment of the chromatogram. If this occurs in the segment, the median deviation from the mean within this segment is determined. This deviation, which is the average random deviation of the segment, is then divided by the square root of the mean abundance of the segment. This calculation is the noise factor (N_f) and is displayed in Equation 1.

Equation 1. Noise factor calculation

$$N_f = \frac{average\ random\ deviation}{\sqrt{average\ signal\ in\ segment}}$$

In the component perception step, peaks are selected and marked for potential compound identification. Peak heights of potential ion chromatogram peaks need to be larger than four noise units to be classified as a peak. A noise unit is the square root of the signal multiplied by median N_f values of the entire chromatogram. The spectral "deconvolution" part of the process is where the noise and any other neighboring peaks are subtracted from the peak signal selected for identification. AMDIS will remove ions that have different peak shapes and retention times. The remaining ions are then grouped as a component based on similarities in peak shape and retention time. Spectrometry purists say that this step is truly known as peak fitting. Lastly, the compound is identified based on how well the spectra matches a known spectra in the NIST database.[57–60] In this analysis, the spectra in question must match the NIST database by a score of >85% to be identified as a compound. Changing this value affects the stringency at which analytes are identified. Lower cutoffs will increase the number of metabolites in the matrix because multiple isomers of the same compound will be matched to one peak. Conversely, if this match score is increased, there are too few metabolites identified. Analysis and conclusions downstream could be affected. The 85% score was empirically derived. When this value was set at an 80% score was produced a list containing multiple

calls of the same metabolite. Conversely, a cutoff score of 90% failed to identify analytes that matched compounds in the database with 87% and 88% scores.

Once AMDIS is opened, select a chromatogram file to be analyzed. Under "Analyze," click "Analyze GC/MS Data" and a box found shown in Figure 6 appears. The "Simple" type of analysis indicates that only mass spectra are used to determine a matched target.[57]

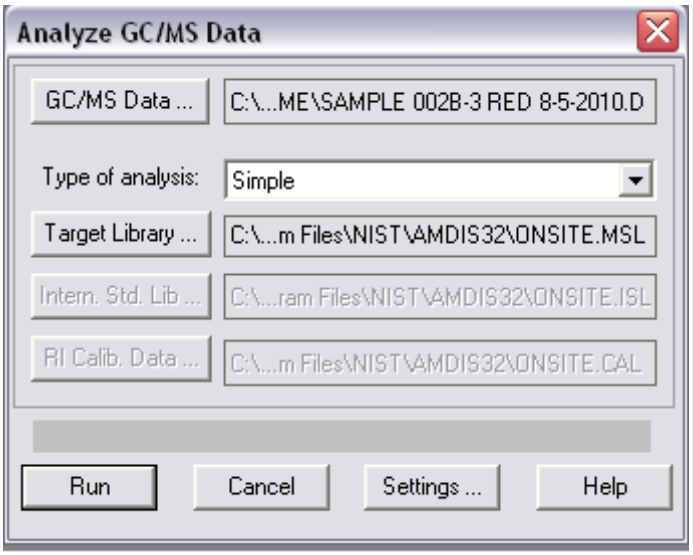


Figure 6. Initial parameters of AMDIS analysis.

Figure 7 shows a series of settings that need to be selected in order to properly analyze the chromatogram. The settings need to be saved each time a change is made. Figure 7A shows the identification tab where the user can set limits on the results that will be displayed. The minimum match factor of 60 is a default and indicates the lowest

value at which a target will be given an identity from the selected library. A target in the chromatogram that has perfect match to a compound in the library will be given a score of 100. The “Multiple identification per compound” box is selected; a target can now be identified and listed as multiple isomers. Figure 7B allows the user to set the instrument and data file types. Both of the boxes for the m/z range are auto selected indicating that they are defaulted to the minimum and maximum m/z values found in the data. With the threshold option of “Off” selected, the signal threshold value used in the chromatogram analysis is from the data file. Selecting a “High to Low” scan direction means that the peaks from the data file are acquired from higher to lower. Data file format is set to Agilent Files while the Instrument type is Quadruple. Figure 7C, the deconvolution tab, allows the user to set the parameters for the deconvolution process. The component width, defaulted at 12, is the number of scans AMDIS will perform on a peak before a spectrum is extracted. All of the parameters in the bottom section of the deconvolution tab are a default selection. Adjacent peak subtraction set to “One” means that the computer will subtract out the ion that would interfere most with the peak in question. The resolution parameter determines the separation between peaks. The sensitivity parameter will tell the computer to differentiate between peak width and noise. The shape requirement will determine how a peak shape must look like to be considered for analysis. Figure 7D, the Libraries tab, allows the user to select a library to be used as a comparison for the chromatogram targets. The QA/QC (Quality Control) tab, displayed in Figure 7E, allows the user to check whether or not the instrument is properly

functioning. For this analysis, both parameters are not selected. The Scan Sets tab, shown in Figure 7F, allows the user to gather data from set m/z ranges.[57,61]

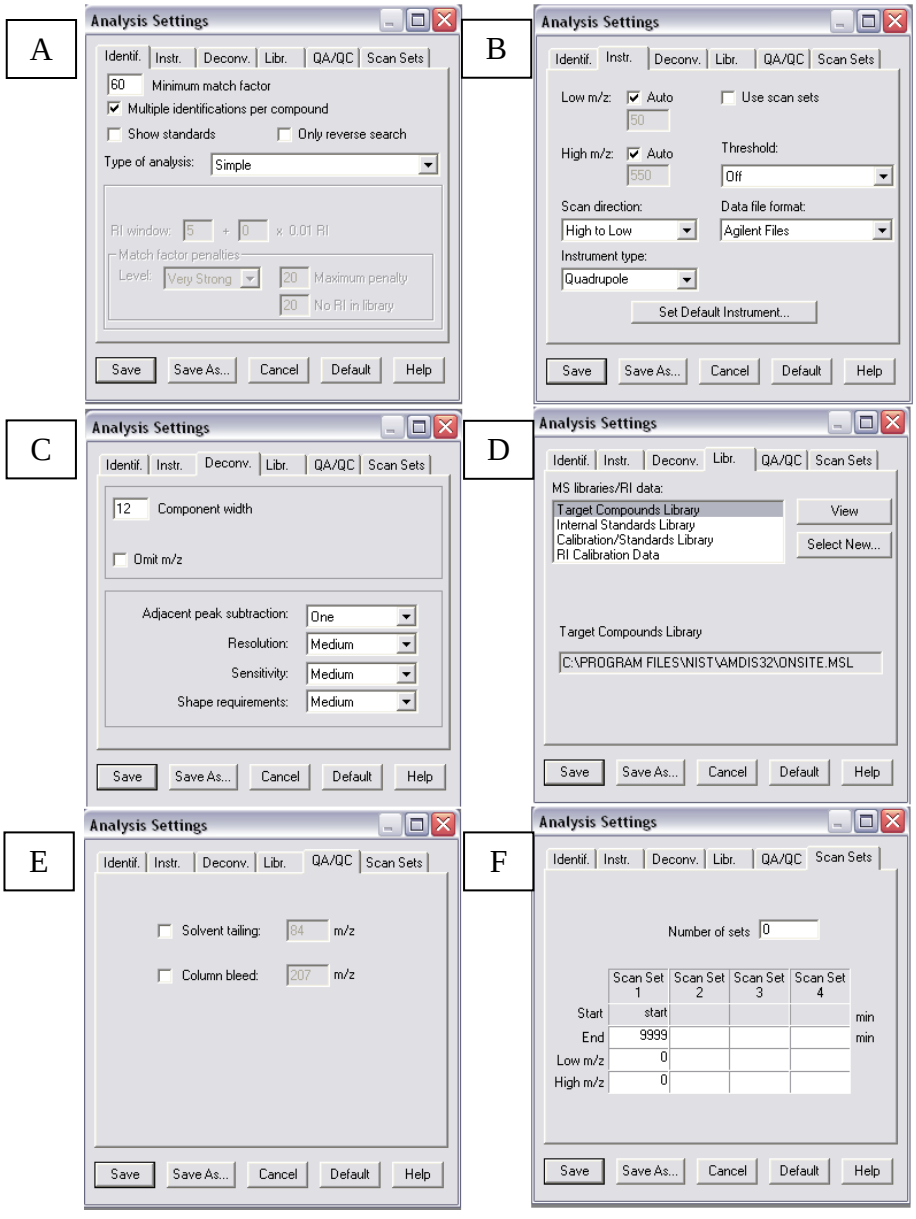
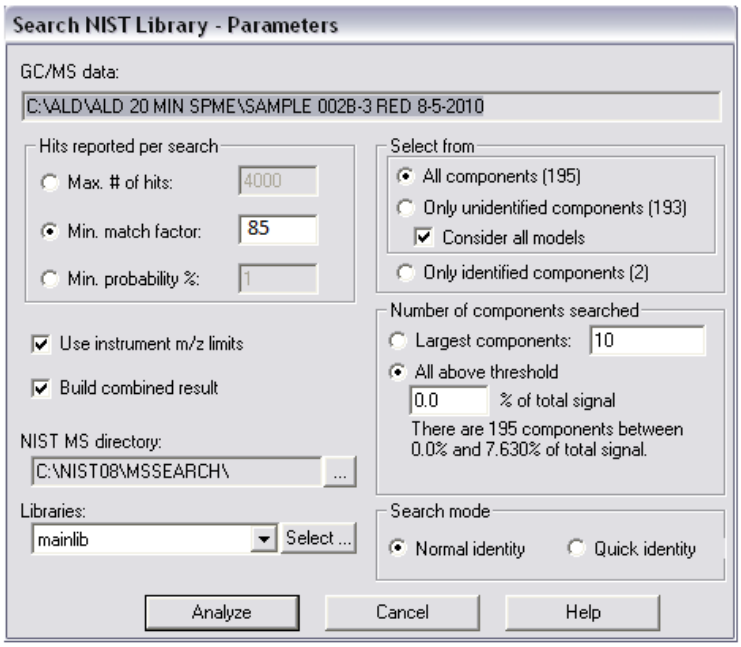


Figure 7. Analysis settings for AMDIS chromatogram analysis. A) Identification tab B) Instrument tab C) Deconvolution tab D) Library tab E) QA/QC tab F) Scan sets tab

Figure 8 shows the parameters for the NIST software used in the compound identification process. In Figure 8A, an 85 match factor is chosen. Again, this setting dictates that the compound in question must match a compound the NIST database by 85%. Figure 8B allows the user to save the results of the NIST analysis to as a specific file.

A



B

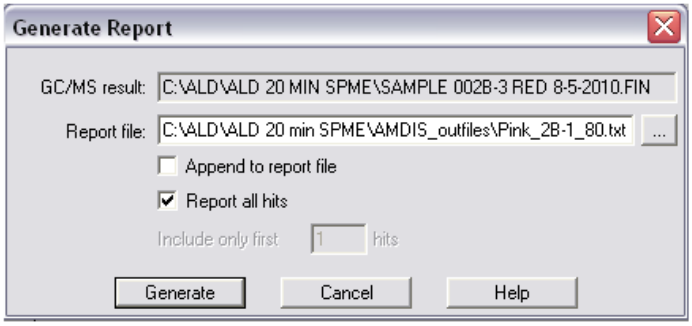


Figure 8. NIST analysis parameters. A) NIST match factor parameters B) Generating report as a text file

Data Filtering

Once all the targets have been identified the corresponding peak intensities need to be culled such that all replicate metabolites are removed from the data set. Multiple calls of the same metabolite will greatly affect the statistical analysis downstream.

Additionally, the data set will then be focused around a core set of metabolites. All of the following steps can be completed in one Microsoft Excel workbook.

First, set up the data table such that each row is a new sample and each column is a new metabolite. Next, divide the data set into the two cohorts, i.e. home v. endoscopy, healthy v. alcoholic, etc. For each metabolite, determine the mean, median, (mean-median)/median, count, and overall count. These six categories are placed beneath each metabolite. The count value displays the frequency of the metabolite in the total number of samples. Use the Excel equation, =COUNTIF(base peak range, ">0") for this cell. Furthermore, the overall count keeps track of the frequency of a metabolite in both cohorts. This is the sum of the "count" value of each metabolite in both cohorts. In a new sheet, create a histogram using the frequency, in percentage form, of the metabolite found in all samples. The histogram will help determine the "one off cutoff" score for a metabolite to be included in the statistical analysis. In essence, "one offs" are those metabolites that are anomalies which have no statistical weight since they do not appear in samples at a certain percentage. Consequently, this accounts for the dietary variability from patient to patient. Based on the shape of the histogram, the lower percentages are outliers and mask the statistical relevance that the more ubiquitous metabolites hold. Therefore, a metabolite must reach a "one off cutoff" score of 21% to be included in the analysis. Table 3 describes the frequency of a metabolite in all samples to reach the 21% one off cutoff score.

Table 3. Frequency of metabolites in each sample to reach 21% one off cutoff score

File type	Collection method	Extraction duration	Total number of samples	Frequency of metabolite in each sample
Healthy v alcoholic	Home	18 hr	47	10
Healthy v alcoholic	Endo	18 hr	34	8
Healthy v alcoholic	Home	20 min	45	10
Healthy v alcoholic	Endo	20 min	26	6
Healthy	Home and Endo	18 hr	32	7
Healthy	Home and Endo	20 min	26	6

After the metabolites that do not meet one off cutoff criteria are removed from the data, empty cells are filled with the median value of that specific metabolite in that specific cohort.[62] The median value was chosen to replace empty cells because this gives the best representation of the data if the standard deviation is large. Equation 2 is then calculated for each metabolite in each cohort.

Equation 2. Data filtering in metabolite matrix

$$\frac{(mean - median)}{median} > 1.5$$

If a metabolite has a (mean-median)/median value > 1.5, then those metabolites are removed. The value of 1.5 represents a 2.5 fold difference in the mean to median value. Any value greater than 1.5 is considered an outlier. Use the conditional formatting function in Excel to highlight these high values, and then sort the matrix by cell color as these steps aid in visualizing which metabolites to remove. To further cull the metabolite list, replicate metabolite identities are removed. In the next step, the sums of each metabolite base peak are compared. If a set of metabolites return identical base peak sums, then one metabolite is retained and the remaining are deleted. Use the conditional formatting function in Excel to find and highlight duplicate values of the metabolite base peak sums. Sort the matrix by cell color and remove duplicate metabolites. The last step in removing replicates of metabolites is to use the conditional formatting function in Excel to find and highlight identical values in the first three samples of each metabolite. Select the first sample and use the conditional formatting function in Excel to find and highlight identical values of the base peaks. Complete the same steps for the next two samples and be sure to choose different colors for each row when highlighting the values. Sort the matrix by color and then remove replicate metabolites. Finally, Z-score normalization of the data was used on the matrix. Equation 3. Z-score normalization shows the formula used for this step.[63] X is the value of any cell in the matrix. The mean and the standard deviation values used in this equation are of a specific metabolite corresponding to the cell being normalized.

Equation 3. Z-score normalization. The value of x is any cell in the given matrix while the mean and standard deviation are calculated from that specific metabolite.

$$Z - score = \frac{x - mean}{standard\ deviation}$$

Statistical Analysis: Principal Component Analysis

Once the data set has been properly filtered and normalized, statistical analysis of the data using a PCA (Principal Component Analysis) plot was completed. This was performed using the Excel add-in, XLSTAT (ver 2012.6.02). The PCA plot is a multivariate analysis that identifies patterns in large data sets in addition to reducing the number of factors that dictate how the data is presented.[64] PCA plots present the variance, or the distance from the average, for each sample in the data.[63] This means that plotted points that are far away from one another represent samples that have variables with differing values. Conversely, data points that are plotted closer in proximity on the graph represent samples that have similar values. The position of each data point on a PCA plot is determined from the correlation between the factor (metabolite) values in each sample in relation to all the other samples. Therefore, one can look at the plot to see the correlation between the samples.

The axis values on PCA plots are ascertained from the principal components of the correlation matrix, a derivation of the original matrix. Principal components (PCs), which are equal to the number of factors that are in the matrix, display how much the correlation matrix has been transformed from the original matrix. Correlation matrices show correlation values that compare how two factors change together. PCs are

ultimately determined from the eigenvalues of the eigenvectors of the correlation matrix. Each of the axes must be unrelated and orthogonal. Eigenvectors are the axes for the new matrix, essentially a least squares fitting of the plotted data, while the eigenvalue is the amount the new matrix has been transformed.[65,66] The first few PCs account for the most variance in the data set and hold the most meaning for interpreting the data. Therefore these axes are most commonly used for the plots. PC1 accounts for the highest possible variation in the data, while PC2 accounts for the second most variation in the data. This is done until 100% of the variation in the data is explained.[65,67]

With the metabolite matrix open in Excel, click on the XLSTAT button under the "Add-Ins" tab. Then choose the "Analyzing Data" button followed by "Principal Component Analysis" option. In the General tab, perform the following operations: select all metabolites and corresponding base peaks for the "Observed/variables table;" for the "Data format" choices, select "Observations/variables;" select "Pearson (n)" for "PCA type;" check the box that reads "Variable labels;" for the "Observation labels" choose all the samples in the matrix. Under the "Options" tab make sure to limit the number of factors to five. In the "Charts" tab, select all boxes except for "Colored labels" and "Filter," while choosing the "Correlation biplot" for the "Type of biplot." The next box allows the user to choose which factors of the analysis to use. Once this is completed, the results for the PCA are displayed in a new sheet.

Statistical Analysis: Fold Change

Fold change analysis was performed using the online tool MetaboAnalyst2.0. The user may upload data and perform a multitude of statistical tests. The data must be saved

in a CSV format. All of the files that were uploaded were of the "Peak intensity table" data type. Since the data sets were already normalized and filtered prior to statistical analysis, the "Missing value" process is skipped, and "None" is selected on the "Data filter." On the "Data normalization" page, select "None" for each of the options. To view the fold change results, click on the "Fold change" link under "Statistics" on the left side of the screen. The percentage of metabolites that were above each fold change threshold was calculated for each file type.[68]

Statistical Analysis: Partial Least Squares-Discriminant Analysis

To determine which metabolites contributed the most to the alcoholic v. healthy data set, a partial least squares-discriminant analysis (PLS-DA) was executed using the Excel add-on, XLSTAT (ver 2012.6.02). This two part analysis consists of a PLS regression and a discriminant analysis. PLS is used to determine the maximum amount of covariance between the samples, while DA creates predictive models when the data set contains multiple collinear variables. Collinear variables are those that are highly correlated to one another.[69–73] In the case of this study, samples can be predicted to be part of the alcoholic or healthy cohorts. PCA computes vectors with the largest variance, while the PLS-DA vectors are computed to best differentiate between the two classes. This is done by rotating the PCA axes to obtain maximum sample covariance. In turn, this creates the greatest separation between the two cohorts.[73,74] Additionally, the newly rotated axis must be correlated to the data in the Y-space (cohorts), to ensure the best chance at a future prediction. This is done by monitoring the magnitude of a sample in the X-space (variables) and tracking the corresponding value in the Y-Space.[75] Similar to

PCA plots, samples on PLS-DA plots are positioned such that those with like metabolites and metabolite levels are close in proximity.[76] Based on this plot, variables important in the projection (VIP) can be determined from the calculation shown in Equation 4. Essentially, this calculation is the weighted sum of squares of the PLS data, where the sum of squares measures the variation within a data set. These are metabolites that contribute the most to the separation in the data set.[77–79]

Equation 4. Variables Important in the Projection. The number of predictors is equal to the number of variables in the data set.[79]

$$VIP = \sqrt{(weighted\ average) * (\#\ of\ predictors)}$$

With the metabolite matrix open in Excel, click on the XLSTAT button under the "Add-Ins" tab. Then choose the "Modeling Data" button followed by "PLS Regression" option. In the "General" tab, select the column containing the data labels, either "alcoholic" or "healthy," for the "Y / Dependant variables." The metabolite identities and corresponding values are selected for the "X / Independent variables." Be sure to select "PLS-DA" under "Method," as well as selecting "Sheet" and checking the "Variable labels" box. The sample names fall under the "Observation labels" option. In the "Options" tab, make sure that "Automatic" is selected. Lastly, in the "Charts" tab, select the "Color labels" option to produce plots that are easier to visualize.

Data Consolidation Using PERL Scripts

Practical Extraction and Report Language (PERL) code allows the computer to read, gather information, and parse text files.[80] The results of the algorithm are then printed for the user to interpret. PERL scripts were written to group all the metabolites in a matrix into 13 chemical classes. The classes grouped were: alcohols; acids; aldehydes; alkanes; alkenes; alkynes; thiols; sulfides; indoles; ketones; nitriles; amines; and others. The different metabolites were sorted into its specific chemical class based on the key word shown in Table 4. The corresponding peak abundances were summed and the cumulative total was tracked by sample. Biplots were then made from the data in the output matrix to see if any specific chemical class could be used to differentiate the two cohorts. This type of PCA plots illustrate how the samples are plotted based on the magnitude of a specific metabolite as they contain both the observations (samples) and variables (metabolites) on the same graph.[81]

Table 4. Sorting Metabolites by Chemical Class

Chemical Class	Must contain to be sorted
Alcohol	-ol
Aldehyde	-al
Acid	acid
Acid	ester
Acid	-ate
Ketone	-one
Alkane	-ane
Alkene	-ene
Alkyne	-yne
Thiol	thiol
Indole	-ole
Sulfide	sulfide

Amine	amine
Other	Remaining metabolites

The code for the algorithm is shown in Appendix A. The input file must be saved in a tab delimited format. The matrix should be arranged such that each column is a new sample while each row is a new metabolite. Lines 11-14 are used to open the file to be parsed after the user input the selection. Each of the chemical classes and counts for the summed values are initialized, shown in lines 19-48. Lines 62-75 is where the file is read, parsed, and closed. Line 66 explains that the original matrix is to be divided by each tab and store that information into array "row," denoted by "@row." The formation of the 2D matrix occurs in line 67 using an array of arrays.

The output file is printed in a tab delimited format and has each of the samples in a new column. Each position in the output file must be initialized as well, which is displayed in lines 80-802. Therefore, the number of samples in the infile matrix must match the number of rows in the output file matrix. Line 83 initializes the position in the new matrix, $out_array, while $columns determines the column where the pertinent information is gathered. The numbers indicates the row and column positions in the matrix, respectively. Therefore, line 83 states that the first row in column 3 has been initialized.

The number of samples for each file is shown in Table 3. Thus, each Perl script must be modified to accommodate each infile depending on the number of samples. This includes initializing the columns in the output matrix as well as the number of times the

algorithm sorts the metabolites into chemical classes. The 13 chemical classes are in a new row, very similar to the original matrix in the input file. In between the samples and the chemical classes, a "Label" row indicates the type of sample. Choices here are "home," "endo," "alcoholic," and "healthy."

"For loops" are employed in lines 803-4118 to sort each metabolite and corresponding value into one of the 13 chemical classes. Every sample and its metabolites must be parsed and therefore the number of "for loops" must match the number of samples in the matrix. Line 904 states that if the second sample in the matrix has a metabolite that is part of the alcohols, then its corresponding value will be sorted into the third row, third column in the new matrix. If this metabolite does not match the specifications of the regular expression to sort alcohol, then it will try to be matched with aldehydes. If this is not the case, then the metabolite will be tried to be matched with one of the other chemical classes or be placed in the "other" category. This process is finished with all the metabolites have been sorted by chemical class and the corresponding values have been summed together.

The code to print the out matrix is show in lines 4119-4366. Essentially, this part of the code prints out every position in the new matrix. Just like the "for loops," the print statements must be adjusted for the number of samples in the matrix. Lines 4124-4125 show how the output file is named.

Results and Discussion

This section presents and discusses the results for the ALD study. Three different SPME fibers (CAR-PDMS, PA, and DVB-CAR-PDMS) were exposed to the fecal

samples for 20 minutes or 18 hours to extract VOCs. Once all the metabolites and their values were obtained from the samples, the spreadsheets were filtered. This was to eliminate metabolites that were outliers, one-offs, and multiple calls. PCA and PLS-DA plots were generated from the data to compare sample collection methods as well as patient types.

Analyzing the Effect of Sample Collection from Healthy Patients and Determining Optimal Extraction Duration: Home vs. Endoscopy

Fecal samples were collected from each patient through endoscopies and home collections. Thus, a total of 34 fecal samples were obtained from 17 healthy volunteers. Exposing the SPME fibers to the samples for 20 minutes resulted in a combined total of 1371 metabolites extracted from the samples collected by endoscopy. When the SPME fibers were exposed to the home collected samples for the same duration, the combined total increased to 1404 metabolites. During 18 hour exposures, the SPME fibers extracted 2097 metabolites from the endoscopy class. The home cohort yielded a total of 2190 metabolites when the SPME fibers were exposed to these samples for 18 hours. These identified metabolites are categorized as alcohols, aldehydes, acids, ketones, hydrocarbons, thiols, indoles, sulfides, or amines to name a few. A sample spreadsheet consisting of the samples, filtered metabolites, and corresponding values is presented in Appendix B.

The metabolites extracted from home and endoscopy collected fecal samples were compared. Figure 9 displays the PCA plot of an 18 hour fecal extraction comparing collection methods. Both the endoscopy and home collected samples cluster amongst themselves. At the same time, endoscopy and home cohorts are separate. This

segregation of the two cohorts signifies that metabolite identities or their corresponding levels are different from one another. PC1 shows the separation between the two collection methods. On the other hand, PC2 displays the intra-cohort variance, or the distance from the mean between samples within the same class.

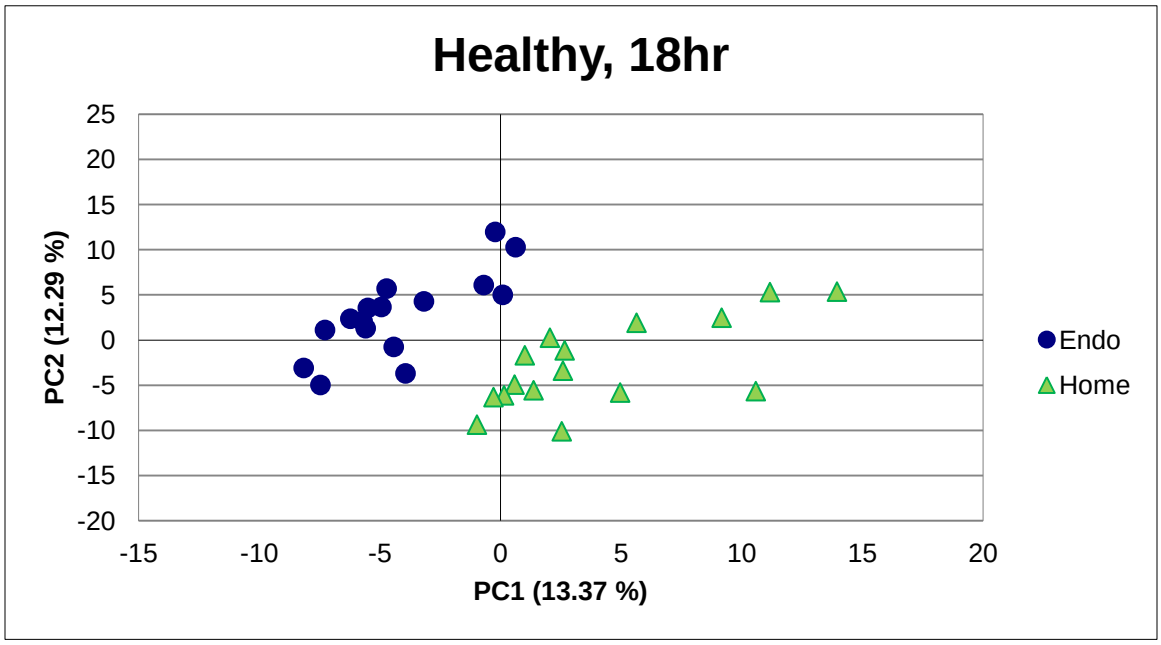


Figure 9. Collection method PCA plot – 18 hour extraction. This PCA plot is an analysis of the two types of collection methods in this study, endoscopy (endo) and home collection given by healthy volunteers. SPME fibers were exposed to the fecal samples for 18 hours. The two cohorts do not cluster amongst each other due to metabolite variation in the data. This indicates that there are differences in the two collection techniques.

Biplots (an example is seen in Figure 10) were produced from matrices after all the metabolites were grouped by chemical class and their corresponding peak values were summed. Metabolites were grouped by chemical classes to ascertain metabolite resolution when plotted. Additionally, these biplots can be used to determine if a certain

class dictates how samples are plotted. On the biplot, the variables (chemical classes) are plotted in red with vectors attached, while samples are plain markers. The length of the vector describes the variance of the variable such that those with longer lines contribute more variance to the data set. The angle of the lines between variables indicates their correlation. Angles closer to 90 or 270° have weaker correlations. Conversely, variables that are strongly related, will be at angles closer to 0 or 180°, which equate to a correlation of 1 or -1, respectively.[82] Figure 10 portrays the biplot of the 18 hour metabolite extraction. In this biplot, showing both the individual samples and variables, there is no real separation between the cohorts meaning that the two collection methods are similar with respect to the chemical classes isolated from the feces. Figure 9 is only a PCA and not a biplot because this data set results in a biplot containing a cluster of the metabolites due to the large amount of variables. This bunching of metabolites makes it difficult to decipher individual analytes.

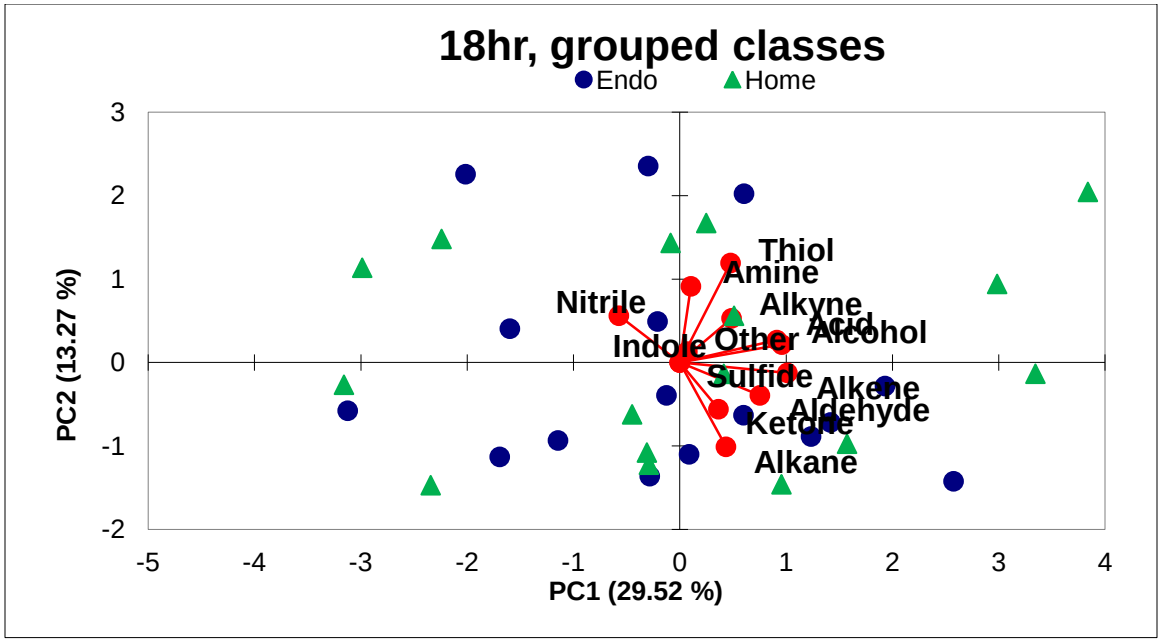


Figure 10. Collection method biplot – 18 hour extraction. This biplot, showing both the samples (blue and green markers) as well as the variables (red markers with attached vectors) is the analysis of the two collection methods, endoscopy (endo) and home collection. Similar to Figure 9, SPME fibers were exposed to the samples for 18 hours. An algorithm grouped each variable into one of thirteen chemical classes. This was to determine if a particular chemical class dictates the manner in which samples are plotted. Opposite to the PCA plot seen in Figure 9, the two cohorts do not show clear separation. This indicates that the two collection methods are similar when the metabolites are grouped into 1 of 13 chemical classes.

Figure 11 and Figure 12 show the statistical analysis of the 20 minute data set comparing collection methods, where the former displays the PCA plot and the latter exhibits the biplot. These 20 minute extractions were performed to verify that the number of metabolites identified would be less than the number of metabolites identified from the 18 hour extractions. This is predicted because the 20 minute time point on the extraction profile (see Figure 2) is located in the beginning stages as opposed to the 18 hour extractions, which is located after the VOCs have reached equilibrium. The 18 hour extractions yielded more metabolite identities (2097 endoscopy identified metabolites

and 2190 home identified metabolites) than the 20 minute extractions (1371 endoscopy identified metabolites and 1404 home identified metabolites). Similar to the 18 hour extraction data seen in Figure 9, the PCA results seen in Figure 11 show that the cohort similar samples cluster among each other, but the two cohorts clearly segregate. This separation between the cohorts is caused by differences in the metabolite identities or values. PC1 distinguishes the two collection techniques while PC2 exposes the variation within each cohort.

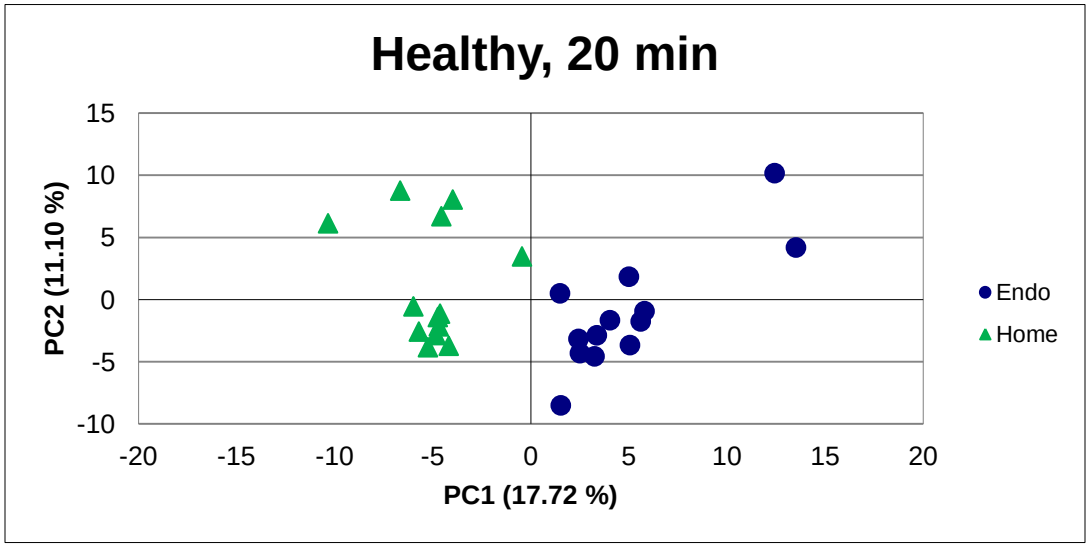


Figure 11. Collection method PCA plot – 20 minute extraction. Samples from healthy patients were analyzed and the results are displayed in this PCA plot. SPME fibers were exposed to the feces, which were home and endoscopy collected, for 20 minutes. The two cohorts have clear separation due to the differences in their metabolite identities or values.

Results in Figure 12 show that there is separation between the two cohorts. Moreover, the alkanes, aldehydes, and other metabolites groupings contribute more to the home collected samples. At the same time, the positions of the endoscopy collected samples on the biplot are influenced more by metabolites that are nitriles, acids, and alcohols. PC1 discriminates between the cohort variance while PC2 exhibits the variation in the collection methods.

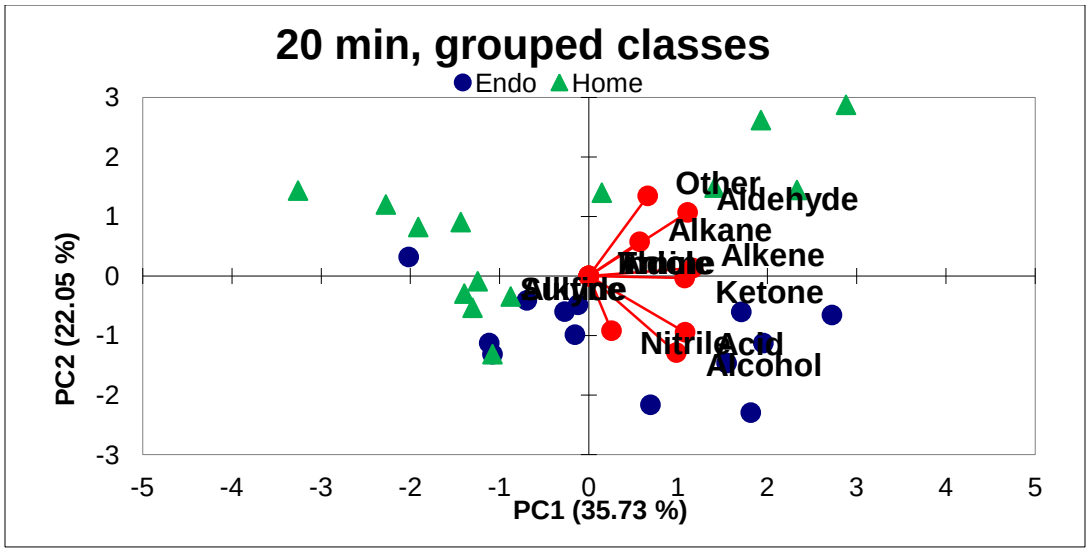


Figure 12. Collection method biplot – 20 minute extraction. Analogous to Figure 10, this biplot is the analysis of home and endoscopy collected samples donated by healthy volunteers after the metabolites have been grouped into chemical classes. Fecal VOCs were extracted by SPME fibers for 20 minutes. The identified metabolites and the corresponding values cause separation between the two cohorts. Additionally, the home collected samples are plotted based on the contributions from the alkanes, aldehydes, and other groups. The endoscopy collected samples are positioned due to the influences from are nitriles, acids, and alcohols.

In comparing the home and endoscopy collected samples, both of the resulting PCAs of the 18 hour and 20 minute extractions show that the two cohorts clearly segregate indicating differences in the metabolite identities or values. When evaluating

the 18 hour and 20 minute extractions, the 20 minute data has less sample to sample variation when compared to the 18 hour data set. This is observed in a tighter clustering of the samples on the PCA plot from the 20 minute data set in comparison to the samples on the 18 hour plot, which are more dispersed. The online tool MetaboAnalyst was used to perform a fold change analysis on the home vs. endo and healthy vs. alcoholic data sets. This was to determine which metabolites have values that greatly differ between each cohort. Fold change is calculated by dividing the mean of the specific case by the mean of the control group.[83] In terms of this study, the metabolite ratio was either home:endoscopy or alcoholic:healthy. When a calculated fold change value exceeds the set fold change threshold of 2, the metabolite level in one group is more than double that in the other. Both the 18 hour and 20 minute data sets were analyzed and the results are displayed in Table 5 and Figure 13.

Table 5. Fold Change Analysis for ALD Data

File type	Collection method	Extraction duration	Total number of metabolites	Percentage of metabolites whose FC > 2
Healthy v alcoholic	Home	18 hr	254	14%
Healthy v alcoholic	Home	20 min	199	73%
Healthy v alcoholic	Endoscopy	18 hr	215	30%
Healthy v alcoholic	Endoscopy	20 min	256	41%
Healthy	Home and Endoscopy	18 hr	234	24%
Healthy	Home and Endoscopy	20 min	195	27%

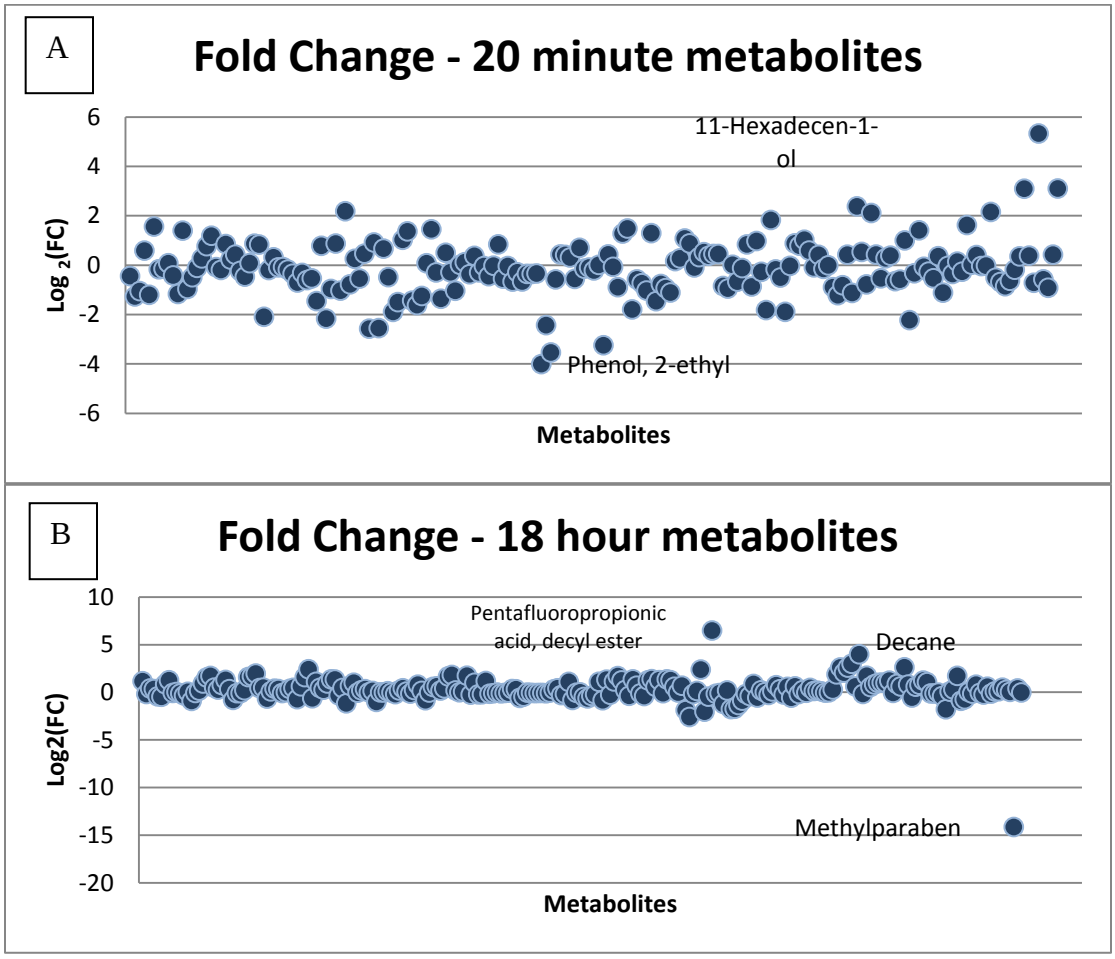


Figure 13. Fold change analysis of metabolites collected from home vs. endoscopy collected samples. A) Fold change analysis of metabolites collected after a 20 minute extraction. B) Fold change analysis of metabolites collected after an 18 hour extraction. Metabolites that had significant FC after 20 minutes extractions were considered insignificant after 18 hour extractions.

When comparing the 18 hour and 20 minute data sets, the latter contained a greater percentage of metabolites that exceeded the fold change threshold of 2.

Furthermore, the metabolites present in the 20 minute data set containing significant fold change had little fold change and were deemed insignificant in the 18 hour data set. This is indicated by the underlined metabolites presented in Table 7, the fold change analysis results. Looking at the tables, only 2 metabolites (underlined in both tables) have significant fold change in both the 20 minute and 18 hour data sets. In other words, the same metabolites that had large fold change values in the 20 minute data set, have an small fold change at extended durations, rendering them inconsequential. Thus, these metabolites with low fold change scores are insignificant in terms of physiological differences between the cohorts. Since 18 hour extraction occurred at equilibrium (see Figure 2), it can be said, with greater confidence, that these metabolites are ones of importance due to the large fold change. Additionally, the metabolites that were extracted after 20 minutes were only considered significant because of experimental conditions and not based on metabolome differences. This is due to the time point on the curve, shown in Figure 2, located in the portion with the greatest slope. Therefore, the number of metabolites extracted at this time is prone to more fluctuation in comparison to an extraction performed after VOC equilibrium has been established at 18 hours. In opposition to other reports[24,31] where fecal VOCs were extracted for 20 minutes, this signifies that extracting fecal VOCs must be completed after equilibrium of the analyte partitioning between the SPME fiber, sample headspace, and sample matrix is reached, occurring at 18 hours.

Table 6. Fold change analysis for metabolites identified in healthy home vs. endoscopy samples after 20 minute extractions. Metabolites in bold are found in both fold change analyses. Negative log2(FC) values indicate metabolites that are more abundant in endoscopy collected samples. Positive log2(FC) values are indicative of metabolites with greater abundances in home collected samples.

	Fold Change	log2(FC)
11-Hexadecen-1-ol, (Z)-	40.232	5.330
Phenol, 2-ethyl-	0.062	-4.008
Phenol, 3-ethyl-	0.086	-3.537
11-Tetradecen-1-ol, (E)-	0.106	-3.244
Azulene, 1,2,3,4,5,6,7,8-octahydro-1,4-dimethyl-7-(1-methylethenyl)-, [1S-(1a,4a,7a)]-	8.597	3.104
Naphthalene, 1,2,3,4,4a,5,6,8a-octahydro-4a,8-dimethyl-2-(1-methylethenyl)-, [2R-(2a,4aa,8aa)]-	8.544	3.095
Butanoic acid, 5-methyl-2-(1-methylethyl)cyclohexyl ester	0.168	-2.573
cis-11-Tetradecen-1-ol	0.171	-2.550
4-Ethylphenyl acetate	0.186	-2.430
Sulfur dioxide	5.206	2.380
E-7-Tetradecen-1-ol	0.214	-2.226
Azulene, 1,2,3,5,6,7,8,8a-octahydro-1,4-dimethyl-7-(1-methylethenyl)-, [1S-(1a,7a,8aa)]-	4.539	2.182
1-Octadecanol, methyl ether	0.223	-2.167
Phthalic acid, cycloheptyl isobutyl ester	4.451	2.154
Aminomethanesulfonic acid	4.320	2.111
Menthol	0.235	-2.092
10-Heneicosene (c,t)	0.269	-1.893
Benzeneacetic acid, a-oxo-, methyl ester	0.270	-1.891
Phthalic acid, isobutyl 4-octyl ester	3.554	1.829
1,3,8-p-Menthatriene	0.283	-1.820
Trichloroacetic acid, pentadecyl ester	0.288	-1.794
Oleyl Alcohol	3.084	1.625
Dichloroacetic acid, 4-hexadecyl ester	0.326	-1.618
Acetophenone	2.954	1.563
9-Nonadecene	0.354	-1.497
2-Butenal, 2-methyl-	2.798	1.485
2-Tetradecene, (E)-	0.364	-1.460
Z-5-Nonadecene	0.364	-1.456
1-Butanone, 1-phenyl-	2.734	1.451
Trichloroacetic acid, tridecyl ester	0.369	-1.439
Decyl trifluoroacetate	2.658	1.410
18-Nonadecen-1-ol	2.634	1.397
1,2,4,5-Tetroxane, 3,3,6,6-tetraphenyl-	2.582	1.369

1-Hexadecene	0.389	-1.362
5-Benzoylpentanoic acid	2.456	1.296
2-Butenal, 3-methyl-	2.428	1.280
7-Tetradecene	0.415	-1.268
Cyclotetradecane	0.421	-1.249
Phenylglyoxal	0.430	-1.218
Pentadecyl pentafluoropropionate	0.435	-1.202
pentadecanal	2.284	1.191
n-Tetracosanol-1	0.445	-1.167
1-Butanol, 2-methyl-	0.458	-1.127
1-Decanol, 2-hexyl-	0.462	-1.115
Cyclopropane, propyl-	0.463	-1.110
Phthalic acid, isobutyl octyl ester	2.114	1.080
Z-8-Hexadecene	0.483	-1.051
a-Phellandrene.1	0.487	-1.038
Nonadecyl acetate	0.489	-1.033
Hexanoic acid	2.041	1.029
3-Heptadecene, (Z)-	0.490	-1.029
Benzophenone	2.038	1.027
2-Methyl-1-undecanol	2.001	1.001

Table 7. Fold change analysis for metabolites identified in healthy home vs. endoscopy samples after 18 hour extractions. Metabolites in bold are found in both fold change analyses. Negative log2(FC) values indicate metabolites that are more abundant in endoscopy collected samples. Positive log2(FC) values are indicative of metabolites with greater abundances in home collected samples.

	Fold Change	**log2(FC)**
Methylparaben	5.55E-05	-14.136
Pentafluoropropionic acid, decyl ester	89.194	6.479
Decane	15.890	3.990
2-Fluorobenzoic acid, 4-nitrophenyl ester	7.964	2.993
Cyclobutyl phenyl ketone	6.166	2.624
3-Buten-1-one, 2,2-dimethyl-1-phenyl-	6.099	2.609
2-Fluorobenzoic acid, 3,4-dichlorophenyl ester	6.042	2.595
1,4-Benzenedicarboxaldehyde	0.166	-2.595
Pentanoic acid, 5-hydroxy-, 2,4-di-t-butylphenyl esters	5.344	2.418
Butanetetrone, diphenyl-, 2,3-dioxime	5.222	2.385

Aminomethanesulfonic acid	4.173	2.061
Ethanone, 1-(6-methyl-3-pyridinyl)-	0.244	-2.034
Octadecanoic acid, 2-(2-hydroxyethoxy)ethyl ester	3.911	1.968
Sulfur dioxide	3.688	1.883
Isophthalaldehyde	0.273	-1.874
2-Benzoyloxyacetophenone	3.529	1.819
E-11(13-Methyl)tetradecen-1-ol acetate	0.291	-1.782
Pentadecane, 7-methyl-	0.295	-1.763
Cyclodecane, methyl-	3.338	1.739
Ethanone, 2-chloro-1,2-diphenyl-	3.328	1.735
Dodecane, 2,7,10-trimethyl-	3.311	1.727
Naphthalene, 1,2,4a,5,8,8a-hexahydro-4,7-dimethyl-1-(1-methylethyl)-, [1S-(1a,4aa,8aa)]-	0.302	-1.727
5(4H)-Oxazolone, 2-phenyl-4-(phenylmethylene)-, (Z)-	3.283	1.715
Benzoic acid 5-methyl-2-phenyl-2H-pyrazol-3-yl ester	3.246	1.699
octadecanoic acid	3.218	1.686
Benzoyl isothiocyanate	3.177	1.668
Benzoylformic acid	3.026	1.597
Phenol, 2,4-bis(1,1-dimethylethyl)-	3.021	1.595
Ethanone, 2,2,2-trifluoro-1-phenyl-	2.932	1.552
Methanol, oxo-, benzoate	2.597	1.377
Ethanone, 2-(formyloxy)-1-phenyl-	2.566	1.360
Benzenepropanoic acid, silver(1+) salt	2.564	1.359
Dodecane, 4,6-dimethyl-	2.561	1.357
Propanoic acid, 2-(benzoylthio)-, ethyl ester	2.534	1.341
Naphthalene, 1,2,3,5,6,8a-hexahydro-4,7-dimethyl-1-(1-methylethyl)-, (1S-cis)-	0.396	-1.335
2-Cyclohexene-1,4-dione, 5,6-dibromo-2,6-dimethyl-, 1-oxime, o-benzoyl-	2.508	1.327
Methanone, (4-bromo-5-methyl-2-nitro-3-thienyl)(phenyl)-	2.457	1.297
4-Nitrosophenyl-a-phenylpropionate	2.419	1.274
Benzenepropanoic acid	2.412	1.270
Cyclobutane-1,1-dicarboxamide, N,N'-di-benzoyloxy-	2.404	1.265
4-Hydroxy-3-methylacetophenone	2.387	1.255
Benzoic acid, (4-benzoyloxy-2-chlorophenyl) ester	2.370	1.245
Butyric acid, octadecyl ester	2.349	1.232
N-Benzyloxy-2,2-bis(trifluoromethyl)aziridine	2.334	1.223
1,14-Tetradecanediol	0.430	-1.217
(d)-(+)-(2R,3R)-2,3-Dibenzoyltartaric acid	2.274	1.185
Benzoyl bromide	2.235	1.161
Hexanedioic acid, dioctyl ester	0.448	-1.159

Pentanediamide, N,N'-di-benzoyloxy-	2.207	1.142
4,4-Dimethylpent-2-enal	2.148	1.103
Ethanone, 2-hydroxy-1-phenyl-	2.139	1.097
Benzeneacetonitrile, a-hydroxy-	2.134	1.094
Butyric acid, pentadecyl ester	2.083	1.059
2-(2-Oxo-2-phenyl-ethyl)-malononitrile	2.026	1.019
Eicosane, 10-methyl-	0.498	-1.007
Hexanedioic acid, bis(2-ethylhexyl) ester	2.002	1.002

Summary of Fecal Collection and VOC Extraction for 18 Hours and 20 Minutes

The purpose of analyzing fecal VOCs collected from healthy patients was to compare endoscopy and home collected samples and determine if aerobic exposure would alter the metabolome. This was also to determine if home collected samples could substitute endoscopy collected samples to diagnose a patient since endoscopies are an expensive and uncomfortable procedure. Preferably, collecting fecal samples after home passage would be the better option. The PCA plots generated display that the two cohorts segregate from one another. This could be caused when the fecal samples are exposed to oxygen. It can be speculated that the aerobic bacteria in the stool could produce different metabolites and metabolite levels. These metabolite differences cause separation between the two cohorts on the PCA plot. In terms of this study, this means that one collection method cannot be used as a replacement for the other. Furthermore, these results justify an endoscopic collection of stool samples if they are needed for a metabolomics study. More information regarding fecal sample collection technique will be discussed in the following section with respect to diagnostic purposes.

In regards to the extraction duration, this portion of the study was to determine if the length of time that the SPME fiber is exposed to the sample will affect the number of metabolites isolated. There have been reports[24,31] that have implemented 20 minute extraction durations even though the extraction profile for fecal VOCs is hyperbolic and does not reach equilibrium until a later time (see Figure 2). The results from this study show that the longer extraction duration (18 hours) yields a higher amount of metabolites than the shorter extraction time (20 minutes), which contains roughly 20% of the 18 hour data set. Moreover, after a 20 minute extraction, the analytes with a high fold change value (FC>2) that are considered important due to drastic values between the cohorts (see Table 6. Fold change analysis for metabolites identified in healthy home vs. endoscopy samples after 20 minute extractions. **Metabolites in bold are found in both fold change analyses. Negative log2(FC) values indicate metabolites that are more abundant in endoscopy collected samples.** did not have high fold change values when they were extracted for 18 hours (see Table 7). Since the shorter extraction time is more variable due to the position on the extraction curve, the 20 minute SPME fiber exposure time does bias the results. Therefore, longer extraction times, 18 hours, should be used in fecal VOC metabolomics studies.

Although most identified metabolites are present in both cohorts, there are a few that are highly associated with a specific cohort (see Table 7). Further examination of the metabolites needs to be completed to justify the detection these analytes. A few metabolite rationales are as follows. Methylparaben, found in the endoscopic samples, is used as a preservative in lubricating jelly.[10] This analyte may have been added to the

feces during the endoscopy procedure. Aminomethanesulfonic acid is more abundant in the home collected samples and has a significant fold change value. Aminomethanesulfonic acid is found in the large intestine and is a primary component of bile.[84–86] A possible explanation as to why this analyte is detected in higher abundance in the home collected samples is could be that the stool sample comes in contact with the large intestine more during the home passage. Thus, the analyte could have the opportunity to accumulate more in these samples. Of the conserved metabolites, only a few analytes have abundances that differ. This could be a reflection of VOCs that have escaped from the fecal samples before the analysis was performed.

Analysis of Feces from Different Patient Types: Healthy vs. Alcoholic

This portion of the study was to determine if a fecal VOC metabolomics approach could be used to differentiate alcoholic and healthy patients. Since abnormally smelling feces is indicative of GI disease, stool VOCs could potentially be used to diagnose an intestinal ailment.[8,31] Three SPME fibers (CAR-PDMS, PA, and DVB-CAR-PDMS) were exposed to the fecal samples for only the 18 hour duration, since 20 minute extractions isolate VOCs whose abundances can be hypervariable, as previously discussed. Samples were collected from 16 alcoholic patients and 18 healthy patients by way of endoscopy and 22 alcoholic patients and 25 healthy patients donated samples after home passage. From the endoscopy collected samples, 2659 metabolites were identified while 2883 metabolites were identified from the home collected samples. The metabolites can be primarily classified as alcohols, aldehydes, acids, ketones, hydrocarbons, thiols, indoles, sulfides, or amines. Again, once the metabolite data was acquired, the

spreadsheets were filtered as stated above in the methods section. PCA and PLS-DA were then performed on the filtered data.

A 3D PCA plot of the home collection method, shown in Figure 14, compares healthy and alcoholic patients, after VOCs were isolated for 18 hours. The top 3 PCs are represented in this plot on the x, y, and z-axes, respectively. Figure 15 shows the biplot of the two cohorts. Both of these plots are from 18 hour extraction data. The results in Figure 14, generated using the statistical programming software R, show that there is clear separation between the two patient types, alcoholic and healthy. Moreover, excluding one sample, the rest of alcoholic samples have a tighter cluster than the healthy samples. As seen in Figure 14, there is separation between the two cohorts indicates that the metabolite identities or their corresponding levels are drastic in value.

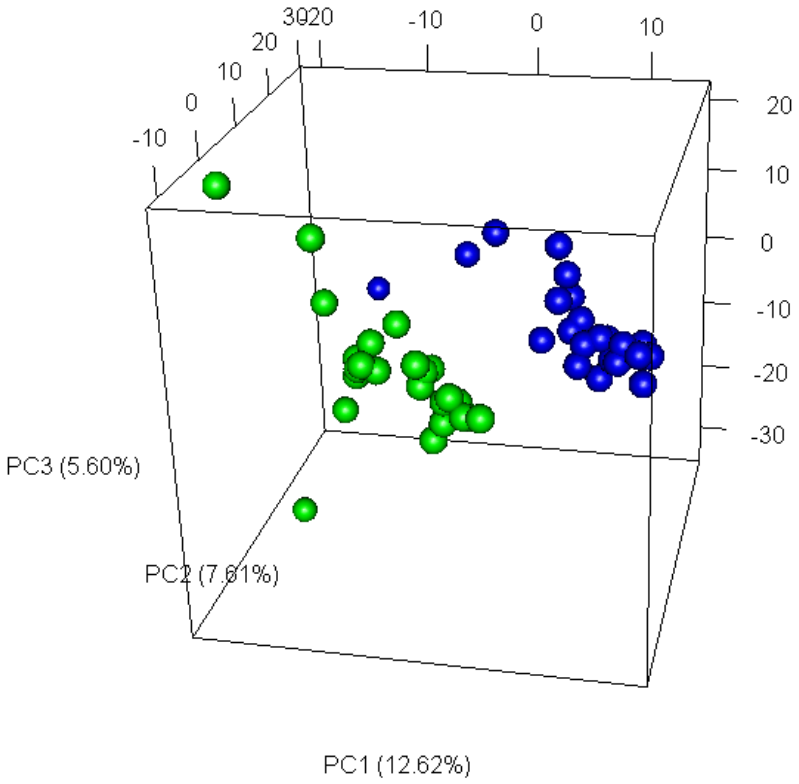


Figure 14. Home collection 3D PCA plot – 18 hour extraction. This 3D PCA plot is an analysis of the two types of patients in this study, alcoholic (blue circles) and healthy (green circles) when the samples were collected after home passage. SPME fibers were exposed to the fecal samples for 18 hours. The two cohorts show separation due to metabolite variation in the data. This indicates that there are differences in the two patient types.

In Figure 15, the two cohorts cluster together on the biplot. For this data set, metabolites were grouped into 1 of 13 chemical classes, by an algorithm, to determine if any metabolite group affected how the samples were plotted. This reduction of variables consequently reduces the resolution of the plot, causing the samples to cluster. It can be determined that the ketones and others groups have a strong negatively correlation with the nitriles.

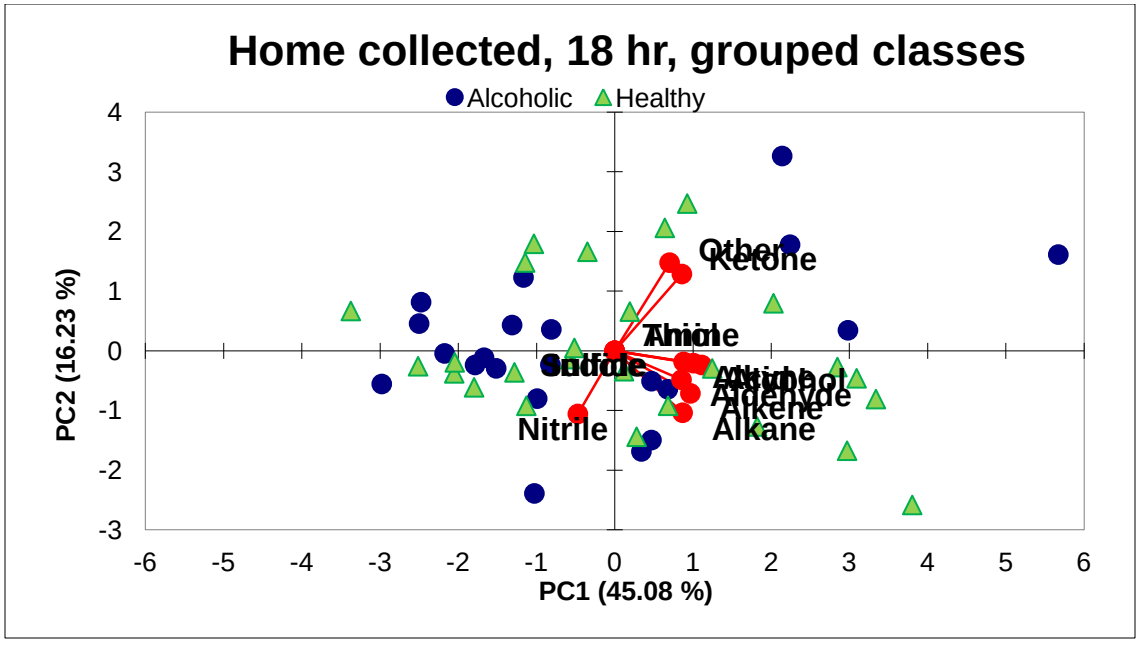


Figure 15. Home collection biplot – 18 hour extraction. This biplot, showing both the samples (blue and green markers) as well as the variables (red markers with attached vectors) is analysis of the two patient types, alcoholic (blue circles) and healthy (green circles). SPME fibers were exposed to the samples for 18 hours. An algorithm grouped each variable into one of thirteen chemical classes. The two cohorts show minimal separation; however, the ketones and others groups are negatively correlated to the nitriles group.

PLS-DA was used to build a predictive model, for potential ALD diagnosis, from the existing data. Moreover, based on the VIP calculation, specific metabolites could be selected, which cause the greatest separation between the two cohorts. These metabolites could potentially be used as biomarkers for disease. Table 8 presents the PLS-DA model quality results of the home collected samples comparing healthy and alcoholic cohorts. These values indicate how well the model can be used to predict an outcome. The Q^2 value is a measure of the overall fit, while the R^2Y and R^2X are indicators of the class and metabolite fits, respectively. These values range from 0.000 to 1.000 with values closer to 1.000 indicating a better fit model.[87] For metabolomic studies, R^2 and Q^2 scores

> 50% are considered good.[88] The Q^2 and R^2Y scores are very high indicating that this is a reliable model and the PLS analysis summarizes the Y variables, alcoholic and healthy, well. The R^2X score is low due to the large number of metabolites.

Table 8. Model quality of PLS-DA alcoholic vs. healthy home collected samples. Comp1, Comp2, and Comp 3 are equate to PC1, PC2, and PC3.

Index	Comp1	Comp2	Comp3
Q^2 cumulative	0.609	0.846	0.862
R^2Y cumulative	0.789	0.950	0.981
R^2X cumulative	0.102	0.209	0.263

The PLS-DA scores plot for the home collected samples is shown in Figure 16, which compares the healthy and alcoholic cohorts. Similar to the corresponding PCA plot of this data set, displayed in Figure 14, this PLS-DA plot shows clear separation between the two cohorts. Since there is enough covariance in the data according to this analysis, to show clear separation between the two cohorts, this model can also be used in diagnosing an alcoholic patient.

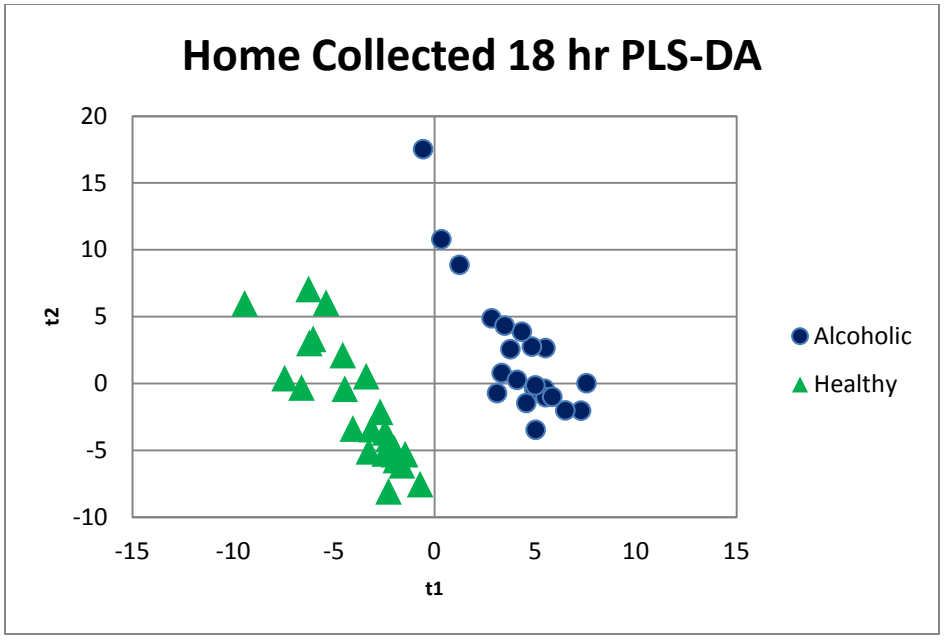


Figure 16. PLS-DA scores plot of home collected samples. This PLS-DA plot is the analysis of the alcoholic (blue circles) and healthy (green triangles) patients. Samples were collected from these patients after home passage and VOCs were extracted from the feces for 18 hours. The plot shows that the two cohorts separate from one another, indicating cohort variance due to their metabolite differences.

Table 9 displays the top 10 VIP metabolites calculated from the PLS-DA of the data set comparing alcoholic vs. healthy patients when stool samples were collected after home passage. These VOCs are more abundant in the alcoholic cohort when compared to the healthy. Additionally, they cause the greatest separation between the two classes and thus are the most indicative of alcoholism based on the calculation. Therefore these metabolites, when extracted from samples collected after home passage, are indicative of an alcoholic patient.

Table 9. Variables Important in the Projection (VIP). The listed metabolites, isolated from home collected fecal samples, have the highest VIP score. Moreover, these metabolites have values greater in the alcoholic cohort when compared to healthy cohort.

Benzene, (1-methyl-1-propenyl)-, (E)-
N(N'-Methyl-N'-nitroso(aminomethyl))benzamide
Azulene, 1,2,3,4,5,6,7,8-octahydro-1,4-dimethyl-7-(1-methylethenyl)-, [1S-(1a,4a,7a)]-
Bicyclo[4.2.0]octa-1,3,5-triene, 2,4-dimethyl-
Cycloheptane, 4-methylene-1-methyl-2-(2-methyl-1-propen-1-yl)-1-vinyl-
1-Pentanol, 4-methyl-
Pentadecane, 2,6,10-trimethyl-
4-Nitrosophenyl-a-phenylpropionate
1-Hexanol
Tetradecane, 6,9-dimethyl-

Fecal samples were collected from a total of 34 patients (16 alcoholic and 18 healthy) via endoscopy. Figure 17 shows the PCA plot for this analysis after SPME fibers were exposed to the samples 18 hours. This plot shows that there is clear segregation between the two cohorts. Patient types are discriminated by PC1, while PC2 shows the metabolite variance of each cohort.

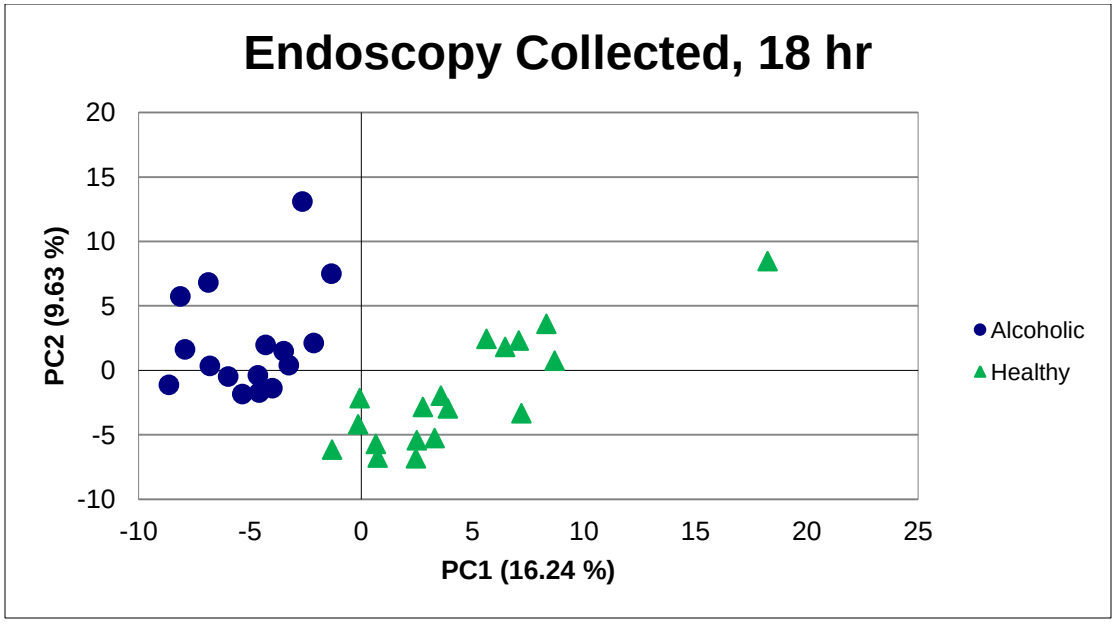


Figure 17. Endoscopy collection PCA plot – 18 hour extraction. These endoscopy collected samples, donated by alcoholic (blue circles) and healthy (green triangles) patients, were exposed to SPME fibers for 18 hours. The PCA plot displays the two classes are separate from one another indicating metabolite variance between the cohorts.

Figure 18 is the biplot of the data after metabolites have been grouped by chemical class. There is some separation between the two cohorts. The limited segregation of the classes occurs because the metabolites have been condensed into 1 of 13 chemical classes. In turn, this reduces the area and resolution of the plot as seen in Figure 15. The healthy samples have greater variance values with respect to the alkene, alcohol, and acid classes. Meanwhile, the alcoholic samples in this data set have metabolites that contain greater variance of the alkane, ketone, aldehyde, and other groups. The listed metabolite classes have the longest vectors since they provide the greatest amount of variance in the data set.

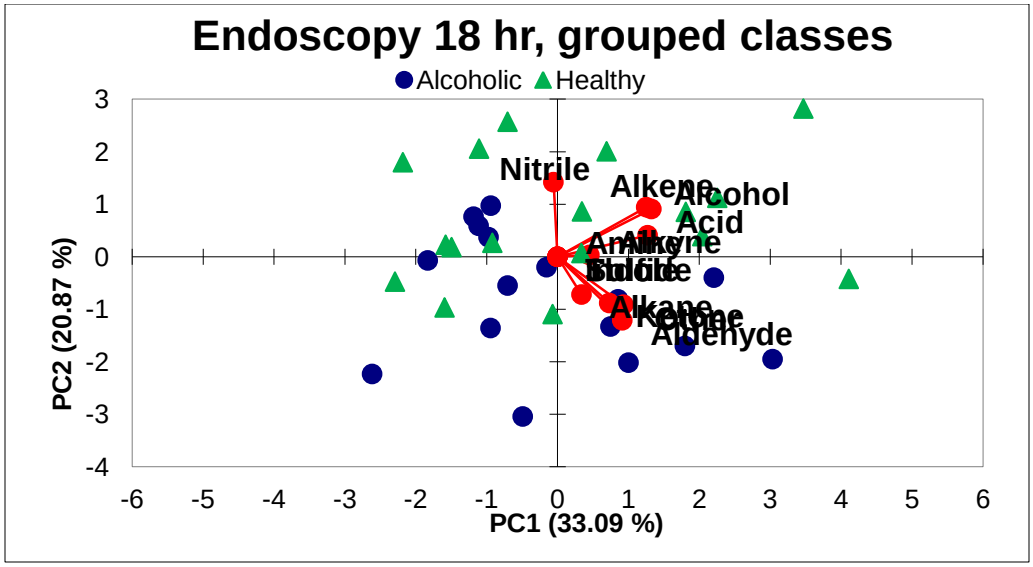


Figure 18. Endoscopy collection biplot – 18 hour extraction. This biplot is analysis of the two patient types, alcoholic (blue circles) and healthy (green circles) when fecal samples were collected via endoscopy. SPME fibers were exposed to these samples for 18 hours. An algorithm grouped each variable into one of thirteen chemical classes. This analysis shows the two cohorts having some separation between each other. Moreover, the healthy samples are affected by the alkene, alcohol, and acid classes. The positions of the alcoholic samples are influenced by the alkane, ketone, aldehyde, and other groups.

Similar to the home collected samples, a PLS-DA was performed on the data collected from the endoscopy collected samples. Table 10 exhibits the PLS-DA model quality results comparing the healthy and alcoholic cohorts. The PLS-DA score plot for the endoscopy data set is displayed in Figure 19. Similar to the corresponding PCA plot seen in Figure 17, the samples of two classes cluster amongst themselves, while the cohorts, as a whole, remain separate from one another. This indicates that the metabolite values between the two cohorts differ enough to be detected by the statistical analysis and create the separation on the plot. Therefore, this model could be used to diagnose an alcoholic patient.

Table 10. Model quality of PLS-DA alcoholic vs. healthy endoscopy collected samples. Comp1, Comp2, and Comp 3 are equate to PC1, PC2, and PC3.

Index	Comp1	Comp2	Comp3
Q^2 cumulative	0.835	0.894	0.907
R^2Y cumulative	0.885	0.969	0.988
R^2X cumulative	0.154	0.243	0.304

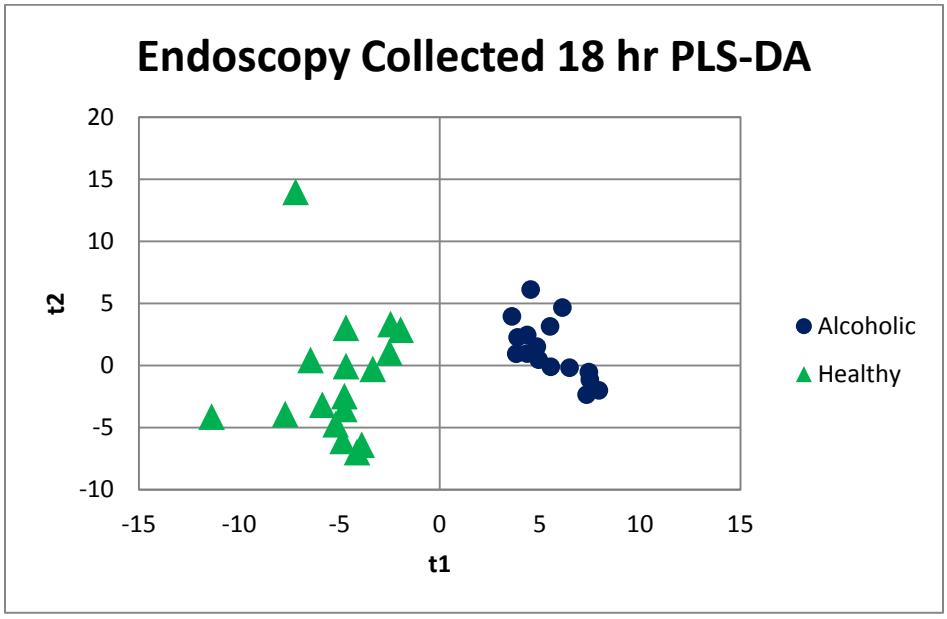


Figure 19. PLS-DA scores plot of endoscopy collected samples. This PLS-DA plot compares alcoholic (blue circles) and healthy (green triangles) patients. These fecal samples were collected via endoscopy and the VOCs were extracted for 18 hours. The two cohorts in this scores plot segregate from each other, which is indicative of a difference in the metabolite variance between classes.

Metabolites to highlight in this analysis are presented in Table 11. These are the top 10 VIP analytes calculated from the PLS-DA of the data set comparing alcoholic vs.

healthy patients when the fecal samples were collected via endoscopy. These listed VOCs were more abundant in the alcoholic samples when compared to the healthy cohort.

Table 11. Variables Important in the Projection (VIP). The listed metabolites, isolated from endoscopy collected samples, have the highest VIP score in this data set. Their values were greater in the alcoholic cohort in comparison to the healthy cohort.

Heptafluorobutyric acid,n-tridecyl ester
Benzamide, N-benzoyl-N-(2-trifluoromethylphenyl)-
2-Cyclohexene-1,4-dione, 5,6-dichloro-2,3-dimethyl-, 1-oxime, o-benzoyl-
Pentafluoropropionic acid, hexyl ester
Propan-1-one, 3-nitro-1-phenyl-
1-Pentanol, 4-methyl-
Chloroacetic acid, pentadecyl ester
1,3-Benzenediol, o-methoxycarbonyl-o'-(2-methoxybenzoyl)-
Hexadecane, 7-methyl-
Undecyl heptafluorobutyrate

Based on the PLS-DA results of the alcoholic vs. healthy data sets, it can be determined that both models can effectively diagnose a person for alcoholism. This can be said because both analyses produce score plots in which both cohorts clearly segregate from one another. Additionally, the model quality values, which validate the models as well as explain the goodness of fit for predicting an outcome, were high. If given the choice, between two collection methods, the home sample collection is less invasive and cheaper than the endoscopy and therefore would be preferred.

Summary of the Alcoholic vs. Healthy Patient Types Analysis

The purpose of this portion of the study was to determine if there were differences in the fecal VOC metabolomes of healthy and alcoholic persons in an attempt to diagnose alcoholism. As previously stated, both home and endoscopy collected samples can be

differentiated based on their extracted VOCs. Because of the invasive and expensive nature of an endoscopy procedure, home collection of fecal sample to diagnose disease would be preferred. Samples were collected after home passage and through an endoscopic procedure. VOCs were extracted from those samples for 18 hours.

When the fecal samples were collected at home and VOCs were extracted for 18 hours, the 3D PCA plot (see Figure 14) shows clear separation between the two cohorts, indicating large metabolite variance between the cohorts. After these metabolites were grouped by chemical class, the biplot results (see Figure 15) showed limited cohort separation as well. The PLS-DA (Figure 16) displays clear separation as well between the two cohorts. The data from the endoscopic collected samples, extracted for 18 hours, results in separated cohorts on a PCA plot (see Figure 17). This is indicative of variance between the two cohorts due to their metabolites. Grouping the metabolites by chemical class results in a biplot (see Figure 18) that has some cohort separation. Additionally, the PLS-DA results (Figure 19) show the two patient types to be segregated. Samples collected after home passage and via endoscopy contain metabolite values that differ to cause separation between the two cohorts on a PCA or PLS-DA plot. Since VOCs in either endoscopy or home collected samples can be used to differentiate between healthy and alcoholic patients, home collection is preferred because it is less invasive and inexpensive in comparison to an endoscopy collection. Therefore, collecting fecal samples after home passage is a viable way to diagnose an alcoholic patient.

Both 1-hexanol and 1-Pentanol, 4-methyl- were detected in the alcoholic cohort with high abundance. The metabolite, 1-hexanol, was more abundant in the alcoholic

cohort within the home collected samples. Additionally, 1-Pentanol, 4-methyl- was more abundant in alcoholics in both endoscopy and home collected samples. These metabolites can be found in alcoholic beverages since they are by-products of grain fermentation.[89] This could give insight as to why they were identified in alcoholics since these patients would imbibe more alcoholic beverages in comparison to healthy persons. As previously stated, more rationales for identified metabolites are needed in future studies.

Conclusion

Metabolomics, which is the collection and analysis of small molecules from a sample matrix, is a multi-faceted tool with uses ranging from environmental to human-health related issues. VOCs can be extracted using SPME fibers from various samples and be analyzed as part of a metabolomics investigation. Since each SPME fiber is composed of different polymers, a multi-fiber approach is needed to extract a full suite of metabolites from a sample. Coupled with a GC-MS analysis, a metabolomics investigation utilizing SPME fiber extraction of VOCs from human feces has been described in this thesis.

Chronic liver disease is the 12th highest cause of death in the US. Moreover, ALD was cause of death for nearly 16,000 people in 2010. As of now, biopsies and other invasive procedures are the only methods to diagnose the disease. The purpose of this study is to find a list of metabolites that would lead to a less invasive diagnosis of ALD.

The first aim of this study was to determine if home collection of samples could substitute for endoscopy collection. Additionally, this part of the study was used to justify the invasive and expensive endoscopic process to collect fecal samples. Comparing

metabolites extracted from the home and endoscopy samples, the resultant PCA plots (Figure 9 and Figure 11) show both of the cohorts clearly segregate from one another indicating differences in analyte identities or values. This could be due to the presence or absence of oxygen in regards to the home and endoscopy fecal collection methods. Additionally, differences in collection containers (plastic vials for endoscopy collected samples and plastic bags with anaerobic GasPacks for home collected samples) could have affected the metabolite numbers and values. Therefore, collecting samples via endoscopy or after home passage does affect the extracted metabolites identified and corresponding values. Moreover, this justifies stool collection via endoscopy if it is needed for a fecal VOC metabolomics investigation.

The second objective of this study was to establish an extraction time (20 minutes or 18 hours) for fecal VOC metabolomics. To determine a proper extraction time for fecal VOCs, PCA and fold change analysis were performed on the data sets. In comparing the collection method of samples from healthy patients, the PCA plots of both the 18 hour and 20 minute data sets (Figure 9Figure 11) show clear separation of the two cohorts. Moreover, the samples from the 20 minute data set clustered tighter on the PCA plot indicating smaller metabolite variance within the class. Although these results show bias toward the 20 minute extractions, the fold change analysis shows otherwise. This analysis shows that metabolites extracted after 20 minutes and contain significant fold change between cohorts were not deemed important in the 18 hour data set (Table 6Table 7). Since the longer extraction occurs at equilibrium, this list of important metabolites would be more reliable. Furthermore, the great fold change values linked to the

metabolites extracted after 20 minutes are attributed to experimental conditions and not metabolome differences. Additionally, this extraction duration produces a limited data set (~750 less metabolites identified than in the 18 hour data set). Therefore, it was determined that extractions of fecal VOCs should be completed at 18 hours, as opposed to 20 minute extractions. In an 18 hour SPME fiber extraction, it is noteworthy however, that some of the VOCs might degrade and derivatize due to the elevated temperatures. Additionally, during the longer extractions, some analytes with higher binding affinities may displace those with lower affinities on the SPME fibers.

Lastly, the third goal of this study was to ascertain if fecal VOCs could be used to differentiate between healthy and diseased (alcoholic) states. Using PCA and PLS-DA (Figure 14,Figure 16, Figure 17, and Figure 19) to statistically analyze the alcoholic vs. healthy data sets, it was determined that both collection techniques can be used to diagnose an alcoholic patient. This can be concluded since samples from either fecal collection contain identified metabolites that differ in value to create separation between the two cohorts on the PCA or PLS-DA score plot. Moreover, the PLS-DA could identify certain metabolites (Table 9Table 11), which are indicative of an alcoholic patient. Since fecal samples, using either collection technique, contain differences in metabolite identities and values to distinguish between healthy and diseased patients, the home collection method would be preferred because of the less invasive and inexpensive nature when compared to an endoscopy.

PART II - APPLICATIONS: WHIPWORM INFECTION AND THE EFFECT ON PORCINE FECAL VOCS

Introduction

Commonly found in pigs, the whipworm *Trichuris suis* (*T. suis*) can cause several illnesses which include diarrhea, anorexia, and retarded growth. Not only do these helminth parasites damage a pig's health, they also greatly affect global food availability in countries such as Japan, Kenya, the Netherlands, and Uganda.[22] Adult worms are expelled from the porcine intestine nine weeks after infection. At this point, parts of the pig body, especially the blood and intestine, will undergo conformations that hamper its function.[22] These parasites are typically spread in areas of poor sanitation through contaminated water, soil, or food. Humans will come in contact with the helminths via agricultural practices, hunting, and animal farming. A majority of the helminths are not able to use human as hosts due to genetic differences.[90]

Although parasite infestation causes detrimental effects to the host, these worms have been used to treat various ailments. Studies that were conducted in Ethiopia and Vietnam infected their subjects with hookworm.[91,92] The results from these investigations showed that the patients had a decreased number of asthmatic cases and decreased allergic reaction to dust mites, respectively.[91,92] Helminth parasite infestation research, namely that of *Trichuris trichiura* (*T. trichiura*), has provided promising results in regards to human health. Beneficial effects from a *T. trichuria* infection are providing aid

in host immune responses, and anti-inflammatory remedies.[22,90] Furthermore, these parasites have been used to assess the "inflammatory bowel disease (IBD) hygiene hypothesis," treat Crohn's disease, in addition to many other immune-mediated diseases.[22,90] The "IBD hygiene hypothesis" states that children, raised in extremely clean environments, will have negative effects to their immune system due to the living conditions.[90] These children are then predisposed to develop IBD and other immunological diseases later on in their lifetime.[90] Approximately 1-2 million people in the US are affected by the common forms of IBD, Crohn's disease and ulcerative colitis, which are the inflammation at any point of the GI tract.[90,91,93]

Data has been gathered that supports the use of helminths to reduce the chances of IBD.[22,91,93] The parasites are believed to be highly associated with type 2 T helper (Th2) cell response. The Th2 response inhibits the production of Th1 cytokines, which is correlated to autoimmune disease and Crohn's disease.[91,93] Th1 cells activate macrophages that mediate cell immunity and phagocyte responses.[94] Th2 cells generate various interleukins that are responsible for antibody production and eosinophil activation, while inhibiting some macrophage function.[91,94] This type of therapy has been performed in clinical trials where parasites were used to treat IBD. The infestation resulted in an improved GI tract.[90] Although many of these parasite treatments have shown positive outcomes, it has still received negative feedback due to the potential danger from the *T. suis* infection. These parasites can cause tissue damage, leading to pathogenic microbes invading the host; a form of secondary infections.[22] Other potential

pitfalls for the helminth treatment include liver fibrosis, portal hypertension, and decreasing the efficacy of a vaccine.[90]

The GI tracts of pigs and humans are similar; additionally, there is cross-reactivity between *T. suis* and *T. trichiura*, an intestinal nematode that infects 1 billion people worldwide.[22] Therefore, the interactions between *T. suis*, the intestinal microbiome, and the host pig, can be used as a model for the relationship between *T. trichiura* and humans. Furthermore, studying the indirect effects from the parasites on the intestinal microbes can be used to obtain a better grasp of the expression of diseases, such as Crohn's disease, ulcerative colitis, and allergic rhinitis. In turn, this could provide new optimization techniques regarding the treatment of IBD with helminth parasites. A GC-MS metabolomics analysis was performed on porcine fecal samples to investigate metabolite changes in the intestine caused by a *T. suis* infection. This was done to validate and supply reasons for biosynthetic differences between the pigs. The results of this investigation have been published in the Journal of Infection and Immunity.[22]

Materials and Methods

The following section describes the methods for the metabolomic study performed on pig feces, which was a collaborative effort with the United States Department of Agriculture (USDA). This study investigated the infection of the whipworm *T. suis* in the pigs and how this interaction would affect the bacteria in the colon.

Animal Preparation

Seven female pigs from four different litters were obtained for this study from Beltsville Area Swine Facility (Beltsville, MD). The pigs were kept in an indoor facility where they had access to water as needed and the volume of their food, consisting of corn/soybean meal, was monitored. When then pigs reached 3 months of age, 4 of the pigs were inoculated with the *T. suis* eggs while the other 3 were orally administered phosphate-buffered saline (PBS) by mouth. All seven pigs were killed 21 days after inoculation and luminal contents were collected from the proximal colon at ~30 cm from the ileal/cecal junction.[22]

Fecal Samples

Samples were collected by the USDA team, snap frozen in liquid nitrogen, and delivered overnight in dry ice for analysis. While still frozen, approximately 0.2 g of pig fecal samples was dispensed into amber vials in a sterile biosafety cabinet environment. Samples were stored at -80°C until they were analyzed.

Headspace Solid-phase Microextraction Procedure

The SPME fibers were preconditioned in a GC inlet based on manufacturer's specifications. Preconditioning durations and temperatures are shown in Table 2. Samples were first heated to 60°C for 30 minutes. The entire SPME fiber assembly was manually positioned above the feces to extract the VOCs for 18 hours. Fibers that were used in this experiment were PA 85 µm, DVB/CAR/PDMS 50/30 µm, and CAR/PDMS 75 µm. The SPME fiber was then inserted into a heated GC inlet port and the metabolites were desorbed onto the column. Since the stationary phase was non-polar, the polar

metabolites eluted first, while non-polar metabolites eluted towards the end of the analysis.

Instruments

Samples were analyzed using an Agilent 7890A GC and 5975C inert XL mass selective detector (MSD) with triple axis detector (Agilent, Palo Alto, CA). The GC-MS was equipped with a DB5-MS capillary column (Agilent), 30 m in length, 0.25 mm ID, and 0.25 mm film thickness, and a 0.75 mm ID SPME injection port liner operated in splitless mode at varying inlet temperatures, dependent upon the SPME fiber used.

GC-MS Conditions

For the GC-MS, helium carrier gas was set to 1.17 mL/min flow rate and the GC oven was held at an initial temperature of 35°C for 1 min, ramped to 80°C at 3°C/min, then to 120°C at 10°C /min, and finally to 260°C at 40°C /min. The final temperature of 260°C was held for 1.5 min. The total run time for the analysis method was 25.0 min. The Agilent 5975C MSD was scanned from 30 to 550 amu at a rate of 2.81 scans/s.

AMDIS and NIST Analyses

Analytes from a chromatogram were identified by comparing the spectra to those found in the NIST database. Resultant spectra needed to match the spectra in the NIST database by a score of 90% to be identified. AMDIS software was used to prepare the spectra for identification by subtracting the noise, and selecting peaks to be named as metabolites. These analyses are described in detail above (see Part I – Applications: Fecal VOCs and Alcohol Liver Disease).

Statistical Analysis: Principal Component Analysis

A matrix of the healthy and infected pig metabolites and the corresponding levels extracted from fecal VOCs was produced. The data set was filtered first, followed by a statistical analysis using PCA. A detailed description of both procedures can be seen in Part I – Applications: Fecal VOCs and Alcohol Liver Disease.

Statistical Analysis: Fold Change

Fold change analysis was performed using the Microsoft Excel. Using the data filtered table, the averages of each metabolite indentified from the specific cohort were calculated. The average peak abundance of a metabolite from cohort 1 (healthy) was divided by the average peak abundance of a metabolite from cohort 2 (infected). The values were then log transformed to be plotted in an xy-scatter plot.

Metabolome Analysis

The two cohorts were compared to determine how the *T. suis* infestation affects the metabolome of the pig intestine. Metabolites were identified using the NIST database and those that were exclusive to each cohort were highlighted. The unique analytes were then searched through the Kyoto Encyclopedia of Genes and Genomes (KEGG) database to determine how they are used in metabolic processes.

Results and Discussion

Four piglets were orally administered a dose of *T. suis* eggs, while 3 piglets were administered PBS as a placebo (7 female piglets in total). All the pigs were killed 21 days after the PBS or eggs were given. Fecal samples were then collected from the colon and stored at -80°C. SPME fibers were used to extract VOCs, which were subsequently analyzed on a GC-MS. A total of 199 metabolites were identified from the healthy pigs,

while 197 metabolites were identified from the pigs infected with whipworm. There were 162 metabolites (85% of all metabolites) that were ubiquitous to both cohorts. These metabolites are listed in Appendix C. Furthermore, 21 metabolites were only found in the healthy cohort, while 23 analytes were unique to the infected pigs.

Statistical Analysis: Principal Component Analysis

PCA was performed on the data set to determine if any samples, based on their metabolites, cluster together. These results, displayed in Figure 20, show that the infected samples are closer to one another than the healthy samples. The latter of the two cohorts had large distances between the respective data points. Overall, there is little to no clustering at all in this plot indicating large amounts of metabolite variance between the individual samples. This could be attributed to differences in analyte abundances of the large number of core metabolites detected in both cohorts. Moreover, minimal information about the metabolomes can be elicited from the plot.

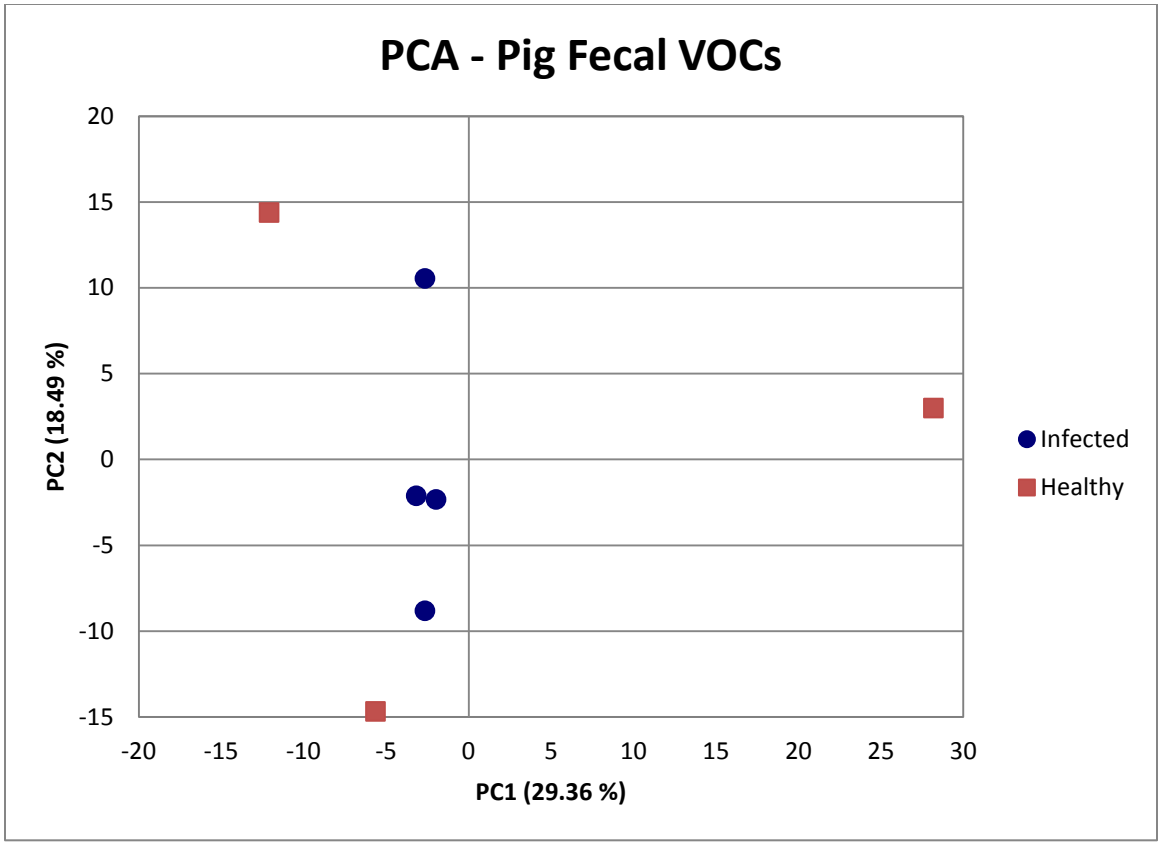


Figure 20. PCA plot of pig fecal VOCs extracted for 18 hours. This PCA plot compares the VOCs extracted from infected (blue circles) and healthy (red squares) pig fecal samples. SPME fibers were utilized to extract the analytes from the samples for 18 hours. The results show that infected samples are plotted closer together in comparison to the healthy samples. Data points from healthy samples are spread apart.

Statistical Analysis: Fold Change Analysis

A fold change analysis was performed on the metabolites to determine which metabolites change the most between cohorts. The results are displayed in Figure 21. Metabolites of note are 2-octene, 2-methyl, 2-butenal, and oleic acid. The first two analytes are unique to the healthy cohort, while the third is unique to the infected cohort. Rationales of how these VOCs could be linked to healthy and disease conditions are explained below.

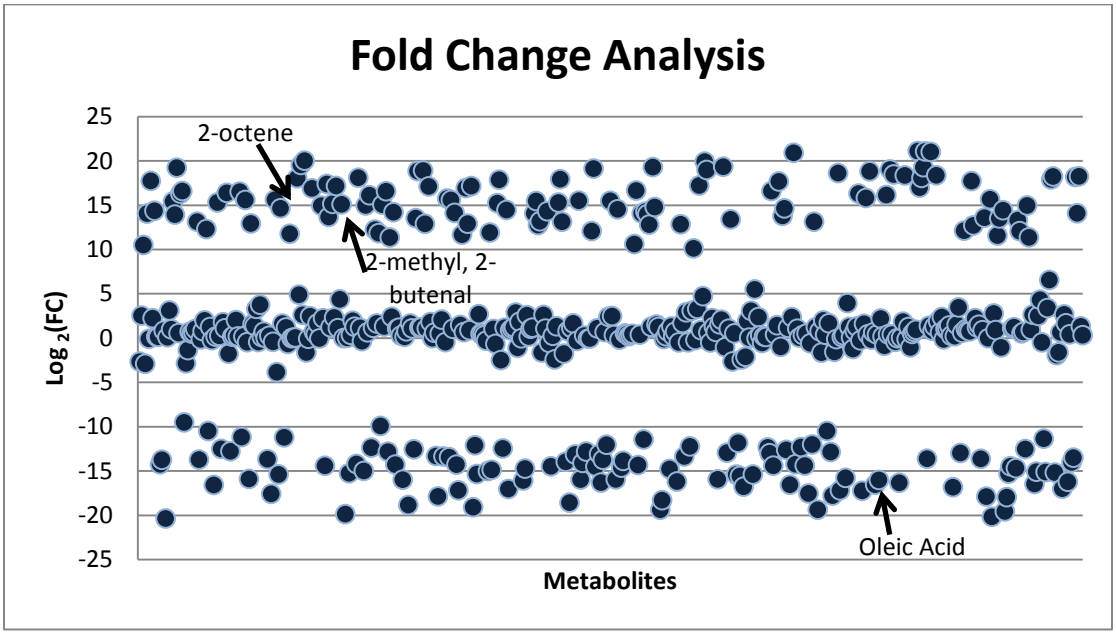


Figure 21. Fold change analysis of VOC from pig feces. Fold change is calculated as the log transformation of the average abundance of a metabolite in healthy cohort relative to the infected cohort.

Metabolome Analysis

Of the metabolites only found in all 4 of the infected pigs, oleic acid was the most noteworthy. The remaining metabolites are displayed in Table 12. Oleic acid is not detected in the healthy cohort because it is normally absorbed throughout the pig intestine.[22] Oleic acid, potentially derived from the GI bacteria, has antibacterial activity. It can be speculated that this metabolite could explain the significant microbiota changes, at both the phylum and genus levels, detected between the two cohorts. The detection of this metabolite could also provide insight to the inflammation of the GI tract seen in the infected pigs.[22] However, further tests are required to confirm these hypotheses.

Table 12. At or above a cutoff score of 90%, metabolites that are unique to the infected pig cohort

CAS number	Metabolite
112801	Oleic Acid
105760	2-Butenedioic acid, dibutyl ester
123080	Benzaldehyde, 4-hydroxy-
106683	3-Octanone
541731	Benzene, 1,3-dichloro-
611701	Isopropyl phenyl ketone
579077	1,2-Propanedione, 1-phenyl-
6919615	N-Methoxy-N-methylbenzamide
532274	Acetophenone, 2-chloro-
614164	Benzenepropanenitrile, á-oxo-
98919	Benzenecarbothioic acid
2648615	à,à-Dichloroacetophenone
611734	Benzoylformic acid
618326	Benzoyl bromide
75503	Trimethylamine
18409171	2-Octen-1-ol, (E)-
71363	1-Butanol
288391	1,2,5-Thiadiazole
1074120	Phenylglyoxal
101973	Benzeneacetic acid, ethyl ester
91190	Quinoxaline
3050699	n-Caproic acid vinyl ester
75036	Ethane, iodo-

Table 13 presents the metabolites that are unique to the healthy cohort. The metabolites of interest are 2-butenal, 2-methyl, 2,6-dimethyl-4-heptanone, and 2-octene. KEGG states that 2-methyl-2-butenal is a derivative of crotonaldehyde, which comes from crotonoyl-coenzyme A (CoA).[22] Crotonyl-CoA is part of the butanoate metabolism

of carbohydrates and also is essential in the production of the amino acids lysine and tryptophan.[22] Thus, the infected pigs could have a reduction in carbohydrate metabolism due to the absence of 2-methyl-2-butenal. A product of butyric acid and uniquely found in the healthy cohort, 2,6-dimethyl-4-heptanone is generated by anaerobic bacteria fermentation.[22] Butyrate and other short chain fatty acids (SCFA) are end products of fermentation.[22] Moreover, SCFA concentrations are higher in pigs with diets high in fermentable carbohydrates.[22] The number of open reading frames identified for carbohydrate metabolism was reduced in the infected pigs when compared to the healthy pigs.[22] Therefore, absence of 2,6-dimethyl-4-heptanone in the infected pigs could signify that the *T. suis* prevents the microbiota to utilize carbohydrates. The metabolite 2-octene, which was absent in the infected pigs, is known to also be missing in human patients with an inflamed colon.[22] Additionally, 2-octene is absent in those who are infected with *Campylobacter jejuni*, bacteria known to cause food poisoning.[31,95] It can be speculated that the lack of 2-octene could provide reasoning as to why the infected pigs developed an inflamed intestinal tract since this metabolite is also absent in humans with an inflamed GI tract, however, further studies are required.

Table 13. At or above a cutoff score of 90%, metabolites that are unique to the healthy pig cohort.

CAS number	Metabolite
1115113	2-Butenal, 2-methyl-
108838	4-Heptanone, 2,6-dimethyl-
111671	2-Octene
75092	Methylene Chloride
1191997	2,3-Dihydrofuran
623427	Butanoic acid, methyl ester

98000	2-Furanmethanol
111273	1-Hexanol
7492388	4-Octanone, 2-methyl-
35447995	Bicyclo[4.2.0]octa-1,3,5-trien-7-ol
620235	Benzaldehyde, 3-methyl-
103093	Acetic acid, 2-ethylhexyl ester
591786	2-Hexanone
10281568	R(-)3,7-Dimethyl-1,6-octadiene
128370	Butylated Hydroxytoluene
107028	2-Propenal
EPA-330476	1,2-Benzenediol, O-(1-naphthoyl)-O'-pentadecafluorooctanoyl-
EPA-307828	1-Naphthoic acid, 3,4-dichlorophenyl ester
EPA-325942	1,2-Benzenediol, O-(1-naphthoyl)-
EPA-331186	1-Naphthoic acid, 2-formyl-4,6-dichlorophenyl ester
EPA-307829	1-Naphthoic acid, 4-nitrophenyl ester

Conclusion

Analysis of VOCs extracted from pig feces, which was in collaboration with the USDA, was to determine the effects of the whipworm, *T. suis*, on metabolites and consequently the intestinal metabolome. This study showed that there are harmful side effects to the GI tract when the helminth parasite is introduced, such as decreased lysine production. This result seen in the infected pigs could be due to the lack of 2-methyl-2-butenal, a metabolite that assists in lysine production. Knowing the unique metabolites extracted from samples of each cohort, some of the deficiencies can be rationalized. Oleic acid was the most noteworthy metabolite that was unique to the infected pigs, while 2-butenal, 2-methyl, 2,6-dimethyl-4-heptanone, and 2-octene were analytes of interest identified in the healthy cohort.

The PCA plot comparing the infected and healthy samples did not show great separation between the two cohorts. Moreover, the individual samples had little to no amounts of clustering, although some form of grouping was seen in the infected samples. This signifies large amounts of variance in the data set between the metabolites extracted from the healthy and infected pigs.

Overall, these results and secondary effects in the GI tract help in understanding the interactions between the host, microflora, and whipworm. Furthermore, this data can be used to optimize techniques where parasites are employed to treat inflammation due to IBD in pigs and humans. One of the proposed methods to remedy the lack and shift of important metabolites would be to simultaneously administer dietary supplements with the helminth parasite to the patient.

PART III - IMPROVEMENTS: MULTIPLE-EXTRACTION DEVICE

Introduction

As stated in Part I, to extract a full range of analytes from the sample headspace, it is a necessity that multiple SPME fibers with different polymer chemistries are utilized.[8] While this method improves breadth of the VOC metabolome, sample requirements and throughput become restrictive. To address these issues, a multiple-extraction device (MED) (Patent #20120264227) was developed to permit the simultaneous extraction of VOCs from the sample headspace using multiple fibers. Coined simultaneous-multiple h-SPME (SIMULTI h-SPME) in this thesis, sample throughput would potentially be increased while maintaining analytical capability and ensuring the collection of comprehensive metabolomes.

If an analysis called for VOCs to be extracted from a sample using three different fibers, three different aliquots would need to be prepared in three separate vials. Then three separate analyses of those samples would need to be performed, taking into account extraction using the different fibers. If VOCs were extracted from the sample headspace for 20 minutes, the entire analysis, from fiber precondition to sample analysis would be completed in 4 hours. However, using the MED coupled with the SIMULTI h-SPME approach, a three fiber extraction of VOCs from one sample would be completed in 1.5 hours. Not only would this increase sample throughput, it potentially decreases the amount of sample that would need to be aliquoted as well. Exhibited in Figure 1, the

MED and SIMULTI h-SPME steps are executed in the "Sample Prep" stage of a metabolomics pipeline, which is briefly discussed in Part I of this thesis.

Sample preparation includes derivatizing and extracting analytes to optimize the sample to obtain the best results. Examples of sample derivatization include silylation,[88] microwave assisted,[88] and microwave-accelerated derivatization.[88] In terms of analyte extraction, metabolomics investigations generally differentiate non-volatile compounds (analyzed by LC-MS/MS) from those that are volatile (analyzed by GC-MS). Differences between non-volatile compounds and volatile organic compounds require unique extraction techniques. The isolation of nonvolatile metabolites may be accomplished by a variety of techniques, but typically involve solvent extraction,[6,96,97] solid phase extraction (SPE),[98–100] or matrix solid-phase dispersion (MSPD).[101–103] Studies focused on volatile metabolites often employ various complex devices such as spirometers and Tedlar bags[17–20] for trapping of the analytes. Headspace-solid phase microextraction (h-SPME),[4,8,11,12,14,15,21–23] can also be used to isolate VOCs for metabolomics analyses. Both SPE and h-SPME techniques concentrate the sample in order to allow the metabolites to reach detection limits.[98,99] By exploiting differences in the composition of the polymer coatings, VOC metabolomics analyses can be tailored to the analytes of interest.

Chromatographic separation encompasses the instrumentation, column selection, and metabolite identification. Samples can be analyzed using NMR;[6] however, a chromatography approach coupled with a mass spectrometer detector (MSD) has greater sensitivity and greater metabolome coverage. Typically, reversed-phase columns,[104] such as C18 alkyl chains,[98] are used in LC to separate non-polar compounds due to the

hydrophobic stationary phase. Conversely, hydrophilic-interaction columns (HILIC),[104] such as pentafluorophenylpropyl (PFPP) columns,[105] are utilized if the sample consists of polar compounds. Non-polar columns, such as DB5-MS capillary column, can be used to separate primarily non-polar analytes in an investigation using GC.[22] Polar analytes can be separated using an HP-INNOWAX capillary column, a polar GC column.[106] Analytes are then typically detected by either flame ionization[8,22] or MSD.[6,8,22,23] Mass analyzers, such as a quadrupole,[88] triple quadrupole (QqQ),[104] and a quadrupole time-of-flight (QTOF),[104] are coupled with the detector to separate and sort the ions. Metabolite identification of the detected analytes can be completed through the Automated Mass Spectral Deconvolution Identification System (AMDIS),[22] MathDMAP,[104] MetAlign,[104] MZMine,[104] and XCMS.[104] Once the samples have been analyzed, the compounds need to be classified by matching the resultant mass spectra against those in databases. Finally, data analysis focuses on the normalization and statistics of the metabolites to gather conclusions about the samples. Common statistical analyses for metabolomics studies include principal component analysis (PCA),[6,14,23,72] and partial least squares-discriminant analysis (PLS-DA).[72,107,108]

Testing the analytical capabilities of this new technique was completed by extracting decane, naphthalene, and biphenyl at various concentrations from a multiple-analyte cocktail (MAC). VOCs from human fecal samples were extracted and chromatograms were compared to determine if SIMULTI h-SPME is a viable option to extract biological samples and increase sample throughput. Two sets of three different

wine types were sampled to determine if there was a list of metabolites that were unique to wines of lesser quality.

Materials and Methods

The following section describes the procedures to calibrate, operate, and clean the MED. This device can be used to aid with sample throughput for a metabolomics investigation, or to examine a complex sample by analytical means.

Multi-Analyte Cocktail

A master mix cocktail of 0.1 M decane, 0.01 M naphthalene, and 0.01 M biphenyl was made to test the analytical properties of the device. Decane, at a volume of 195.9 µL, 12.8 mg of naphthalene, and 15.4 mg of biphenyl were added to a 10 mL volumetric flask. A decane aliquot of 1 mL was taken from stock solution and then 195.9 µL was added to the master mix. A few mL of ethanol was used to dissolve the naphthalene and biphenyl before the solution was diluted to the 10 mL mark. This master mix was wrapped in aluminum foil and placed in a 4°C refrigerator.

Four dilutions of this master mix were prepared to create a standard curve. The concentrations of these dilutions are displayed in Table 14. These dilutions of the master mix need to be performed at room temperature. All dilutions had small pieces of paper clips acting as stir bars added to the amber vials to create a homogenous mixture while extracting the VOCs.

Dilutions of the MAC were analyzed in increasing concentration to diminish the chance for analyte carryover in the device. An amber vial containing the MAC was placed in the bottom portion of the multiple-extraction device (see Figure 22) and sealed.

Then, the device was placed on a hot plate, which was at a temperature of 60°C and a stir setting of 150 rpm, for the duration of the analyte extraction. The fibers were placed in the device and exposed to the sample for 1 hour. Extractions were completed using the PDMS/DVB 65, DVB/CAR/PDMS 50/30 µm, and CAR/PDMS 75 µm fibers in duplicate.

Table 14. Concentrations of solutions for calibration curve. The Master Mix was prepared in a 100 mL volumetric flask. The Working Solution was prepared in a volumetric flask using the master mix. Each of the four dilutions used for the calibration curve were prepared in 4 mL amber vials using the Working Solution.

	Master Mix	Working Solution	Dilution 1	Dilution 2	Dilution 3	Dilution 4
Decane (M)	0.1	0.01	$6.0x10^{-3}$	$5.0x10^{-3}$	$3.5x10^{-3}$	$2.5x10^{-3}$
Naphthalene (M)	0.01	0.001	$6.0x10^{-4}$	$5.0x10^{-4}$	$3.5x10^{-4}$	$2.5x10^{-4}$
Biphenyl (M)	0.01	0.001	$6.0x10^{-4}$	$5.0x10^{-4}$	$3.5x10^{-4}$	$2.5x10^{-4}$

To test the multi-fiber standard curve, the second weakest dilution was prepared and analyzed in the same fashion as those samples used to create the standard curve. The % error between the abundance obtained and the abundance from the standard curve needed to be 5% or less. Portions of paper clips were cut and added to each vial. These acted as stir bars to create a homogenous mixture while the sample was being extracted. Samples were stirred at 150 rpm.

Fecal Samples

While still frozen, approximately 0.2 g of feces was added to vials in a sterile biosafety cabinet environment. These samples were snap frozen in liquid nitrogen and

stored at -80°C until they were ready to be analyzed. Vials were placed in the bottom piece of the device. The MED was then sealed and placed on a hot plate, which was set to 60°C. Single and SIMULTI fiber extractions using the PDMS/DVB 65 µm, DVB/CAR/PDMS 50/30 µm, and CAR/PDMS 75 µm fibers of the fecal VOCs were performed. Metabolites from fecal samples were extracted for 18 hours.

Wine Samples

Two Zinfandel, two Merlot, and two Chardonnay wines were studied in this investigation. One of each wine type was from Oak Leaf Vineyards (Ripon, CA). The other wines were a Gnarly Head Vintage Old Vine Zinfandel (Lodi, CA), a HRM Rex-Goliath Merlot (Madera, CA), and a Black Swan Chardonnay (Australia). Aliquots of each wine (0.3 mL) were added to amber vials. Samples were stored at -80°C until the analysis was to be performed. Similar to the MAC samples, paper clips were added to each sample to act as stir bars. Sample vials were placed in the bottom piece of the device. The device was sealed and placed on a hot plate, which was set to a temperature of 60°C and a stir setting of 150 rpm. The DVB/CAR/PDMS 50/30 µm, CAR/PDMS 75 µm, and PA 85 µm fibers were used to extract VOCs from wine samples. Analytes from wine samples were extracted for 16 hours. All samples were dispensed into 4 mL WISP style screw thread amber glass vials and sealed with Black Top Hat PTFE/Silicone caps (J.G. Finneran, Vineland, NJ).

Headspace Solid-Phase Microextraction Procedure

SPME fibers were preconditioned for 10 minutes each at 250°C. Fibers were then inserted into the MED and exposed to the headspace of the sample to collect VOCs for

the allotted period of time. After the analyte extraction period concluded, the SPME fiber was removed from the MED and inserted into the heated GC inlet port. Analytes were allowed to desorb onto the head of the column for 10 seconds. Analysis of the VOCs, using the GC-MS, began directly after the 10 second desorption.

Instrument

Samples were analyzed using an Agilent 7890A GC and 5975C inert XL mass selective detector (MSD) with triple axis detector (Agilent, Palo Alto, CA). The GC-MS was equipped with a DB5-MS capillary column (Agilent), 30 m in length, 0.25 mm ID, and 0.25 mm film thickness, and a 0.75 mm ID SPME injection port liner operated in splitless mode at varying inlet temperatures, dependent upon the SPME fiber used.

GC-MS conditions

Three different analysis methods were used for the GC-MS. The first method was to analyze the MAC. Here, the helium carrier gas was set to 69.3 mL/min flow rate and the GC oven was held at an initial temperature of 35°C for 1 min, ramped to 80°C at 3°C /min, then to 120°C at 10°C /min, and finally to 260°C at 40°C /min. The final temperature of 260°C was held for 1.5 min. The inlet temperature for this method was 240°C. The total run time for the analysis method was 25.0 min.

To analyze the fecal samples, the helium carrier gas was set to 53.3 mL/min flow rate and the GC oven was held at an initial temperature of 35°C for 1 min, ramped to 80°C at 3°C /min, then to 120°C at 10°C /min, and then to 260°C at 40C/min held for 1.50 min, finally to 290°C at 30 °C /min, held at 5 min. The final temperature of 290°C

was held for 5.0 min. The inlet temperature for this method was 250°C. The total run time for the analysis method was 31.0 min.

Finally, to analyze the wine samples, the helium carrier gas was set to 69.3 mL/min flow rate and the GC oven was held at an initial temperature of 35°C for 1 min, ramped to 60°C at 5°C /min, then to 150°C at 35°C /min, and then to 270°C at 8°C/min. The final temperature of GC oven was 270°C. The inlet temperature for this method was 250°C. The total run time for the analysis method was 23.57 min.

The Agilent 5975C MSD was scanned from 30 to 550 amu at a rate of 2.81 scans/s. All compounds in each of the three sample types were identified using the National Institute of Standards and Technology (NIST, Washington, DC) Automated Mass Spectral Deconvolution and Identification System (AMDIS, ver 2.69) software and mass spectral library (NIST08). To be qualified for identification, the probability of the analyte matching a molecule the NIST library needed to be 85% or greater. This analyte identification procedure is described in detail above (see Part I – Applications: Fecal VOCs and Alcohol Liver Disease).

SIMULTI Headspace Solid-Phase Microextraction Procedure

In the SIMULTI h-SPME fiber analysis, after analyte extraction was complete, each fiber was inserted into the heated inlet port of the GC for 10 seconds. The fibers were kept in the multiple-extraction device and were exposed to the sample until ready for analyte desorption. This SIMULTI h-SPME fiber desorption process allowed identical VOCs to accumulate at the head of the column before the instrument method was started.

Cleaning and Storage of the Multiple-Extraction Device

After the fibers were analyzed, the device was disassembled and the sample was removed from the bottom piece. A propane flame was applied to the faces of the bottom two pieces that touch each other to remove any VOCs that may have adsorbed to the aluminum in the bottom two pieces. All of the septa were removed from the middle piece during this process, cleaned with water, and dried with Kimwipes. The device was then placed back on the hot plate and allowed to cool to 60°C before the new sample was to be analyzed.

At the end of each day, the torch was applied to the device and placed on a hot plate in the Volatile Extraction Chamber. Both the hot plate and chamber are at a constant temperature of 60°C.

Statistical Analysis: Principal Component Analysis

A matrix containing samples from Oak Leaf vs. other wineries and the metabolites with corresponding values was produced. Data filtering, explained in detail above (see Part I – Applications: Fecal VOCs and Alcohol Liver Disease) was performed on this matrix. The filtered data was then analyzed by PCA as described above (see Part I – Applications: Fecal VOCs and Alcohol Liver Disease).

Results and Discussion

The following sections are results for the method development of the MED, which include the MAC dilutions, fecal samples, and wine samples.

MED Description

As seen in Figure 22A, the assembled extraction device was made out of an aluminum block and metal bolts. Figure 22B shows the block was separated into 3 pieces.

The bottom holds the sample, an uncovered 4 mL amber vial. The middle section has 8 ports that run the length of the section to allow the fibers to extend and become exposed to the sample. The area between these pieces acts as the sample headspace where the VOCs can pool and become easier to extract. At the top of the middle section sit rubber septa to trap the volatiles in the device while allowing SPME fibers to breach the cover and extract analytes. The top portion has 8 holes, which line up with the 8 holes in the middle portion, to insert and stabilize the fibers into the device.

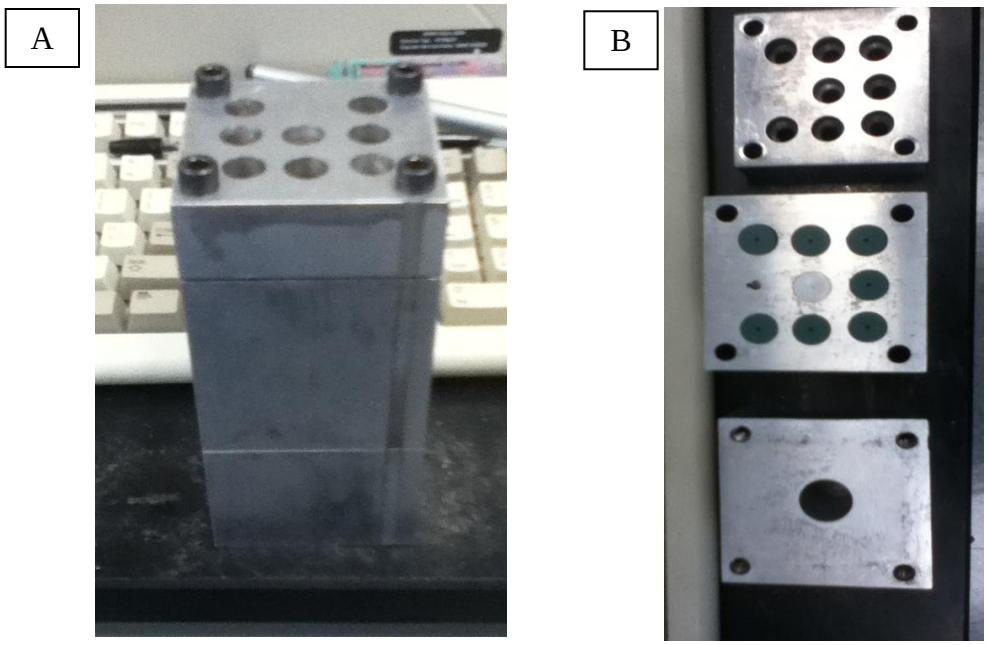


Figure 22. Multiple-Extraction Device. A) The assembled extraction device. An Allen wrench is used to screw the four bolts into place and seal the device. B) The MED separated into 3 pieces. The top portion is used to stabilize the SPME fibers when they are exposed in the device. The green circles in the middle piece are rubber septa, similar to ones that cover a GC inlet. These septa help contain the VOCs in the MED and can be removed for cleaning. The sample vial sits in hole in the bottom piece.

Multiple-Analyte Cocktail (MAC)

Analysis of the MAC, consisting of decane, naphthalene, and biphenyl was executed using the MED, coupled with h-SPME and SIMULTI h-SPME fiber extractions. Standard curves of these results then were produced. The 65 µm DVB-PDMS, 50/30 µm CAR-DVB-PDMS, and 85 µm CAR-PDMS fibers were used to extract the analytes from the sample headspace. Figure 23, Figure 24, and Figure 25 show the standard curves of the decane, naphthalene, and biphenyl in the MAC extracted by the SIMULTI h-SPME fiber technique. The standard curves in Figure 23, Figure 24, and Figure 25 were used to calibrate the device. These calibration results are displayed in Table 15.

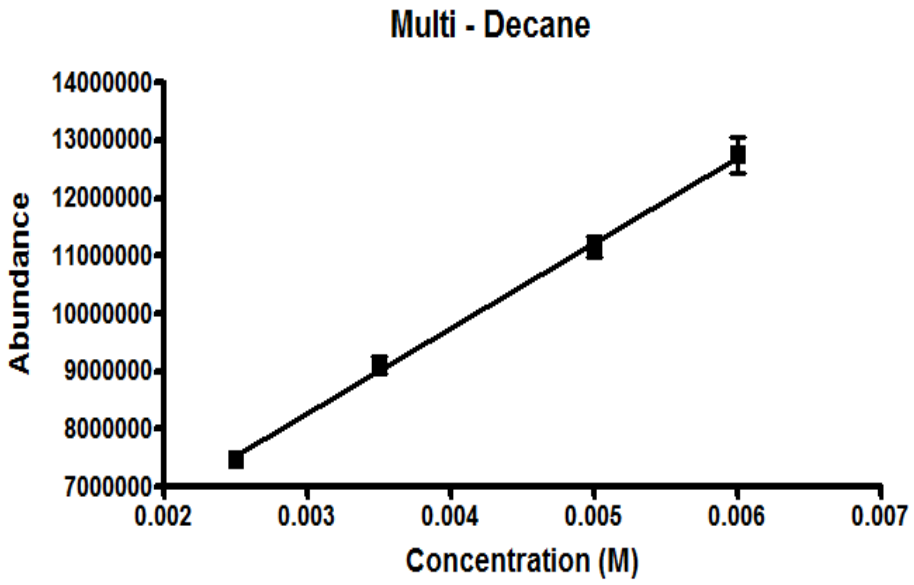


Figure 23. Standard curve of the 65 µm DVB-PDMS, 50/30 µm CAR-DVB-PDMS, and 85 µm CAR-PDMS fibers exposed to decane. The R^2 value for this curve was 99.90%. VOCs were extracted for 1 hour in the MED using the SIMULTI h-SPME technique.

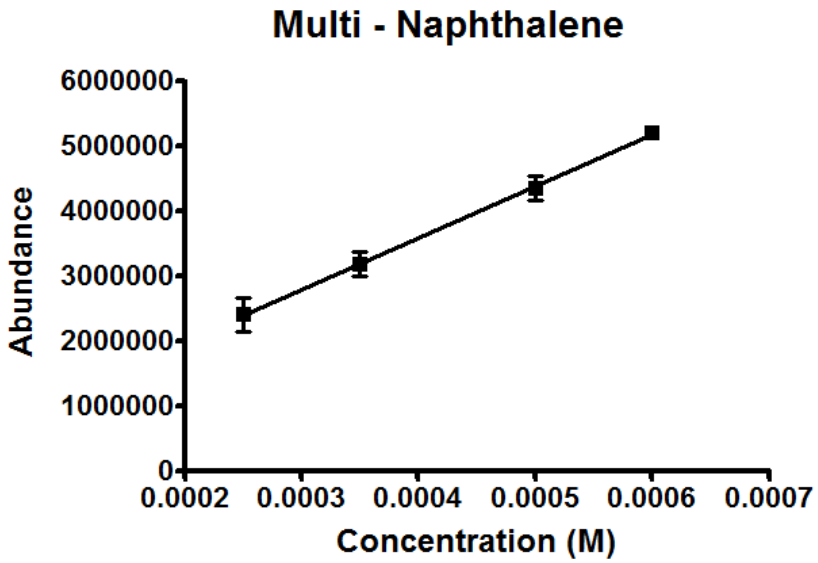


Figure 24. Standard curve of the 65 µm DVB-PDMS, 50/30 µm CAR-DVB-PDMS, and 85 µm CAR-PDMS fibers exposed to naphthalene. The R^2 value for this curve was 99.95%. VOCs were extracted for 1 hour in the MED using the SIMULTI h-SPME technique.

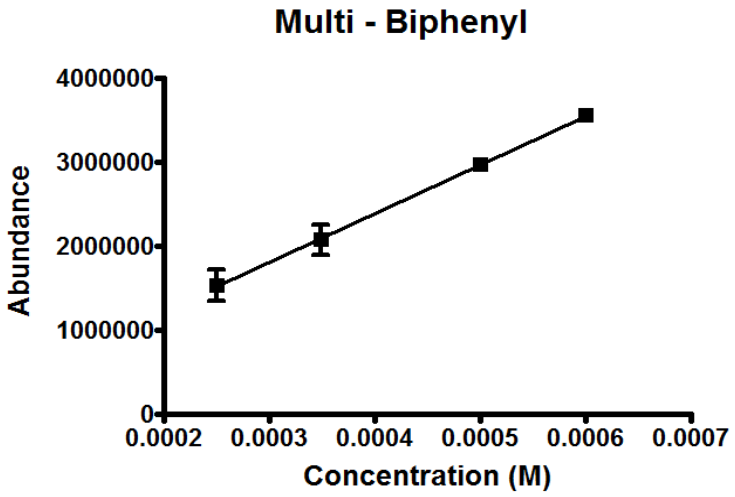


Figure 25. Standard curve of the 65 µm DVB-PDMS, 50/30 µm CAR-DVB-PDMS, and 85 µm CAR-PDMS fibers exposed to biphenyl. The R^2 value for this curve was 99.97%. VOCs were extracted for 1 hour in the MED using the SIMULTI h-SPME technique.

Table 15. Concentrations and % Errors of Decane, Naphthalene, and Biphenyl in the multi-analyte cocktail. The concentration of decane should be $3.5x10^{-3}$M, while the concentrations of naphthalene and biphenyl should be $3.5x10^{-4}$M.

Analyte	Concentration (M)		
Decane	3.51E-03	3.54E-03	3.66E-03
Naphthalene	3.31E-04	3.45E-04	3.57E-04
Biphenyl	3.32E-04	3.35E-04	3.40E-04
Analyte		**% Error**	
Decane	**0.37%**	**1.23%**	**4.51%**
Naphthalene	**5.44%**	**1.54%**	**1.98%**
Biphenyl	**5.06%**	**4.22%**	**2.91%**

Figure 26-Figure 34 are to show that when the 65 µm DVB-PDMS, 50/30 µm CAR-DVB-PDMS, and 85 µm CAR-PDMS fibers are individually exposed to the MAC at specific concentrations, then the abundance will increase in a linear fashion. Figure 35

depicts the chromatogram of the three analytes after a SIMULTI fiber extraction using the 65 µm DVB-PDMS, 50/30 µm CAR-DVB-PDMS, and 85 µm CAR-PDMS fibers.

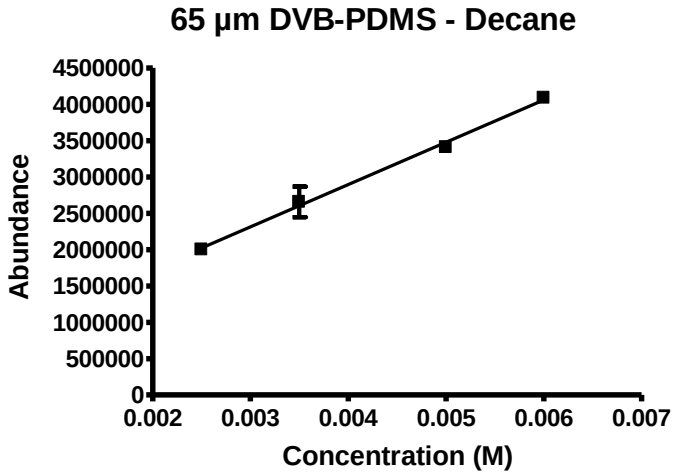


Figure 26. Standard curve of the 65 µm DVB-PDMS fiber exposed to decane. VOCs were extracted for 1 hour in the MED using a single fiber.

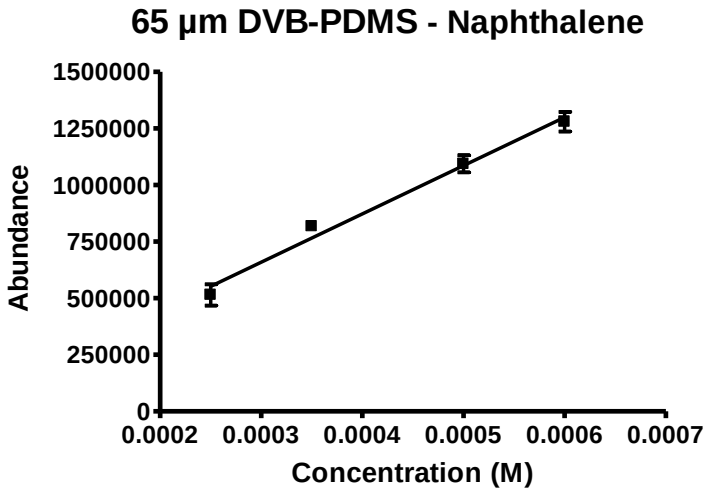


Figure 27. Standard curve of the 65 µm DVB-PDMS fiber exposed to naphthalene. VOCs were extracted for 1 hour in the MED using a single fiber.

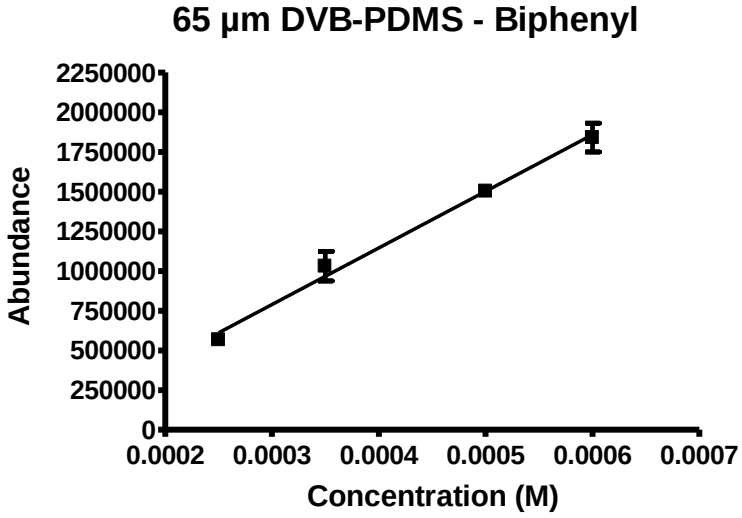


Figure 28. Standard curve of the 65 µm DVB-PDMS fiber exposed to biphenyl. VOCs were extracted for 1 hour in the MED using a single fiber.

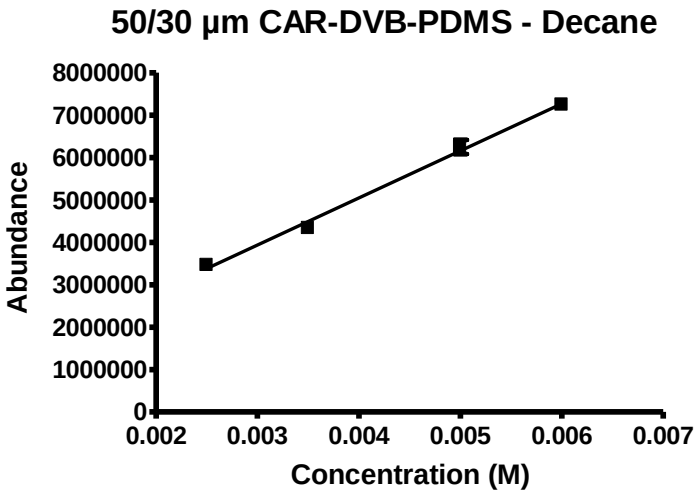


Figure 29. Standard curve of the 50/30 µm CAR-DVB-PDMS fiber exposed to decane. VOCs were extracted for 1 hour in the MED using a single fiber.

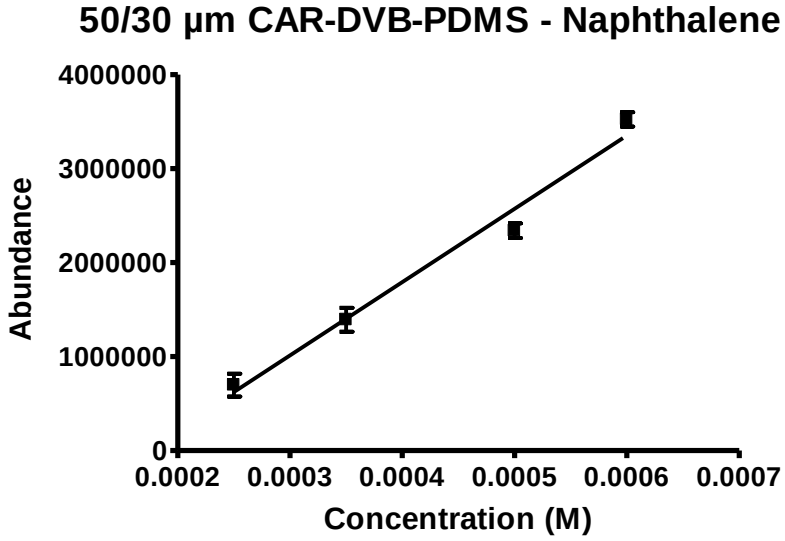


Figure 30. Standard curve of the 50/30 µm CAR-DVB-PDMS fiber exposed to naphthalene. VOCs were extracted for 1 hour in the MED using a single fiber.

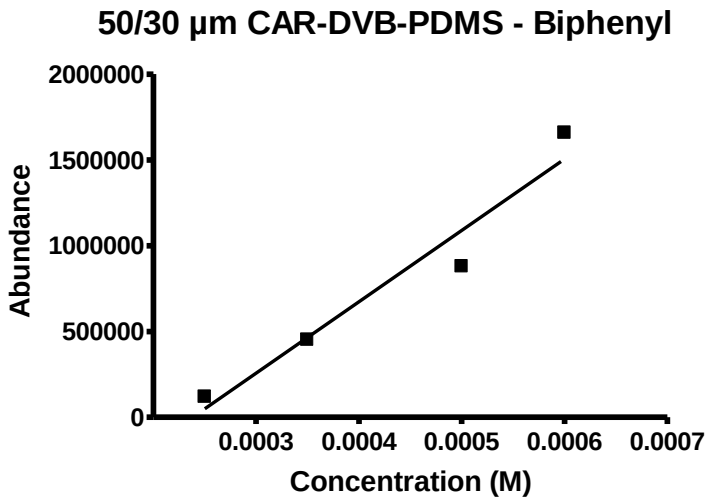


Figure 31. Standard curve of the 50/30 µm CAR-DVB-PDMS fiber exposed to biphenyl. VOCs were extracted for 1 hour in the MED using a single fiber.

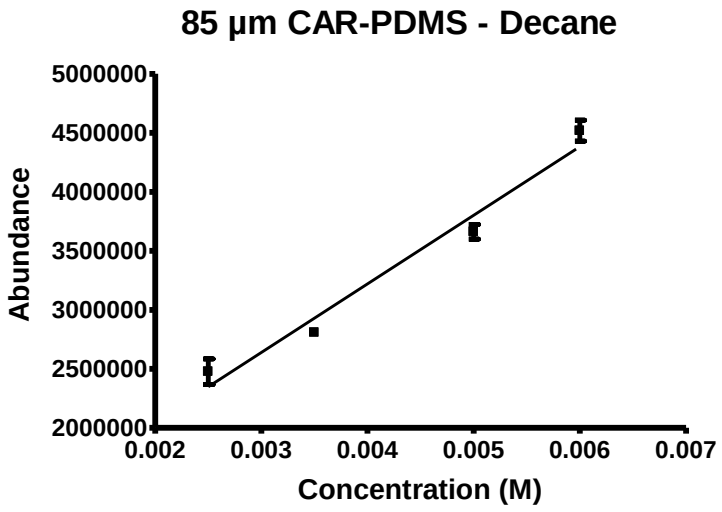


Figure 32. Standard curve of the 85 µm CAR-PDMS fiber exposed to decane. VOCs were extracted for 1 hour in the MED using a single fiber.

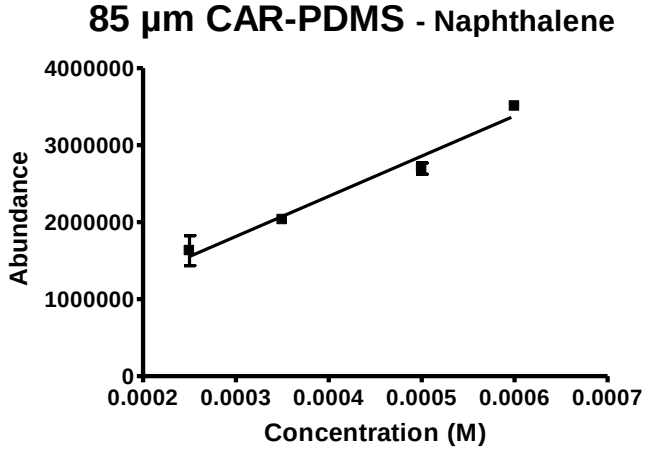


Figure 33. Standard curve of the 85 µm CAR-PDMS fiber exposed to naphthalene. VOCs were extracted for 1 hour in the MED using a single fiber.

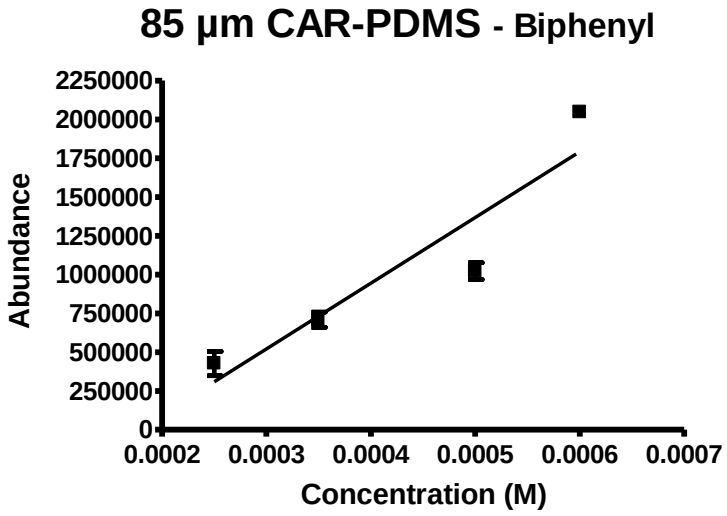


Figure 34. Standard curve of the 85 µm CAR-PDMS fiber exposed to biphenyl. VOCs were extracted for 1 hour in the MED using a single fiber.

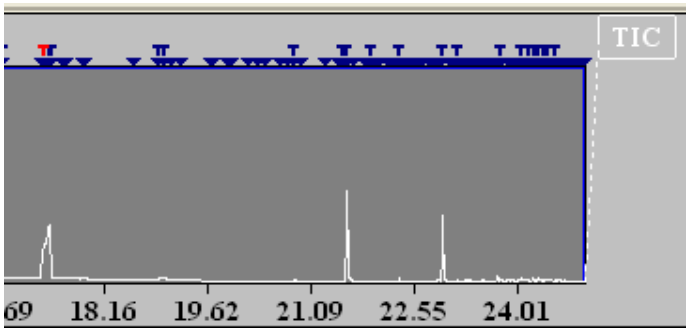


Figure 35. Chromatogram of the 65 µm DVB-PDMS, 50/30 µm CAR-DVB-PDMS, and 85 µm CAR-PDMS fibers exposed to decane, naphthalene, and biphenyl. The retention times of decane, naphthalene, and biphenyl are 17.7, 21.6, and 23.1 minutes, respectively. VOCs were extracted for 1 hour in the MED using a SIMULTI h-SPME technique.

The MED can be used to analytically quantify concentrations of analytes in a complex mixture using the SIMULTI h-SPME fiber technique. These results are seen in Table 15. The device was calibrated and a known concentration was prepared and extracted in the device. The VOCs extracted from the mixture, correlated to % errors of 5% or less. In all three trials, decane produced a higher signal than that used to make the calibration curve, therefore extracting a concentration greater than $3.5x10^{-3}$M. Naphthalene and biphenyl signals were lower than expected except for the naphthalene level in the third trial. The naphthalene and biphenyl parts of the solution were made from solids while decane was prepared from a liquid sample. The two solids may have degraded quicker than decane, thus producing slightly lower signals.

Fecal Samples

Figure 36A-D shows the chromatograms of fecal VOCs extracted in the MED for 18 hours. Figure 36A-C are the chromatograms of after the 65 µm DVB-PDMS, 50/30 µm CAR-DVB-PDMS, and 85 µm CAR-PDMS fibers individually extracted VOCs from fecal samples, respectively. Figure 36D displays the chromatogram of all three fibers simultaneously exposed to a fecal sample.

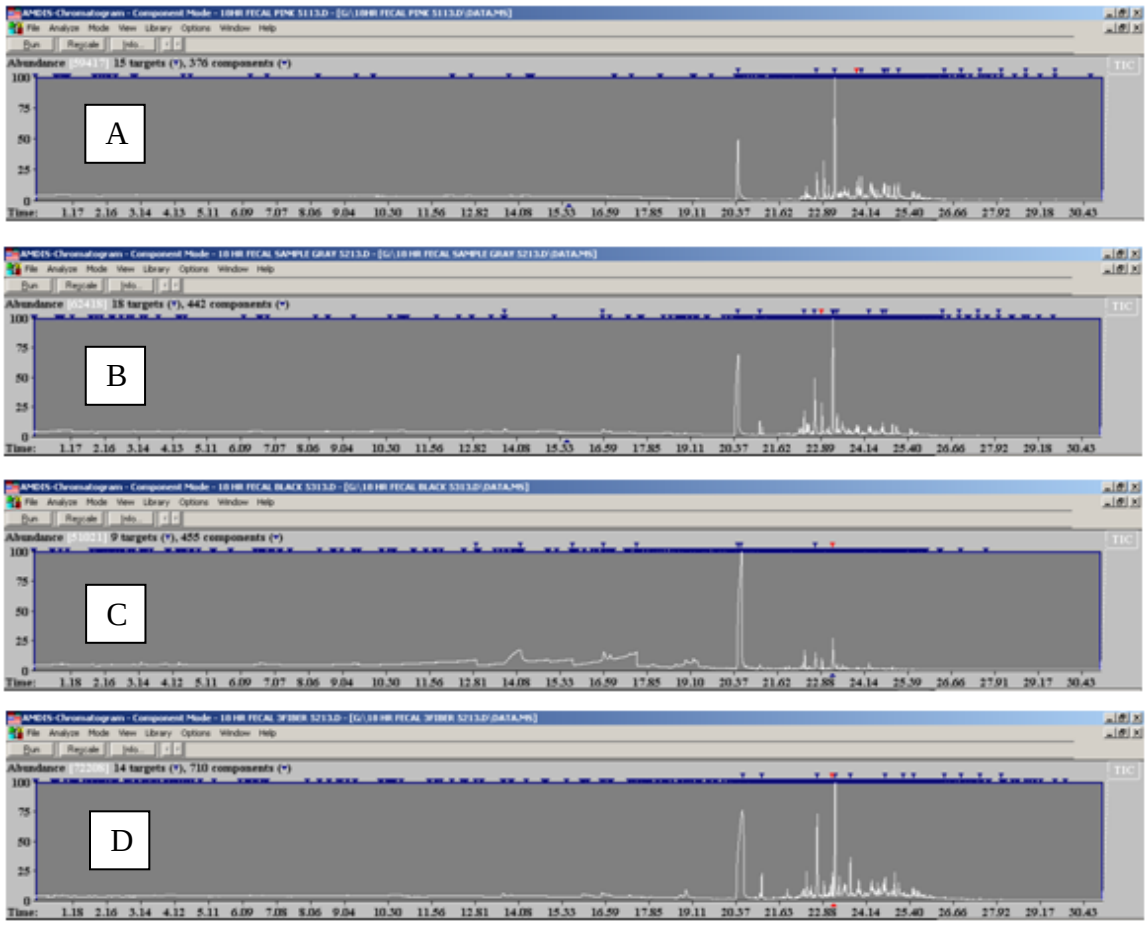


Figure 36. Chromatograms of fecal samples extracted in the MED. SPME fibers were exposed to the samples for 18 hours. A) Chromatogram of the 65 µm DVB-PDMS fiber exposed to fecal sample. B) Chromatogram of the 50/30 µm CAR-DVB-PDMS fiber exposed to fecal sample. C) Chromatogram of the 85 µm CAR-PDMS fiber exposed to fecal sample. D) Chromatogram of the 65 µm DVB-PDMS, 50/30 µm CAR-DVB-PDMS, and 85 µm CAR-PDMS fibers exposed to fecal sample.

As seen in Figure 36D, the resultant chromatogram after a SIMULTI h-SPME fiber extraction of fecal VOCs, the displayed peaks are at greater amplitudes than those shown in the individual fiber extractions. This is due to an increased number of binding sites on the fibers compared to a single fiber. Thus, more VOCs can be extracted from the sample and analyzed on the GC/MS. The peak at the 20.37 minute mark exemplifies this

well. In the individual fiber extractions, shown in Figure 36A-C, this peak is smaller in comparison to the peak at the same retention time in Figure 36D. In addition to the greater peak amplitude, the chromatogram in Figure 36D contains an increased number of peaks when compared to the chromatogram shown in Figure 36C, a fecal VOC extraction using the 85 µm CAR-PDMS fiber. If only the 85 µm CAR-PDMS fiber were to be used in the analysis, it is possible that many analytes that would not be extracted from the sample. This could lead to false conclusions in regards to analyzing samples for metabolomics purposes. Therefore, the process of SIMLUTI h-SPME fiber extractions using the MED can reduce the amount of time and increase the throughput for analyzing biological samples. Furthermore, the amount of sample aliquots needed for the analysis is decreased.

Wine Samples

Two sets of zinfandels, merlots, and chardonnays from different wineries were studied using the MED and SIMULTI h-SPME fiber technique. The SPME fibers used were the DVB/CAR/PDMS 50/30 µm, CAR/PDMS 75 µm, and PA 85 µm fibers. This was to ascertain if it was possible to differentiate wines from various wineries and determine a unique metabolomic fingerprint from the Oak Leaf wines. A total of 551 metabolites were identified from the wine samples. Figure 37 displays the PCA plot for the wine analysis. This plot shows that the two cohorts segregate from one another due to the amount of variance seen in the metabolite levels. Table 16 shows the metabolites that were uniquely found in the Oak Creek wines. Post data filtering, 43 of the 287 metabolites in the data set were only found in samples from the Oak Creek winery.

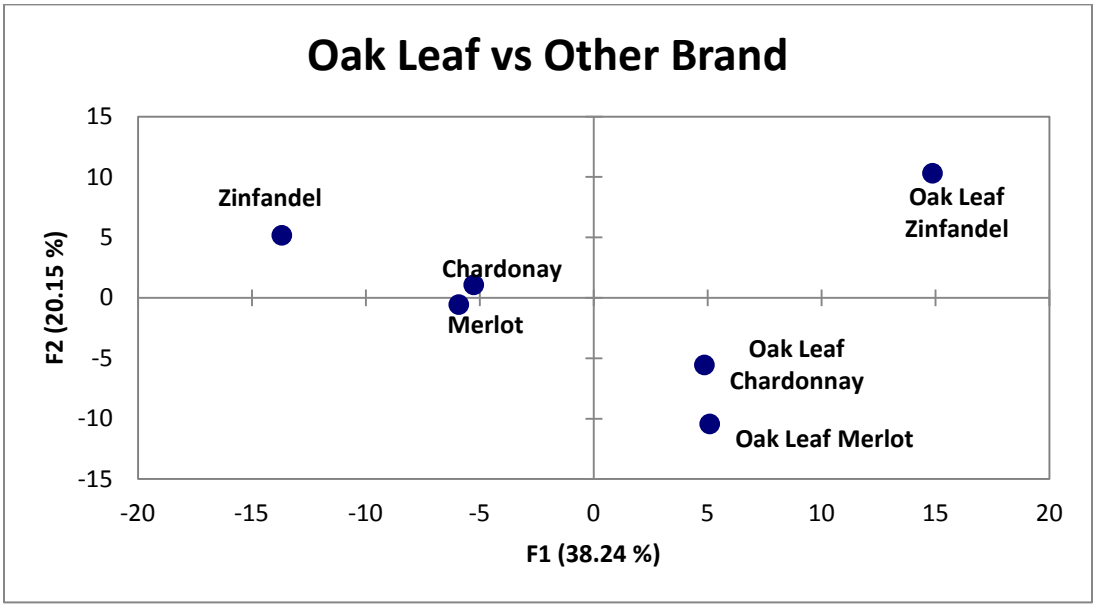


Figure 37. PCA plot for wine analysis. This PCA plot displays the products of the different wineries, Oak Leaf vs. Other Brand. Utilizing the MED and SIMULTI h-SPME technique, SPME fibers were exposed to the wine samples for 16 hours. The results show that the two cohorts clearly segregate from one other, indicating metabolite variance in the data set.

These metabolites, listed in Table 16, cause separation between the cohorts and have squared cosine values >90%. This list was provided by XLSTAT as additional results for the PCA analysis. A squared cosine value indicates the contribution of specific metabolites on how the samples are plotted on the PC. It is derived from the cosine of the angle between the data point, origin, and PC axis. A metabolite with a large squared cosine value has a greater influence on the positioning of the samples with respect to the specific PC. All of the metabolites listed in Table 16 have high squared cosine values in relation to PC1. These metabolites dictate that the Oak Creek samples will be plotted with higher importance in relation to PC1.[109,110] Therefore, by using a SIMULTI h-SPME

fiber approach to extract VOCs from a sample in the MED, metabolites indicative of lesser quality wines were identified.

Table 16. Metabolites unique to Oak Wines causing separation between the cohorts (squared cosine values >90%)

Cyclopentane, pentyl-	2-Dodecene, (Z)-
n-Dodecyl acetate	Cyclohexadecane
Ethane	Dichloroacetic acid, dodecyl ester
Cyclotridecane	E-11,13-Tetradecadien-1-ol
á-1,5-O-Dibenzoyl-ribofuranose	Cyclodecane
1-Dodecanethiol	1-Undecene
5-Eicosene, (E)-	2-Tetradecene, (E)-
Fluoroacetic acid, dodecyl ester	Benzoic acid
Trifluoroacetic acid, n-octadecyl ester	Cyclobutane-1,1-dicarboxamide, N,N'-di-benzoyloxy-
2-Butenedioic acid (Z)-, monododecyl ester	Cyclotetradecane
Pentadecane	1,2-Benzenedicarboxylic acid, butyl 2-ethylhexyl ester
2-Chloroethyl benzoate	Pentafluoropropionic acid, dodecyl ester
Cyclopropane, octyl-	Cyclopentadecane
Pentadecane, 2,6,10-trimethyl-	(d)-(+)-(2R,3R)-2,3-Dibenzoyltartaric acid
9-Eicosene, (E)-	Tetradecane, 6,9-dimethyl-
1-Undecanol	Cyclodecane, methyl-
Octadecane	1-Tridecanol
Heneicosane	Dichloroacetic acid, tridecyl ester
Nonadecane	Cycloeicosane
Cyclododecane	2-Butenal, (E)-
1-Decanol	1-Tetradecene

Conclusion

Multiple SPME fibers with different polymer chemistries are needed to extract the greatest amount of analytes from a sample. Although this approach yields a complete set

of results, limitations, such as time and sample amount, hinder the proficiency of the analysis. A MED was fabricated from an aluminum block to allow for a SIMULTI h-SPME fiber extraction and was tested with MAC dilutions, feces, and wine samples. The results show that the use of the MED can increase throughput of analyzing complex biological and food samples, displayed through the fecal and wine samples. This could especially be beneficial in a metabolomics study. Using this approach also allows for the reduction of sample aliquots, which could be in limited supply. Additionally, this technique still maintains analytical specifications, exhibited through the standard curves and analysis of the MAC.

OVERALL CONCLUSION

VOC metabolomics is a growing field with the increase and development of technology to separate and analyze molecules of interest from a sample. Alcohol liver disease is currently diagnosed through invasive procedures. Analyzing human fecal VOC metabolites, collected after home passage, showed enough differences in metabolite identities and values, between the healthy and alcoholic cohorts, to indicate an alcoholic patient. Fecal VOCs from porcine infested with the whipworm, *Trichuris suis*, showed absences of vital cofactors for carbohydrate and lysine synthesis, and a surplus of oleic acid. This caused fatty acids to be absorbed, leading to an inflamed colon. Results from this study can be used to optimize treatments of IBD using helminth parasites. A multiple extraction device was fabricated to address the issue of sample throughput limiting an analysis. This device implemented the use of a SIMULTI h-SPME extraction technique that expedites sample analysis while still preserving analytical regulations. Therefore, the work completed in this thesis greatly exhibits progress in the field, primarily through the use of h-SPME to extract metabolites from sample headspaces, which are then analyzed by GC/MS.

APPENDIX A

```
1. #!usr/bi    n/perl -w
2. #use strict 'refs';
3. use warnings;

4. #use diagnostics;

5. #author: Karl A. Navarro
6. #file: group_metabolites_47.pl
7. #date: 11/6/12

8. #purpose: group metabolites into their specific chemical classes
   (alcohols, esters, ketones, etc.)
9. # and obtain sums of their peak abundance obtained in the GC-MS
   analysis

10.#obtain input on filename

11.$filename = $ARGV[0];   #the user to inputs the filename

12.# Open the file given as the first argument on the command line
13.#the tab delimited files need to have samples across the top in
   every column (for this code, it needs 47 samples) and the
   metabolites down the rows.
14.open(INFILE, $filename) or die "Can't open $filename\n";

15.#variable declarations

16.my $label_name = "Label";
17.my $line;
18.#my $metabolite_counts = 0;   #initialize number of metabolite
   count
19.my $alcohol   = "Alcohol";    #ends in -ol is an alcohol
20.my $acid      = "Acid";       #this will define the chemical
   classes on the left most column
21.my $aldehyde  = "Aldehyde";
22.my $ester     = "Ester";
23.my $alkene    = "Alkene";
24.my $alkane    = "Alkane";
25.my $alkyne    = "Alkyne";
26.my $thiol     = "Thiol";
27.my $sulfide   = "Sulfide";
```

```
28. my $indole    = "Indole";
29. my $ketone   = "Ketone";
30. my $nitrile   = "Nitrile";
31. my $amine     = "Amine";
32. my $other     = "Other";    #other metabolites
33. my $alcsum    = 0;            #ends in -ol is an alcohol. starts
    the alc sum count
34. my $othersum  = 0;            #the other category is where if the
    metabolite name isn't caught by one of the regular expressions
35. # from the other classes, it will go here. Starts the other sum
    count
36. my $acidsum   = 0;            #starts the acid count
37. my $aldsum    = 0;            #starts the aldehyde count
38. my $estersum  = 0;            #starts the ester count
39. my $alkenesum = 0;            #starts the alkene count
40. my $alkanesum = 0;            #starts the alkane count
41. my $alkynesum = 0;            #starts the alkyne count
42. my $thiolsum  = 0;            #starts the thiol count
43. my $sulfsum   = 0;            #starts the sulfide count
44. my $indolesum = 0;            #starts the indole count
45. my $keysum    = 0;            #starts the keytone count
46. my $nitsum    = 0;            #starts the nitrile count
47. my $aminesum  = 0;            #starts the amine count
48. my $total_metabolites = 0;    #initialize metabolite count
49. #my $row_count        = 0;    #initialize count for the rows
50. #my @new_array = ( );
51. my @out_array = ();           #initialize out_array where the
    output of the algorithm will go
52. #my $new_array;
53. my $Chemical_classes = "Chemical Classes"; #hard codes "Chemical
    Classes" into the outfile
54. my $Sum_of_abundances = "Sum of Abundances"; #Hard codes "Sum of
    Abundances" into the outfile
55. my $sample_name = "Sample"; #hard codes "Sample" into the outfile
56. #my @final_array = ();
57. #my $out_array;
58. #my $all_samples;
59. my @columns = ();       # initialize an empty columns array,
    we'll fill it with rows later
60. my @row = ();     #initialize an empty rows array
61. my $samples_count = 0; #initialize the count for the number of
    samples

62. while ( my $line = <INFILE> ) {     #while the infile is open,
    we're going to do a couple of things.
63. # 1. set up the 2D array 2. Count the number of metabolites
64. # 3. count the number of rows 4. count the number of samples in
    the file

65. chomp $line;      #remove any excess lines at the end of the
    input file
66. @row = split( /\t/, $line );  # split file at each tab
67. push( @columns, [ @row ] ); # put this row into the columns
    matrix as an array reference
```

```
#get total number of rows
foreach ($row[0])  {
next  if ($row[0] =~ /(^Sample|^Label)/); #skip if line starts with the word "Sample" or "Label"
$total_metabolites++; #increases the count by 1
}

$samples_count = $#{$columns[0]}; #counts the number of samples in the file; essentially the number of rows

}
close( INFILE );  #closes the infile


#define positions in array for matrix of chemical classes and sum of abundances
#this sets up the output array for the outfile
#the out_array is set up row x column. So out_array[0][0] is the very 1st row, very 1st column.
# to put it into excel spreadsheet format, this position would be cell A1.

$out_array[0][0] = "$sample_name";  #Hard codes "Sample name" into position (0,0) into the outarray
$out_array[1][0] = "$label_name";   #Hard codes "Label name" into position (1,0) into the outarray
$out_array[0][1] = "$columns[0][1]"; #row 0 column 1. row 0 (first row in the outfile) has all the sample names
$out_array[0][2] = "$columns[0][2]";
$out_array[0][3] = "$columns[0][3]";
$out_array[0][4] = "$columns[0][4]";
$out_array[0][5] = "$columns[0][5]";
$out_array[0][6] = "$columns[0][6]";
$out_array[0][7] = "$columns[0][7]";
$out_array[0][8] = "$columns[0][8]";
$out_array[0][9] = "$columns[0][9]";
$out_array[0][10] = "$columns[0][10]";
$out_array[0][11] = "$columns[0][11]";
$out_array[0][12] = "$columns[0][12]";
$out_array[0][13] = "$columns[0][13]";
$out_array[0][14] = "$columns[0][14]";
$out_array[0][15] = "$columns[0][15]";
$out_array[0][16] = "$columns[0][16]";
$out_array[0][17] = "$columns[0][17]";
$out_array[0][18] = "$columns[0][18]";
     $out_array[0][19] = "$columns[0][19]";
     $out_array[0][20] = "$columns[0][20]";
     $out_array[0][21] = "$columns[0][21]";
     $out_array[0][22] = "$columns[0][22]";
     $out_array[0][23] = "$columns[0][23]";
     $out_array[0][24] = "$columns[0][24]";
     $out_array[0][25] = "$columns[0][25]";
     $out_array[0][26] = "$columns[0][26]";
     $out_array[0][27] = "$columns[0][27]";
     $out_array[0][28] = "$columns[0][28]";
```

```
110.      $out_array[0][29] = "$columns[0][29]";
111.      $out_array[0][30] = "$columns[0][30]";
112.      $out_array[0][31] = "$columns[0][31]";
113.      $out_array[0][32] = "$columns[0][32]";
114.      $out_array[0][33] = "$columns[0][33]";
115.      $out_array[0][34] = "$columns[0][34]";
116.      $out_array[0][35] = "$columns[0][35]";
117.      $out_array[0][36] = "$columns[0][36]";
118.      $out_array[0][37] = "$columns[0][37]";
119.      $out_array[0][38] = "$columns[0][38]";
120.      $out_array[0][39] = "$columns[0][39]";
121.      $out_array[0][40] = "$columns[0][40]";
122.      $out_array[0][41] = "$columns[0][41]";
123.      $out_array[0][42] = "$columns[0][42]";
124.      $out_array[0][43] = "$columns[0][43]";
125.      $out_array[0][44] = "$columns[0][44]";
126.      $out_array[0][45] = "$columns[0][45]";
127.      $out_array[0][46] = "$columns[0][46]";
128.      $out_array[0][47] = "$columns[0][47]";

129.      $out_array[1][1] = "$columns[1][1]"; #row 1 column 1. Row 1
   has the sample type: alcoholic or healthy
130.      $out_array[1][2] = "$columns[1][2]";
131.      $out_array[1][3] = "$columns[1][3]";
132.      $out_array[1][4] = "$columns[1][4]";
133.      $out_array[1][5] = "$columns[1][5]";
134.      $out_array[1][6] = "$columns[1][6]";
135.      $out_array[1][7] = "$columns[1][7]";
136.      $out_array[1][8] = "$columns[1][8]";
137.      $out_array[1][9] = "$columns[1][9]";
138.      $out_array[1][10] = "$columns[1][10]";
139.      $out_array[1][11] = "$columns[1][11]";
140.      $out_array[1][12] = "$columns[1][12]";
141.      $out_array[1][13] = "$columns[1][13]";
142.      $out_array[1][14] = "$columns[1][14]";
143.      $out_array[1][15] = "$columns[1][15]";
144.      $out_array[1][16] = "$columns[1][16]";
145.      $out_array[1][17] = "$columns[1][17]";
146.      $out_array[1][18] = "$columns[1][18]";
147.      $out_array[1][19] = "$columns[1][19]";
148.      $out_array[1][20] = "$columns[1][20]";
149.      $out_array[1][21] = "$columns[1][21]";
150.      $out_array[1][22] = "$columns[1][22]";
151.      $out_array[1][23] = "$columns[1][23]";
152.      $out_array[1][24] = "$columns[1][24]";
153.      $out_array[1][25] = "$columns[1][25]";
154.      $out_array[1][26] = "$columns[1][26]";
155.      $out_array[1][27] = "$columns[1][27]";
156.      $out_array[1][28] = "$columns[1][28]";
157.      $out_array[1][29] = "$columns[1][29]";
158.      $out_array[1][30] = "$columns[1][30]";
159.      $out_array[1][31] = "$columns[1][31]";
160.      $out_array[1][32] = "$columns[1][32]";
161.      $out_array[1][33] = "$columns[1][33]";
```

```
$out_array[1][34] = "$columns[1][34]";
$out_array[1][35] = "$columns[1][35]";
$out_array[1][36] = "$columns[1][36]";
$out_array[1][37] = "$columns[1][37]";
$out_array[1][38] = "$columns[1][38]";
$out_array[1][39] = "$columns[1][39]";
$out_array[1][40] = "$columns[1][40]";
$out_array[1][41] = "$columns[1][41]";
$out_array[1][42] = "$columns[1][42]";
$out_array[1][43] = "$columns[1][43]";
$out_array[1][44] = "$columns[1][44]";
$out_array[1][45] = "$columns[1][45]";
$out_array[1][46] = "$columns[1][46]";
$out_array[1][47] = "$columns[1][47]";

#Places all of the names of the chemical classes in the 1st column (position 0).
$out_array[2][0] = $alcohol;
$out_array[3][0] = $acid;
$out_array[4][0] = $aldehyde;
$out_array[5][0] = $alkene;
$out_array[6][0] = $alkane;
$out_array[7][0] = $alkyne;
$out_array[8][0] = $thiol;
$out_array[9][0] = $sulfide;
$out_array[10][0] = $indole;
$out_array[11][0] = $ketone;
$out_array[12][0] = $nitrile;
$out_array[13][0] = $amine;
$out_array[14][0] = $other;

#defining the rest of the matrix. every posistion (for every sample and every chemical class) needs to be initialized.
#initially all of the positions have a value of 0.
$out_array[2][1] = $alcsum;
$out_array[3][1] = $acidsum;
$out_array[4][1] = $aldsum;
$out_array[5][1] = $alkenesum;
$out_array[6][1] = $alkanesum;
$out_array[7][1] = $alkynesum;
$out_array[8][1] = $thiolsum;
$out_array[9][1] = $sulfsum;
$out_array[10][1] = $indolesum;
$out_array[11][1] = $keysum;
$out_array[12][1] = $nitsum;
$out_array[13][1] = $aminesum;
$out_array[14][1] = $othersum;

$out_array[2][2] = $alcsum;
$out_array[3][2] = $acidsum;
$out_array[4][2] = $aldsum;
$out_array[5][2] = $alkenesum;
$out_array[6][2] = $alkanesum;
$out_array[7][2] = $alkynesum;
```

```
211.      $out_array[8][2] = $thiolsum;
212.      $out_array[9][2] = $sulfsum;
213.      $out_array[10][2] = $indolesum;
214.      $out_array[11][2] = $keysum;
215.      $out_array[12][2] = $nitsum;
216.      $out_array[13][2] = $aminesum;
217.      $out_array[14][2] = $othersum;

218.      $out_array[2][3] = $alcsum;
219.      $out_array[3][3] = $acidsum;
220.      $out_array[4][3] = $aldsum;
221.      $out_array[5][3] = $alkenesum;
222.      $out_array[6][3] = $alkanesum;
223.      $out_array[7][3] = $alkynesum;
224.      $out_array[8][3] = $thiolsum;
225.      $out_array[9][3] = $sulfsum;
226.      $out_array[10][3] = $indolesum;
227.      $out_array[11][3] = $keysum;
228.      $out_array[12][3] = $nitsum;
229.      $out_array[13][3] = $aminesum;
230.      $out_array[14][3] = $othersum;

231.      $out_array[2][4] = $alcsum;
232.      $out_array[3][4] = $acidsum;
233.      $out_array[4][4] = $aldsum;
234.      $out_array[5][4] = $alkenesum;
235.      $out_array[6][4] = $alkanesum;
236.      $out_array[7][4] = $alkynesum;
237.      $out_array[8][4] = $thiolsum;
238.      $out_array[9][4] = $sulfsum;
239.      $out_array[10][4] = $indolesum;
240.      $out_array[11][4] = $keysum;
241.      $out_array[12][4] = $nitsum;
242.      $out_array[13][4] = $aminesum;
243.      $out_array[14][4] = $othersum;

244.      $out_array[2][5] = $alcsum;
245.      $out_array[3][5] = $acidsum;
246.      $out_array[4][5] = $aldsum;
247.      $out_array[5][5] = $alkenesum;
248.      $out_array[6][5] = $alkanesum;
249.      $out_array[7][5] = $alkynesum;
250.      $out_array[8][5] = $thiolsum;
251.      $out_array[9][5] = $sulfsum;
252.      $out_array[10][5] = $indolesum;
253.      $out_array[11][5] = $keysum;
254.      $out_array[12][5] = $nitsum;
255.      $out_array[13][5] = $aminesum;
256.      $out_array[14][5] = $othersum;

257.      $out_array[2][6] = $alcsum;
258.      $out_array[3][6] = $acidsum;
259.      $out_array[4][6] = $aldsum;
260.      $out_array[5][6] = $alkenesum;
```

```
261.      $out_array[6][6] = $alkanesum;
262.      $out_array[7][6] = $alkynesum;
263.      $out_array[8][6] = $thiolsum;
264.      $out_array[9][6] = $sulfsum;
265.      $out_array[10][6] = $indolesum;
266.      $out_array[11][6] = $keysum;
267.      $out_array[12][6] = $nitsum;
268.      $out_array[13][6] = $aminesum;
269.      $out_array[14][6] = $othersum;

270.      $out_array[2][7] = $alcsum;
271.      $out_array[3][7] = $acidsum;
272.      $out_array[4][7] = $aldsum;
273.      $out_array[5][7] = $alkenesum;
274.      $out_array[6][7] = $alkanesum;
275.      $out_array[7][7] = $alkynesum;
276.      $out_array[8][7] = $thiolsum;
277.      $out_array[9][7] = $sulfsum;
278.      $out_array[10][7] = $indolesum;
279.      $out_array[11][7] = $keysum;
280.      $out_array[12][7] = $nitsum;
281.      $out_array[13][7] = $aminesum;
282.      $out_array[14][7] = $othersum;

283.      $out_array[2][8] = $alcsum;
284.      $out_array[3][8] = $acidsum;
285.      $out_array[4][8] = $aldsum;
286.      $out_array[5][8] = $alkenesum;
287.      $out_array[6][8] = $alkanesum;
288.      $out_array[7][8] = $alkynesum;
289.      $out_array[8][8] = $thiolsum;
290.      $out_array[9][8] = $sulfsum;
291.      $out_array[10][8] = $indolesum;
292.      $out_array[11][8] = $keysum;
293.      $out_array[12][8] = $nitsum;
294.      $out_array[13][8] = $aminesum;
295.      $out_array[14][8] = $othersum;

296.      $out_array[2][9] = $alcsum;
297.      $out_array[3][9] = $acidsum;
298.      $out_array[4][9] = $aldsum;
299.      $out_array[5][9] = $alkenesum;
300.      $out_array[6][9] = $alkanesum;
301.      $out_array[7][9] = $alkynesum;
302.      $out_array[8][9] = $thiolsum;
303.      $out_array[9][9] = $sulfsum;
304.      $out_array[10][9] = $indolesum;
305.      $out_array[11][9] = $keysum;
306.      $out_array[12][9] = $nitsum;
307.      $out_array[13][9] = $aminesum;
308.      $out_array[14][9] = $othersum;

309.      $out_array[2][10] = $alcsum;
310.      $out_array[3][10] = $acidsum;
```

```
311.      $out_array[4][10] = $aldsum;
312.      $out_array[5][10] = $alkenesum;
313.      $out_array[6][10] = $alkanesum;
314.      $out_array[7][10] = $alkynesum;
315.      $out_array[8][10] = $thiolsum;
316.      $out_array[9][10] = $sulfsum;
317.      $out_array[10][10] = $indolesum;
318.      $out_array[11][10] = $keysum;
319.      $out_array[12][10] = $nitsum;
320.      $out_array[13][10] = $aminesum;
321.      $out_array[14][10] = $othersum;

322.      $out_array[2][11] = $alcsum;
323.      $out_array[3][11] = $acidsum;
324.      $out_array[4][11] = $aldsum;
325.      $out_array[5][11] = $alkenesum;
326.      $out_array[6][11] = $alkanesum;
327.      $out_array[7][11] = $alkynesum;
328.      $out_array[8][11] = $thiolsum;
329.      $out_array[9][11] = $sulfsum;
330.      $out_array[10][11] = $indolesum;
331.      $out_array[11][11] = $keysum;
332.      $out_array[12][11] = $nitsum;
333.      $out_array[13][11] = $aminesum;
334.      $out_array[14][11] = $othersum;

335.      $out_array[2][12] = $alcsum;
336.      $out_array[3][12] = $acidsum;
337.      $out_array[4][12] = $aldsum;
338.      $out_array[5][12] = $alkenesum;
339.      $out_array[6][12] = $alkanesum;
340.      $out_array[7][12] = $alkynesum;
341.      $out_array[8][12] = $thiolsum;
342.      $out_array[9][12] = $sulfsum;
343.      $out_array[10][12] = $indolesum;
344.      $out_array[11][12] = $keysum;
345.      $out_array[12][12] = $nitsum;
346.      $out_array[13][12] = $aminesum;
347.      $out_array[14][12] = $othersum;

348.      $out_array[2][13] = $alcsum;
349.      $out_array[3][13] = $acidsum;
350.      $out_array[4][13] = $aldsum;
351.      $out_array[5][13] = $alkenesum;
352.      $out_array[6][13] = $alkanesum;
353.      $out_array[7][13] = $alkynesum;
354.      $out_array[8][13] = $thiolsum;
355.      $out_array[9][13] = $sulfsum;
356.      $out_array[10][13] = $indolesum;
357.      $out_array[11][13] = $keysum;
358.      $out_array[12][13] = $nitsum;
359.      $out_array[13][13] = $aminesum;
360.      $out_array[14][13] = $othersum;
```

```
361.      $out_array[2][14] = $alcsum;
362.      $out_array[3][14] = $acidsum;
363.      $out_array[4][14] = $aldsum;
364.      $out_array[5][14] = $alkenesum;
365.      $out_array[6][14] = $alkanesum;
366.      $out_array[7][14] = $alkynesum;
367.      $out_array[8][14] = $thiolsum;
368.      $out_array[9][14] = $sulfsum;
369.      $out_array[10][14] = $indolesum;
370.      $out_array[11][14] = $keysum;
371.      $out_array[12][14] = $nitsum;
372.      $out_array[13][14] = $aminesum;
373.      $out_array[14][14] = $othersum;

374.      $out_array[2][15] = $alcsum;
375.      $out_array[3][15] = $acidsum;
376.      $out_array[4][15] = $aldsum;
377.      $out_array[5][15] = $alkenesum;
378.      $out_array[6][15] = $alkanesum;
379.      $out_array[7][15] = $alkynesum;
380.      $out_array[8][15] = $thiolsum;
381.      $out_array[9][15] = $sulfsum;
382.      $out_array[10][15] = $indolesum;
383.      $out_array[11][15] = $keysum;
384.      $out_array[12][15] = $nitsum;
385.      $out_array[13][15] = $aminesum;
386.      $out_array[14][15] = $othersum;

387.      $out_array[2][16] = $alcsum;
388.      $out_array[3][16] = $acidsum;
389.      $out_array[4][16] = $aldsum;
390.      $out_array[5][16] = $alkenesum;
391.      $out_array[6][16] = $alkanesum;
392.      $out_array[7][16] = $alkynesum;
393.      $out_array[8][16] = $thiolsum;
394.      $out_array[9][16] = $sulfsum;
395.      $out_array[10][16] = $indolesum;
396.      $out_array[11][16] = $keysum;
397.      $out_array[12][16] = $nitsum;
398.      $out_array[13][16] = $aminesum;
399.      $out_array[14][16] = $othersum;

400.      $out_array[2][17] = $alcsum;
401.      $out_array[3][17] = $acidsum;
402.      $out_array[4][17] = $aldsum;
403.      $out_array[5][17] = $alkenesum;
404.      $out_array[6][17] = $alkanesum;
405.      $out_array[7][17] = $alkynesum;
406.      $out_array[8][17] = $thiolsum;
407.      $out_array[9][17] = $sulfsum;
408.      $out_array[10][17] = $indolesum;
409.      $out_array[11][17] = $keysum;
410.      $out_array[12][17] = $nitsum;
411.      $out_array[13][17] = $aminesum;
```

```
412.      $out_array[14][17] = $othersum;

413.      $out_array[2][18] = $alcsum;
414.      $out_array[3][18] = $acidsum;
415.      $out_array[4][18] = $aldsum;
416.      $out_array[5][18] = $alkenesum;
417.      $out_array[6][18] = $alkanesum;
418.      $out_array[7][18] = $alkynesum;
419.      $out_array[8][18] = $thiolsum;
420.      $out_array[9][18] = $sulfsum;
421.      $out_array[10][18] = $indolesum;
422.      $out_array[11][18] = $keysum;
423.      $out_array[12][18] = $nitsum;
424.      $out_array[13][18] = $aminesum;
425.      $out_array[14][18] = $othersum;

426.      $out_array[2][19] = $alcsum;
427.      $out_array[3][19] = $acidsum;
428.      $out_array[4][19] = $aldsum;
429.      $out_array[5][19] = $alkenesum;
430.      $out_array[6][19] = $alkanesum;
431.      $out_array[7][19] = $alkynesum;
432.      $out_array[8][19] = $thiolsum;
433.      $out_array[9][19] = $sulfsum;
434.      $out_array[10][19] = $indolesum;
435.      $out_array[11][19] = $keysum;
436.      $out_array[12][19] = $nitsum;
437.      $out_array[13][19] = $aminesum;
438.      $out_array[14][19] = $othersum;

439.      $out_array[2][20] = $alcsum;
440.      $out_array[3][20] = $acidsum;
441.      $out_array[4][20] = $aldsum;
442.      $out_array[5][20] = $alkenesum;
443.      $out_array[6][20] = $alkanesum;
444.      $out_array[7][20] = $alkynesum;
445.      $out_array[8][20] = $thiolsum;
446.      $out_array[9][20] = $sulfsum;
447.      $out_array[10][20] = $indolesum;
448.      $out_array[11][20] = $keysum;
449.      $out_array[12][20] = $nitsum;
450.      $out_array[13][20] = $aminesum;
451.      $out_array[14][20] = $othersum;

452.      $out_array[2][21] = $alcsum;
453.      $out_array[3][21] = $acidsum;
454.      $out_array[4][21] = $aldsum;
455.      $out_array[5][21] = $alkenesum;
456.      $out_array[6][21] = $alkanesum;
457.      $out_array[7][21] = $alkynesum;
458.      $out_array[8][21] = $thiolsum;
459.      $out_array[9][21] = $sulfsum;
460.      $out_array[10][21] = $indolesum;
461.      $out_array[11][21] = $keysum;
```

```
462.      $out_array[12][21] = $nitsum;
463.      $out_array[13][21] = $aminesum;
464.      $out_array[14][21] = $othersum;

465.      $out_array[2][22] = $alcsum;
466.      $out_array[3][22] = $acidsum;
467.      $out_array[4][22] = $aldsum;
468.      $out_array[5][22] = $alkenesum;
469.      $out_array[6][22] = $alkanesum;
470.      $out_array[7][22] = $alkynesum;
471.      $out_array[8][22] = $thiolsum;
472.      $out_array[9][22] = $sulfsum;
473.      $out_array[10][22] = $indolesum;
474.      $out_array[11][22] = $keysum;
475.      $out_array[12][22] = $nitsum;
476.      $out_array[13][22] = $aminesum;
477.      $out_array[14][22] = $othersum;

478.      $out_array[2][23] = $alcsum;
479.      $out_array[3][23] = $acidsum;
480.      $out_array[4][23] = $aldsum;
481.      $out_array[5][23] = $alkenesum;
482.      $out_array[6][23] = $alkanesum;
483.      $out_array[7][23] = $alkynesum;
484.      $out_array[8][23] = $thiolsum;
485.      $out_array[9][23] = $sulfsum;
486.      $out_array[10][23] = $indolesum;
487.      $out_array[11][23] = $keysum;
488.      $out_array[12][23] = $nitsum;
489.      $out_array[13][23] = $aminesum;
490.      $out_array[14][23] = $othersum;

491.      $out_array[2][24] = $alcsum;
492.      $out_array[3][24] = $acidsum;
493.      $out_array[4][24] = $aldsum;
494.      $out_array[5][24] = $alkenesum;
495.      $out_array[6][24] = $alkanesum;
496.      $out_array[7][24] = $alkynesum;
497.      $out_array[8][24] = $thiolsum;
498.      $out_array[9][24] = $sulfsum;
499.      $out_array[10][24] = $indolesum;
500.      $out_array[11][24] = $keysum;
501.      $out_array[12][24] = $nitsum;
502.      $out_array[13][24] = $aminesum;
503.      $out_array[14][24] = $othersum;

504.      $out_array[2][25] = $alcsum;
505.      $out_array[3][25] = $acidsum;
506.      $out_array[4][25] = $aldsum;
507.      $out_array[5][25] = $alkenesum;
508.      $out_array[6][25] = $alkanesum;
509.      $out_array[7][25] = $alkynesum;
510.      $out_array[8][25] = $thiolsum;
511.      $out_array[9][25] = $sulfsum;
```

```
512.      $out_array[10][25] = $indolesum;
513.      $out_array[11][25] = $keysum;
514.      $out_array[12][25] = $nitsum;
515.      $out_array[13][25] = $aminesum;
516.      $out_array[14][25] = $othersum;

517.      $out_array[2][26] = $alcsum;
518.      $out_array[3][26] = $acidsum;
519.      $out_array[4][26] = $aldsum;
520.      $out_array[5][26] = $alkenesum;
521.      $out_array[6][26] = $alkanesum;
522.      $out_array[7][26] = $alkynesum;
523.      $out_array[8][26] = $thiolsum;
524.      $out_array[9][26] = $sulfsum;
525.      $out_array[10][26] = $indolesum;
526.      $out_array[11][26] = $keysum;
527.      $out_array[12][26] = $nitsum;
528.      $out_array[13][26] = $aminesum;
529.      $out_array[14][26] = $othersum;

530.      $out_array[2][27] = $alcsum;
531.      $out_array[3][27] = $acidsum;
532.      $out_array[4][27] = $aldsum;
533.      $out_array[5][27] = $alkenesum;
534.      $out_array[6][27] = $alkanesum;
535.      $out_array[7][27] = $alkynesum;
536.      $out_array[8][27] = $thiolsum;
537.      $out_array[9][27] = $sulfsum;
538.      $out_array[10][27] = $indolesum;
539.      $out_array[11][27] = $keysum;
540.      $out_array[12][27] = $nitsum;
541.      $out_array[13][27] = $aminesum;
542.      $out_array[14][27] = $othersum;

543.      $out_array[2][28] = $alcsum;
544.      $out_array[3][28] = $acidsum;
545.      $out_array[4][28] = $aldsum;
546.      $out_array[5][28] = $alkenesum;
547.      $out_array[6][28] = $alkanesum;
548.      $out_array[7][28] = $alkynesum;
549.      $out_array[8][28] = $thiolsum;
550.      $out_array[9][28] = $sulfsum;
551.      $out_array[10][28] = $indolesum;
552.      $out_array[11][28] = $keysum;
553.      $out_array[12][28] = $nitsum;
554.      $out_array[13][28] = $aminesum;
555.      $out_array[14][28] = $othersum;

556.      $out_array[2][29] = $alcsum;
557.      $out_array[3][29] = $acidsum;
558.      $out_array[4][29] = $aldsum;
559.      $out_array[5][29] = $alkenesum;
560.      $out_array[6][29] = $alkanesum;
561.      $out_array[7][29] = $alkynesum;
```

```
562.      $out_array[8][29] = $thiolsum;
563.      $out_array[9][29] = $sulfsum;
564.      $out_array[10][29] = $indolesum;
565.      $out_array[11][29] = $keysum;
566.      $out_array[12][29] = $nitsum;
567.      $out_array[13][29] = $aminesum;
568.      $out_array[14][29] = $othersum;

569.      $out_array[2][30] = $alcsum;
570.      $out_array[3][30] = $acidsum;
571.      $out_array[4][30] = $aldsum;
572.      $out_array[5][30] = $alkenesum;
573.      $out_array[6][30] = $alkanesum;
574.      $out_array[7][30] = $alkynesum;
575.      $out_array[8][30] = $thiolsum;
576.      $out_array[9][30] = $sulfsum;
577.      $out_array[10][30] = $indolesum;
578.      $out_array[11][30] = $keysum;
579.      $out_array[12][30] = $nitsum;
580.      $out_array[13][30] = $aminesum;
581.      $out_array[14][30] = $othersum;

582.      $out_array[2][31] = $alcsum;
583.      $out_array[3][31] = $acidsum;
584.      $out_array[4][31] = $aldsum;
585.      $out_array[5][31] = $alkenesum;
586.      $out_array[6][31] = $alkanesum;
587.      $out_array[7][31] = $alkynesum;
588.      $out_array[8][31] = $thiolsum;
589.      $out_array[9][31] = $sulfsum;
590.      $out_array[10][31] = $indolesum;
591.      $out_array[11][31] = $keysum;
592.      $out_array[12][31] = $nitsum;
593.      $out_array[13][31] = $aminesum;
594.      $out_array[14][31] = $othersum;

595.      $out_array[2][32] = $alcsum;
596.      $out_array[3][32] = $acidsum;
597.      $out_array[4][32] = $aldsum;
598.      $out_array[5][32] = $alkenesum;
599.      $out_array[6][32] = $alkanesum;
600.      $out_array[7][32] = $alkynesum;
601.      $out_array[8][32] = $thiolsum;
602.      $out_array[9][32] = $sulfsum;
603.      $out_array[10][32] = $indolesum;
604.      $out_array[11][32] = $keysum;
605.      $out_array[12][32] = $nitsum;
606.      $out_array[13][32] = $aminesum;
607.      $out_array[14][32] = $othersum;

608.      $out_array[2][33] = $alcsum;
609.      $out_array[3][33] = $acidsum;
610.      $out_array[4][33] = $aldsum;
611.      $out_array[5][33] = $alkenesum;
```

```
612.      $out_array[6][33] = $alkanesum;
613.      $out_array[7][33] = $alkynesum;
614.      $out_array[8][33] = $thiolsum;
615.      $out_array[9][33] = $sulfsum;
616.      $out_array[10][33] = $indolesum;
617.      $out_array[11][33] = $keysum;
618.      $out_array[12][33] = $nitsum;
619.      $out_array[13][33] = $aminesum;
620.      $out_array[14][33] = $othersum;

621.      $out_array[2][34] = $alcsum;
622.      $out_array[3][34] = $acidsum;
623.      $out_array[4][34] = $aldsum;
624.      $out_array[5][34] = $alkenesum;
625.      $out_array[6][34] = $alkanesum;
626.      $out_array[7][34] = $alkynesum;
627.      $out_array[8][34] = $thiolsum;
628.      $out_array[9][34] = $sulfsum;
629.      $out_array[10][34] = $indolesum;
630.      $out_array[11][34] = $keysum;
631.      $out_array[12][34] = $nitsum;
632.      $out_array[13][34] = $aminesum;
633.      $out_array[14][34] = $othersum;

634.      $out_array[2][35] = $alcsum;
635.      $out_array[3][35] = $acidsum;
636.      $out_array[4][35] = $aldsum;
637.      $out_array[5][35] = $alkenesum;
638.      $out_array[6][35] = $alkanesum;
639.      $out_array[7][35] = $alkynesum;
640.      $out_array[8][35] = $thiolsum;
641.      $out_array[9][35] = $sulfsum;
642.      $out_array[10][35] = $indolesum;
643.      $out_array[11][35] = $keysum;
644.      $out_array[12][35] = $nitsum;
645.      $out_array[13][35] = $aminesum;
646.      $out_array[14][35] = $othersum;

647.      $out_array[2][36] = $alcsum;
648.      $out_array[3][36] = $acidsum;
649.      $out_array[4][36] = $aldsum;
650.      $out_array[5][36] = $alkenesum;
651.      $out_array[6][36] = $alkanesum;
652.      $out_array[7][36] = $alkynesum;
653.      $out_array[8][36] = $thiolsum;
654.      $out_array[9][36] = $sulfsum;
655.      $out_array[10][36] = $indolesum;
656.      $out_array[11][36] = $keysum;
657.      $out_array[12][36] = $nitsum;
658.      $out_array[13][36] = $aminesum;
659.      $out_array[14][36] = $othersum;

660.      $out_array[2][37] = $alcsum;
661.      $out_array[3][37] = $acidsum;
```

```
662.      $out_array[4][37] = $aldsum;
663.      $out_array[5][37] = $alkenesum;
664.      $out_array[6][37] = $alkanesum;
665.      $out_array[7][37] = $alkynesum;
666.      $out_array[8][37] = $thiolsum;
667.      $out_array[9][37] = $sulfsum;
668.      $out_array[10][37] = $indolesum;
669.      $out_array[11][37] = $keysum;
670.      $out_array[12][37] = $nitsum;
671.      $out_array[13][37] = $aminesum;
672.      $out_array[14][37] = $othersum;

673.      $out_array[2][38] = $alcsum;
674.      $out_array[3][38] = $acidsum;
675.      $out_array[4][38] = $aldsum;
676.      $out_array[5][38] = $alkenesum;
677.      $out_array[6][38] = $alkanesum;
678.      $out_array[7][38] = $alkynesum;
679.      $out_array[8][38] = $thiolsum;
680.      $out_array[9][38] = $sulfsum;
681.      $out_array[10][38] = $indolesum;
682.      $out_array[11][38] = $keysum;
683.      $out_array[12][38] = $nitsum;
684.      $out_array[13][38] = $aminesum;
685.      $out_array[14][38] = $othersum;

686.      $out_array[2][39] = $alcsum;
687.      $out_array[3][39] = $acidsum;
688.      $out_array[4][39] = $aldsum;
689.      $out_array[5][39] = $alkenesum;
690.      $out_array[6][39] = $alkanesum;
691.      $out_array[7][39] = $alkynesum;
692.      $out_array[8][39] = $thiolsum;
693.      $out_array[9][39] = $sulfsum;
694.      $out_array[10][39] = $indolesum;
695.      $out_array[11][39] = $keysum;
696.      $out_array[12][39] = $nitsum;
697.      $out_array[13][39] = $aminesum;
698.      $out_array[14][39] = $othersum;

699.      $out_array[2][40] = $alcsum;
700.      $out_array[3][40] = $acidsum;
701.      $out_array[4][40] = $aldsum;
702.      $out_array[5][40] = $alkenesum;
703.      $out_array[6][40] = $alkanesum;
704.      $out_array[7][40] = $alkynesum;
705.      $out_array[8][40] = $thiolsum;
706.      $out_array[9][40] = $sulfsum;
707.      $out_array[10][40] = $indolesum;
708.      $out_array[11][40] = $keysum;
709.      $out_array[12][40] = $nitsum;
710.      $out_array[13][40] = $aminesum;
711.      $out_array[14][40] = $othersum;
```

```
712.      $out_array[2][41] = $alcsum;
713.      $out_array[3][41] = $acidsum;
714.      $out_array[4][41] = $aldsum;
715.      $out_array[5][41] = $alkenesum;
716.      $out_array[6][41] = $alkanesum;
717.      $out_array[7][41] = $alkynesum;
718.      $out_array[8][41] = $thiolsum;
719.      $out_array[9][41] = $sulfsum;
720.      $out_array[10][41] = $indolesum;
721.      $out_array[11][41] = $keysum;
722.      $out_array[12][41] = $nitsum;
723.      $out_array[13][41] = $aminesum;
724.      $out_array[14][41] = $othersum;


725.      $out_array[2][42] = $alcsum;
726.      $out_array[3][42] = $acidsum;
727.      $out_array[4][42] = $aldsum;
728.      $out_array[5][42] = $alkenesum;
729.      $out_array[6][42] = $alkanesum;
730.      $out_array[7][42] = $alkynesum;
731.      $out_array[8][42] = $thiolsum;
732.      $out_array[9][42] = $sulfsum;
733.      $out_array[10][42] = $indolesum;
734.      $out_array[11][42] = $keysum;
735.      $out_array[12][42] = $nitsum;
736.      $out_array[13][42] = $aminesum;
737.      $out_array[14][42] = $othersum;

738.      $out_array[2][43] = $alcsum;
739.      $out_array[3][43] = $acidsum;
740.      $out_array[4][43] = $aldsum;
741.      $out_array[5][43] = $alkenesum;
742.      $out_array[6][43] = $alkanesum;
743.      $out_array[7][43] = $alkynesum;
744.      $out_array[8][43] = $thiolsum;
745.      $out_array[9][43] = $sulfsum;
746.      $out_array[10][43] = $indolesum;
747.      $out_array[11][43] = $keysum;
748.      $out_array[12][43] = $nitsum;
749.      $out_array[13][43] = $aminesum;
750.      $out_array[14][43] = $othersum;

751.      $out_array[2][44] = $alcsum;
752.      $out_array[3][44] = $acidsum;
753.      $out_array[4][44] = $aldsum;
754.      $out_array[5][44] = $alkenesum;
755.      $out_array[6][44] = $alkanesum;
756.      $out_array[7][44] = $alkynesum;
757.      $out_array[8][44] = $thiolsum;
758.      $out_array[9][44] = $sulfsum;
759.      $out_array[10][44] = $indolesum;
760.      $out_array[11][44] = $keysum;
761.      $out_array[12][44] = $nitsum;
```

```
762.      $out_array[13][44] = $aminesum;
763.      $out_array[14][44] = $othersum;

764.      $out_array[2][45] = $alcsum;
765.      $out_array[3][45] = $acidsum;
766.      $out_array[4][45] = $aldsum;
767.      $out_array[5][45] = $alkenesum;
768.      $out_array[6][45] = $alkanesum;
769.      $out_array[7][45] = $alkynesum;
770.      $out_array[8][45] = $thiolsum;
771.      $out_array[9][45] = $sulfsum;
772.      $out_array[10][45] = $indolesum;
773.      $out_array[11][45] = $keysum;
774.      $out_array[12][45] = $nitsum;
775.      $out_array[13][45] = $aminesum;
776.      $out_array[14][45] = $othersum;

777.      $out_array[2][46] = $alcsum;
778.      $out_array[3][46] = $acidsum;
779.      $out_array[4][46] = $aldsum;
780.      $out_array[5][46] = $alkenesum;
781.      $out_array[6][46] = $alkanesum;
782.      $out_array[7][46] = $alkynesum;
783.      $out_array[8][46] = $thiolsum;
784.      $out_array[9][46] = $sulfsum;
785.      $out_array[10][46] = $indolesum;
786.      $out_array[11][46] = $keysum;
787.      $out_array[12][46] = $nitsum;
788.      $out_array[13][46] = $aminesum;
789.      $out_array[14][46] = $othersum;

790.      $out_array[2][47] = $alcsum;
791.      $out_array[3][47] = $acidsum;
792.      $out_array[4][47] = $aldsum;
793.      $out_array[5][47] = $alkenesum;
794.      $out_array[6][47] = $alkanesum;
795.      $out_array[7][47] = $alkynesum;
796.      $out_array[8][47] = $thiolsum;
797.      $out_array[9][47] = $sulfsum;
798.      $out_array[10][47] = $indolesum;
799.      $out_array[11][47] = $keysum;
800.      $out_array[12][47] = $nitsum;
801.      $out_array[13][47] = $aminesum;
802.      $out_array[14][47] = $othersum;

803.      #for loop to sum abundances in 1st sample in list
804.      #[row][columns]
805.      # for loop to go through and look at each of the
   metabolites, group them into chemical class,
806.      # add the sum abundance
807.      # $metabolite_counts keeps track of which metabolite is
   being grouped
808.      # $total_metabolites is the total number of metabolites in
   the file
```

```
809.      # for the $out_array assignment, need to group the
   (original matrix and sum of abundances for that chemical class)
810.      # and have those assigned to the specific cell in the
   out_array matrix.

811.      for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
812.      my $BB = 0;
813.      #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
814.      next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
815.      #           for (@columns){
816.      #                 if ($columns[$metabolite_counts][0] =~
   /(^"Sample"|^"Label"|^"alcoholic")/){
817.      #                       $BB += 1;
818.      #                       next
   $columns[$metabolite_counts][0];
819.      #                 }
820.      #           }
821.      if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ #if
   the metabolite ends in "-ol" or has "-ol," in the name
822.      #           if ( $columns[$metabolite_counts][0] =~
   m/((.*?ol)$)|((.*?ol, ))/ #and $columns[$metabolite_counts][0] =~
   m/(.*diol)?/
823.      and $columns[$metabolite_counts][0] !~ m/thiol/){
          # regular experssion to find alcohols. this is more complex
   than the other matches
824.      # because the match 'ol' encompasses indole and thiol


825.      #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
826.      $out_array[2][1] += ($columns[$metabolite_counts][1] +
   $alcsum);   # if there is a metabolite match, add the abundance
   to a cumulative
827.      # sum in out_array in the alcohol group under sample 1
828.      $BB += 1;   #this counter keeps track of which regular
   experssion the metabolite is being sorted into.
829.      # Since the BB counter increases, there is no need for the
   metabolite to be sent to the next if statement (regular
   experssion match)
830.      #                 $metabolite_counts++;
831.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) { #if
   the metabolite has "al" in the name
832.      # and doesn't contain "nitrile"
833.      $out_array[4][1] += ($columns[$metabolite_counts][1] +
   $aldsum);
                  # then add the corresponding abundance to the
   aldehyde sum
834.      $BB += 1;
835.      #                 $metabolite_counts++;
```

```
836.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {  #if the metabolite has "aldehyde" in the name
837.      $out_array[4][1] += ($columns[$metabolite_counts][1] +
   $aldsum);                #then add the corresponding abundance to
   the aldehyde sum
838.      $BB += 1;
839.      #                 $metabolite_counts++;
840.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) { #if the metabolite has "acid" in the name
841.      $out_array[3][1] += ($columns[$metabolite_counts][1] +
   $acidsum);  #then add the corresponding abundance to the acid sum
842.      $BB += 1;
843.      #                 $metabolite_counts++;
844.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) { #if the metabolite has "-ate" in the name
845.      $out_array[3][1] += ($columns[$metabolite_counts][1] +
   $acidsum);  #then add the corresponding abundance to the acid sum
846.      $BB += 1;
847.      #                 $metabolite_counts++;
848.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {     #if the metabolite has "ester" in the name
849.      $out_array[3][1] += ($columns[$metabolite_counts][1] +
   $acidsum);  #then add the corresponding abundance to the acid sum
850.      $BB += 1;
851.      #                 $metabolite_counts++;
852.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) { #if the metabolite has "-one" in the name
853.      $out_array[11][1] += ($columns[$metabolite_counts][1] +
   $keysum);   #then add the corresponding abundance to the ketone
   sum
854.      $BB += 1;
855.      #                 $metabolite_counts++;
856.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/anediol/)) {     #if the
   metabolite has "-ane" in the name
857.      #and the metabolite does not contain "thiol" or "-anediol"
858.      $out_array[6][1] += ($columns[$metabolite_counts][1] +
   $alkanesum);

                                        #then add the corresponding
   abundance to the alkane sum
859.      $BB += 1;
860.      #                 $metabolite_counts++;
861.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
         #if the metabolite has "-ene" in the name
862.      #and the metabolite does not contain "nitrile"
863.      $out_array[5][1] += ($columns[$metabolite_counts][1] +
   $alkenesum);
                                  #then add the corresponding
   abundance to the alkene sum
864.      $BB += 1;
865.      #                 $metabolite_counts++;
866.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) { #if the metabolite has "-yne" in the name
```

```
867.      $out_array[7][1] += ($columns[$metabolite_counts][1] +
   $alkynesum);      #then add the corresponding abundance to the
   alkyne sum
868.      $BB += 1;
869.      #                 $metabolite_counts++;
870.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {   #if the metabolite has "nitrile" in the name
871.      $out_array[12][1] += ($columns[$metabolite_counts][1] +
   $nitsum);         #then add the corresponding abundance to the
   nitrile sum
872.      $BB += 1;
873.      #                 $metabolite_counts++;
874.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {     #if the metabolite has "thiol" in the name
875.      $out_array[8][1] += ($columns[$metabolite_counts][1] +
   $thiolsum);       #then add the corresponding abundance to the
   thiol sum
876.      $BB += 1;
877.      #                 $metabolite_counts++;
878.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {       #if the metabolite has "-ole" in the name
879.      $out_array[10][1] += ($columns[$metabolite_counts][1] +
   $indolesum);      #then add the corresponding abundance to the
   indole sum
880.      $BB += 1;
881.      #                 $metabolite_counts++;
882.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {         #if the metabolite has "sulfide" in the
   name
883.      $out_array[9][1] += ($columns[$metabolite_counts][1] +
   $sulfsum);              #then add the corresponding abundance to
   the sulfide sum
884.      $BB += 1;
885.      #                 $metabolite_counts++;
886.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {     #if the metabolite has "amine" in the name
887.      $out_array[13][1] += ($columns[$metabolite_counts][1] +
   $aminesum);       #then add the corresponding abundance to the
   amine sum
888.      $BB += 1;
889.      #                 $metabolite_counts++;
890.      } elsif (( $BB == 0 )) {
891.      $out_array[14][1] += ($columns[$metabolite_counts][1]+
   $othersum); #if the metabolite doesn't meet any of the criteria
   for the other
892.      # regular expressions, the corresponding abundance is
   summed here
893.      $BB += 1;
894.      }
895.      }
896.      #this continues until all of the samples are completed, in
   this case, until all 47 samples have been parsed
897.      #for loop to sum abundances in 2nd sample in list.
898.      for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
899.      next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
900.      my $BB = 0;
901.      #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
902.      if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/) {
903.      #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
904.      $out_array[2][2] += ($columns[$metabolite_counts][2] +
   $alcsum);
905.      $BB += 1;
906.      #                 $metabolite_counts++;
907.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/ )) {
908.      $out_array[4][2] += ($columns[$metabolite_counts][2] +
   $aldsum);
909.      $BB += 1;
910.      #                 $metabolite_counts++;
911.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
912.      $out_array[4][2] += ($columns[$metabolite_counts][2] +
   $aldsum);
913.      $BB += 1;
914.      #                 $metabolite_counts++;
915.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
916.      $out_array[3][2] += ($columns[$metabolite_counts][2] +
   $acidsum);
917.      $BB += 1;
918.      #                 $metabolite_counts++;
919.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
920.      $out_array[3][2] += ($columns[$metabolite_counts][2] +
   $acidsum);
921.      $BB += 1;
922.      #                 $metabolite_counts++;
923.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
924.      $out_array[3][2] += ($columns[$metabolite_counts][2] +
   $acidsum);
925.      $BB += 1;
926.      #                 $metabolite_counts++;
927.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
928.      $out_array[11][2] += ($columns[$metabolite_counts][2] +
   $keysum);
929.      $BB += 1;
930.      #                 $metabolite_counts++;
931.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
932.      $out_array[6][2] += ($columns[$metabolite_counts][2] +
   $alkanesum);
933.      $BB += 1;
934.      #                 $metabolite_counts++;
935.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
936.      $out_array[5][2] += ($columns[$metabolite_counts][2] +
   $alkenesum);
937.      $BB += 1;
938.      #                 $metabolite_counts++;
939.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
940.      $out_array[7][2] += ($columns[$metabolite_counts][2] +
   $alkynesum);
941.      $BB += 1;
942.      #                 $metabolite_counts++;
943.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
944.      $out_array[12][2] += ($columns[$metabolite_counts][2] +
   $nitsum);
945.      $BB += 1;
946.      #                 $metabolite_counts++;
947.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
948.      $out_array[8][2] += ($columns[$metabolite_counts][2] +
   $thiolsum);
949.      $BB += 1;
950.      #                 $metabolite_counts++;
951.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
952.      $out_array[10][2] += ($columns[$metabolite_counts][2] +
   $indolesum);
953.      $BB += 1;
954.      #                 $metabolite_counts++;
955.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
956.      $out_array[9][2] += ($columns[$metabolite_counts][2] +
   $sulfsum);
957.      $BB += 1;
958.      #                 $metabolite_counts++;
959.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
960.      $out_array[13][2] += ($columns[$metabolite_counts][2] +
   $aminesum);
961.      $BB += 1;
962.      #                 $metabolite_counts++;
963.      } elsif (( $BB == 0 )) {
964.      $out_array[14][2] += ($columns[$metabolite_counts][2]+
   $othersum);
965.      $BB += 1;
966.      }
967.      }
968.      for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
969.      next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
970.      my $BB = 0;
971.      #           next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);
972.      #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
973.      if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/) {
974.      #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
975.      $out_array[2][3] += ($columns[$metabolite_counts][3] +
   $alcsum);
976.      $BB += 1;
977.      #                 $metabolite_counts++;
978.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
979.      $out_array[4][3] += ($columns[$metabolite_counts][3] +
   $aldsum);
980.      $BB += 1;
981.      #                 $metabolite_counts++;
982.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
983.      $out_array[4][3] += ($columns[$metabolite_counts][3] +
   $aldsum);
984.      $BB += 1;
985.      #                 $metabolite_counts++;
986.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
987.      $out_array[3][3] += ($columns[$metabolite_counts][3] +
   $acidsum);
988.      $BB += 1;
989.      #                 $metabolite_counts++;
990.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
991.      $out_array[3][3] += ($columns[$metabolite_counts][3] +
   $acidsum);
992.      $BB += 1;
993.      #                 $metabolite_counts++;
994.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
995.      $out_array[3][3] += ($columns[$metabolite_counts][3] +
   $acidsum);
996.      $BB += 1;
997.      #                 $metabolite_counts++;
998.      } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
999.      $out_array[11][3] += ($columns[$metabolite_counts][3] +
   $keysum);
1000.     $BB += 1;
1001.     #                 $metabolite_counts++;
```

```
1002.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
1003.     $out_array[6][3] += ($columns[$metabolite_counts][3] +
   $alkanesum);
1004.     $BB += 1;
1005.     #                 $metabolite_counts++;
1006.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1007.     $out_array[5][3] += ($columns[$metabolite_counts][3] +
   $alkenesum);
1008.     $BB += 1;
1009.     #                 $metabolite_counts++;
1010.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1011.     $out_array[7][3] += ($columns[$metabolite_counts][3] +
   $alkynesum);
1012.     $BB += 1;
1013.     #                 $metabolite_counts++;
1014.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1015.     $out_array[12][3] += ($columns[$metabolite_counts][3] +
   $nitsum);
1016.     $BB += 1;
1017.     #                 $metabolite_counts++;
1018.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1019.     $out_array[8][3] += ($columns[$metabolite_counts][3] +
   $thiolsum);
1020.     $BB += 1;
1021.     #                 $metabolite_counts++;
1022.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1023.     $out_array[10][3] += ($columns[$metabolite_counts][3] +
   $indolesum);
1024.     $BB += 1;
1025.     #                 $metabolite_counts++;
1026.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1027.     $out_array[9][3] += ($columns[$metabolite_counts][3] +
   $sulfsum);
1028.     $BB += 1;
1029.     #                 $metabolite_counts++;
1030.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1031.     $out_array[13][3] += ($columns[$metabolite_counts][3] +
   $aminesum);
1032.     $BB += 1;
1033.     #                 $metabolite_counts++;
1034.     } elsif (( $BB == 0 )) {
1035.     $out_array[14][3] += ($columns[$metabolite_counts][4]+
   $othersum);
1036.     $BB += 1;
1037.     }
1038.     }
1039.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1040.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1041.     my $BB = 0;
1042.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1043.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {       #regular
   expression for alcohols
1044.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1045.     $out_array[2][4] += ($columns[$metabolite_counts][4] +
   $alcsum);
1046.     $BB += 1;
1047.     #                 $metabolite_counts++;
1048.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/ )) {
1049.     $out_array[4][4] += ($columns[$metabolite_counts][4] +
   $aldsum);
1050.     $BB += 1;
1051.     #                 $metabolite_counts++;
1052.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1053.     $out_array[4][4] += ($columns[$metabolite_counts][4] +
   $aldsum);
1054.     $BB += 1;
1055.     #                 $metabolite_counts++;
1056.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1057.     $out_array[3][4] += ($columns[$metabolite_counts][4] +
   $acidsum);
1058.     $BB += 1;
1059.     #                 $metabolite_counts++;
1060.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1061.     $out_array[3][4] += ($columns[$metabolite_counts][4] +
   $acidsum);
1062.     $BB += 1;
1063.     #                 $metabolite_counts++;
1064.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1065.     $out_array[3][4] += ($columns[$metabolite_counts][4] +
   $acidsum);
1066.     $BB += 1;
1067.     #                 $metabolite_counts++;
1068.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1069.     $out_array[11][4] += ($columns[$metabolite_counts][4] +
   $keysum);
1070.     $BB += 1;
1071.     #                 $metabolite_counts++;
1072.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1073.     $out_array[6][4] += ($columns[$metabolite_counts][4] +
   $alkanesum);
1074.     $BB += 1;
1075.     #                 $metabolite_counts++;
1076.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1077.     $out_array[5][4] += ($columns[$metabolite_counts][4] +
   $alkenesum);
1078.     $BB += 1;
1079.     #                 $metabolite_counts++;
1080.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1081.     $out_array[7][4] += ($columns[$metabolite_counts][4] +
   $alkynesum);
1082.     $BB += 1;
1083.     #                 $metabolite_counts++;
1084.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1085.     $out_array[12][4] += ($columns[$metabolite_counts][4] +
   $nitsum);
1086.     $BB += 1;
1087.     #                 $metabolite_counts++;
1088.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1089.     $out_array[8][4] += ($columns[$metabolite_counts][4] +
   $thiolsum);
1090.     $BB += 1;
1091.     #                 $metabolite_counts++;
1092.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1093.     $out_array[10][4] += ($columns[$metabolite_counts][4] +
   $indolesum);
1094.     $BB += 1;
1095.     #                 $metabolite_counts++;
1096.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1097.     $out_array[9][4] += ($columns[$metabolite_counts][4] +
   $sulfsum);
1098.     $BB += 1;
1099.     #                 $metabolite_counts++;
1100.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1101.     $out_array[13][4] += ($columns[$metabolite_counts][4] +
   $aminesum);
1102.     $BB += 1;
1103.     #                 $metabolite_counts++;
1104.     } elsif (( $BB == 0 )) {
1105.     $out_array[14][4] += ($columns[$metabolite_counts][5]+
   $othersum);
1106.     $BB += 1;
1107.     }
1108.     }
1109.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1110.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1111.     my $BB = 0;
1112.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1113.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1114.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1115.     $out_array[2][5] += ($columns[$metabolite_counts][5] +
   $alcsum);
1116.     $BB += 1;
1117.     #                 $metabolite_counts++;
1118.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1119.     $out_array[4][5] += ($columns[$metabolite_counts][5] +
   $aldsum);
1120.     $BB += 1;
1121.     #                 $metabolite_counts++;
1122.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1123.     $out_array[4][5] += ($columns[$metabolite_counts][5] +
   $aldsum);
1124.     $BB += 1;
1125.     #                 $metabolite_counts++;
1126.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1127.     $out_array[3][5] += ($columns[$metabolite_counts][5] +
   $acidsum);
1128.     $BB += 1;
1129.     #                 $metabolite_counts++;
1130.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1131.     $out_array[3][5] += ($columns[$metabolite_counts][5] +
   $acidsum);
1132.     $BB += 1;
1133.     #                 $metabolite_counts++;
1134.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1135.     $out_array[3][5] += ($columns[$metabolite_counts][5] +
   $acidsum);
1136.     $BB += 1;
1137.     #                 $metabolite_counts++;
1138.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1139.     $out_array[11][5] += ($columns[$metabolite_counts][5] +
   $keysum);
1140.     $BB += 1;
1141.     #                 $metabolite_counts++;
1142.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1143.     $out_array[6][5] += ($columns[$metabolite_counts][5] +
   $alkanesum);
1144.     $BB += 1;
1145.     #                 $metabolite_counts++;
1146.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1147.     $out_array[5][5] += ($columns[$metabolite_counts][5] +
   $alkenesum);
1148.     $BB += 1;
1149.     #                 $metabolite_counts++;
1150.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1151.     $out_array[7][5] += ($columns[$metabolite_counts][5] +
   $alkynesum);
1152.     $BB += 1;
1153.     #                 $metabolite_counts++;
1154.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1155.     $out_array[12][5] += ($columns[$metabolite_counts][5] +
   $nitsum);
1156.     $BB += 1;
1157.     #                 $metabolite_counts++;
1158.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1159.     $out_array[8][5] += ($columns[$metabolite_counts][5] +
   $thiolsum);
1160.     $BB += 1;
1161.     #                 $metabolite_counts++;
1162.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1163.     $out_array[10][5] += ($columns[$metabolite_counts][5] +
   $indolesum);
1164.     $BB += 1;
1165.     #                 $metabolite_counts++;
1166.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1167.     $out_array[9][5] += ($columns[$metabolite_counts][5] +
   $sulfsum);
1168.     $BB += 1;
1169.     #                 $metabolite_counts++;
1170.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1171.     $out_array[13][5] += ($columns[$metabolite_counts][5] +
   $aminesum);
1172.     $BB += 1;
1173.     #                 $metabolite_counts++;
1174.     } elsif (( $BB == 0 )) {
1175.     $out_array[14][5] += ($columns[$metabolite_counts][5]+
   $othersum);
1176.     $BB += 1;
1177.     }
1178.     }
1179.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1180.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1181.     my $BB = 0;
1182.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1183.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1184.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1185.     $out_array[2][6] += ($columns[$metabolite_counts][6] +
   $alcsum);
1186.     $BB += 1;
1187.     #                 $metabolite_counts++;
1188.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/ )) {
1189.     $out_array[4][6] += ($columns[$metabolite_counts][6] +
   $aldsum);
1190.     $BB += 1;
1191.     #                 $metabolite_counts++;
1192.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1193.     $out_array[4][6] += ($columns[$metabolite_counts][6] +
   $aldsum);
1194.     $BB += 1;
1195.     #                 $metabolite_counts++;
1196.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1197.     $out_array[3][6] += ($columns[$metabolite_counts][6] +
   $acidsum);
1198.     $BB += 1;
1199.     #                 $metabolite_counts++;
1200.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1201.     $out_array[3][6] += ($columns[$metabolite_counts][6] +
   $acidsum);
1202.     $BB += 1;
1203.     #                 $metabolite_counts++;
1204.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1205.     $out_array[3][6] += ($columns[$metabolite_counts][6] +
   $acidsum);
1206.     $BB += 1;
1207.     #                 $metabolite_counts++;
1208.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1209.     $out_array[11][6] += ($columns[$metabolite_counts][6] +
   $keysum);
1210.     $BB += 1;
1211.     #                 $metabolite_counts++;
1212.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1213.     $out_array[6][6] += ($columns[$metabolite_counts][6] +
   $alkanesum);
1214.     $BB += 1;
1215.     #                 $metabolite_counts++;
1216.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1217.     $out_array[5][6] += ($columns[$metabolite_counts][6] +
   $alkenesum);
1218.     $BB += 1;
1219.     #                 $metabolite_counts++;
1220.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1221.     $out_array[7][6] += ($columns[$metabolite_counts][6] +
   $alkynesum);
1222.     $BB += 1;
1223.     #                 $metabolite_counts++;
1224.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1225.     $out_array[12][6] += ($columns[$metabolite_counts][6] +
   $nitsum);
1226.     $BB += 1;
1227.     #                 $metabolite_counts++;
1228.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1229.     $out_array[8][6] += ($columns[$metabolite_counts][6] +
   $thiolsum);
1230.     $BB += 1;
1231.     #                 $metabolite_counts++;
1232.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1233.     $out_array[10][6] += ($columns[$metabolite_counts][6] +
   $indolesum);
1234.     $BB += 1;
1235.     #                 $metabolite_counts++;
1236.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1237.     $out_array[9][6] += ($columns[$metabolite_counts][6] +
   $sulfsum);
1238.     $BB += 1;
1239.     #                 $metabolite_counts++;
1240.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1241.     $out_array[13][6] += ($columns[$metabolite_counts][6] +
   $aminesum);
1242.     $BB += 1;
1243.     #                 $metabolite_counts++;
1244.     } elsif (( $BB == 0 )) {
1245.     $out_array[14][6] += ($columns[$metabolite_counts][6]+
   $othersum);
1246.     $BB += 1;
1247.     }
1248.     }
1249.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1250.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1251.     my $BB = 0;
1252.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1253.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1254.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1255.     $out_array[2][7] += ($columns[$metabolite_counts][7] +
   $alcsum);
1256.     $BB += 1;
1257.     #                 $metabolite_counts++;
1258.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1259.     $out_array[4][7] += ($columns[$metabolite_counts][7] +
   $aldsum);
1260.     $BB += 1;
1261.     #                 $metabolite_counts++;
1262.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1263.     $out_array[4][7] += ($columns[$metabolite_counts][7] +
   $aldsum);
1264.     $BB += 1;
1265.     #                 $metabolite_counts++;
1266.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1267.     $out_array[3][7] += ($columns[$metabolite_counts][7] +
   $acidsum);
1268.     $BB += 1;
1269.     #                 $metabolite_counts++;
1270.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1271.     $out_array[3][7] += ($columns[$metabolite_counts][7] +
   $acidsum);
1272.     $BB += 1;
1273.     #                 $metabolite_counts++;
1274.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1275.     $out_array[3][7] += ($columns[$metabolite_counts][7] +
   $acidsum);
1276.     $BB += 1;
1277.     #                 $metabolite_counts++;
1278.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1279.     $out_array[11][7] += ($columns[$metabolite_counts][7] +
   $keysum);
1280.     $BB += 1;
1281.     #                 $metabolite_counts++;
1282.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1283.     $out_array[6][7] += ($columns[$metabolite_counts][7] +
   $alkanesum);
1284.     $BB += 1;
1285.     #                 $metabolite_counts++;
1286.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1287.     $out_array[5][7] += ($columns[$metabolite_counts][7] +
   $alkenesum);
1288.     $BB += 1;
1289.     #                 $metabolite_counts++;
1290.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1291.     $out_array[7][7] += ($columns[$metabolite_counts][7] +
   $alkynesum);
1292.     $BB += 1;
1293.     #                 $metabolite_counts++;
1294.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1295.     $out_array[12][7] += ($columns[$metabolite_counts][7] +
   $nitsum);
1296.     $BB += 1;
1297.     #                 $metabolite_counts++;
1298.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1299.     $out_array[8][7] += ($columns[$metabolite_counts][7] +
   $thiolsum);
1300.     $BB += 1;
1301.     #                 $metabolite_counts++;
1302.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1303.     $out_array[10][7] += ($columns[$metabolite_counts][7] +
   $indolesum);
1304.     $BB += 1;
1305.     #                 $metabolite_counts++;
1306.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1307.     $out_array[9][7] += ($columns[$metabolite_counts][7] +
   $sulfsum);
1308.     $BB += 1;
1309.     #                 $metabolite_counts++;
1310.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1311.     $out_array[13][7] += ($columns[$metabolite_counts][7] +
   $aminesum);
1312.     $BB += 1;
1313.     #                 $metabolite_counts++;
1314.     } elsif (( $BB == 0 )) {
1315.     $out_array[14][7] += ($columns[$metabolite_counts][7]+
   $othersum);
1316.     $BB += 1;
1317.     }
1318.     }
1319.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1320.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1321.     my $BB = 0;
1322.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1323.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1324.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1325.     $out_array[2][8] += ($columns[$metabolite_counts][8] +
   $alcsum);
1326.     $BB += 1;
1327.     #                 $metabolite_counts++;
1328.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1329.     $out_array[4][8] += ($columns[$metabolite_counts][8] +
   $aldsum);
1330.     $BB += 1;
1331.     #                 $metabolite_counts++;
1332.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1333.     $out_array[4][8] += ($columns[$metabolite_counts][8] +
   $aldsum);
1334.     $BB += 1;
1335.     #                 $metabolite_counts++;
1336.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1337.     $out_array[3][8] += ($columns[$metabolite_counts][8] +
   $acidsum);
1338.     $BB += 1;
1339.     #                 $metabolite_counts++;
1340.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1341.     $out_array[3][8] += ($columns[$metabolite_counts][8] +
   $acidsum);
1342.     $BB += 1;
1343.     #                 $metabolite_counts++;
1344.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1345.     $out_array[3][8] += ($columns[$metabolite_counts][8] +
   $acidsum);
1346.     $BB += 1;
1347.     #                 $metabolite_counts++;
1348.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1349.     $out_array[11][8] += ($columns[$metabolite_counts][8] +
   $keysum);
1350.     $BB += 1;
1351.     #                 $metabolite_counts++;
1352.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1353.     $out_array[6][8] += ($columns[$metabolite_counts][8] +
   $alkanesum);
1354.     $BB += 1;
1355.     #                 $metabolite_counts++;
1356.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1357.     $out_array[5][8] += ($columns[$metabolite_counts][8] +
   $alkenesum);
1358.     $BB += 1;
1359.     #                 $metabolite_counts++;
1360.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1361.     $out_array[7][8] += ($columns[$metabolite_counts][8] +
   $alkynesum);
1362.     $BB += 1;
1363.     #                 $metabolite_counts++;
1364.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1365.     $out_array[12][8] += ($columns[$metabolite_counts][8] +
   $nitsum);
1366.     $BB += 1;
1367.     #                 $metabolite_counts++;
1368.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1369.     $out_array[8][8] += ($columns[$metabolite_counts][8] +
   $thiolsum);
1370.     $BB += 1;
1371.     #                 $metabolite_counts++;
1372.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1373.     $out_array[10][8] += ($columns[$metabolite_counts][8] +
   $indolesum);
1374.     $BB += 1;
1375.     #                 $metabolite_counts++;
1376.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1377.     $out_array[9][8] += ($columns[$metabolite_counts][8] +
   $sulfsum);
1378.     $BB += 1;
1379.     #                 $metabolite_counts++;
1380.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1381.     $out_array[13][8] += ($columns[$metabolite_counts][8] +
   $aminesum);
1382.     $BB += 1;
1383.     #                 $metabolite_counts++;
1384.     } elsif (( $BB == 0 )) {
1385.     $out_array[14][8] += ($columns[$metabolite_counts][8]+
   $othersum);
1386.     $BB += 1;
1387.     }
1388.     }
1389.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1390.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1391.     my $BB = 0;
1392.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1393.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1394.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1395.     $out_array[2][9] += ($columns[$metabolite_counts][9] +
   $alcsum);
1396.     $BB += 1;
1397.     #                 $metabolite_counts++;
1398.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/ and
   $columns[$metabolite_counts][0] !~ m/alkane/ and
   $columns[$metabolite_counts][0] !~ m/alkene/ and
   $columns[$metabolite_counts][0] !~ m/alkyne/)) {
1399.     $out_array[4][9] += ($columns[$metabolite_counts][9] +
   $aldsum);
1400.     $BB += 1;
1401.     #                 $metabolite_counts++;
1402.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1403.     $out_array[4][9] += ($columns[$metabolite_counts][9] +
   $aldsum);
1404.     $BB += 1;
1405.     #                 $metabolite_counts++;
1406.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1407.     $out_array[3][9] += ($columns[$metabolite_counts][9] +
   $acidsum);
1408.     $BB += 1;
1409.     #                 $metabolite_counts++;
1410.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1411.     $out_array[3][9] += ($columns[$metabolite_counts][9] +
   $acidsum);
1412.     $BB += 1;
1413.     #                 $metabolite_counts++;
1414.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1415.     $out_array[3][9] += ($columns[$metabolite_counts][9] +
   $acidsum);
1416.     $BB += 1;
1417.     #                 $metabolite_counts++;
1418.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1419.     $out_array[11][9] += ($columns[$metabolite_counts][9] +
   $keysum);
1420.     $BB += 1;
1421.     #                 $metabolite_counts++;
```

```
1422.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
1423.     $out_array[6][9] += ($columns[$metabolite_counts][9] +
   $alkanesum);
1424.     $BB += 1;
1425.     #                 $metabolite_counts++;
1426.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1427.     $out_array[5][9] += ($columns[$metabolite_counts][9] +
   $alkenesum);
1428.     $BB += 1;
1429.     #                 $metabolite_counts++;
1430.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1431.     $out_array[7][9] += ($columns[$metabolite_counts][9] +
   $alkynesum);
1432.     $BB += 1;
1433.     #                 $metabolite_counts++;
1434.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1435.     $out_array[12][9] += ($columns[$metabolite_counts][9] +
   $nitsum);
1436.     $BB += 1;
1437.     #                 $metabolite_counts++;
1438.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1439.     $out_array[8][9] += ($columns[$metabolite_counts][9] +
   $thiolsum);
1440.     $BB += 1;
1441.     #                 $metabolite_counts++;
1442.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1443.     $out_array[10][9] += ($columns[$metabolite_counts][9] +
   $indolesum);
1444.     $BB += 1;
1445.     #                 $metabolite_counts++;
1446.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1447.     $out_array[9][9] += ($columns[$metabolite_counts][9] +
   $sulfsum);
1448.     $BB += 1;
1449.     #                 $metabolite_counts++;
1450.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1451.     $out_array[13][9] += ($columns[$metabolite_counts][9] +
   $aminesum);
1452.     $BB += 1;
1453.     #                 $metabolite_counts++;
1454.     } elsif (( $BB == 0 )) {
1455.     $out_array[14][9] += ($columns[$metabolite_counts][9]+
   $othersum);
1456.     $BB += 1;
1457.     }
1458.     }
1459.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1460.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1461.     my $BB = 0;
1462.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1463.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1464.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1465.     $out_array[2][10] += ($columns[$metabolite_counts][10] +
   $alcsum);
1466.     $BB += 1;
1467.     #                 $metabolite_counts++;
1468.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1469.     $out_array[4][10] += ($columns[$metabolite_counts][10] +
   $aldsum);
1470.     $BB += 1;
1471.     #                 $metabolite_counts++;
1472.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1473.     $out_array[4][10] += ($columns[$metabolite_counts][10] +
   $aldsum);
1474.     $BB += 1;
1475.     #                 $metabolite_counts++;
1476.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1477.     $out_array[3][10] += ($columns[$metabolite_counts][10] +
   $acidsum);
1478.     $BB += 1;
1479.     #                 $metabolite_counts++;
1480.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1481.     $out_array[3][10] += ($columns[$metabolite_counts][10] +
   $acidsum);
1482.     $BB += 1;
1483.     #                 $metabolite_counts++;
1484.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1485.     $out_array[3][10] += ($columns[$metabolite_counts][10] +
   $acidsum);
1486.     $BB += 1;
1487.     #                 $metabolite_counts++;
1488.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1489.     $out_array[11][10] += ($columns[$metabolite_counts][10] +
   $keysum);
1490.     $BB += 1;
1491.     #                 $metabolite_counts++;
1492.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1493.     $out_array[6][10] += ($columns[$metabolite_counts][10] +
   $alkanesum);
1494.     $BB += 1;
1495.     #                 $metabolite_counts++;
1496.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1497.     $out_array[5][10] += ($columns[$metabolite_counts][10] +
   $alkenesum);
1498.     $BB += 1;
1499.     #                 $metabolite_counts++;
1500.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1501.     $out_array[7][10] += ($columns[$metabolite_counts][10] +
   $alkynesum);
1502.     $BB += 1;
1503.     #                 $metabolite_counts++;
1504.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1505.     $out_array[12][10] += ($columns[$metabolite_counts][10] +
   $nitsum);
1506.     $BB += 1;
1507.     #                 $metabolite_counts++;
1508.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1509.     $out_array[8][10] += ($columns[$metabolite_counts][10] +
   $thiolsum);
1510.     $BB += 1;
1511.     #                 $metabolite_counts++;
1512.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1513.     $out_array[10][10] += ($columns[$metabolite_counts][10] +
   $indolesum);
1514.     $BB += 1;
1515.     #                 $metabolite_counts++;
1516.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1517.     $out_array[9][10] += ($columns[$metabolite_counts][10] +
   $sulfsum);
1518.     $BB += 1;
1519.     #                 $metabolite_counts++;
1520.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1521.     $out_array[13][10] += ($columns[$metabolite_counts][10] +
   $aminesum);
1522.     $BB += 1;
1523.     #                 $metabolite_counts++;
1524.     } elsif (( $BB == 0 )) {
1525.     $out_array[14][10] += ($columns[$metabolite_counts][10]+
   $othersum);
1526.     $BB += 1;
1527.     }
1528.     }
1529.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1530.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1531.     my $BB = 0;
1532.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1533.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1534.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1535.     $out_array[2][11] += ($columns[$metabolite_counts][11] +
   $alcsum);
1536.     $BB += 1;
1537.     #                 $metabolite_counts++;
1538.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1539.     $out_array[4][11] += ($columns[$metabolite_counts][11] +
   $aldsum);
1540.     $BB += 1;
1541.     #                 $metabolite_counts++;
1542.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1543.     $out_array[4][11] += ($columns[$metabolite_counts][11] +
   $aldsum);
1544.     $BB += 1;
1545.     #                 $metabolite_counts++;
1546.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1547.     $out_array[3][11] += ($columns[$metabolite_counts][11] +
   $acidsum);
1548.     $BB += 1;
1549.     #                 $metabolite_counts++;
1550.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1551.     $out_array[3][11] += ($columns[$metabolite_counts][11] +
   $acidsum);
1552.     $BB += 1;
1553.     #                 $metabolite_counts++;
1554.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1555.     $out_array[3][11] += ($columns[$metabolite_counts][11] +
   $acidsum);
1556.     $BB += 1;
1557.     #                 $metabolite_counts++;
1558.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1559.     $out_array[11][11] += ($columns[$metabolite_counts][11] +
   $keysum);
1560.     $BB += 1;
1561.     #                 $metabolite_counts++;
1562.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1563.     $out_array[6][11] += ($columns[$metabolite_counts][11] +
   $alkanesum);
1564.     $BB += 1;
1565.     #                 $metabolite_counts++;
1566.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1567.     $out_array[5][11] += ($columns[$metabolite_counts][11] +
   $alkenesum);
1568.     $BB += 1;
1569.     #                 $metabolite_counts++;
1570.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1571.     $out_array[7][11] += ($columns[$metabolite_counts][11] +
   $alkynesum);
1572.     $BB += 1;
1573.     #                 $metabolite_counts++;
1574.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1575.     $out_array[12][11] += ($columns[$metabolite_counts][11] +
   $nitsum);
1576.     $BB += 1;
1577.     #                 $metabolite_counts++;
1578.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1579.     $out_array[8][11] += ($columns[$metabolite_counts][11] +
   $thiolsum);
1580.     $BB += 1;
1581.     #                 $metabolite_counts++;
1582.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1583.     $out_array[10][11] += ($columns[$metabolite_counts][11] +
   $indolesum);
1584.     $BB += 1;
1585.     #                 $metabolite_counts++;
1586.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1587.     $out_array[9][11] += ($columns[$metabolite_counts][11] +
   $sulfsum);
1588.     $BB += 1;
1589.     #                 $metabolite_counts++;
1590.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1591.     $out_array[13][11] += ($columns[$metabolite_counts][11] +
   $aminesum);
1592.     $BB += 1;
1593.     #                 $metabolite_counts++;
1594.     } elsif (( $BB == 0 )) {
1595.     $out_array[14][11] += ($columns[$metabolite_counts][11]+
   $othersum);
1596.     $BB += 1;
1597.     }
1598.     }
1599.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1600.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1601.     my $BB = 0;
1602.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1603.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1604.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1605.     $out_array[2][12] += ($columns[$metabolite_counts][12] +
   $alcsum);
1606.     $BB += 1;
1607.     #                 $metabolite_counts++;
1608.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1609.     $out_array[4][12] += ($columns[$metabolite_counts][12] +
   $aldsum);
1610.     $BB += 1;
1611.     #                 $metabolite_counts++;
1612.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1613.     $out_array[4][12] += ($columns[$metabolite_counts][12] +
   $aldsum);
1614.     $BB += 1;
1615.     #                 $metabolite_counts++;
1616.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1617.     $out_array[3][12] += ($columns[$metabolite_counts][12] +
   $acidsum);
1618.     $BB += 1;
1619.     #                 $metabolite_counts++;
1620.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1621.     $out_array[3][12] += ($columns[$metabolite_counts][12] +
   $acidsum);
1622.     $BB += 1;
1623.     #                 $metabolite_counts++;
1624.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1625.     $out_array[3][12] += ($columns[$metabolite_counts][12] +
   $acidsum);
1626.     $BB += 1;
1627.     #                 $metabolite_counts++;
1628.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1629.     $out_array[11][12] += ($columns[$metabolite_counts][12] +
   $keysum);
1630.     $BB += 1;
1631.     #                 $metabolite_counts++;
1632.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1633.     $out_array[6][12] += ($columns[$metabolite_counts][12] +
   $alkanesum);
1634.     $BB += 1;
1635.     #                 $metabolite_counts++;
1636.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1637.     $out_array[5][12] += ($columns[$metabolite_counts][12] +
   $alkenesum);
1638.     $BB += 1;
1639.     #                 $metabolite_counts++;
1640.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1641.     $out_array[7][12] += ($columns[$metabolite_counts][12] +
   $alkynesum);
1642.     $BB += 1;
1643.     #                 $metabolite_counts++;
1644.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1645.     $out_array[12][12] += ($columns[$metabolite_counts][12] +
   $nitsum);
1646.     $BB += 1;
1647.     #                 $metabolite_counts++;
1648.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1649.     $out_array[8][12] += ($columns[$metabolite_counts][12] +
   $thiolsum);
1650.     $BB += 1;
1651.     #                 $metabolite_counts++;
1652.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1653.     $out_array[10][12] += ($columns[$metabolite_counts][12] +
   $indolesum);
1654.     $BB += 1;
1655.     #                 $metabolite_counts++;
1656.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1657.     $out_array[9][12] += ($columns[$metabolite_counts][12] +
   $sulfsum);
1658.     $BB += 1;
1659.     #                 $metabolite_counts++;
1660.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1661.     $out_array[13][12] += ($columns[$metabolite_counts][12] +
   $aminesum);
1662.     $BB += 1;
1663.     #                 $metabolite_counts++;
1664.     } elsif (( $BB == 0 )) {
1665.     $out_array[14][12] += ($columns[$metabolite_counts][12]+
   $othersum);
1666.     $BB += 1;
1667.     }
1668.     }
1669.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1670.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1671.     my $BB = 0;
1672.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1673.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1674.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1675.     $out_array[2][13] += ($columns[$metabolite_counts][13] +
   $alcsum);
1676.     $BB += 1;
1677.     #                 $metabolite_counts++;
1678.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1679.     $out_array[4][13] += ($columns[$metabolite_counts][13] +
   $aldsum);
1680.     $BB += 1;
1681.     #                 $metabolite_counts++;
1682.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1683.     $out_array[4][13] += ($columns[$metabolite_counts][13] +
   $aldsum);
1684.     $BB += 1;
1685.     #                 $metabolite_counts++;
1686.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1687.     $out_array[3][13] += ($columns[$metabolite_counts][13] +
   $acidsum);
1688.     $BB += 1;
1689.     #                 $metabolite_counts++;
1690.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1691.     $out_array[3][13] += ($columns[$metabolite_counts][13] +
   $acidsum);
1692.     $BB += 1;
1693.     #                 $metabolite_counts++;
1694.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1695.     $out_array[3][13] += ($columns[$metabolite_counts][13] +
   $acidsum);
1696.     $BB += 1;
1697.     #                 $metabolite_counts++;
1698.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1699.     $out_array[11][13] += ($columns[$metabolite_counts][13] +
   $keysum);
1700.     $BB += 1;
1701.     #                 $metabolite_counts++;
1702.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1703.     $out_array[6][13] += ($columns[$metabolite_counts][13] +
   $alkanesum);
1704.     $BB += 1;
1705.     #                 $metabolite_counts++;
1706.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1707.     $out_array[5][13] += ($columns[$metabolite_counts][13] +
   $alkenesum);
1708.     $BB += 1;
1709.     #                 $metabolite_counts++;
1710.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1711.     $out_array[7][13] += ($columns[$metabolite_counts][13] +
   $alkynesum);
1712.     $BB += 1;
1713.     #                 $metabolite_counts++;
1714.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1715.     $out_array[12][13] += ($columns[$metabolite_counts][13] +
   $nitsum);
1716.     $BB += 1;
1717.     #                 $metabolite_counts++;
1718.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1719.     $out_array[8][13] += ($columns[$metabolite_counts][13] +
   $thiolsum);
1720.     $BB += 1;
1721.     #                 $metabolite_counts++;
1722.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1723.     $out_array[10][13] += ($columns[$metabolite_counts][13] +
   $indolesum);
1724.     $BB += 1;
1725.     #                 $metabolite_counts++;
1726.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1727.     $out_array[9][13] += ($columns[$metabolite_counts][13] +
   $sulfsum);
1728.     $BB += 1;
1729.     #                 $metabolite_counts++;
1730.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1731.     $out_array[13][13] += ($columns[$metabolite_counts][13] +
   $aminesum);
1732.     $BB += 1;
1733.     #                 $metabolite_counts++;
1734.     } elsif (( $BB == 0 )) {
1735.     $out_array[14][13] += ($columns[$metabolite_counts][13]+
   $othersum);
1736.     $BB += 1;
1737.     }
1738.     }
1739.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1740.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1741.     my $BB = 0;
1742.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1743.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1744.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1745.     $out_array[2][14] += ($columns[$metabolite_counts][14] +
   $alcsum);
1746.     $BB += 1;
1747.     #                 $metabolite_counts++;
1748.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1749.     $out_array[4][14] += ($columns[$metabolite_counts][14] +
   $aldsum);
1750.     $BB += 1;
1751.     #                 $metabolite_counts++;
1752.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1753.     $out_array[4][14] += ($columns[$metabolite_counts][14] +
   $aldsum);
1754.     $BB += 1;
1755.     #                 $metabolite_counts++;
1756.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1757.     $out_array[3][14] += ($columns[$metabolite_counts][14] +
   $acidsum);
1758.     $BB += 1;
1759.     #                 $metabolite_counts++;
1760.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1761.     $out_array[3][14] += ($columns[$metabolite_counts][14] +
   $acidsum);
1762.     $BB += 1;
1763.     #                 $metabolite_counts++;
1764.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1765.     $out_array[3][14] += ($columns[$metabolite_counts][14] +
   $acidsum);
1766.     $BB += 1;
1767.     #                 $metabolite_counts++;
1768.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1769.     $out_array[11][14] += ($columns[$metabolite_counts][14] +
   $keysum);
1770.     $BB += 1;
1771.     #                 $metabolite_counts++;
1772.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1773.     $out_array[6][14] += ($columns[$metabolite_counts][14] +
   $alkanesum);
1774.     $BB += 1;
1775.     #                 $metabolite_counts++;
1776.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1777.     $out_array[5][14] += ($columns[$metabolite_counts][14] +
   $alkenesum);
1778.     $BB += 1;
1779.     #                 $metabolite_counts++;
1780.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1781.     $out_array[7][14] += ($columns[$metabolite_counts][14] +
   $alkynesum);
1782.     $BB += 1;
1783.     #                 $metabolite_counts++;
1784.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1785.     $out_array[12][14] += ($columns[$metabolite_counts][14] +
   $nitsum);
1786.     $BB += 1;
1787.     #                 $metabolite_counts++;
1788.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1789.     $out_array[8][14] += ($columns[$metabolite_counts][14] +
   $thiolsum);
1790.     $BB += 1;
1791.     #                 $metabolite_counts++;
1792.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1793.     $out_array[10][14] += ($columns[$metabolite_counts][14] +
   $indolesum);
1794.     $BB += 1;
1795.     #                 $metabolite_counts++;
1796.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1797.     $out_array[9][14] += ($columns[$metabolite_counts][14] +
   $sulfsum);
1798.     $BB += 1;
1799.     #                 $metabolite_counts++;
1800.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1801.     $out_array[13][14] += ($columns[$metabolite_counts][14] +
   $aminesum);
1802.     $BB += 1;
1803.     #                 $metabolite_counts++;
1804.     } elsif (( $BB == 0 )) {
1805.     $out_array[14][14] += ($columns[$metabolite_counts][14]+
   $othersum);
1806.     $BB += 1;
1807.     }
1808.     }
1809.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1810.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1811.     my $BB = 0;
1812.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1813.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1814.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1815.     $out_array[2][15] += ($columns[$metabolite_counts][15] +
   $alcsum);
1816.     $BB += 1;
1817.     #                 $metabolite_counts++;
1818.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1819.     $out_array[4][15] += ($columns[$metabolite_counts][15] +
   $aldsum);
1820.     $BB += 1;
1821.     #                 $metabolite_counts++;
1822.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1823.     $out_array[4][15] += ($columns[$metabolite_counts][15] +
   $aldsum);
1824.     $BB += 1;
1825.     #                 $metabolite_counts++;
1826.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1827.     $out_array[3][15] += ($columns[$metabolite_counts][15] +
   $acidsum);
1828.     $BB += 1;
1829.     #                 $metabolite_counts++;
1830.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1831.     $out_array[3][15] += ($columns[$metabolite_counts][15] +
   $acidsum);
1832.     $BB += 1;
1833.     #                 $metabolite_counts++;
1834.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1835.     $out_array[3][15] += ($columns[$metabolite_counts][15] +
   $acidsum);
1836.     $BB += 1;
1837.     #                 $metabolite_counts++;
1838.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1839.     $out_array[11][15] += ($columns[$metabolite_counts][15] +
   $keysum);
1840.     $BB += 1;
1841.     #                 $metabolite_counts++;
1842.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1843.     $out_array[6][15] += ($columns[$metabolite_counts][15] +
   $alkanesum);
1844.     $BB += 1;
1845.     #                 $metabolite_counts++;
1846.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1847.     $out_array[5][15] += ($columns[$metabolite_counts][15] +
   $alkenesum);
1848.     $BB += 1;
1849.     #                 $metabolite_counts++;
1850.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1851.     $out_array[7][15] += ($columns[$metabolite_counts][15] +
   $alkynesum);
1852.     $BB += 1;
1853.     #                 $metabolite_counts++;
1854.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1855.     $out_array[12][15] += ($columns[$metabolite_counts][15] +
   $nitsum);
1856.     $BB += 1;
1857.     #                 $metabolite_counts++;
1858.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1859.     $out_array[8][15] += ($columns[$metabolite_counts][15] +
   $thiolsum);
1860.     $BB += 1;
1861.     #                 $metabolite_counts++;
1862.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1863.     $out_array[10][15] += ($columns[$metabolite_counts][15] +
   $indolesum);
1864.     $BB += 1;
1865.     #                 $metabolite_counts++;
1866.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1867.     $out_array[9][15] += ($columns[$metabolite_counts][15] +
   $sulfsum);
1868.     $BB += 1;
1869.     #                 $metabolite_counts++;
1870.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1871.     $out_array[13][15] += ($columns[$metabolite_counts][15] +
   $aminesum);
1872.     $BB += 1;
1873.     #                 $metabolite_counts++;
1874.     } elsif (( $BB == 0 )) {
1875.     $out_array[14][15] += ($columns[$metabolite_counts][15]+
   $othersum);
1876.     $BB += 1;
1877.     }
1878.     }
1879.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1880.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1881.     my $BB = 0;
1882.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1883.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1884.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1885.     $out_array[2][16] += ($columns[$metabolite_counts][16] +
   $alcsum);
1886.     $BB += 1;
1887.     #                 $metabolite_counts++;
1888.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1889.     $out_array[4][16] += ($columns[$metabolite_counts][16] +
   $aldsum);
1890.     $BB += 1;
1891.     #                 $metabolite_counts++;
1892.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1893.     $out_array[4][16] += ($columns[$metabolite_counts][16] +
   $aldsum);
1894.     $BB += 1;
1895.     #                 $metabolite_counts++;
1896.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1897.     $out_array[3][16] += ($columns[$metabolite_counts][16] +
   $acidsum);
1898.     $BB += 1;
1899.     #                 $metabolite_counts++;
1900.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1901.     $out_array[3][16] += ($columns[$metabolite_counts][16] +
   $acidsum);
1902.     $BB += 1;
1903.     #                 $metabolite_counts++;
1904.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1905.     $out_array[3][16] += ($columns[$metabolite_counts][16] +
   $acidsum);
1906.     $BB += 1;
1907.     #                 $metabolite_counts++;
1908.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1909.     $out_array[11][16] += ($columns[$metabolite_counts][16] +
   $keysum);
1910.     $BB += 1;
1911.     #                 $metabolite_counts++;
1912.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1913.     $out_array[6][16] += ($columns[$metabolite_counts][16] +
   $alkanesum);
1914.     $BB += 1;
1915.     #                 $metabolite_counts++;
1916.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1917.     $out_array[5][16] += ($columns[$metabolite_counts][16] +
   $alkenesum);
1918.     $BB += 1;
1919.     #                 $metabolite_counts++;
1920.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1921.     $out_array[7][16] += ($columns[$metabolite_counts][16] +
   $alkynesum);
1922.     $BB += 1;
1923.     #                 $metabolite_counts++;
1924.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1925.     $out_array[12][16] += ($columns[$metabolite_counts][16] +
   $nitsum);
1926.     $BB += 1;
1927.     #                 $metabolite_counts++;
1928.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1929.     $out_array[8][16] += ($columns[$metabolite_counts][16] +
   $thiolsum);
1930.     $BB += 1;
1931.     #                 $metabolite_counts++;
1932.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
1933.     $out_array[10][16] += ($columns[$metabolite_counts][16] +
   $indolesum);
1934.     $BB += 1;
1935.     #                 $metabolite_counts++;
1936.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
1937.     $out_array[9][16] += ($columns[$metabolite_counts][16] +
   $sulfsum);
1938.     $BB += 1;
1939.     #                 $metabolite_counts++;
1940.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
1941.     $out_array[13][16] += ($columns[$metabolite_counts][16] +
   $aminesum);
1942.     $BB += 1;
1943.     #                 $metabolite_counts++;
1944.     } elsif (( $BB == 0 )) {
1945.     $out_array[14][16] += ($columns[$metabolite_counts][16]+
   $othersum);
1946.     $BB += 1;
1947.     }
1948.     }
1949.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
1950.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
1951.     my $BB = 0;
1952.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
1953.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
1954.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
1955.     $out_array[2][17] += ($columns[$metabolite_counts][17] +
   $alcsum);
1956.     $BB += 1;
1957.     #                 $metabolite_counts++;
1958.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1959.     $out_array[4][17] += ($columns[$metabolite_counts][17] +
   $aldsum);
1960.     $BB += 1;
1961.     #                 $metabolite_counts++;
1962.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
1963.     $out_array[4][17] += ($columns[$metabolite_counts][17] +
   $aldsum);
1964.     $BB += 1;
1965.     #                 $metabolite_counts++;
1966.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
1967.     $out_array[3][17] += ($columns[$metabolite_counts][17] +
   $acidsum);
1968.     $BB += 1;
1969.     #                 $metabolite_counts++;
1970.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
1971.     $out_array[3][17] += ($columns[$metabolite_counts][17] +
   $acidsum);
1972.     $BB += 1;
1973.     #                 $metabolite_counts++;
1974.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
1975.     $out_array[3][17] += ($columns[$metabolite_counts][17] +
   $acidsum);
1976.     $BB += 1;
1977.     #                 $metabolite_counts++;
1978.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
1979.     $out_array[11][17] += ($columns[$metabolite_counts][17] +
   $keysum);
1980.     $BB += 1;
1981.     #                 $metabolite_counts++;
1982.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
1983.     $out_array[6][17] += ($columns[$metabolite_counts][17] +
   $alkanesum);
1984.     $BB += 1;
1985.     #                 $metabolite_counts++;
1986.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
1987.     $out_array[5][17] += ($columns[$metabolite_counts][17] +
   $alkenesum);
1988.     $BB += 1;
1989.     #                 $metabolite_counts++;
1990.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
1991.     $out_array[7][17] += ($columns[$metabolite_counts][17] +
   $alkynesum);
1992.     $BB += 1;
1993.     #                 $metabolite_counts++;
1994.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
1995.     $out_array[12][17] += ($columns[$metabolite_counts][17] +
   $nitsum);
1996.     $BB += 1;
1997.     #                 $metabolite_counts++;
1998.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
1999.     $out_array[8][17] += ($columns[$metabolite_counts][17] +
   $thiolsum);
2000.     $BB += 1;
2001.     #                 $metabolite_counts++;
2002.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2003.     $out_array[10][17] += ($columns[$metabolite_counts][17] +
   $indolesum);
2004.     $BB += 1;
2005.     #                 $metabolite_counts++;
2006.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2007.     $out_array[9][17] += ($columns[$metabolite_counts][17] +
   $sulfsum);
2008.     $BB += 1;
2009.     #                 $metabolite_counts++;
2010.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2011.     $out_array[13][17] += ($columns[$metabolite_counts][17] +
   $aminesum);
2012.     $BB += 1;
2013.     #                 $metabolite_counts++;
2014.     } elsif (( $BB == 0 )) {
2015.     $out_array[14][17] += ($columns[$metabolite_counts][17]+
   $othersum);
2016.     $BB += 1;
2017.     }
2018.     }
2019.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2020.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2021.     my $BB = 0;
2022.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2023.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2024.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2025.     $out_array[2][18] += ($columns[$metabolite_counts][18] +
   $alcsum);
2026.     $BB += 1;
2027.     #                 $metabolite_counts++;
2028.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2029.     $out_array[4][18] += ($columns[$metabolite_counts][18] +
   $aldsum);
2030.     $BB += 1;
2031.     #                 $metabolite_counts++;
2032.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2033.     $out_array[4][18] += ($columns[$metabolite_counts][18] +
   $aldsum);
2034.     $BB += 1;
2035.     #                 $metabolite_counts++;
2036.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2037.     $out_array[3][18] += ($columns[$metabolite_counts][18] +
   $acidsum);
2038.     $BB += 1;
2039.     #                 $metabolite_counts++;
2040.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2041.     $out_array[3][18] += ($columns[$metabolite_counts][18] +
   $acidsum);
2042.     $BB += 1;
2043.     #                 $metabolite_counts++;
2044.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2045.     $out_array[3][18] += ($columns[$metabolite_counts][18] +
   $acidsum);
2046.     $BB += 1;
2047.     #                 $metabolite_counts++;
2048.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2049.     $out_array[11][18] += ($columns[$metabolite_counts][18] +
   $keysum);
2050.     $BB += 1;
2051.     #                 $metabolite_counts++;
2052.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2053.     $out_array[6][18] += ($columns[$metabolite_counts][18] +
   $alkanesum);
2054.     $BB += 1;
2055.     #                 $metabolite_counts++;
2056.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2057.     $out_array[5][18] += ($columns[$metabolite_counts][18] +
   $alkenesum);
2058.     $BB += 1;
2059.     #                 $metabolite_counts++;
2060.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2061.     $out_array[7][18] += ($columns[$metabolite_counts][18] +
   $alkynesum);
2062.     $BB += 1;
2063.     #                 $metabolite_counts++;
2064.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2065.     $out_array[12][18] += ($columns[$metabolite_counts][18] +
   $nitsum);
2066.     $BB += 1;
2067.     #                 $metabolite_counts++;
2068.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2069.     $out_array[8][18] += ($columns[$metabolite_counts][18] +
   $thiolsum);
2070.     $BB += 1;
2071.     #                 $metabolite_counts++;
2072.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2073.     $out_array[10][18] += ($columns[$metabolite_counts][18] +
   $indolesum);
2074.     $BB += 1;
2075.     #                 $metabolite_counts++;
2076.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2077.     $out_array[9][18] += ($columns[$metabolite_counts][18] +
   $sulfsum);
2078.     $BB += 1;
2079.     #                 $metabolite_counts++;
2080.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2081.     $out_array[13][18] += ($columns[$metabolite_counts][18] +
   $aminesum);
2082.     $BB += 1;
2083.     #                 $metabolite_counts++;
2084.     } elsif (( $BB == 0 )) {
2085.     $out_array[14][18] += ($columns[$metabolite_counts][18]+
   $othersum);
2086.     $BB += 1;
2087.     }
2088.     }
2089.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2090.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2091.     my $BB = 0;
2092.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2093.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2094.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2095.     $out_array[2][19] += ($columns[$metabolite_counts][19] +
   $alcsum);
2096.     $BB += 1;
2097.     #                 $metabolite_counts++;
2098.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/ )) {
2099.     $out_array[4][19] += ($columns[$metabolite_counts][19] +
   $aldsum);
2100.     $BB += 1;
2101.     #                 $metabolite_counts++;
2102.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2103.     $out_array[4][19] += ($columns[$metabolite_counts][19] +
   $aldsum);
2104.     $BB += 1;
2105.     #                 $metabolite_counts++;
2106.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2107.     $out_array[3][19] += ($columns[$metabolite_counts][19] +
   $acidsum);
2108.     $BB += 1;
2109.     #                 $metabolite_counts++;
2110.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2111.     $out_array[3][19] += ($columns[$metabolite_counts][19] +
   $acidsum);
2112.     $BB += 1;
2113.     #                 $metabolite_counts++;
2114.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2115.     $out_array[3][19] += ($columns[$metabolite_counts][19] +
   $acidsum);
2116.     $BB += 1;
2117.     #                 $metabolite_counts++;
2118.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2119.     $out_array[11][19] += ($columns[$metabolite_counts][19] +
   $keysum);
2120.     $BB += 1;
2121.     #                 $metabolite_counts++;
2122.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2123.     $out_array[6][19] += ($columns[$metabolite_counts][19] +
   $alkanesum);
2124.     $BB += 1;
2125.     #                 $metabolite_counts++;
2126.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2127.     $out_array[5][19] += ($columns[$metabolite_counts][19] +
   $alkenesum);
2128.     $BB += 1;
2129.     #                 $metabolite_counts++;
2130.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2131.     $out_array[7][19] += ($columns[$metabolite_counts][19] +
   $alkynesum);
2132.     $BB += 1;
2133.     #                 $metabolite_counts++;
2134.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2135.     $out_array[12][19] += ($columns[$metabolite_counts][19] +
   $nitsum);
2136.     $BB += 1;
2137.     #                 $metabolite_counts++;
2138.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2139.     $out_array[8][19] += ($columns[$metabolite_counts][19] +
   $thiolsum);
2140.     $BB += 1;
2141.     #                 $metabolite_counts++;
2142.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2143.     $out_array[10][19] += ($columns[$metabolite_counts][19] +
   $indolesum);
2144.     $BB += 1;
2145.     #                 $metabolite_counts++;
2146.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2147.     $out_array[9][19] += ($columns[$metabolite_counts][19] +
   $sulfsum);
2148.     $BB += 1;
2149.     #                 $metabolite_counts++;
2150.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2151.     $out_array[13][19] += ($columns[$metabolite_counts][19] +
   $aminesum);
2152.     $BB += 1;
2153.     #                 $metabolite_counts++;
2154.     } elsif (( $BB == 0 )) {
2155.     $out_array[14][19] += ($columns[$metabolite_counts][19]+
   $othersum);
2156.     $BB += 1;
2157.     }
2158.     }
2159.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2160.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2161.     my $BB = 0;
2162.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2163.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2164.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2165.     $out_array[2][20] += ($columns[$metabolite_counts][20] +
   $alcsum);
2166.     $BB += 1;
2167.     #                 $metabolite_counts++;
2168.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2169.     $out_array[4][20] += ($columns[$metabolite_counts][20] +
   $aldsum);
2170.     $BB += 1;
2171.     #                 $metabolite_counts++;
2172.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2173.     $out_array[4][20] += ($columns[$metabolite_counts][20] +
   $aldsum);
2174.     $BB += 1;
2175.     #                 $metabolite_counts++;
2176.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2177.     $out_array[3][20] += ($columns[$metabolite_counts][20] +
   $acidsum);
2178.     $BB += 1;
2179.     #                 $metabolite_counts++;
2180.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2181.     $out_array[3][20] += ($columns[$metabolite_counts][20] +
   $acidsum);
2182.     $BB += 1;
2183.     #                 $metabolite_counts++;
2184.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2185.     $out_array[3][20] += ($columns[$metabolite_counts][20] +
   $acidsum);
2186.     $BB += 1;
2187.     #                 $metabolite_counts++;
2188.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2189.     $out_array[11][20] += ($columns[$metabolite_counts][20] +
   $keysum);
2190.     $BB += 1;
2191.     #                 $metabolite_counts++;
2192.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2193.     $out_array[6][20] += ($columns[$metabolite_counts][20] +
   $alkanesum);
2194.     $BB += 1;
2195.     #                 $metabolite_counts++;
2196.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2197.     $out_array[5][20] += ($columns[$metabolite_counts][20] +
   $alkenesum);
2198.     $BB += 1;
2199.     #                 $metabolite_counts++;
2200.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2201.     $out_array[7][20] += ($columns[$metabolite_counts][20] +
   $alkynesum);
2202.     $BB += 1;
2203.     #                 $metabolite_counts++;
2204.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2205.     $out_array[12][20] += ($columns[$metabolite_counts][20] +
   $nitsum);
2206.     $BB += 1;
2207.     #                 $metabolite_counts++;
2208.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2209.     $out_array[8][20] += ($columns[$metabolite_counts][20] +
   $thiolsum);
2210.     $BB += 1;
2211.     #                 $metabolite_counts++;
2212.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2213.     $out_array[10][20] += ($columns[$metabolite_counts][20] +
   $indolesum);
2214.     $BB += 1;
2215.     #                 $metabolite_counts++;
2216.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2217.     $out_array[9][20] += ($columns[$metabolite_counts][20] +
   $sulfsum);
2218.     $BB += 1;
2219.     #                 $metabolite_counts++;
2220.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2221.     $out_array[13][20] += ($columns[$metabolite_counts][20] +
   $aminesum);
2222.     $BB += 1;
2223.     #                 $metabolite_counts++;
2224.     } elsif (( $BB == 0 )) {
2225.     $out_array[14][20] += ($columns[$metabolite_counts][20]+
   $othersum);
2226.     $BB += 1;
2227.     }
2228.     }
2229.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2230.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2231.     my $BB = 0;
2232.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2233.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2234.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2235.     $out_array[2][21] += ($columns[$metabolite_counts][21] +
   $alcsum);
2236.     $BB += 1;
2237.     #                 $metabolite_counts++;
2238.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2239.     $out_array[4][21] += ($columns[$metabolite_counts][21] +
   $aldsum);
2240.     $BB += 1;
2241.     #                 $metabolite_counts++;
2242.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2243.     $out_array[4][21] += ($columns[$metabolite_counts][21] +
   $aldsum);
2244.     $BB += 1;
2245.     #                 $metabolite_counts++;
2246.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2247.     $out_array[3][21] += ($columns[$metabolite_counts][21] +
   $acidsum);
2248.     $BB += 1;
2249.     #                 $metabolite_counts++;
2250.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2251.     $out_array[3][21] += ($columns[$metabolite_counts][21] +
   $acidsum);
2252.     $BB += 1;
2253.     #                 $metabolite_counts++;
2254.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2255.     $out_array[3][21] += ($columns[$metabolite_counts][21] +
   $acidsum);
2256.     $BB += 1;
2257.     #                 $metabolite_counts++;
2258.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2259.     $out_array[11][21] += ($columns[$metabolite_counts][21] +
   $keysum);
2260.     $BB += 1;
2261.     #                 $metabolite_counts++;
2262.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2263.     $out_array[6][21] += ($columns[$metabolite_counts][21] +
   $alkanesum);
2264.     $BB += 1;
2265.     #                 $metabolite_counts++;
2266.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2267.     $out_array[5][21] += ($columns[$metabolite_counts][21] +
   $alkenesum);
2268.     $BB += 1;
2269.     #                 $metabolite_counts++;
2270.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2271.     $out_array[7][21] += ($columns[$metabolite_counts][21] +
   $alkynesum);
2272.     $BB += 1;
2273.     #                 $metabolite_counts++;
2274.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2275.     $out_array[12][21] += ($columns[$metabolite_counts][21] +
   $nitsum);
2276.     $BB += 1;
2277.     #                 $metabolite_counts++;
2278.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2279.     $out_array[8][21] += ($columns[$metabolite_counts][21] +
   $thiolsum);
2280.     $BB += 1;
2281.     #                 $metabolite_counts++;
2282.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2283.     $out_array[10][21] += ($columns[$metabolite_counts][21] +
   $indolesum);
2284.     $BB += 1;
2285.     #                 $metabolite_counts++;
2286.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2287.     $out_array[9][21] += ($columns[$metabolite_counts][21] +
   $sulfsum);
2288.     $BB += 1;
2289.     #                 $metabolite_counts++;
2290.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2291.     $out_array[13][21] += ($columns[$metabolite_counts][21] +
   $aminesum);
2292.     $BB += 1;
2293.     #                 $metabolite_counts++;
2294.     } elsif (( $BB == 0 )) {
2295.     $out_array[14][21] += ($columns[$metabolite_counts][21]+
   $othersum);
2296.     $BB += 1;
2297.     }
2298.     }
2299.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2300.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2301.     my $BB = 0;
2302.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2303.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2304.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2305.     $out_array[2][22] += ($columns[$metabolite_counts][22] +
   $alcsum);
2306.     $BB += 1;
2307.     #                 $metabolite_counts++;
2308.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2309.     $out_array[4][22] += ($columns[$metabolite_counts][22] +
   $aldsum);
2310.     $BB += 1;
2311.     #                 $metabolite_counts++;
2312.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2313.     $out_array[4][22] += ($columns[$metabolite_counts][22] +
   $aldsum);
2314.     $BB += 1;
2315.     #                 $metabolite_counts++;
2316.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2317.     $out_array[3][22] += ($columns[$metabolite_counts][22] +
   $acidsum);
2318.     $BB += 1;
2319.     #                 $metabolite_counts++;
2320.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2321.     $out_array[3][22] += ($columns[$metabolite_counts][22] +
   $acidsum);
2322.     $BB += 1;
2323.     #                 $metabolite_counts++;
2324.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2325.     $out_array[3][22] += ($columns[$metabolite_counts][22] +
   $acidsum);
2326.     $BB += 1;
2327.     #                 $metabolite_counts++;
2328.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2329.     $out_array[11][22] += ($columns[$metabolite_counts][22] +
   $keysum);
2330.     $BB += 1;
2331.     #                 $metabolite_counts++;
2332.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2333.     $out_array[6][22] += ($columns[$metabolite_counts][22] +
   $alkanesum);
2334.     $BB += 1;
2335.     #                 $metabolite_counts++;
2336.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2337.     $out_array[5][22] += ($columns[$metabolite_counts][22] +
   $alkenesum);
2338.     $BB += 1;
2339.     #                 $metabolite_counts++;
2340.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2341.     $out_array[7][22] += ($columns[$metabolite_counts][22] +
   $alkynesum);
2342.     $BB += 1;
2343.     #                 $metabolite_counts++;
2344.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2345.     $out_array[12][22] += ($columns[$metabolite_counts][22] +
   $nitsum);
2346.     $BB += 1;
2347.     #                 $metabolite_counts++;
2348.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2349.     $out_array[8][22] += ($columns[$metabolite_counts][22] +
   $thiolsum);
2350.     $BB += 1;
2351.     #                 $metabolite_counts++;
2352.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2353.     $out_array[10][22] += ($columns[$metabolite_counts][22] +
   $indolesum);
2354.     $BB += 1;
2355.     #                 $metabolite_counts++;
2356.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2357.     $out_array[9][22] += ($columns[$metabolite_counts][22] +
   $sulfsum);
2358.     $BB += 1;
2359.     #                 $metabolite_counts++;
2360.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2361.     $out_array[13][22] += ($columns[$metabolite_counts][22] +
   $aminesum);
2362.     $BB += 1;
2363.     #                 $metabolite_counts++;
2364.     } elsif (( $BB == 0 )) {
2365.     $out_array[14][22] += ($columns[$metabolite_counts][22]+
   $othersum);
2366.     $BB += 1;
2367.     }
2368.     }
2369.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2370.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2371.     my $BB = 0;
2372.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2373.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2374.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2375.     $out_array[2][23] += ($columns[$metabolite_counts][23] +
   $alcsum);
2376.     $BB += 1;
2377.     #                 $metabolite_counts++;
2378.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2379.     $out_array[4][23] += ($columns[$metabolite_counts][23] +
   $aldsum);
2380.     $BB += 1;
2381.     #                 $metabolite_counts++;
2382.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2383.     $out_array[4][23] += ($columns[$metabolite_counts][23] +
   $aldsum);
2384.     $BB += 1;
2385.     #                 $metabolite_counts++;
2386.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2387.     $out_array[3][23] += ($columns[$metabolite_counts][23] +
   $acidsum);
2388.     $BB += 1;
2389.     #                 $metabolite_counts++;
2390.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2391.     $out_array[3][23] += ($columns[$metabolite_counts][23] +
   $acidsum);
2392.     $BB += 1;
2393.     #                 $metabolite_counts++;
2394.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2395.     $out_array[3][23] += ($columns[$metabolite_counts][23] +
   $acidsum);
2396.     $BB += 1;
2397.     #                 $metabolite_counts++;
2398.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2399.     $out_array[11][23] += ($columns[$metabolite_counts][23] +
   $keysum);
2400.     $BB += 1;
2401.     #                 $metabolite_counts++;
2402.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2403.     $out_array[6][23] += ($columns[$metabolite_counts][23] +
   $alkanesum);
2404.     $BB += 1;
2405.     #                 $metabolite_counts++;
2406.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2407.     $out_array[5][23] += ($columns[$metabolite_counts][23] +
   $alkenesum);
2408.     $BB += 1;
2409.     #                 $metabolite_counts++;
2410.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2411.     $out_array[7][23] += ($columns[$metabolite_counts][23] +
   $alkynesum);
2412.     $BB += 1;
2413.     #                 $metabolite_counts++;
2414.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2415.     $out_array[12][23] += ($columns[$metabolite_counts][23] +
   $nitsum);
2416.     $BB += 1;
2417.     #                 $metabolite_counts++;
2418.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2419.     $out_array[8][23] += ($columns[$metabolite_counts][23] +
   $thiolsum);
2420.     $BB += 1;
2421.     #                 $metabolite_counts++;
2422.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2423.     $out_array[10][23] += ($columns[$metabolite_counts][23] +
   $indolesum);
2424.     $BB += 1;
2425.     #                 $metabolite_counts++;
2426.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2427.     $out_array[9][23] += ($columns[$metabolite_counts][23] +
   $sulfsum);
2428.     $BB += 1;
2429.     #                 $metabolite_counts++;
2430.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2431.     $out_array[13][23] += ($columns[$metabolite_counts][23] +
   $aminesum);
2432.     $BB += 1;
2433.     #                 $metabolite_counts++;
2434.     } elsif (( $BB == 0 )) {
2435.     $out_array[14][23] += ($columns[$metabolite_counts][23]+
   $othersum);
2436.     $BB += 1;
2437.     }
2438.     }
2439.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2440.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2441.     my $BB = 0;
2442.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2443.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2444.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2445.     $out_array[2][24] += ($columns[$metabolite_counts][24] +
   $alcsum);
2446.     $BB += 1;
2447.     #                 $metabolite_counts++;
2448.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/ )) {
2449.     $out_array[4][24] += ($columns[$metabolite_counts][24] +
   $aldsum);
2450.     $BB += 1;
2451.     #                 $metabolite_counts++;
2452.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2453.     $out_array[4][24] += ($columns[$metabolite_counts][24] +
   $aldsum);
2454.     $BB += 1;
2455.     #                 $metabolite_counts++;
2456.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2457.     $out_array[3][24] += ($columns[$metabolite_counts][24] +
   $acidsum);
2458.     $BB += 1;
2459.     #                 $metabolite_counts++;
2460.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2461.     $out_array[3][24] += ($columns[$metabolite_counts][24] +
   $acidsum);
2462.     $BB += 1;
2463.     #                 $metabolite_counts++;
2464.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2465.     $out_array[3][24] += ($columns[$metabolite_counts][24] +
   $acidsum);
2466.     $BB += 1;
2467.     #                 $metabolite_counts++;
2468.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2469.     $out_array[11][24] += ($columns[$metabolite_counts][24] +
   $keysum);
2470.     $BB += 1;
2471.     #                 $metabolite_counts++;
2472.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2473.     $out_array[6][24] += ($columns[$metabolite_counts][24] +
   $alkanesum);
2474.     $BB += 1;
2475.     #                 $metabolite_counts++;
2476.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2477.     $out_array[5][24] += ($columns[$metabolite_counts][24] +
   $alkenesum);
2478.     $BB += 1;
2479.     #                 $metabolite_counts++;
2480.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2481.     $out_array[7][24] += ($columns[$metabolite_counts][24] +
   $alkynesum);
2482.     $BB += 1;
2483.     #                 $metabolite_counts++;
2484.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2485.     $out_array[12][24] += ($columns[$metabolite_counts][24] +
   $nitsum);
2486.     $BB += 1;
2487.     #                 $metabolite_counts++;
2488.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2489.     $out_array[8][24] += ($columns[$metabolite_counts][24] +
   $thiolsum);
2490.     $BB += 1;
2491.     #                 $metabolite_counts++;
2492.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2493.     $out_array[10][24] += ($columns[$metabolite_counts][24] +
   $indolesum);
2494.     $BB += 1;
2495.     #                 $metabolite_counts++;
2496.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2497.     $out_array[9][24] += ($columns[$metabolite_counts][24] +
   $sulfsum);
2498.     $BB += 1;
2499.     #                 $metabolite_counts++;
2500.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2501.     $out_array[13][24] += ($columns[$metabolite_counts][24] +
   $aminesum);
2502.     $BB += 1;
2503.     #                 $metabolite_counts++;
2504.     } elsif (( $BB == 0 )) {
2505.     $out_array[14][24] += ($columns[$metabolite_counts][24]+
   $othersum);
2506.     $BB += 1;
2507.     }
2508.     }
2509.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2510.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2511.     my $BB = 0;
2512.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2513.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2514.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2515.     $out_array[2][25] += ($columns[$metabolite_counts][25] +
   $alcsum);
2516.     $BB += 1;
2517.     #                 $metabolite_counts++;
2518.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2519.     $out_array[4][25] += ($columns[$metabolite_counts][25] +
   $aldsum);
2520.     $BB += 1;
2521.     #                 $metabolite_counts++;
2522.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2523.     $out_array[4][25] += ($columns[$metabolite_counts][25] +
   $aldsum);
2524.     $BB += 1;
2525.     #                 $metabolite_counts++;
2526.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2527.     $out_array[3][25] += ($columns[$metabolite_counts][25] +
   $acidsum);
2528.     $BB += 1;
2529.     #                 $metabolite_counts++;
2530.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2531.     $out_array[3][25] += ($columns[$metabolite_counts][25] +
   $acidsum);
2532.     $BB += 1;
2533.     #                 $metabolite_counts++;
2534.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2535.     $out_array[3][25] += ($columns[$metabolite_counts][25] +
   $acidsum);
2536.     $BB += 1;
2537.     #                 $metabolite_counts++;
2538.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2539.     $out_array[11][25] += ($columns[$metabolite_counts][25] +
   $keysum);
2540.     $BB += 1;
2541.     #                 $metabolite_counts++;
2542.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2543.     $out_array[6][25] += ($columns[$metabolite_counts][25] +
   $alkanesum);
2544.     $BB += 1;
2545.     #                 $metabolite_counts++;
2546.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2547.     $out_array[5][25] += ($columns[$metabolite_counts][25] +
   $alkenesum);
2548.     $BB += 1;
2549.     #                 $metabolite_counts++;
2550.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2551.     $out_array[7][25] += ($columns[$metabolite_counts][25] +
   $alkynesum);
2552.     $BB += 1;
2553.     #                 $metabolite_counts++;
2554.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2555.     $out_array[12][25] += ($columns[$metabolite_counts][25] +
   $nitsum);
2556.     $BB += 1;
2557.     #                 $metabolite_counts++;
2558.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2559.     $out_array[8][25] += ($columns[$metabolite_counts][25] +
   $thiolsum);
2560.     $BB += 1;
2561.     #                 $metabolite_counts++;
2562.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2563.     $out_array[10][25] += ($columns[$metabolite_counts][25] +
   $indolesum);
2564.     $BB += 1;
2565.     #                 $metabolite_counts++;
2566.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2567.     $out_array[9][25] += ($columns[$metabolite_counts][25] +
   $sulfsum);
2568.     $BB += 1;
2569.     #                 $metabolite_counts++;
2570.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2571.     $out_array[13][25] += ($columns[$metabolite_counts][25] +
   $aminesum);
2572.     $BB += 1;
2573.     #                 $metabolite_counts++;
2574.     } elsif (( $BB == 0 )) {
2575.     $out_array[14][25] += ($columns[$metabolite_counts][25]+
   $othersum);
2576.     $BB += 1;
2577.     }
2578.     }
2579.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2580.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2581.     my $BB = 0;
2582.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2583.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2584.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2585.     $out_array[2][26] += ($columns[$metabolite_counts][26] +
   $alcsum);
2586.     $BB += 1;
2587.     #                 $metabolite_counts++;
2588.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2589.     $out_array[4][26] += ($columns[$metabolite_counts][26] +
   $aldsum);
2590.     $BB += 1;
2591.     #                 $metabolite_counts++;
2592.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2593.     $out_array[4][26] += ($columns[$metabolite_counts][26] +
   $aldsum);
2594.     $BB += 1;
2595.     #                 $metabolite_counts++;
2596.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2597.     $out_array[3][26] += ($columns[$metabolite_counts][26] +
   $acidsum);
2598.     $BB += 1;
2599.     #                 $metabolite_counts++;
2600.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2601.     $out_array[3][26] += ($columns[$metabolite_counts][26] +
   $acidsum);
2602.     $BB += 1;
2603.     #                 $metabolite_counts++;
2604.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2605.     $out_array[3][26] += ($columns[$metabolite_counts][26] +
   $acidsum);
2606.     $BB += 1;
2607.     #                 $metabolite_counts++;
2608.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2609.     $out_array[11][26] += ($columns[$metabolite_counts][26] +
   $keysum);
2610.     $BB += 1;
2611.     #                 $metabolite_counts++;
2612.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2613.     $out_array[6][26] += ($columns[$metabolite_counts][26] +
   $alkanesum);
2614.     $BB += 1;
2615.     #                 $metabolite_counts++;
2616.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2617.     $out_array[5][26] += ($columns[$metabolite_counts][26] +
   $alkenesum);
2618.     $BB += 1;
2619.     #                 $metabolite_counts++;
2620.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2621.     $out_array[7][26] += ($columns[$metabolite_counts][26] +
   $alkynesum);
2622.     $BB += 1;
2623.     #                 $metabolite_counts++;
2624.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2625.     $out_array[12][26] += ($columns[$metabolite_counts][26] +
   $nitsum);
2626.     $BB += 1;
2627.     #                 $metabolite_counts++;
2628.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2629.     $out_array[8][26] += ($columns[$metabolite_counts][26] +
   $thiolsum);
2630.     $BB += 1;
2631.     #                 $metabolite_counts++;
2632.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2633.     $out_array[10][26] += ($columns[$metabolite_counts][26] +
   $indolesum);
2634.     $BB += 1;
2635.     #                 $metabolite_counts++;
2636.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2637.     $out_array[9][26] += ($columns[$metabolite_counts][26] +
   $sulfsum);
2638.     $BB += 1;
2639.     #                 $metabolite_counts++;
2640.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2641.     $out_array[13][26] += ($columns[$metabolite_counts][26] +
   $aminesum);
2642.     $BB += 1;
2643.     #                 $metabolite_counts++;
2644.     } elsif (( $BB == 0 )) {
2645.     $out_array[14][26] += ($columns[$metabolite_counts][26]+
   $othersum);
2646.     $BB += 1;
2647.     }
2648.     }
2649.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2650.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2651.     my $BB = 0;
2652.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2653.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2654.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2655.     $out_array[2][27] += ($columns[$metabolite_counts][27] +
   $alcsum);
2656.     $BB += 1;
2657.     #                 $metabolite_counts++;
2658.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2659.     $out_array[4][27] += ($columns[$metabolite_counts][27] +
   $aldsum);
2660.     $BB += 1;
2661.     #                 $metabolite_counts++;
2662.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2663.     $out_array[4][27] += ($columns[$metabolite_counts][27] +
   $aldsum);
2664.     $BB += 1;
2665.     #                 $metabolite_counts++;
2666.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2667.     $out_array[3][27] += ($columns[$metabolite_counts][27] +
   $acidsum);
2668.     $BB += 1;
2669.     #                 $metabolite_counts++;
2670.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2671.     $out_array[3][27] += ($columns[$metabolite_counts][27] +
   $acidsum);
2672.     $BB += 1;
2673.     #                 $metabolite_counts++;
2674.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2675.     $out_array[3][27] += ($columns[$metabolite_counts][27] +
   $acidsum);
2676.     $BB += 1;
2677.     #                 $metabolite_counts++;
2678.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2679.     $out_array[11][27] += ($columns[$metabolite_counts][27] +
   $keysum);
2680.     $BB += 1;
2681.     #                 $metabolite_counts++;
2682.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2683.     $out_array[6][27] += ($columns[$metabolite_counts][27] +
   $alkanesum);
2684.     $BB += 1;
2685.     #                 $metabolite_counts++;
2686.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2687.     $out_array[5][27] += ($columns[$metabolite_counts][27] +
   $alkenesum);
2688.     $BB += 1;
2689.     #                 $metabolite_counts++;
2690.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2691.     $out_array[7][27] += ($columns[$metabolite_counts][27] +
   $alkynesum);
2692.     $BB += 1;
2693.     #                 $metabolite_counts++;
2694.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2695.     $out_array[12][27] += ($columns[$metabolite_counts][27] +
   $nitsum);
2696.     $BB += 1;
2697.     #                 $metabolite_counts++;
2698.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2699.     $out_array[8][27] += ($columns[$metabolite_counts][27] +
   $thiolsum);
2700.     $BB += 1;
2701.     #                 $metabolite_counts++;
2702.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2703.     $out_array[10][27] += ($columns[$metabolite_counts][27] +
   $indolesum);
2704.     $BB += 1;
2705.     #                 $metabolite_counts++;
2706.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2707.     $out_array[9][27] += ($columns[$metabolite_counts][27] +
   $sulfsum);
2708.     $BB += 1;
2709.     #                 $metabolite_counts++;
2710.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2711.     $out_array[13][27] += ($columns[$metabolite_counts][27] +
   $aminesum);
2712.     $BB += 1;
2713.     #                 $metabolite_counts++;
2714.     } elsif (( $BB == 0 )) {
2715.     $out_array[14][27] += ($columns[$metabolite_counts][27]+
   $othersum);
2716.     $BB += 1;
2717.     }
2718.     }
2719.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2720.     next  if ($columns[$metabolite_counts][0] =~
   /(^Sample|^Label)/);    # skip over lines starts with "Label"
   "Sample"
2721.     my $BB = 0;
2722.     #           next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);
   # skip over lines starts with "Label" "Sample" "alcoholic" or
   "healthy"
2723.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2724.     #                 next line if
   ($columns[$metabolite_counts][0] =~
   /(^Label|^Sample|^alcoholic)/);
2725.     $out_array[2][28] += ($columns[$metabolite_counts][28] +
   $alcsum);
2726.     $BB += 1;
2727.     #                 $metabolite_counts++;
2728.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/al/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2729.     $out_array[4][28] += ($columns[$metabolite_counts][28] +
   $aldsum);
2730.     $BB += 1;
2731.     #                 $metabolite_counts++;
2732.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2733.     $out_array[4][28] += ($columns[$metabolite_counts][28] +
   $aldsum);
2734.     $BB += 1;
2735.     #                 $metabolite_counts++;
2736.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2737.     $out_array[3][28] += ($columns[$metabolite_counts][28] +
   $acidsum);
2738.     $BB += 1;
2739.     #                 $metabolite_counts++;
2740.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2741.     $out_array[3][28] += ($columns[$metabolite_counts][28] +
   $acidsum);
2742.     $BB += 1;
2743.     #                 $metabolite_counts++;
2744.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2745.     $out_array[3][28] += ($columns[$metabolite_counts][28] +
   $acidsum);
2746.     $BB += 1;
2747.     #                 $metabolite_counts++;
2748.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2749.     $out_array[11][28] += ($columns[$metabolite_counts][28] +
   $keysum);
2750.     $BB += 1;
2751.     #                 $metabolite_counts++;
2752.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2753.     $out_array[6][28] += ($columns[$metabolite_counts][28] +
   $alkanesum);
2754.     $BB += 1;
2755.     #                 $metabolite_counts++;
2756.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2757.     $out_array[5][28] += ($columns[$metabolite_counts][28] +
   $alkenesum);
2758.     $BB += 1;
2759.     #                 $metabolite_counts++;
2760.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2761.     $out_array[7][28] += ($columns[$metabolite_counts][28] +
   $alkynesum);
2762.     $BB += 1;
2763.     #                 $metabolite_counts++;
2764.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2765.     $out_array[12][28] += ($columns[$metabolite_counts][28] +
   $nitsum);
2766.     $BB += 1;
2767.     #                 $metabolite_counts++;
2768.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2769.     $out_array[8][28] += ($columns[$metabolite_counts][28] +
   $thiolsum);
2770.     $BB += 1;
2771.     #                 $metabolite_counts++;
2772.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2773.     $out_array[10][28] += ($columns[$metabolite_counts][28] +
   $indolesum);
2774.     $BB += 1;
2775.     #                 $metabolite_counts++;
2776.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2777.     $out_array[9][28] += ($columns[$metabolite_counts][28] +
   $sulfsum);
2778.     $BB += 1;
2779.     #                 $metabolite_counts++;
2780.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2781.     $out_array[13][28] += ($columns[$metabolite_counts][28] +
   $aminesum);
2782.     $BB += 1;
2783.     #                 $metabolite_counts++;
2784.     } elsif (( $BB == 0 )) {
2785.     $out_array[14][28] += ($columns[$metabolite_counts][28]+
   $othersum);
2786.     $BB += 1;
2787.     }
2788.     }
2789.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
```

```
2790.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
2791.     my $BB = 0;
2792.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
2793.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2794.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
2795.     $out_array[2][29] += ($columns[$metabolite_counts][29] +
   $alcsum);
2796.     $BB += 1;
2797.     #                   $metabolite_counts++;
2798.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2799.     $out_array[4][29] += ($columns[$metabolite_counts][29] +
   $aldsum);
2800.     $BB += 1;
2801.     #                   $metabolite_counts++;
2802.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2803.     $out_array[4][29] += ($columns[$metabolite_counts][29] +
   $aldsum);
2804.     $BB += 1;
2805.     #                   $metabolite_counts++;
2806.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2807.     $out_array[3][29] += ($columns[$metabolite_counts][29] +
   $acidsum);
2808.     $BB += 1;
2809.     #                   $metabolite_counts++;
2810.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2811.     $out_array[3][29] += ($columns[$metabolite_counts][29] +
   $acidsum);
2812.     $BB += 1;
2813.     #                   $metabolite_counts++;
2814.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2815.     $out_array[3][29] += ($columns[$metabolite_counts][29] +
   $acidsum);
2816.     $BB += 1;
2817.     #                   $metabolite_counts++;
2818.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2819.     $out_array[11][29] += ($columns[$metabolite_counts][29] +
   $keysum);
2820.     $BB += 1;
2821.     #                   $metabolite_counts++;
2822.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
```

```
2823.     $out_array[6][29] += ($columns[$metabolite_counts][29] +
   $alkanesum);
2824.     $BB += 1;
2825.     #                   $metabolite_counts++;
2826.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2827.     $out_array[5][29] += ($columns[$metabolite_counts][29] +
   $alkenesum);
2828.     $BB += 1;
2829.     #                   $metabolite_counts++;
2830.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2831.     $out_array[7][29] += ($columns[$metabolite_counts][29] +
   $alkynesum);
2832.     $BB += 1;
2833.     #                   $metabolite_counts++;
2834.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2835.     $out_array[12][29] += ($columns[$metabolite_counts][29] +
   $nitsum);
2836.     $BB += 1;
2837.     #                   $metabolite_counts++;
2838.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2839.     $out_array[8][29] += ($columns[$metabolite_counts][29] +
   $thiolsum);
2840.     $BB += 1;
2841.     #                   $metabolite_counts++;
2842.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2843.     $out_array[10][29] += ($columns[$metabolite_counts][29] +
   $indolesum);
2844.     $BB += 1;
2845.     #                   $metabolite_counts++;
2846.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2847.     $out_array[9][29] += ($columns[$metabolite_counts][29] +
   $sulfsum);
2848.     $BB += 1;
2849.     #                   $metabolite_counts++;
2850.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2851.     $out_array[13][29] += ($columns[$metabolite_counts][29] +
   $aminesum);
2852.     $BB += 1;
2853.     #                   $metabolite_counts++;
2854.     } elsif (( $BB == 0 )) {
2855.     $out_array[14][29] += ($columns[$metabolite_counts][29]+
   $othersum);
2856.     $BB += 1;
2857.     }
2858.     }
```

```
2859.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
2860.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
2861.     my $BB = 0;
2862.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
2863.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2864.     #                  next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
2865.     $out_array[2][30] += ($columns[$metabolite_counts][30] +
   $alcsum);
2866.     $BB += 1;
2867.     #                  $metabolite_counts++;
2868.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2869.     $out_array[4][30] += ($columns[$metabolite_counts][30] +
   $aldsum);
2870.     $BB += 1;
2871.     #                  $metabolite_counts++;
2872.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2873.     $out_array[4][30] += ($columns[$metabolite_counts][30] +
   $aldsum);
2874.     $BB += 1;
2875.     #                  $metabolite_counts++;
2876.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2877.     $out_array[3][30] += ($columns[$metabolite_counts][30] +
   $acidsum);
2878.     $BB += 1;
2879.     #                  $metabolite_counts++;
2880.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2881.     $out_array[3][30] += ($columns[$metabolite_counts][30] +
   $acidsum);
2882.     $BB += 1;
2883.     #                  $metabolite_counts++;
2884.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2885.     $out_array[3][30] += ($columns[$metabolite_counts][30] +
   $acidsum);
2886.     $BB += 1;
2887.     #                  $metabolite_counts++;
2888.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2889.     $out_array[11][30] += ($columns[$metabolite_counts][30] +
   $keysum);
2890.     $BB += 1;
2891.     #                  $metabolite_counts++;
```

```
2892.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
2893.     $out_array[6][30] += ($columns[$metabolite_counts][30] +
   $alkanesum);
2894.     $BB += 1;
2895.     #                   $metabolite_counts++;
2896.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2897.     $out_array[5][30] += ($columns[$metabolite_counts][30] +
   $alkenesum);
2898.     $BB += 1;
2899.     #                   $metabolite_counts++;
2900.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2901.     $out_array[7][30] += ($columns[$metabolite_counts][30] +
   $alkynesum);
2902.     $BB += 1;
2903.     #                   $metabolite_counts++;
2904.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2905.     $out_array[12][30] += ($columns[$metabolite_counts][30] +
   $nitsum);
2906.     $BB += 1;
2907.     #                   $metabolite_counts++;
2908.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2909.     $out_array[8][30] += ($columns[$metabolite_counts][30] +
   $thiolsum);
2910.     $BB += 1;
2911.     #                   $metabolite_counts++;
2912.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2913.     $out_array[10][30] += ($columns[$metabolite_counts][30] +
   $indolesum);
2914.     $BB += 1;
2915.     #                   $metabolite_counts++;
2916.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2917.     $out_array[9][30] += ($columns[$metabolite_counts][30] +
   $sulfsum);
2918.     $BB += 1;
2919.     #                   $metabolite_counts++;
2920.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2921.     $out_array[13][30] += ($columns[$metabolite_counts][30] +
   $aminesum);
2922.     $BB += 1;
2923.     #                   $metabolite_counts++;
2924.     } elsif (( $BB == 0 )) {
2925.     $out_array[14][30] += ($columns[$metabolite_counts][30]+
   $othersum);
2926.     $BB += 1;
```

```
2927.     }
2928.     }
2929.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
2930.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
2931.     my $BB = 0;
2932.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
2933.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
2934.     #                  next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
2935.     $out_array[2][31] += ($columns[$metabolite_counts][31] +
   $alcsum);
2936.     $BB += 1;
2937.     #                  $metabolite_counts++;
2938.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2939.     $out_array[4][31] += ($columns[$metabolite_counts][31] +
   $aldsum);
2940.     $BB += 1;
2941.     #                  $metabolite_counts++;
2942.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
2943.     $out_array[4][31] += ($columns[$metabolite_counts][31] +
   $aldsum);
2944.     $BB += 1;
2945.     #                  $metabolite_counts++;
2946.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
2947.     $out_array[3][31] += ($columns[$metabolite_counts][31] +
   $acidsum);
2948.     $BB += 1;
2949.     #                  $metabolite_counts++;
2950.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
2951.     $out_array[3][31] += ($columns[$metabolite_counts][31] +
   $acidsum);
2952.     $BB += 1;
2953.     #                  $metabolite_counts++;
2954.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
2955.     $out_array[3][31] += ($columns[$metabolite_counts][31] +
   $acidsum);
2956.     $BB += 1;
2957.     #                  $metabolite_counts++;
2958.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
2959.     $out_array[11][31] += ($columns[$metabolite_counts][31] +
   $keysum);
```

```
2960.     $BB += 1;
2961.     #                   $metabolite_counts++;
2962.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
2963.     $out_array[6][31] += ($columns[$metabolite_counts][31] +
   $alkanesum);
2964.     $BB += 1;
2965.     #                   $metabolite_counts++;
2966.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
2967.     $out_array[5][31] += ($columns[$metabolite_counts][31] +
   $alkenesum);
2968.     $BB += 1;
2969.     #                   $metabolite_counts++;
2970.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
2971.     $out_array[7][31] += ($columns[$metabolite_counts][31] +
   $alkynesum);
2972.     $BB += 1;
2973.     #                   $metabolite_counts++;
2974.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
2975.     $out_array[12][31] += ($columns[$metabolite_counts][31] +
   $nitsum);
2976.     $BB += 1;
2977.     #                   $metabolite_counts++;
2978.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
2979.     $out_array[8][31] += ($columns[$metabolite_counts][31] +
   $thiolsum);
2980.     $BB += 1;
2981.     #                   $metabolite_counts++;
2982.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
2983.     $out_array[10][31] += ($columns[$metabolite_counts][31] +
   $indolesum);
2984.     $BB += 1;
2985.     #                   $metabolite_counts++;
2986.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
2987.     $out_array[9][31] += ($columns[$metabolite_counts][31] +
   $sulfsum);
2988.     $BB += 1;
2989.     #                   $metabolite_counts++;
2990.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
2991.     $out_array[13][31] += ($columns[$metabolite_counts][31] +
   $aminesum);
2992.     $BB += 1;
2993.     #                   $metabolite_counts++;
2994.     } elsif (( $BB == 0 )) {
```

```
2995.     $out_array[14][31] += ($columns[$metabolite_counts][31] +
   $othersum);
2996.     $BB += 1;
2997.     }
2998.     }
2999.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3000.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
3001.     my $BB = 0;
3002.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3003.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3004.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3005.     $out_array[2][32] += ($columns[$metabolite_counts][32] +
   $alcsum);
3006.     $BB += 1;
3007.     #                   $metabolite_counts++;
3008.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/ )) {
3009.     $out_array[4][32] += ($columns[$metabolite_counts][32] +
   $aldsum);
3010.     $BB += 1;
3011.     #                   $metabolite_counts++;
3012.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3013.     $out_array[4][32] += ($columns[$metabolite_counts][32] +
   $aldsum);
3014.     $BB += 1;
3015.     #                   $metabolite_counts++;
3016.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3017.     $out_array[3][32] += ($columns[$metabolite_counts][32] +
   $acidsum);
3018.     $BB += 1;
3019.     #                   $metabolite_counts++;
3020.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3021.     $out_array[3][32] += ($columns[$metabolite_counts][32] +
   $acidsum);
3022.     $BB += 1;
3023.     #                   $metabolite_counts++;
3024.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3025.     $out_array[3][32] += ($columns[$metabolite_counts][32] +
   $acidsum);
3026.     $BB += 1;
3027.     #                   $metabolite_counts++;
```

```
3028.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3029.     $out_array[11][32] += ($columns[$metabolite_counts][32] +
   $keysum);
3030.     $BB += 1;
3031.     #                   $metabolite_counts++;
3032.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3033.     $out_array[6][32] += ($columns[$metabolite_counts][32] +
   $alkanesum);
3034.     $BB += 1;
3035.     #                   $metabolite_counts++;
3036.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/ )) {
3037.     $out_array[5][32] += ($columns[$metabolite_counts][32] +
   $alkenesum);
3038.     $BB += 1;
3039.     #                   $metabolite_counts++;
3040.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3041.     $out_array[7][32] += ($columns[$metabolite_counts][32] +
   $alkynesum);
3042.     $BB += 1;
3043.     #                   $metabolite_counts++;
3044.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3045.     $out_array[12][32] += ($columns[$metabolite_counts][32] +
   $nitsum);
3046.     $BB += 1;
3047.     #                   $metabolite_counts++;
3048.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3049.     $out_array[8][32] += ($columns[$metabolite_counts][32] +
   $thiolsum);
3050.     $BB += 1;
3051.     #                   $metabolite_counts++;
3052.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3053.     $out_array[10][32] += ($columns[$metabolite_counts][32] +
   $indolesum);
3054.     $BB += 1;
3055.     #                   $metabolite_counts++;
3056.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3057.     $out_array[9][32] += ($columns[$metabolite_counts][32] +
   $sulfsum);
3058.     $BB += 1;
3059.     #                   $metabolite_counts++;
3060.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3061.     $out_array[13][32] += ($columns[$metabolite_counts][32] +
   $aminesum);
```

```
3062.     $BB += 1;
3063.     #                   $metabolite_counts++;
3064.     } elsif (( $BB == 0 )) {
3065.     $out_array[14][32] += ($columns[$metabolite_counts][32] +
   $othersum);
3066.     $BB += 1;
3067.     }
3068.     }
3069.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3070.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
3071.     my $BB = 0;
3072.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3073.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3074.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3075.     $out_array[2][33] += ($columns[$metabolite_counts][33] +
   $alcsum);
3076.     $BB += 1;
3077.     #                   $metabolite_counts++;
3078.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3079.     $out_array[4][33] += ($columns[$metabolite_counts][33] +
   $aldsum);
3080.     $BB += 1;
3081.     #                   $metabolite_counts++;
3082.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3083.     $out_array[4][33] += ($columns[$metabolite_counts][33] +
   $aldsum);
3084.     $BB += 1;
3085.     #                   $metabolite_counts++;
3086.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3087.     $out_array[3][33] += ($columns[$metabolite_counts][33] +
   $acidsum);
3088.     $BB += 1;
3089.     #                   $metabolite_counts++;
3090.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3091.     $out_array[3][33] += ($columns[$metabolite_counts][33] +
   $acidsum);
3092.     $BB += 1;
3093.     #                   $metabolite_counts++;
3094.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3095.     $out_array[3][33] += ($columns[$metabolite_counts][33] +
   $acidsum);
```

```
3096.     $BB += 1;
3097.     #                   $metabolite_counts++;
3098.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3099.     $out_array[11][33] += ($columns[$metabolite_counts][33] +
   $keysum);
3100.     $BB += 1;
3101.     #                   $metabolite_counts++;
3102.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3103.     $out_array[6][33] += ($columns[$metabolite_counts][33] +
   $alkanesum);
3104.     $BB += 1;
3105.     #                   $metabolite_counts++;
3106.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3107.     $out_array[5][33] += ($columns[$metabolite_counts][33] +
   $alkenesum);
3108.     $BB += 1;
3109.     #                   $metabolite_counts++;
3110.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3111.     $out_array[7][33] += ($columns[$metabolite_counts][33] +
   $alkynesum);
3112.     $BB += 1;
3113.     #                   $metabolite_counts++;
3114.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3115.     $out_array[12][33] += ($columns[$metabolite_counts][33] +
   $nitsum);
3116.     $BB += 1;
3117.     #                   $metabolite_counts++;
3118.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3119.     $out_array[8][33] += ($columns[$metabolite_counts][33] +
   $thiolsum);
3120.     $BB += 1;
3121.     #                   $metabolite_counts++;
3122.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3123.     $out_array[10][33] += ($columns[$metabolite_counts][33] +
   $indolesum);
3124.     $BB += 1;
3125.     #                   $metabolite_counts++;
3126.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3127.     $out_array[9][33] += ($columns[$metabolite_counts][33] +
   $sulfsum);
3128.     $BB += 1;
3129.     #                   $metabolite_counts++;
3130.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
```

```
3131.     $out_array[13][33] += ($columns[$metabolite_counts][33] +
   $aminesum);
3132.     $BB += 1;
3133.     #                   $metabolite_counts++;
3134.     } elsif (( $BB == 0 )) {
3135.     $out_array[14][33] += ($columns[$metabolite_counts][33] +
   $othersum);
3136.     $BB += 1;
3137.     }
3138.     }
3139.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3140.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
3141.     my $BB = 0;
3142.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3143.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/  and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3144.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3145.     $out_array[2][34] += ($columns[$metabolite_counts][34] +
   $alcsum);
3146.     $BB += 1;
3147.     #                   $metabolite_counts++;
3148.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3149.     $out_array[4][34] += ($columns[$metabolite_counts][34] +
   $aldsum);
3150.     $BB += 1;
3151.     #                   $metabolite_counts++;
3152.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3153.     $out_array[4][34] += ($columns[$metabolite_counts][34] +
   $aldsum);
3154.     $BB += 1;
3155.     #                   $metabolite_counts++;
3156.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3157.     $out_array[3][34] += ($columns[$metabolite_counts][34] +
   $acidsum);
3158.     $BB += 1;
3159.     #                   $metabolite_counts++;
3160.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3161.     $out_array[3][34] += ($columns[$metabolite_counts][34] +
   $acidsum);
3162.     $BB += 1;
3163.     #                   $metabolite_counts++;
3164.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
```

```
3165.     $out_array[3][34] += ($columns[$metabolite_counts][34] +
   $acidsum);
3166.     $BB += 1;
3167.     #                   $metabolite_counts++;
3168.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3169.     $out_array[11][34] += ($columns[$metabolite_counts][34] +
   $keysum);
3170.     $BB += 1;
3171.     #                   $metabolite_counts++;
3172.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3173.     $out_array[6][34] += ($columns[$metabolite_counts][34] +
   $alkanesum);
3174.     $BB += 1;
3175.     #                   $metabolite_counts++;
3176.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3177.     $out_array[5][34] += ($columns[$metabolite_counts][34] +
   $alkenesum);
3178.     $BB += 1;
3179.     #                   $metabolite_counts++;
3180.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3181.     $out_array[7][34] += ($columns[$metabolite_counts][34] +
   $alkynesum);
3182.     $BB += 1;
3183.     #                   $metabolite_counts++;
3184.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3185.     $out_array[12][34] += ($columns[$metabolite_counts][34] +
   $nitsum);
3186.     $BB += 1;
3187.     #                   $metabolite_counts++;
3188.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3189.     $out_array[8][34] += ($columns[$metabolite_counts][34] +
   $thiolsum);
3190.     $BB += 1;
3191.     #                   $metabolite_counts++;
3192.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3193.     $out_array[10][34] += ($columns[$metabolite_counts][34] +
   $indolesum);
3194.     $BB += 1;
3195.     #                   $metabolite_counts++;
3196.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3197.     $out_array[9][34] += ($columns[$metabolite_counts][34] +
   $sulfsum);
3198.     $BB += 1;
3199.     #                   $metabolite_counts++;
```

```
3200.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3201.     $out_array[13][34] += ($columns[$metabolite_counts][34] +
   $aminesum);
3202.     $BB += 1;
3203.     #                   $metabolite_counts++;
3204.     } elsif (( $BB == 0 )) {
3205.     $out_array[14][34] += ($columns[$metabolite_counts][34] +
   $othersum);
3206.     $BB += 1;
3207.     }
3208.     }
3209.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3210.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
3211.     my $BB = 0;
3212.     #             next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3213.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3214.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3215.     $out_array[2][35] += ($columns[$metabolite_counts][35] +
   $alcsum);
3216.     $BB += 1;
3217.     #                   $metabolite_counts++;
3218.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3219.     $out_array[4][35] += ($columns[$metabolite_counts][35] +
   $aldsum);
3220.     $BB += 1;
3221.     #                   $metabolite_counts++;
3222.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3223.     $out_array[4][35] += ($columns[$metabolite_counts][35] +
   $aldsum);
3224.     $BB += 1;
3225.     #                   $metabolite_counts++;
3226.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3227.     $out_array[3][35] += ($columns[$metabolite_counts][35] +
   $acidsum);
3228.     $BB += 1;
3229.     #                   $metabolite_counts++;
3230.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3231.     $out_array[3][35] += ($columns[$metabolite_counts][35] +
   $acidsum);
3232.     $BB += 1;
3233.     #                   $metabolite_counts++;
```

```
3234.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3235.     $out_array[3][35] += ($columns[$metabolite_counts][35] +
   $acidsum);
3236.     $BB += 1;
3237.     #                   $metabolite_counts++;
3238.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3239.     $out_array[11][35] += ($columns[$metabolite_counts][35] +
   $keysum);
3240.     $BB += 1;
3241.     #                   $metabolite_counts++;
3242.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3243.     $out_array[6][35] += ($columns[$metabolite_counts][35] +
   $alkanesum);
3244.     $BB += 1;
3245.     #                   $metabolite_counts++;
3246.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3247.     $out_array[5][35] += ($columns[$metabolite_counts][35] +
   $alkenesum);
3248.     $BB += 1;
3249.     #                   $metabolite_counts++;
3250.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3251.     $out_array[7][35] += ($columns[$metabolite_counts][35] +
   $alkynesum);
3252.     $BB += 1;
3253.     #                   $metabolite_counts++;
3254.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3255.     $out_array[12][35] += ($columns[$metabolite_counts][35] +
   $nitsum);
3256.     $BB += 1;
3257.     #                   $metabolite_counts++;
3258.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3259.     $out_array[8][35] += ($columns[$metabolite_counts][35] +
   $thiolsum);
3260.     $BB += 1;
3261.     #                   $metabolite_counts++;
3262.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3263.     $out_array[10][35] += ($columns[$metabolite_counts][35] +
   $indolesum);
3264.     $BB += 1;
3265.     #                   $metabolite_counts++;
3266.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3267.     $out_array[9][35] += ($columns[$metabolite_counts][35] +
   $sulfsum);
```

```
3268.     $BB += 1;
3269.     #                   $metabolite_counts++;
3270.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3271.     $out_array[13][35] += ($columns[$metabolite_counts][35] +
   $aminesum);
3272.     $BB += 1;
3273.     #                   $metabolite_counts++;
3274.     } elsif (( $BB == 0 )) {
3275.     $out_array[14][35] += ($columns[$metabolite_counts][35] +
   $othersum);
3276.     $BB += 1;
3277.     }
3278.     }
3279.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3280.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
         # skip over lines starts with "Label" "Sample"
3281.     my $BB = 0;
3282.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3283.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3284.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3285.     $out_array[2][36] += ($columns[$metabolite_counts][36] +
   $alcsum);
3286.     $BB += 1;
3287.     #                   $metabolite_counts++;
3288.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3289.     $out_array[4][36] += ($columns[$metabolite_counts][36] +
   $aldsum);
3290.     $BB += 1;
3291.     #                   $metabolite_counts++;
3292.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3293.     $out_array[4][36] += ($columns[$metabolite_counts][36] +
   $aldsum);
3294.     $BB += 1;
3295.     #                   $metabolite_counts++;
3296.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3297.     $out_array[3][36] += ($columns[$metabolite_counts][36] +
   $acidsum);
3298.     $BB += 1;
3299.     #                   $metabolite_counts++;
3300.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3301.     $out_array[3][36] += ($columns[$metabolite_counts][36] +
   $acidsum);
```

```
3302.     $BB += 1;
3303.     #                   $metabolite_counts++;
3304.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3305.     $out_array[3][36] += ($columns[$metabolite_counts][36] +
   $acidsum);
3306.     $BB += 1;
3307.     #                   $metabolite_counts++;
3308.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3309.     $out_array[11][36] += ($columns[$metabolite_counts][36] +
   $keysum);
3310.     $BB += 1;
3311.     #                   $metabolite_counts++;
3312.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3313.     $out_array[6][36] += ($columns[$metabolite_counts][36] +
   $alkanesum);
3314.     $BB += 1;
3315.     #                   $metabolite_counts++;
3316.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3317.     $out_array[5][36] += ($columns[$metabolite_counts][36] +
   $alkenesum);
3318.     $BB += 1;
3319.     #                   $metabolite_counts++;
3320.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3321.     $out_array[7][36] += ($columns[$metabolite_counts][36] +
   $alkynesum);
3322.     $BB += 1;
3323.     #                   $metabolite_counts++;
3324.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3325.     $out_array[12][36] += ($columns[$metabolite_counts][36] +
   $nitsum);
3326.     $BB += 1;
3327.     #                   $metabolite_counts++;
3328.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3329.     $out_array[8][36] += ($columns[$metabolite_counts][36] +
   $thiolsum);
3330.     $BB += 1;
3331.     #                   $metabolite_counts++;
3332.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3333.     $out_array[10][36] += ($columns[$metabolite_counts][36] +
   $indolesum);
3334.     $BB += 1;
3335.     #                   $metabolite_counts++;
3336.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
```

```
3337.     $out_array[9][36] += ($columns[$metabolite_counts][36] +
   $sulfsum);
3338.     $BB += 1;
3339.     #                   $metabolite_counts++;
3340.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3341.     $out_array[13][36] += ($columns[$metabolite_counts][36] +
   $aminesum);
3342.     $BB += 1;
3343.     #                   $metabolite_counts++;
3344.     } elsif (( $BB == 0 )) {
3345.     $out_array[14][36] += ($columns[$metabolite_counts][36] +
   $othersum);
3346.     $BB += 1;
3347.     }
3348.     }
3349.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3350.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
3351.     my $BB = 0;
3352.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3353.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3354.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3355.     $out_array[2][37] += ($columns[$metabolite_counts][37] +
   $alcsum);
3356.     $BB += 1;
3357.     #                   $metabolite_counts++;
3358.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3359.     $out_array[4][37] += ($columns[$metabolite_counts][37] +
   $aldsum);
3360.     $BB += 1;
3361.     #                   $metabolite_counts++;
3362.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3363.     $out_array[4][37] += ($columns[$metabolite_counts][37] +
   $aldsum);
3364.     $BB += 1;
3365.     #                   $metabolite_counts++;
3366.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3367.     $out_array[3][37] += ($columns[$metabolite_counts][37] +
   $acidsum);
3368.     $BB += 1;
3369.     #                   $metabolite_counts++;
3370.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
```

```
3371.     $out_array[3][37] += ($columns[$metabolite_counts][37] +
   $acidsum);
3372.     $BB += 1;
3373.     #                   $metabolite_counts++;
3374.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3375.     $out_array[3][37] += ($columns[$metabolite_counts][37] +
   $acidsum);
3376.     $BB += 1;
3377.     #                   $metabolite_counts++;
3378.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3379.     $out_array[11][37] += ($columns[$metabolite_counts][37] +
   $keysum);
3380.     $BB += 1;
3381.     #                   $metabolite_counts++;
3382.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3383.     $out_array[6][37] += ($columns[$metabolite_counts][37] +
   $alkanesum);
3384.     $BB += 1;
3385.     #                   $metabolite_counts++;
3386.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3387.     $out_array[5][37] += ($columns[$metabolite_counts][37] +
   $alkenesum);
3388.     $BB += 1;
3389.     #                   $metabolite_counts++;
3390.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3391.     $out_array[7][37] += ($columns[$metabolite_counts][37] +
   $alkynesum);
3392.     $BB += 1;
3393.     #                   $metabolite_counts++;
3394.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3395.     $out_array[12][37] += ($columns[$metabolite_counts][37] +
   $nitsum);
3396.     $BB += 1;
3397.     #                   $metabolite_counts++;
3398.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3399.     $out_array[8][37] += ($columns[$metabolite_counts][37] +
   $thiolsum);
3400.     $BB += 1;
3401.     #                   $metabolite_counts++;
3402.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3403.     $out_array[10][37] += ($columns[$metabolite_counts][37] +
   $indolesum);
3404.     $BB += 1;
3405.     #                   $metabolite_counts++;
```

```
3406.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3407.     $out_array[9][37] += ($columns[$metabolite_counts][37] +
   $sulfsum);
3408.     $BB += 1;
3409.     #                   $metabolite_counts++;
3410.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3411.     $out_array[13][37] += ($columns[$metabolite_counts][37] +
   $aminesum);
3412.     $BB += 1;
3413.     #                   $metabolite_counts++;
3414.     } elsif (( $BB == 0 )) {
3415.     $out_array[14][37] += ($columns[$metabolite_counts][37] +
   $othersum);
3416.     $BB += 1;
3417.     }
3418.     }
3419.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3420.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
         # skip over lines starts with "Label" "Sample"
3421.     my $BB = 0;
3422.     #             next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3423.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3424.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3425.     $out_array[2][38] += ($columns[$metabolite_counts][38] +
   $alcsum);
3426.     $BB += 1;
3427.     #                   $metabolite_counts++;
3428.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3429.     $out_array[4][38] += ($columns[$metabolite_counts][38] +
   $aldsum);
3430.     $BB += 1;
3431.     #                   $metabolite_counts++;
3432.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3433.     $out_array[4][38] += ($columns[$metabolite_counts][38] +
   $aldsum);
3434.     $BB += 1;
3435.     #                   $metabolite_counts++;
3436.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3437.     $out_array[3][38] += ($columns[$metabolite_counts][38] +
   $acidsum);
3438.     $BB += 1;
3439.     #                   $metabolite_counts++;
```

```
3440.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3441.     $out_array[3][38] += ($columns[$metabolite_counts][38] +
   $acidsum);
3442.     $BB += 1;
3443.     #                   $metabolite_counts++;
3444.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3445.     $out_array[3][38] += ($columns[$metabolite_counts][38] +
   $acidsum);
3446.     $BB += 1;
3447.     #                   $metabolite_counts++;
3448.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3449.     $out_array[11][38] += ($columns[$metabolite_counts][38] +
   $keysum);
3450.     $BB += 1;
3451.     #                   $metabolite_counts++;
3452.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3453.     $out_array[6][38] += ($columns[$metabolite_counts][38] +
   $alkanesum);
3454.     $BB += 1;
3455.     #                   $metabolite_counts++;
3456.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3457.     $out_array[5][38] += ($columns[$metabolite_counts][38] +
   $alkenesum);
3458.     $BB += 1;
3459.     #                   $metabolite_counts++;
3460.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3461.     $out_array[7][38] += ($columns[$metabolite_counts][38] +
   $alkynesum);
3462.     $BB += 1;
3463.     #                   $metabolite_counts++;
3464.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3465.     $out_array[12][38] += ($columns[$metabolite_counts][38] +
   $nitsum);
3466.     $BB += 1;
3467.     #                   $metabolite_counts++;
3468.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3469.     $out_array[8][38] += ($columns[$metabolite_counts][38] +
   $thiolsum);
3470.     $BB += 1;
3471.     #                   $metabolite_counts++;
3472.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3473.     $out_array[10][38] += ($columns[$metabolite_counts][38] +
   $indolesum);
```

```
3474.     $BB += 1;
3475.     #                   $metabolite_counts++;
3476.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3477.     $out_array[9][38] += ($columns[$metabolite_counts][38] +
   $sulfsum);
3478.     $BB += 1;
3479.     #                   $metabolite_counts++;
3480.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3481.     $out_array[13][38] += ($columns[$metabolite_counts][38] +
   $aminesum);
3482.     $BB += 1;
3483.     #                   $metabolite_counts++;
3484.     } elsif (( $BB == 0 )) {
3485.     $out_array[14][38] += ($columns[$metabolite_counts][38] +
   $othersum);
3486.     $BB += 1;
3487.     }
3488.     }
3489.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3490.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
3491.     my $BB = 0;
3492.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3493.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3494.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3495.     $out_array[2][39] += ($columns[$metabolite_counts][39] +
   $alcsum);
3496.     $BB += 1;
3497.     #                   $metabolite_counts++;
3498.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3499.     $out_array[4][39] += ($columns[$metabolite_counts][39] +
   $aldsum);
3500.     $BB += 1;
3501.     #                   $metabolite_counts++;
3502.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3503.     $out_array[4][39] += ($columns[$metabolite_counts][39] +
   $aldsum);
3504.     $BB += 1;
3505.     #                   $metabolite_counts++;
3506.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3507.     $out_array[3][39] += ($columns[$metabolite_counts][39] +
   $acidsum);
```

```
3508.     $BB += 1;
3509.     #                   $metabolite_counts++;
3510.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3511.     $out_array[3][39] += ($columns[$metabolite_counts][39] +
   $acidsum);
3512.     $BB += 1;
3513.     #                   $metabolite_counts++;
3514.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3515.     $out_array[3][39] += ($columns[$metabolite_counts][39] +
   $acidsum);
3516.     $BB += 1;
3517.     #                   $metabolite_counts++;
3518.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3519.     $out_array[11][39] += ($columns[$metabolite_counts][39] +
   $keysum);
3520.     $BB += 1;
3521.     #                   $metabolite_counts++;
3522.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3523.     $out_array[6][39] += ($columns[$metabolite_counts][39] +
   $alkanesum);
3524.     $BB += 1;
3525.     #                   $metabolite_counts++;
3526.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3527.     $out_array[5][39] += ($columns[$metabolite_counts][39] +
   $alkenesum);
3528.     $BB += 1;
3529.     #                   $metabolite_counts++;
3530.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3531.     $out_array[7][39] += ($columns[$metabolite_counts][39] +
   $alkynesum);
3532.     $BB += 1;
3533.     #                   $metabolite_counts++;
3534.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3535.     $out_array[12][39] += ($columns[$metabolite_counts][39] +
   $nitsum);
3536.     $BB += 1;
3537.     #                   $metabolite_counts++;
3538.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3539.     $out_array[8][39] += ($columns[$metabolite_counts][39] +
   $thiolsum);
3540.     $BB += 1;
3541.     #                   $metabolite_counts++;
3542.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
```

```
3543.     $out_array[10][39] += ($columns[$metabolite_counts][39] +
   $indolesum);
3544.     $BB += 1;
3545.     #                   $metabolite_counts++;
3546.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3547.     $out_array[9][39] += ($columns[$metabolite_counts][39] +
   $sulfsum);
3548.     $BB += 1;
3549.     #                   $metabolite_counts++;
3550.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3551.     $out_array[13][39] += ($columns[$metabolite_counts][39] +
   $aminesum);
3552.     $BB += 1;
3553.     #                   $metabolite_counts++;
3554.     } elsif (( $BB == 0 )) {
3555.     $out_array[14][39] += ($columns[$metabolite_counts][39] +
   $othersum);
3556.     $BB += 1;
3557.     }
3558.     }
3559.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3560.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
         # skip over lines starts with "Label" "Sample"
3561.     my $BB = 0;
3562.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3563.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3564.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3565.     $out_array[2][40] += ($columns[$metabolite_counts][40] +
   $alcsum);
3566.     $BB += 1;
3567.     #                   $metabolite_counts++;
3568.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3569.     $out_array[4][40] += ($columns[$metabolite_counts][40] +
   $aldsum);
3570.     $BB += 1;
3571.     #                   $metabolite_counts++;
3572.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3573.     $out_array[4][40] += ($columns[$metabolite_counts][40] +
   $aldsum);
3574.     $BB += 1;
3575.     #                   $metabolite_counts++;
3576.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
```

```
3577.     $out_array[3][40] += ($columns[$metabolite_counts][40] +
   $acidsum);
3578.     $BB += 1;
3579.     #                   $metabolite_counts++;
3580.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3581.     $out_array[3][40] += ($columns[$metabolite_counts][40] +
   $acidsum);
3582.     $BB += 1;
3583.     #                   $metabolite_counts++;
3584.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3585.     $out_array[3][40] += ($columns[$metabolite_counts][40] +
   $acidsum);
3586.     $BB += 1;
3587.     #                   $metabolite_counts++;
3588.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3589.     $out_array[11][40] += ($columns[$metabolite_counts][40] +
   $keysum);
3590.     $BB += 1;
3591.     #                   $metabolite_counts++;
3592.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3593.     $out_array[6][40] += ($columns[$metabolite_counts][40] +
   $alkanesum);
3594.     $BB += 1;
3595.     #                   $metabolite_counts++;
3596.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3597.     $out_array[5][40] += ($columns[$metabolite_counts][40] +
   $alkenesum);
3598.     $BB += 1;
3599.     #                   $metabolite_counts++;
3600.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3601.     $out_array[7][40] += ($columns[$metabolite_counts][40] +
   $alkynesum);
3602.     $BB += 1;
3603.     #                   $metabolite_counts++;
3604.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3605.     $out_array[12][40] += ($columns[$metabolite_counts][40] +
   $nitsum);
3606.     $BB += 1;
3607.     #                   $metabolite_counts++;
3608.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3609.     $out_array[8][40] += ($columns[$metabolite_counts][40] +
   $thiolsum);
3610.     $BB += 1;
3611.     #                   $metabolite_counts++;
```

```
3612.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3613.     $out_array[10][40] += ($columns[$metabolite_counts][40] +
   $indolesum);
3614.     $BB += 1;
3615.     #                   $metabolite_counts++;
3616.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3617.     $out_array[9][40] += ($columns[$metabolite_counts][40] +
   $sulfsum);
3618.     $BB += 1;
3619.     #                   $metabolite_counts++;
3620.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3621.     $out_array[13][40] += ($columns[$metabolite_counts][40] +
   $aminesum);
3622.     $BB += 1;
3623.     #                   $metabolite_counts++;
3624.     } elsif (( $BB == 0 )) {
3625.     $out_array[14][40] += ($columns[$metabolite_counts][40] +
   $othersum);
3626.     $BB += 1;
3627.     }
3628.     }
3629.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3630.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
         # skip over lines starts with "Label" "Sample"
3631.     my $BB = 0;
3632.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3633.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3634.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3635.     $out_array[2][41] += ($columns[$metabolite_counts][41] +
   $alcsum);
3636.     $BB += 1;
3637.     #                   $metabolite_counts++;
3638.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3639.     $out_array[4][41] += ($columns[$metabolite_counts][41] +
   $aldsum);
3640.     $BB += 1;
3641.     #                   $metabolite_counts++;
3642.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3643.     $out_array[4][41] += ($columns[$metabolite_counts][41] +
   $aldsum);
3644.     $BB += 1;
3645.     #                   $metabolite_counts++;
```

```
3646.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3647.     $out_array[3][41] += ($columns[$metabolite_counts][41] +
   $acidsum);
3648.     $BB += 1;
3649.     #                   $metabolite_counts++;
3650.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3651.     $out_array[3][41] += ($columns[$metabolite_counts][41] +
   $acidsum);
3652.     $BB += 1;
3653.     #                   $metabolite_counts++;
3654.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3655.     $out_array[3][41] += ($columns[$metabolite_counts][41] +
   $acidsum);
3656.     $BB += 1;
3657.     #                   $metabolite_counts++;
3658.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3659.     $out_array[11][41] += ($columns[$metabolite_counts][41] +
   $keysum);
3660.     $BB += 1;
3661.     #                   $metabolite_counts++;
3662.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3663.     $out_array[6][41] += ($columns[$metabolite_counts][41] +
   $alkanesum);
3664.     $BB += 1;
3665.     #                   $metabolite_counts++;
3666.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3667.     $out_array[5][41] += ($columns[$metabolite_counts][41] +
   $alkenesum);
3668.     $BB += 1;
3669.     #                   $metabolite_counts++;
3670.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3671.     $out_array[7][41] += ($columns[$metabolite_counts][41] +
   $alkynesum);
3672.     $BB += 1;
3673.     #                   $metabolite_counts++;
3674.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3675.     $out_array[12][41] += ($columns[$metabolite_counts][41] +
   $nitsum);
3676.     $BB += 1;
3677.     #                   $metabolite_counts++;
3678.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3679.     $out_array[8][41] += ($columns[$metabolite_counts][41] +
   $thiolsum);
```

```
3680.     $BB += 1;
3681.     #                   $metabolite_counts++;
3682.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3683.     $out_array[10][41] += ($columns[$metabolite_counts][41] +
   $indolesum);
3684.     $BB += 1;
3685.     #                   $metabolite_counts++;
3686.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3687.     $out_array[9][41] += ($columns[$metabolite_counts][41] +
   $sulfsum);
3688.     $BB += 1;
3689.     #                   $metabolite_counts++;
3690.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3691.     $out_array[13][41] += ($columns[$metabolite_counts][41] +
   $aminesum);
3692.     $BB += 1;
3693.     #                   $metabolite_counts++;
3694.     } elsif (( $BB == 0 )) {
3695.     $out_array[14][41] += ($columns[$metabolite_counts][41] +
   $othersum);
3696.     $BB += 1;
3697.     }
3698.     }
3699.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3700.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
3701.     my $BB = 0;
3702.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3703.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3704.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3705.     $out_array[2][42] += ($columns[$metabolite_counts][42] +
   $alcsum);
3706.     $BB += 1;
3707.     #                   $metabolite_counts++;
3708.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3709.     $out_array[4][42] += ($columns[$metabolite_counts][42] +
   $aldsum);
3710.     $BB += 1;
3711.     #                   $metabolite_counts++;
3712.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3713.     $out_array[4][42] += ($columns[$metabolite_counts][42] +
   $aldsum);
```

```
3714.     $BB += 1;
3715.     #                   $metabolite_counts++;
3716.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3717.     $out_array[3][42] += ($columns[$metabolite_counts][42] +
   $acidsum);
3718.     $BB += 1;
3719.     #                   $metabolite_counts++;
3720.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3721.     $out_array[3][42] += ($columns[$metabolite_counts][42] +
   $acidsum);
3722.     $BB += 1;
3723.     #                   $metabolite_counts++;
3724.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3725.     $out_array[3][42] += ($columns[$metabolite_counts][42] +
   $acidsum);
3726.     $BB += 1;
3727.     #                   $metabolite_counts++;
3728.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3729.     $out_array[11][42] += ($columns[$metabolite_counts][42] +
   $keysum);
3730.     $BB += 1;
3731.     #                   $metabolite_counts++;
3732.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3733.     $out_array[6][42] += ($columns[$metabolite_counts][42] +
   $alkanesum);
3734.     $BB += 1;
3735.     #                   $metabolite_counts++;
3736.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3737.     $out_array[5][42] += ($columns[$metabolite_counts][42] +
   $alkenesum);
3738.     $BB += 1;
3739.     #                   $metabolite_counts++;
3740.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3741.     $out_array[7][42] += ($columns[$metabolite_counts][42] +
   $alkynesum);
3742.     $BB += 1;
3743.     #                   $metabolite_counts++;
3744.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3745.     $out_array[12][42] += ($columns[$metabolite_counts][42] +
   $nitsum);
3746.     $BB += 1;
3747.     #                   $metabolite_counts++;
3748.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
```

```
3749.     $out_array[8][42] += ($columns[$metabolite_counts][42] +
   $thiolsum);
3750.     $BB += 1;
3751.     #                   $metabolite_counts++;
3752.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3753.     $out_array[10][42] += ($columns[$metabolite_counts][42] +
   $indolesum);
3754.     $BB += 1;
3755.     #                   $metabolite_counts++;
3756.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3757.     $out_array[9][42] += ($columns[$metabolite_counts][42] +
   $sulfsum);
3758.     $BB += 1;
3759.     #                   $metabolite_counts++;
3760.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3761.     $out_array[13][42] += ($columns[$metabolite_counts][42] +
   $aminesum);
3762.     $BB += 1;
3763.     #                   $metabolite_counts++;
3764.     } elsif (( $BB == 0 )) {
3765.     $out_array[14][42] += ($columns[$metabolite_counts][42] +
   $othersum);
3766.     $BB += 1;
3767.     }
3768.     }
3769.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3770.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
         # skip over lines starts with "Label" "Sample"
3771.     my $BB = 0;
3772.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3773.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3774.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3775.     $out_array[2][43] += ($columns[$metabolite_counts][43] +
   $alcsum);
3776.     $BB += 1;
3777.     #                   $metabolite_counts++;
3778.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3779.     $out_array[4][43] += ($columns[$metabolite_counts][43] +
   $aldsum);
3780.     $BB += 1;
3781.     #                   $metabolite_counts++;
3782.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
```

```
3783.     $out_array[4][43] += ($columns[$metabolite_counts][43] +
   $aldsum);
3784.     $BB += 1;
3785.     #                   $metabolite_counts++;
3786.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3787.     $out_array[3][43] += ($columns[$metabolite_counts][43] +
   $acidsum);
3788.     $BB += 1;
3789.     #                   $metabolite_counts++;
3790.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3791.     $out_array[3][43] += ($columns[$metabolite_counts][43] +
   $acidsum);
3792.     $BB += 1;
3793.     #                   $metabolite_counts++;
3794.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3795.     $out_array[3][43] += ($columns[$metabolite_counts][43] +
   $acidsum);
3796.     $BB += 1;
3797.     #                   $metabolite_counts++;
3798.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3799.     $out_array[11][43] += ($columns[$metabolite_counts][43] +
   $keysum);
3800.     $BB += 1;
3801.     #                   $metabolite_counts++;
3802.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3803.     $out_array[6][43] += ($columns[$metabolite_counts][43] +
   $alkanesum);
3804.     $BB += 1;
3805.     #                   $metabolite_counts++;
3806.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3807.     $out_array[5][43] += ($columns[$metabolite_counts][43] +
   $alkenesum);
3808.     $BB += 1;
3809.     #                   $metabolite_counts++;
3810.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3811.     $out_array[7][43] += ($columns[$metabolite_counts][43] +
   $alkynesum);
3812.     $BB += 1;
3813.     #                   $metabolite_counts++;
3814.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3815.     $out_array[12][43] += ($columns[$metabolite_counts][43] +
   $nitsum);
3816.     $BB += 1;
3817.     #                   $metabolite_counts++;
```

```
3818.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3819.     $out_array[8][43] += ($columns[$metabolite_counts][43] +
   $thiolsum);
3820.     $BB += 1;
3821.     #                   $metabolite_counts++;
3822.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3823.     $out_array[10][43] += ($columns[$metabolite_counts][43] +
   $indolesum);
3824.     $BB += 1;
3825.     #                   $metabolite_counts++;
3826.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3827.     $out_array[9][43] += ($columns[$metabolite_counts][43] +
   $sulfsum);
3828.     $BB += 1;
3829.     #                   $metabolite_counts++;
3830.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3831.     $out_array[13][43] += ($columns[$metabolite_counts][43] +
   $aminesum);
3832.     $BB += 1;
3833.     #                   $metabolite_counts++;
3834.     } elsif (( $BB == 0 )) {
3835.     $out_array[14][43] += ($columns[$metabolite_counts][43] +
   $othersum);
3836.     $BB += 1;
3837.     }
3838.     }
3839.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3840.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
3841.     my $BB = 0;
3842.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3843.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3844.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3845.     $out_array[2][44] += ($columns[$metabolite_counts][44] +
   $alcsum);
3846.     $BB += 1;
3847.     #                   $metabolite_counts++;
3848.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3849.     $out_array[4][44] += ($columns[$metabolite_counts][44] +
   $aldsum);
3850.     $BB += 1;
3851.     #                   $metabolite_counts++;
```

```
3852.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3853.     $out_array[4][44] += ($columns[$metabolite_counts][44] +
   $aldsum);
3854.     $BB += 1;
3855.     #                   $metabolite_counts++;
3856.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3857.     $out_array[3][44] += ($columns[$metabolite_counts][44] +
   $acidsum);
3858.     $BB += 1;
3859.     #                   $metabolite_counts++;
3860.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
3861.     $out_array[3][44] += ($columns[$metabolite_counts][44] +
   $acidsum);
3862.     $BB += 1;
3863.     #                   $metabolite_counts++;
3864.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
3865.     $out_array[3][44] += ($columns[$metabolite_counts][44] +
   $acidsum);
3866.     $BB += 1;
3867.     #                   $metabolite_counts++;
3868.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
3869.     $out_array[11][44] += ($columns[$metabolite_counts][44] +
   $keysum);
3870.     $BB += 1;
3871.     #                   $metabolite_counts++;
3872.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
3873.     $out_array[6][44] += ($columns[$metabolite_counts][44] +
   $alkanesum);
3874.     $BB += 1;
3875.     #                   $metabolite_counts++;
3876.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3877.     $out_array[5][44] += ($columns[$metabolite_counts][44] +
   $alkenesum);
3878.     $BB += 1;
3879.     #                   $metabolite_counts++;
3880.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
3881.     $out_array[7][44] += ($columns[$metabolite_counts][44] +
   $alkynesum);
3882.     $BB += 1;
3883.     #                   $metabolite_counts++;
3884.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
3885.     $out_array[12][44] += ($columns[$metabolite_counts][44] +
   $nitsum);
```

```
3886.     $BB += 1;
3887.     #                   $metabolite_counts++;
3888.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3889.     $out_array[8][44] += ($columns[$metabolite_counts][44] +
   $thiolsum);
3890.     $BB += 1;
3891.     #                   $metabolite_counts++;
3892.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3893.     $out_array[10][44] += ($columns[$metabolite_counts][44] +
   $indolesum);
3894.     $BB += 1;
3895.     #                   $metabolite_counts++;
3896.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3897.     $out_array[9][44] += ($columns[$metabolite_counts][44] +
   $sulfsum);
3898.     $BB += 1;
3899.     #                   $metabolite_counts++;
3900.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3901.     $out_array[13][44] += ($columns[$metabolite_counts][44] +
   $aminesum);
3902.     $BB += 1;
3903.     #                   $metabolite_counts++;
3904.     } elsif (( $BB == 0 )) {
3905.     $out_array[14][44] += ($columns[$metabolite_counts][44] +
   $othersum);
3906.     $BB += 1;
3907.     }
3908.     }
3909.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3910.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
         # skip over lines starts with "Label" "Sample"
3911.     my $BB = 0;
3912.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3913.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3914.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3915.     $out_array[2][45] += ($columns[$metabolite_counts][45] +
   $alcsum);
3916.     $BB += 1;
3917.     #                   $metabolite_counts++;
3918.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
3919.     $out_array[4][45] += ($columns[$metabolite_counts][45] +
   $aldsum);
```

```
    $BB += 1;
    #                   $metabolite_counts++;
    } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
m/aldehyde/ )) {
    $out_array[4][45] += ($columns[$metabolite_counts][45] +
$aldsum);
    $BB += 1;
    #                   $metabolite_counts++;
    } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
m/acid/ )) {
    $out_array[3][45] += ($columns[$metabolite_counts][45] +
$acidsum);
    $BB += 1;
    #                   $metabolite_counts++;
    } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
m/ate/ )) {
    $out_array[3][45] += ($columns[$metabolite_counts][45] +
$acidsum);
    $BB += 1;
    #                   $metabolite_counts++;
    } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
m/ester/ )) {
    $out_array[3][45] += ($columns[$metabolite_counts][45] +
$acidsum);
    $BB += 1;
    #                   $metabolite_counts++;
    } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
m/one/ )) {
    $out_array[11][45] += ($columns[$metabolite_counts][45] +
$keysum);
    $BB += 1;
    #                   $metabolite_counts++;
    } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
    $out_array[6][45] += ($columns[$metabolite_counts][45] +
$alkanesum);
    $BB += 1;
    #                   $metabolite_counts++;
    } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
    $out_array[5][45] += ($columns[$metabolite_counts][45] +
$alkenesum);
    $BB += 1;
    #                   $metabolite_counts++;
    } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
m/yne/ )) {
    $out_array[7][45] += ($columns[$metabolite_counts][45] +
$alkynesum);
    $BB += 1;
    #                   $metabolite_counts++;
    } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
m/nitrile/ )) {
```

```
3955.     $out_array[12][45] += ($columns[$metabolite_counts][45] +
   $nitsum);
3956.     $BB += 1;
3957.     #                   $metabolite_counts++;
3958.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
3959.     $out_array[8][45] += ($columns[$metabolite_counts][45] +
   $thiolsum);
3960.     $BB += 1;
3961.     #                   $metabolite_counts++;
3962.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
3963.     $out_array[10][45] += ($columns[$metabolite_counts][45] +
   $indolesum);
3964.     $BB += 1;
3965.     #                   $metabolite_counts++;
3966.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
3967.     $out_array[9][45] += ($columns[$metabolite_counts][45] +
   $sulfsum);
3968.     $BB += 1;
3969.     #                   $metabolite_counts++;
3970.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
3971.     $out_array[13][45] += ($columns[$metabolite_counts][45] +
   $aminesum);
3972.     $BB += 1;
3973.     #                   $metabolite_counts++;
3974.     } elsif (( $BB == 0 )) {
3975.     $out_array[14][45] += ($columns[$metabolite_counts][45] +
   $othersum);
3976.     $BB += 1;
3977.     }
3978.     }
3979.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
3980.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
3981.     my $BB = 0;
3982.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
3983.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
3984.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
3985.     $out_array[2][46] += ($columns[$metabolite_counts][46] +
   $alcsum);
3986.     $BB += 1;
3987.     #                   $metabolite_counts++;
3988.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
```

```
3989.     $out_array[4][46] += ($columns[$metabolite_counts][46] +
   $aldsum);
3990.     $BB += 1;
3991.     #                   $metabolite_counts++;
3992.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
3993.     $out_array[4][46] += ($columns[$metabolite_counts][46] +
   $aldsum);
3994.     $BB += 1;
3995.     #                   $metabolite_counts++;
3996.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
3997.     $out_array[3][46] += ($columns[$metabolite_counts][46] +
   $acidsum);
3998.     $BB += 1;
3999.     #                   $metabolite_counts++;
4000.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
4001.     $out_array[3][46] += ($columns[$metabolite_counts][46] +
   $acidsum);
4002.     $BB += 1;
4003.     #                   $metabolite_counts++;
4004.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
4005.     $out_array[3][46] += ($columns[$metabolite_counts][46] +
   $acidsum);
4006.     $BB += 1;
4007.     #                   $metabolite_counts++;
4008.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
4009.     $out_array[11][46] += ($columns[$metabolite_counts][46] +
   $keysum);
4010.     $BB += 1;
4011.     #                   $metabolite_counts++;
4012.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
4013.     $out_array[6][46] += ($columns[$metabolite_counts][46] +
   $alkanesum);
4014.     $BB += 1;
4015.     #                   $metabolite_counts++;
4016.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
4017.     $out_array[5][46] += ($columns[$metabolite_counts][46] +
   $alkenesum);
4018.     $BB += 1;
4019.     #                   $metabolite_counts++;
4020.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
4021.     $out_array[7][46] += ($columns[$metabolite_counts][46] +
   $alkynesum);
4022.     $BB += 1;
4023.     #                   $metabolite_counts++;
```

```
4024.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
4025.     $out_array[12][46] += ($columns[$metabolite_counts][46] +
   $nitsum);
4026.     $BB += 1;
4027.     #                   $metabolite_counts++;
4028.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
4029.     $out_array[8][46] += ($columns[$metabolite_counts][46] +
   $thiolsum);
4030.     $BB += 1;
4031.     #                   $metabolite_counts++;
4032.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
4033.     $out_array[10][46] += ($columns[$metabolite_counts][46] +
   $indolesum);
4034.     $BB += 1;
4035.     #                   $metabolite_counts++;
4036.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
4037.     $out_array[9][46] += ($columns[$metabolite_counts][46] +
   $sulfsum);
4038.     $BB += 1;
4039.     #                   $metabolite_counts++;
4040.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
4041.     $out_array[13][46] += ($columns[$metabolite_counts][46] +
   $aminesum);
4042.     $BB += 1;
4043.     #                   $metabolite_counts++;
4044.     } elsif (( $BB == 0 )) {
4045.     $out_array[14][46] += ($columns[$metabolite_counts][46] +
   $othersum);
4046.     $BB += 1;
4047.     }
4048.     }
4049.     for (my $metabolite_counts = -1; $metabolite_counts <=
   $total_metabolites; $metabolite_counts++) {
4050.     next  if ($columns[$metabolite_counts][0] =~ /(^Sample|^Label)/);
          # skip over lines starts with "Label" "Sample"
4051.     my $BB = 0;
4052.     #            next $columns[$metabolite_counts] if
   ($columns[$metabolite_counts] =~ /(^Label|^Sample|^alcoholic)/);  # skip
   over lines starts with "Label" "Sample" "alcoholic" or "healthy"
4053.     if ($columns[$metabolite_counts][0] =~ m/.*ol|.*ol,/ and
   $columns[$metabolite_counts][0] !~ m/thiol/ ) {
4054.     #                   next line if ($columns[$metabolite_counts][0]
   =~ /(^Label|^Sample|^alcoholic)/);
4055.     $out_array[2][47] += ($columns[$metabolite_counts][47] +
   $alcsum);
4056.     $BB += 1;
4057.     #                   $metabolite_counts++;
```

```
4058.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~ m/al/
   and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
4059.     $out_array[4][47] += ($columns[$metabolite_counts][47] +
   $aldsum);
4060.     $BB += 1;
4061.     #                   $metabolite_counts++;
4062.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/aldehyde/ )) {
4063.     $out_array[4][47] += ($columns[$metabolite_counts][47] +
   $aldsum);
4064.     $BB += 1;
4065.     #                   $metabolite_counts++;
4066.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/acid/ )) {
4067.     $out_array[3][47] += ($columns[$metabolite_counts][47] +
   $acidsum);
4068.     $BB += 1;
4069.     #                   $metabolite_counts++;
4070.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ate/ )) {
4071.     $out_array[3][47] += ($columns[$metabolite_counts][47] +
   $acidsum);
4072.     $BB += 1;
4073.     #                   $metabolite_counts++;
4074.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ester/ )) {
4075.     $out_array[3][47] += ($columns[$metabolite_counts][47] +
   $acidsum);
4076.     $BB += 1;
4077.     #                   $metabolite_counts++;
4078.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/one/ )) {
4079.     $out_array[11][47] += ($columns[$metabolite_counts][47] +
   $keysum);
4080.     $BB += 1;
4081.     #                   $metabolite_counts++;
4082.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ane/ and $columns[$metabolite_counts][0] !~ m/thiol/ )) {
4083.     $out_array[6][47] += ($columns[$metabolite_counts][47] +
   $alkanesum);
4084.     $BB += 1;
4085.     #                   $metabolite_counts++;
4086.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ene/ and $columns[$metabolite_counts][0] !~ m/nitrile/)) {
4087.     $out_array[5][47] += ($columns[$metabolite_counts][47] +
   $alkenesum);
4088.     $BB += 1;
4089.     #                   $metabolite_counts++;
4090.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/yne/ )) {
4091.     $out_array[7][47] += ($columns[$metabolite_counts][47] +
   $alkynesum);
```

```
4092.     $BB += 1;
4093.     #                   $metabolite_counts++;
4094.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/nitrile/ )) {
4095.     $out_array[12][47] += ($columns[$metabolite_counts][47] +
   $nitsum);
4096.     $BB += 1;
4097.     #                   $metabolite_counts++;
4098.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/thiol/ )) {
4099.     $out_array[8][47] += ($columns[$metabolite_counts][47] +
   $thiolsum);
4100.     $BB += 1;
4101.     #                   $metabolite_counts++;
4102.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/ole/ )) {
4103.     $out_array[10][47] += ($columns[$metabolite_counts][47] +
   $indolesum);
4104.     $BB += 1;
4105.     #                   $metabolite_counts++;
4106.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/sulfide/ )) {
4107.     $out_array[9][47] += ($columns[$metabolite_counts][47] +
   $sulfsum);
4108.     $BB += 1;
4109.     #                   $metabolite_counts++;
4110.     } if (( $BB == 0 ) and ( $columns[$metabolite_counts][0] =~
   m/amine/ )) {
4111.     $out_array[13][47] += ($columns[$metabolite_counts][47] +
   $aminesum);
4112.     $BB += 1;
4113.     #                   $metabolite_counts++;
4114.     } elsif (( $BB == 0 )) {
4115.     $out_array[14][47] += ($columns[$metabolite_counts][47] +
   $othersum);
4116.     $BB += 1;
4117.     }
4118.     }


4119.     #print statements
4120.     #print "$filename\n";
4121.     print "Total Metabolite count: $total_metabolites\n";
4122.     #print "$row_count\n";
4123.     print "Total Samples count: $samples_count\n";

4124.     #print outfile to new tab delimited file with OUTPUT followed by
   the filename
4125.     open OUTPUT, ">OUTPUT $filename.txt" or die "Can't open
   $filename.txt\n";
4126.     #prints out every position in the matrix that was defined earlier
```

```
4127.     print OUTPUT "$out_array[0][0]\t $out_array[0][1]\t
   $out_array[0][2]\t
4128.     $out_array[0][3]\t $out_array[0][4]\t $out_array[0][5]\t
4129.     $out_array[0][6]\t $out_array[0][7]\t $out_array[0][8]\t
4130.     $out_array[0][9]\t $out_array[0][10]\t $out_array[0][11]\t
4131.     $out_array[0][12]\t $out_array[0][13]\t $out_array[0][14]\t
4132.     $out_array[0][15]\t $out_array[0][16]\t $out_array[0][17]\t
4133.     $out_array[0][18]\t $out_array[0][19]\t $out_array[0][20]\t
4134.     $out_array[0][21]\t $out_array[0][22]\t $out_array[0][23]\t
4135.     $out_array[0][24]\t $out_array[0][25]\t $out_array[0][26]\t
4136.     $out_array[0][27]\t $out_array[0][28]\t $out_array[0][29]\t
4137.     $out_array[0][30]\t $out_array[0][31]\t $out_array[0][32]\t
4138.     $out_array[0][33]\t $out_array[0][34]\t $out_array[0][35]\t
4139.     $out_array[0][36]\t $out_array[0][37]\t $out_array[0][38]\t
4140.     $out_array[0][39]\t $out_array[0][40]\t $out_array[0][41]\t
4141.     $out_array[0][42]\t $out_array[0][43]\t $out_array[0][44]\t
4142.     $out_array[0][45]\t $out_array[0][46]\t $out_array[0][47]\t
4143.     $out_array[1][0]\t $out_array[1][1]\t $out_array[1][2]\t
4144.     $out_array[1][3]\t $out_array[1][4]\t $out_array[1][5]\t
4145.     $out_array[1][6]\t $out_array[1][7]\t $out_array[1][8]\t
4146.     $out_array[1][9]\t $out_array[1][10]\t $out_array[1][11]\t
4147.     $out_array[1][12]\t $out_array[1][13]\t $out_array[1][14]\t
4148.     $out_array[1][15]\t $out_array[1][16]\t $out_array[1][17]\t
4149.     $out_array[1][18]\t $out_array[1][19]\t $out_array[1][20]\t
4150.     $out_array[1][21]\t $out_array[1][22]\t $out_array[1][23]\t
4151.     $out_array[1][24]\t $out_array[1][25]\t $out_array[1][26]\t
4152.     $out_array[1][27]\t $out_array[1][28]\t $out_array[1][29]\t
4153.     $out_array[1][30]\t $out_array[1][31]\t $out_array[1][32]\t
4154.     $out_array[1][33]\t $out_array[1][34]\t $out_array[1][35]\t
4155.     $out_array[1][36]\t $out_array[1][37]\t $out_array[1][38]\t
4156.     $out_array[1][39]\t $out_array[1][40]\t $out_array[1][41]\t
4157.     $out_array[1][42]\t $out_array[1][43]\t $out_array[1][44]\t
4158.     $out_array[1][45]\t  $out_array[1][46]\t $out_array[1][47]\t
4159.     $out_array[2][0]\t $out_array[2][1]\t $out_array[2][2]\t
4160.     $out_array[2][3]\t $out_array[2][4]\t $out_array[2][5]\t
4161.     $out_array[2][6]\t $out_array[2][7]\t$out_array[2][8]\t
4162.     $out_array[2][9]\t $out_array[2][10]\t $out_array[2][11]\t
4163.     $out_array[2][12]\t $out_array[2][13]\t $out_array[2][14]\t
4164.     $out_array[2][15]\t $out_array[2][16]\t $out_array[2][17]\t
4165.     $out_array[2][18]\t $out_array[2][19]\t $out_array[2][20]\t
4166.     $out_array[2][21]\t $out_array[2][22]\t $out_array[2][23]\t
4167.     $out_array[2][24]\t $out_array[2][25]\t $out_array[2][26]\t
4168.     $out_array[2][27]\t $out_array[2][28]\t $out_array[2][29]\t
4169.     $out_array[2][30]\t $out_array[2][31]\t $out_array[2][32]\t
4170.     $out_array[2][33]\t $out_array[2][34]\t $out_array[2][35]\t
4171.     $out_array[2][36]\t $out_array[2][37]\t $out_array[2][38]\t
4172.     $out_array[2][39]\t $out_array[2][40]\t $out_array[2][41]\t
4173.     $out_array[2][42]\t $out_array[2][43]\t $out_array[2][44]\t
4174.     $out_array[2][45]\t $out_array[2][46]\t $out_array[2][47]\t
4175.     $out_array[3][0]\t $out_array[3][1]\t $out_array[3][2]\t
4176.     $out_array[3][3]\t $out_array[3][4]\t $out_array[3][5]\t
4177.     $out_array[3][6]\t $out_array[3][7]\t $out_array[3][8]\t
```

```
    $out_array[3][9]\t $out_array[3][10]\t $out_array[3][11]\t
     $out_array[3][12]\t $out_array[3][13]\t $out_array[3][14]\t
    $out_array[3][15]\t $out_array[3][16]\t $out_array[3][17]\t
    $out_array[3][18]\t $out_array[3][19]\t $out_array[3][20]\t
    $out_array[3][21]\t $out_array[3][22]\t $out_array[3][23]\t
    $out_array[3][24]\t $out_array[3][25]\t $out_array[3][26]\t
    $out_array[3][27]\t $out_array[3][28]\t $out_array[3][29]\t
    $out_array[3][30]\t $out_array[3][31]\t $out_array[3][32]\t
    $out_array[3][33]\t $out_array[3][34]\t $out_array[3][35]\t
    $out_array[3][36]\t $out_array[3][37]\t $out_array[3][38]\t
    $out_array[3][39]\t $out_array[3][40]\t $out_array[3][41]\t
    $out_array[3][42]\t $out_array[3][43]\t $out_array[3][44]\t
    $out_array[3][45]\t $out_array[3][46]\t $out_array[3][47]\t
    $out_array[4][0]\t $out_array[4][1]\t $out_array[4][2]\t
    $out_array[4][3]\t $out_array[4][4]\t $out_array[4][5]\t
    $out_array[4][6]\t $out_array[4][7]\t $out_array[4][8]\t
    $out_array[4][9]\t $out_array[4][10]\t $out_array[4][11]\t
    $out_array[4][12]\t $out_array[4][13]\t $out_array[4][14]\t
    $out_array[4][15]\t $out_array[4][16]\t $out_array[4][17]\t
    $out_array[4][18]\t $out_array[4][19]\t $out_array[4][20]\t
    $out_array[4][21]\t $out_array[4][22]\t $out_array[4][23]\t
    $out_array[4][24]\t $out_array[4][25]\t $out_array[4][26]\t
    $out_array[4][27]\t $out_array[4][28]\t $out_array[4][29]\t
    $out_array[4][30]\t $out_array[4][31]\t $out_array[4][32]\t
    $out_array[4][33]\t $out_array[4][34]\t $out_array[4][35]\t
    $out_array[4][36]\t $out_array[4][37]\t $out_array[4][38]\t
    $out_array[4][39]\t $out_array[4][40]\t $out_array[4][41]\t
    $out_array[4][42]\t $out_array[4][43]\t $out_array[4][44]\t
    $out_array[4][45]\t $out_array[4][46]\t $out_array[4][47]\t
    $out_array[5][0]\t $out_array[5][1]\t $out_array[5][2]\t
    $out_array[5][3]\t $out_array[5][4]\t $out_array[5][5]\t
    $out_array[5][6]\t $out_array[5][7]\t $out_array[5][8]\t
    $out_array[5][9]\t $out_array[5][10]\t $out_array[5][11]\t
    $out_array[5][12]\t $out_array[5][13]\t $out_array[5][14]\t
    $out_array[5][15]\t $out_array[5][16]\t $out_array[5][17]\t
    $out_array[5][18]\t $out_array[5][19]\t $out_array[5][20]\t
    $out_array[5][21]\t $out_array[5][22]\t $out_array[5][23]\t
    $out_array[5][24]\t $out_array[5][25]\t $out_array[5][26]\t
    $out_array[5][27]\t $out_array[5][28]\t $out_array[5][29]\t
    $out_array[5][30]\t $out_array[5][31]\t $out_array[5][32]\t
    $out_array[5][33]\t $out_array[5][34]\t $out_array[5][35]\t
    $out_array[5][36]\t $out_array[5][37]\t $out_array[5][38]\t
    $out_array[5][39]\t $out_array[5][40]\t $out_array[5][41]\t
    $out_array[5][42]\t $out_array[5][43]\t $out_array[5][44]\t
    $out_array[5][45]\t $out_array[5][46]\t $out_array[5][47]\t
    $out_array[6][0]\t $out_array[6][1]\t $out_array[6][2]\t
    $out_array[6][3]\t $out_array[6][4]\t $out_array[6][5]\t
    $out_array[6][6]\t $out_array[6][7]\t $out_array[6][8]\t
    $out_array[6][9]\t $out_array[6][10]\t $out_array[6][11]\t
    $out_array[6][12]\t $out_array[6][13]\t $out_array[6][14]\t
    $out_array[6][15]\t $out_array[6][16]\t $out_array[6][17]\t
    $out_array[6][18]\t $out_array[6][19]\t $out_array[6][20]\t
```

```
4230.     $out_array[6][21]\t $out_array[6][22]\t $out_array[6][23]\t
4231.     $out_array[6][24]\t $out_array[6][25]\t $out_array[6][26]\t
4232.     $out_array[6][27]\t $out_array[6][28]\t $out_array[6][29]\t
4233.     $out_array[6][30]\t $out_array[6][31]\t $out_array[6][32]\t
4234.     $out_array[6][33]\t $out_array[6][34]\t $out_array[6][35]\t
4235.     $out_array[6][36]\t $out_array[6][37]\t $out_array[6][38]\t
4236.     $out_array[6][39]\t $out_array[6][40]\t $out_array[6][41]\t
4237.     $out_array[6][42]\t $out_array[6][43]\t $out_array[6][44]\t
4238.     $out_array[6][45]\t $out_array[6][46]\t $out_array[6][47]\t
4239.     $out_array[7][0]\t $out_array[7][1]\t $out_array[7][2]\t
4240.     $out_array[7][3]\t $out_array[7][4]\t $out_array[7][5]\t
4241.     $out_array[7][6]\t $out_array[7][7]\t $out_array[7][8]\t
4242.     $out_array[7][9]\t $out_array[7][10]\t $out_array[7][11]\t
4243.     $out_array[7][12]\t $out_array[7][13]\t $out_array[7][14]\t
4244.     $out_array[7][15]\t $out_array[7][16]\t $out_array[7][17]\t
4245.     $out_array[7][18]\t $out_array[7][19]\t $out_array[7][20]\t
4246.     $out_array[7][21]\t $out_array[7][22]\t $out_array[7][23]\t
4247.     $out_array[7][24]\t $out_array[7][25]\t $out_array[7][26]\t
4248.     $out_array[7][27]\t $out_array[7][28]\t $out_array[7][29]\t
4249.     $out_array[7][30]\t $out_array[7][31]\t $out_array[7][32]\t
4250.     $out_array[7][33]\t $out_array[7][34]\t $out_array[7][35]\t
4251.     $out_array[7][36]\t $out_array[7][37]\t $out_array[7][38]\t
4252.     $out_array[7][39]\t $out_array[7][40]\t $out_array[7][41]\t
4253.     $out_array[7][42]\t $out_array[7][43]\t $out_array[7][44]\t
4254.     $out_array[7][45]\t $out_array[7][46]\t $out_array[7][47]\t
4255.     $out_array[8][0]\t $out_array[8][1]\t $out_array[8][2]\t
4256.     $out_array[8][3]\t $out_array[8][4]\t $out_array[8][5]\t
4257.     $out_array[8][6]\t $out_array[8][7]\t $out_array[8][8]\t
4258.     $out_array[8][9]\t $out_array[8][10]\t $out_array[8][11]\t
4259.     $out_array[8][12]\t $out_array[8][13]\t $out_array[8][14]\t
4260.     $out_array[8][15]\t $out_array[8][16]\t $out_array[8][17]\t
4261.     $out_array[8][18]\t $out_array[8][19]\t $out_array[8][20]\t
4262.     $out_array[8][21]\t $out_array[8][22]\t $out_array[8][23]\t
4263.     $out_array[8][24]\t $out_array[8][25]\t $out_array[8][26]\t
4264.     $out_array[8][27]\t $out_array[8][28]\t $out_array[8][29]\t
4265.     $out_array[8][30]\t $out_array[8][31]\t $out_array[8][32]\t
4266.     $out_array[8][33]\t $out_array[8][34]\t  $out_array[8][35]\t
4267.     $out_array[8][36]\t $out_array[8][37]\t $out_array[8][38]\t
4268.     $out_array[8][39]\t $out_array[8][40]\t $out_array[8][41]\t
4269.     $out_array[8][42]\t $out_array[8][43]\t $out_array[8][44]\t
4270.     $out_array[8][45]\t $out_array[8][46]\t $out_array[8][47]\t
4271.     $out_array[9][0]\t $out_array[9][1]\t $out_array[9][2]\t
4272.     $out_array[9][3]\t $out_array[9][4]\t $out_array[9][5]\t
4273.     $out_array[9][6]\t $out_array[9][7]\t $out_array[9][8]\t
4274.     $out_array[9][9]\t $out_array[9][10]\t $out_array[9][11]\t
4275.     $out_array[9][12]\t $out_array[9][13]\t $out_array[9][14]\t
4276.     $out_array[9][15]\t $out_array[9][16]\t $out_array[9][17]\t
4277.     $out_array[9][18]\t $out_array[9][19]\t $out_array[9][20]\t
4278.     $out_array[9][21]\t $out_array[9][22]\t $out_array[9][23]\t
4279.     $out_array[9][24]\t $out_array[9][25]\t $out_array[9][26]\t
4280.     $out_array[9][27]\t $out_array[9][28]\t $out_array[9][29]\t
4281.     $out_array[9][30]\t $out_array[9][31]\t $out_array[9][32]\t
```

```
4282.     $out_array[9][33]\t $out_array[9][34]\t  $out_array[9][35]\t
4283.     $out_array[9][36]\t $out_array[9][37]\t $out_array[9][38]\t
4284.     $out_array[9][39]\t $out_array[9][40]\t $out_array[9][41]\t
4285.     $out_array[9][42]\t $out_array[9][43]\t $out_array[9][44]\t
4286.     $out_array[9][45]\t $out_array[9][46]\t $out_array[9][47]\t
4287.     $out_array[10][0]\t $out_array[10][1]\t $out_array[10][2]\t
4288.     $out_array[10][3]\t $out_array[10][4]\t $out_array[10][5]\t
4289.     $out_array[10][6]\t $out_array[10][7]\t $out_array[10][8]\t
4290.     $out_array[10][9]\t $out_array[10][10]\t $out_array[10][11]\t
4291.     $out_array[10][12]\t $out_array[10][13]\t $out_array[10][14]\t
4292.     $out_array[10][15]\t $out_array[10][16]\t $out_array[10][17]\t
4293.     $out_array[10][18]\t $out_array[10][19]\t $out_array[10][20]\t
4294.     $out_array[10][21]\t $out_array[10][22]\t $out_array[10][23]\t
4295.     $out_array[10][24]\t $out_array[10][25]\t $out_array[10][26]\t
4296.     $out_array[10][27]\t $out_array[10][28]\t $out_array[10][29]\t
4297.     $out_array[10][30]\t $out_array[10][31]\t$out_array[10][32]\t
4298.     $out_array[10][33]\t $out_array[10][34]\t  $out_array[10][35]\t
4299.     $out_array[10][36]\t $out_array[10][37]\t $out_array[10][38]\t
4300.     $out_array[10][39]\t $out_array[10][40]\t $out_array[10][41]\t
4301.     $out_array[10][42]\t $out_array[10][43]\t $out_array[10][44]\t
4302.     $out_array[10][45]\t $out_array[10][46]\t $out_array[10][47]\t
4303.     $out_array[11][0]\t $out_array[11][1]\t $out_array[11][2]\t
4304.     $out_array[11][3]\t $out_array[11][4]\t $out_array[11][5]\t
4305.     $out_array[11][6]\t $out_array[11][7]\t $out_array[11][8]\t
4306.     $out_array[11][9]\t $out_array[11][10]\t $out_array[11][11]\t
4307.     $out_array[11][12]\t $out_array[11][13]\t $out_array[11][14]\t
4308.     $out_array[11][15]\t $out_array[11][16]\t $out_array[11][17]\t
4309.     $out_array[11][18]\t $out_array[11][19]\t $out_array[11][20]\t
4310.     $out_array[11][21]\t $out_array[11][22]\t $out_array[11][23]\t
4311.     $out_array[11][24]\t $out_array[11][25]\t $out_array[11][26]\t
4312.     $out_array[11][27]\t $out_array[11][28]\t $out_array[11][29]\t
4313.     $out_array[11][30]\t $out_array[11][31]\t$out_array[11][32]\t
4314.     $out_array[11][33]\t $out_array[11][34]\t  $out_array[11][35]\t
4315.     $out_array[11][36]\t $out_array[11][37]\t $out_array[11][38]\t
4316.     $out_array[11][39]\t $out_array[11][40]\t $out_array[11][41]\t
4317.     $out_array[11][42]\t $out_array[11][43]\t $out_array[11][44]\t
4318.     $out_array[11][45]\t $out_array[11][46]\t $out_array[11][47]\t
4319.     $out_array[12][0]\t $out_array[12][1]\t $out_array[12][2]\t
4320.     $out_array[12][3]\t $out_array[12][4]\t $out_array[12][5]\t
4321.     $out_array[12][6]\t $out_array[12][7]\t $out_array[12][8]\t
4322.     $out_array[12][9]\t $out_array[12][10]\t $out_array[12][11]\t
4323.     $out_array[12][12]\t $out_array[12][13]\t $out_array[12][14]\t
4324.     $out_array[12][15]\t $out_array[12][16]\t $out_array[12][17]\t
4325.     $out_array[12][18]\t $out_array[12][19]\t $out_array[12][20]\t
4326.     $out_array[12][21]\t $out_array[12][22]\t $out_array[12][23]\t
4327.     $out_array[12][24]\t $out_array[12][25]\t $out_array[12][26]\t
4328.     $out_array[12][27]\t $out_array[12][28]\t $out_array[12][29]\t
4329.     $out_array[12][30]\t $out_array[12][31]\t $out_array[12][32]\t
4330.     $out_array[12][33]\t $out_array[12][34]\t  $out_array[12][35]\t
4331.     $out_array[12][36]\t $out_array[12][37]\t $out_array[12][38]\t
4332.     $out_array[12][39]\t $out_array[12][40]\t $out_array[12][41]\t
4333.     $out_array[12][42]\t $out_array[12][43]\t $out_array[12][44]\t
```

```
4334.     $out_array[12][45]\t $out_array[12][46]\t $out_array[12][47]\t
4335.     $out_array[13][0]\t $out_array[13][1]\t $out_array[13][2]\t
4336.     $out_array[13][3]\t $out_array[13][4]\t $out_array[13][5]\t
4337.     $out_array[13][6]\t $out_array[13][7]\t $out_array[13][8]\t
4338.     $out_array[13][9]\t $out_array[13][10]\t $out_array[13][11]\t
4339.     $out_array[13][12]\t $out_array[13][13]\t $out_array[13][14]\t
4340.     $out_array[13][15]\t $out_array[13][16]\t $out_array[13][17]\t
4341.     $out_array[13][18]\t $out_array[13][19]\t $out_array[13][20]\t
4342.     $out_array[13][21]\t $out_array[13][22]\t $out_array[13][23]\t
4343.     $out_array[13][24]\t $out_array[13][25]\t $out_array[13][26]\t
4344.     $out_array[13][27]\t $out_array[13][28]\t $out_array[13][29]\t
4345.     $out_array[13][30]\t $out_array[13][31]\t $out_array[13][32]\t
4346.     $out_array[13][33]\t $out_array[13][34]\t  $out_array[13][35]\t
4347.     $out_array[13][36]\t $out_array[13][37]\t $out_array[13][38]\t
4348.     $out_array[13][39]\t $out_array[13][40]\t $out_array[13][41]\t
4349.     $out_array[13][42]\t $out_array[13][43]\t $out_array[13][44]\t
4350.     $out_array[13][45]\t $out_array[13][46]\t $out_array[13][47]\t
4351.     $out_array[14][0]\t $out_array[14][1]\t $out_array[14][2]\t
4352.     $out_array[14][3]\t $out_array[14][4]\t $out_array[14][5]\t
4353.     $out_array[14][6]\t $out_array[14][7]\t $out_array[14][8]\t
4354.     $out_array[14][9]\t $out_array[14][10]\t $out_array[14][11]\t
4355.     $out_array[14][12]\t $out_array[14][13]\t $out_array[14][14]\t
4356.     $out_array[14][15]\t$out_array[14][16]\t $out_array[14][17]\t
4357.     $out_array[14][18]\t $out_array[14][19]\t $out_array[14][20]\t
4358.     $out_array[14][21]\t $out_array[14][22]\t $out_array[14][23]\t
4359.     $out_array[14][24]\t $out_array[14][25]\t $out_array[14][26]\t
4360.     $out_array[14][27]\t $out_array[14][28]\t $out_array[14][29]\t
4361.     $out_array[14][30]\t $out_array[14][31]\t $out_array[14][32]\t
4362.     $out_array[14][33]\t $out_array[14][34]\t$out_array[14][35]\t
4363.     $out_array[14][36]\t $out_array[14][37]\t $out_array[14][38]\t
4364.     $out_array[14][39]\t $out_array[14][40]\t $out_array[14][41]\t
4365.     $out_array[14][42]\t $out_array[14][43]\t $out_array[14][44]\t
4366.     $out_array[14][45]\t $out_array[14][46]\t out_array[14][47]\t\n";
```

APPENDIX B

Metabolites	007A	022A	023A
Benzoyl bromide	1930676	1043194	360289.5
Benzene, 1-ethenyl-4-ethyl-	593242	593242	52298
2-Propenal, 3-phenyl-	660315	47634	465345
octadecenal	1631002	313786	1419034
Cyclohexene, 3-methyl-6-(1-methylethenyl)-, (3R-trans)-	235843	664028	1301495
Cyclobutane, 1,3-diisopropenyl-, trans	337098	1301495	1301495
2-Methoxybenzoic acid, 2,3-dichlorophenyl ester	302947.5	206874	302947.5
Cyclobutane-1,1-dicarboxamide, N,N'-di-benzoyloxy-	495473	1292387	1202559
Cyclopentane, decyl-	6309394	1639448	12290212
Tridecyl pentafluoropropionate	6577256	2352816	8818987
Hexadecane, 7-methyl-	1687252	1338252	3496384
Ethanol, 2-(9-octadecenyloxy)-, (E)-	1778874	1778874	1778874
Tetradecyl trifluoroacetate	8054188	4007903	12862769
Heptadecane, 8-methyl-	3578401	5088837	3496384
Z-1,6-Tridecadiene	2105722	855479	2367299
Isocaryophillene	83909	2997594	2997594
Phenacyl thiocyanate	95181	1043194	192246.5
Ethanone, 2,2,2-trifluoro-1-phenyl-	95181	1007199	210724
5(4H)-Oxazolone, 2-phenyl-4-(phenylmethylene)-, (Z)-	15625	15625	15625
Hippuric-benzaldehyde azalactone	42685	42685	42685
2-Tridecene, (E)-	259013.3	259013.3	259013.3
1,4-Cyclohexadiene, 1-methyl-4-(1-methylethyl)-	135969	260484.5	260484.5
Methanone, (4-bromo-5-methyl-2-nitro-3-thienyl)(phenyl)-	95181	418363	85047.5
3,11-Tetradecadien-1-ol	268546.8	268546.8	268546.8
9-Tetradecenal, (Z)-	1064990	807672	807672
7-Hexadecenal, (Z)-	1353006	313786	713987
13-Tetradecenal	1631002	901357	901357
E-11-Hexadecenal	1353006	1133045	1133045
Benzoylformic acid	116026	1043194	242197
octadecanoic acid	86121.25	69120	86121.25
Octadecanoic acid, 2-(2-hydroxyethoxy)ethyl ester	89530	89530	89530
1,5-Cyclooctadiene, 3,4-dimethyl-	337098	664028	706635

Pentafluoropropionic acid, tridecyl ester	261223.8	877698	261223.8
1-Dodecanol, 3,7,11-trimethyl-	676664.8	1639448	676664.8
Heptafluorobutyric acid, n-pentadecyl ester	837575	1639448	837575
Trichloroacetic acid, pentadecyl ester	652920	905078	652920
Dodecanal	3840973	630891	3108335
Tridecanal	5460391	630891	4192629
Oleyl Alcohol	5460391	855479	4060430
E-11-Hexadecen-1-ol	1600251	855479	392550
Naphthalene, 1,2,3,5,6,7,8,8a-octahydro-1,8a-dimethyl-7-(1-methylethenyl)-, [1R-(1a,7a,8aa)]-	1141798	1141798	1141798
benzocycloheptene, 2,4a,5,6,7,8,9,9a-octahydro-3,5,5-trimethyl-9-methylene	4975557	4975557	4975557
Aromadendrene	83909	963217.8	963217.8
Phenol, 2,4-bis(1,1-dimethylethyl)-	2846149	475933.3	475933.3
Pentanoic acid, 5-hydroxy-, 2,4-di-t-butylphenyl esters	285852	285852	285852
6-Benzylquinoline	41145	27540.5	27540.5
Ethanone, 2-hydroxy-1-phenyl-	116026	1043194	206044
Benzylidenemalonaldehyde	458248	47634	454975
N-Benzoylimidazole	201006.5	1007199	201006.5
a,a-Dichloroacetophenone	95181	1043194	191289
Ethanone, 2-(formyloxy)-1-phenyl-	116026	1043194	2808929
Propanoic acid, 2-(benzoylthio)-, ethyl ester	116026	418363	153657.5
Benzene, tert-butyl-	818368	818368	818368
1,3,5-Cycloheptatriene, 3,7,7-trimethyl-	386103	386103	386103
Hexanedioic acid, dioctyl ester	1949366	1949366	1949366
Hexanedioic acid, mono(2-ethylhexyl)ester	354640.8	354640.8	161648
Hexanedioic acid, bis(2-ethylhexyl) ester	237029	237029	233769
Cyclohexene, 1-methyl-4-(1-methylethylidene)-	620229	620229	620229
3-Cyclohexen-1-ol, 4-methyl-1-(1-methylethyl)-, acetate	49933	420805.8	420805.8
Pentafluoropropionic acid, 4-hexadecyl ester	5058226	3191939	4210220
Tridecyl acetate	5870372	846472	8491219
Dodecyl trifluoroacetate	6791505	4007903	9632517
Eicosane, 10-methyl-	3056234	5088837	3056234
Dichloroacetic acid, tridecyl ester	6346479	1639448	7868125
1-Decanol, 2-hexyl-	3603455	3914555	11705973
1-Tetradecyl acetate	7508714	3914555	12862769
(+)-Isomenthol	1280145	1317977	438242.5
cyclohexanol, 5-methyl-2-(1-methylethyl)-	1280145	1317977	460308
Bicyclo[5.3.0]decane, 2-methylene-5-(1-methylvinyl)-8-methyl-	1473757	1473757	1473757
1R,3Z,9s-4,11,11-Trimethyl-8-methylenebicyclo[7.2.0]undec-3-ene	877342	2418422	2997594

Phthalic acid, cyclohexyl isohexyl ester	3538647	3551183	6452036
Phthalic acid, butyl isohexyl ester	5819074	3551183	6452036
Benzene, 4-ethenyl-1,2-dimethyl-	45850.5	45850.5	45850.5
Benzofuran, 2-methyl-	538078	494890	494890
5-Tetradecene, (E)-	909967	909967	841627
E-9-Hexadecenal	1353006	713987	713987
7-Tetradecenal, (Z)-	471483	261596.3	261596.3
Undecanal	473512	473512	473512
Z-2-Dodecenol	430572	78328	1218680
2-Tridecen-1-ol, (E)-	458210	630891	1326264
Cyclododecanol	481322	153454	2358303
Benzoyl isothiocyanate	116026	1043194	244919
2-Benzoyloxyacetophenone	90798	418363	90798
cyclohexene, 1-methyl-5-(1-methylethenyl)-	359936	690656	117601
Trichloroacetic acid, tridecyl ester	870798.5	854710	870798.5
Decyl trifluoroacetate	752358.3	815964	752358.3
Cyclodecane, methyl-	607756.5	815964	607756.5
E-11,13-Tetradecadien-1-ol	1935787	815964	3805772
Benzoic acid 2-bromoethyl ester	229576	97436	1202559
Cyclopropanecarboxamide, N-benzoyloxy-	1202559	1292387	1202559
Butanedioic acid, 2,3-bis(benzoyloxy)-, [S-(R*,R*)]-	1367561	97436	1367561
(d)-(+)-(2R,3R)-2,3-Dibenzoyltartaric acid	486688.5	1292387	1202559
10-Heneicosene (c,t)	73850	2282456	3882452
9-Nonadecene	5060828	3984915	8254002
5-Eicosene, (E)-	6218450	3984915	12239675
Bicyclo[3.1.0]hexane, 4-methyl-1-(1-methylethyl)-, didehydro deriv.	807529	807529	807529
octadecenol	5460391	630891	4020902
n-Tetracosanol-1	4560768	3914555	12329067
Pentanal	658123	2717570	3734693
methyl butanal	658123	3256771	3734693
Benzeneacetic acid	3548739	2396880	2396880
Propanedioic acid, phenyl-	3548739	2003590	2003590
Nonadecyl acetate	4652207	3914555	8285285
9-Eicosene, (E)-	5184794	2352816	8441486
Heptadecyl pentafluoropropionate	6448559	3984915	12862769
1-Heptadecanol, acetate	5873043	3914555	12290212
Pentafluoropropionic acid, tetradecyl ester	7306219	3984915	12239675
1-Heneicosyl formate	6907799	3984915	12239675
Heptafluorobutyric acid, n-tetradecyl ester	7084428	3984915	7334423
1,13-Tetradecadiene	2017593	855479	2358303

Phthalic acid, cycloheptyl isobutyl ester	3538647	606851.3	710714
Phthalic acid, 6-ethyl-3-octyl isobutyl ester	3247347	3551183	6448688
Azulene, 1,2,3,3a,4,5,6,7-octahydro-1,4-dimethyl-7-(1-methylethenyl)-, [1R-(1a,3aa,4a,7a)]-	919596	919596	919596
Benzeneacetonitrile, a-hydroxy-	2429840	243276	565304
1,2-Benzenedicarboxaldehyde	6116534	6879411	6116534
Carbamic acid, methyl-, 3-methylphenyl ester	10555951	4521422	11317366
Benzyl Alcohol	10620998	10444353	11382929
1-Pentanol	22204	74545	192766
1-Butanol, 3-methyl-	26644	91190	192766
2-Pentene, 3-ethyl-2-methyl-	70284.75	70284.75	70284.75
Benzoic acid, 4-methylphenyl ester	195883	543039	195883
4,4-Dimethylpent-2-enal	71682.75	71682.75	142254
Benzoic acid, 2-methylphenyl ester	387458.5	579034	195883
2-Cyclohexene-1,4-dione, 5,6-dibromo-2,6-dimethyl-, 1-oxime, o-benzoyl-	71178	67442	71178
11-Tetradecen-1-ol, acetate, (Z)-	442552	442552	442552
Pentanediamide, N,N'-di-benzoyloxy-	116026	418363	116026
Benzoic acid 5-methyl-2-phenyl-2H-pyrazol-3-yl ester	74914	418363	74914
Benzoic acid, (4-benzoyloxy-2-chlorophenyl) ester	116026	418363	151208
Dodecane, 2,6,10-trimethyl-	407912	407912	407912
Benzene, 1-butenyl-, (E)-	626130.5	626130.5	626130.5
Dodecane, 4,6-dimethyl-	263502	68753	263502
4-Hydroxy-2-methylacetophenone	775667	775667	1244359
trans-11-Tetradecenyl acetate	783817.5	855479	783817.5
E-7-Tetradecen-1-ol	2230557	2230557	2358303
4-Hydroxy-3-methylacetophenone	481655.5	481655.5	1244359
Methanol, oxo-, benzoate	244756	1292387	1202559
5-Nonadecen-1-ol	1171173	1171173	392550
4-Nitrosophenyl-a-phenylpropionate	879935.5	879935.5	1083461
Ammonium acetate	1974023	439667	3322096
Benzenepropanoic acid, silver(1+) salt	676410	676410	1083461
Benzenepropanoic acid	561790.5	561790.5	3193100
Benzeneacetaldehyde, a-ethylidene-	2105738	3334560	8685215
Acetic acid	2159001	439667	3438348
2-Propenal, 2-methyl-3-phenyl-	308328	3334560	8685215
Isophthalaldehyde	20324	20324	20324
1,4-Benzenedicarboxaldehyde	35028.75	35028.75	35028.75
Spiro[2.4]hepta-4,6-diene	59914	104514	138674
Cyclobutene, 2-propenylidene-	18029	65586.5	138674
Butanetetrone, diphenyl-, 2,3-dioxime	20845	67442	63525

Ethanone, 1-(6-methyl-3-pyridinyl)-	527351	1406360	527351
Cyclopropane, 1-methyl-2-octyl-	189751	815964	189751
Pentafluoropropionic acid, decyl ester	3411	3411	3411
11-Dodecenol	143097	143097	143097
E-9-Tetradecenal	199212	199212	199212
1,14-Tetradecanediol	1004344	1004344	1004344
9-Octadecen-1-ol, (E)-	792602	855479	2505415
Pentadecane, 7-methyl-	1257310	1338252	1257310
Naphthalene, 1,2,4a,5,8,8a-hexahydro-4, 7-dimethyl-1-(1-methylethyl)-, [1S-(1a,4aa,8aa)]-	831559	831559	831559
Naphthalene, 1,2,3,5,6,8a-hexahydro-4, 7-dimethyl-1-(1-methylethyl)-, (1S-cis)-	990326	990326	990326
Acetophenone, 4'-amino-	426971.3	1406360	426971.3
6-Tetradecene, (E)-	531795.5	531795.5	531795.5
1,2-Benzenediol, o-(4-methoxybezoyl)-o'-(5-chlorovaleryl)-	406094	206874	406094
3-Nitro-1-phenyl-1-butanone	210724	418363	210724
Phthalic acid, butyl tridec-2-yn-1-yl ester	759529	759529	759529
Naphthalene, 1,2,3,4,4a,5,6,8a-octahydro-4a, 8-dimethyl-2-(1-methylethenyl)-, [2R-(2a,4aa,8aa)]-	1306798	1306798	1306798
Phthalic acid, butyl hex-2-yn-4-yl ester	1082353	1082353	1082353
Phthalic acid, butyl 2-ethylcyclohexyl ester	3247347	1396895	1207909
Bicyclo[2.2.1]heptane, 2,2-dimethyl-3-methylene-, (1R)-	715378	715378	715378
E-15-Octadecacen-1-ol acetate	413539	413539	413539
Ethanone, 1-(3-aminophenyl)-	598816.8	1406360	598816.8
Bicyclo[2.2.1]heptane, 7,7-dimethyl-2-methylene-	848029	848029	848029
Z-8-Hexadecen-1-ol acetate	1117846	1117846	1117846
Cycloundecane, 1,1,2-trimethyl-	2995281	2995281	140875
1,2-Octadecanediol	681953.5	792976	681953.5
Nonadecyl trifluoroacetate	2534574	1559840	2534574
Acetic acid n-octadecyl ester	1559840	1559840	3882452
Phthalic acid, butyl non-5-yn-3-yl ester	5819074	1534934	1207909
Eicosyl trifluoroacetate	1418911	1559840	1418911
Heptafluorobutyric acid, n-octadecyl ester	2951219	3191939	3882452
Octadecyl pentafluoropropionate	2071221	1559840	4210220
Cyclopentadecanol	458210	2358303	2358303
Heptadecane	4039786	5088837	3496384
Octadecane	4600020	5088837	6930444
Tetradecane	118462	1874261	1376297
Sulfur dioxide	36136	77272	33150.5
3-Buten-1-one, 2,2-dimethyl-1-phenyl-	48192.75	67442	48192.75
Aminomethanesulfonic acid	36136	77272	30165

2-Fluorobenzoic acid, 3,4-dichlorophenyl ester	30112	29356.5	29356.5
2-Fluorobenzoic acid, 4-nitrophenyl ester	30112	30112	30112
Undecane	96497	68753	96497
Decane	20294.75	20294.75	20294.75
1-Eicosyne	366729.8	366729.8	366729.8
Ethanone, 2-chloro-1,2-diphenyl-	116026	418363	75640.5
Benzoic acid, anhydride	185942	418363	185942
6-Benzamido-4-benzoyl-1,2,4-triazine-3,5(2H,4H)-dione	95181	975752	147665.5
2-Ethylacrolein	32801	233839	286316
2-(2-Oxo-2-phenyl-ethyl)-malononitrile	176137	176137	176137
2-Butenal, 3-methyl-	166234.3	233839	414227
N-Benzyloxy-2,2-bis(trifluoromethyl)aziridine	116026	418363	182845.5
Dodecane	280721	68753	280721
N-(1H-Tetrazol-5-yl)benzamide	173212	975752	173212
1-Ethyl-2-phenylpyrazolium bromide	240196.5	975752	240196.5
Cyclobutyl phenyl ketone	67442	67442	67442
1-Propanone, 1-phenyl-	176137	418363	176137
E-7-Tetradecenol	720025	720025	720025
Naphthalene, 1,2,3,4,4a,7-hexahydro-1,6-dimethyl-4-(1-methylethyl)-	330676	330676	330676
9-Eicosyne	357794	357794	357794
Butyric acid, octadecyl ester	291475	291475	291475
Butyric acid, pentadecyl ester	268518	268518	374020
Pentafluoropropionic acid, undecyl ester	939347	792976	939347
(2-Aziridinylethyl)amine	33539	749136	768315
Phenol, 4-[2-(methylamino)ethyl]-	605335.8	7111592	769733
8-Dodecen-1-ol, (Z)-	1761999	1761999	2942541
E-11(13-Methyl)tetradecen-1-ol acetate	4636419	4636419	4636419
8-Heptadecene	4780840	1639448	8491219
2-Methoxy-4-vinylphenol	895028.8	895028.8	2029427
Dodecane, 2,7,10-trimethyl-	191462.3	191462.3	191462.3
Z-10-Pentadecen-1-ol acetate	2143228	2143228	2143228
naphthalene, 1,2,3,4,4a,5,6,8a-octahydro-7-methyl-4-methylene-1-(1-methylethyl)	2058312	2058312	2058312
1,9-Tetradecadiene	2015163	2015163	2358303
Ylangene	1483797	1483797	1483797
Acetic acid, chloro-, octadecyl ester	984920.3	984920.3	984920.3
13-Tetradecen-1-ol acetate	2210792	2282456	2210792
Hexadecane, 2,6,10,14-tetramethyl-	3203269	5088837	3496384
Pentanoic acid	2615298	677121	822127
Z-6-Pentadecen-1-ol acetate	4370759	4370759	4370759

1-Hexadecanethiol	4076227	3191939	3451971
Carbonic acid, hexadecyl methyl ester	4100628	4100628	4100628
Hexanoic acid	2673153	467490	373409
Pentafluoropropionic acid, dodecyl ester	4478204	1639448	7791445
Pentafluoropropionic acid, heptadecyl ester	6545243	2352816	12862769
Methylparaben	2493032	2493032	2493032
Methyl nitrite	6937.5	3506	6937.5
methyl furan	32608.75	21190	104733

APPENDIX C

At or above a cutoff score of 90%, metabolites that are common to both pig cohorts:

CAS number	Metabolite
1111780	Carbamic acid, monoammonium salt
75070	Acetaldehyde
74931	Methanethiol
64197	Acetic acid
503286	Diazene, dimethyl-
67641	Acetone
75150	Carbon disulfide
7446095	Sulfur dioxide
13881919	Aminomethanesulfonic acid
78842	Propanal, 2-methyl-
78933	2-Butanone
534225	methyl furan
141786	Ethyl Acetate
4170303	2-Butenal
590863	methyl butanal
631618	Ammonium acetate
107879	2-Pentanone
3208160	Furan, 2-ethyl-
105373	Propanoic acid, ethyl ester
624920	Disulfide, dimethyl
108883	Toluene
765468	Spiro[2.4]hepta-4,6-diene
71410	1-Pentanol
123513	1-Butanol, 3-methyl-
498602	3-Furaldehyde
503742	methyl butanoic acid
106423	p-Xylene
108383	Benzene, 1,3-dimethyl-
95476	o-Xylene
110430	2-Heptanone
100425	Styrene

629209	1,3,5,7-Cyclooctatetraene
694871	Bicyclo[4.2.0]octa-1,3,5-triene
111717	Heptanal
3268493	Propanal, 3-(methylthio)-
109524	Pentanoic acid
57266861	2-Heptenal, (Z)-
100527	Benzaldehyde
3658808	Dimethyl trisulfide
110930	5-Hepten-2-one, 6-methyl-
108952	Phenol
33795185	Phosphonic acid, (p-hydroxyphenyl)-
3777693	Furan, 2-pentyl-
124130	Octanal
4313035	2,4-Heptadienal, (E,E)-
104767	1-Hexanol, 2-ethyl-
58175578	2-Propyl-1-pentanol
1669449	3-Octen-2-one
122781	Benzeneacetaldehyde
2548870	2-Octenal, (E)-
98862	Acetophenone
25740801	S-Benzoyl(thiohydroxylamine)
106445	methyl phenol
124196	Nonanal
60128	Phenylethyl Alcohol
103797	Benzyl methyl ketone
20600548	Benzene, 1-isocyano-3-methyl-
4432637	2-Phenylpropenal
123079	Phenol, 4-ethyl-
693549	2-Decanone
65850	Benzoic acid
EPA-305652	Methanol, oxo-, benzoate
EPA-253264	Heptanediamide, N,N'-di-benzoyloxy-
91203	Naphthalene
1120065	2-Decanol
13679419	Furan, 3-phenyl-
112129	2-Undecanone
91225	Quinoline
119653	Isoquinoline
1885387	2-Propenenitrile, 3-phenyl-, (E)-
495103	Benzeneacetonitrile, à-methylene-

93107	2-Quinolinecarboxylic acid
3435516	Benzonitrile, 4-ethenyl-
629505	Tridecane
629594	Tetradecane
1120214	Undecane
120729	indole
1450722	Ethanone, 1-(2-hydroxy-5-methylphenyl)-
2021285	Benzenepropanoic acid, ethyl ester
501520	Benzenepropanoic acid
334485	n-Decanoic acid
148243	8-Hydroxyquinoline
59314	2(1H)-Quinolinone
121335	Vanillin
127413	3-Buten-2-one, 4-(2,6,6-trimethyl-2-cyclohexen-1-yl)-, (E)-
124254	Tetradecanal
629801	Hexadecanal
23676097	Benzoic acid, 4-ethoxy-, ethyl ester
36653824	1-Hexadecanol
112709	n-Tridecan-1-ol
629765	n-Pentadecanol
4727177	Cyclopentadecanol
55153123	Ethanone, 2-(formyloxy)-1-phenyl-
769788	Vinyl benzoate
136367	1,3-Benzenediol, monobenzoate
15206550	Benzeneacetic acid, à-oxo-, methyl ester
532558	Benzoyl isothiocyanate
5062306	Phenacylidene diacetate
2114003	1-Propanone, 2-bromo-1-phenyl-
67714345	3-(Benzoylthio)-2-methylpropanoic acid
93992	Benzoic acid, phenyl ester
643754	Propanetrione, diphenyl-
134816	Ethanedione, diphenyl-
582241	Ethanone, 2-hydroxy-1-phenyl-
544-63-8	Tetradecanoic acid
1002842	Pentadecanoic acid
84695	1,2-Benzenedicarboxylic acid, bis(2-methylpropyl) ester
EPA-315476	Phthalic acid, butyl 2-pentyl ester
17851535	1,2-Benzenedicarboxylic acid, butyl 2-methylpropyl ester
84640	1,2-Benzenedicarboxylic acid, butyl cyclohexyl ester
84742	Dibutyl phthalate

2423101	octadecenal
56219046	hexadecenal
112-39-0	Hexadecanoic acid, methyl ester
82304663	7,9-Di-tert-butyl-1-oxaspiro(4,5)deca-6,9-diene-2,8-dione
57103	n-Hexadecanoic acid
628-97-7	Hexadecanoic acid, ethyl ester
4376209	1,2-Benzenedicarboxylic acid, mono(2-ethylhexyl) ester
27554263	1,2-Benzenedicarboxylic acid, diisooctyl ester
117840	Di-n-octyl phthalate
117817	Bis(2-ethylhexyl) phthalate
64175	Ethanol
79312	Propanoic acid, 2-methyl-
105668	Butanoic acid, propyl ester
3391864	1-Octen-3-ol
1125219	2,6,6-Trimethyl-2-cyclohexene-1,4-dione
124072	Octanoic Acid
116267	1,3-Cyclohexadiene-1-carboxaldehyde, 2,6,6-trimethyl-
EPA-307691	2-Fluorobenzoic acid, 4-nitrophenyl ester
638539	Tridecanoic acid
64437474	Hexadecen-1-ol, trans-9-
112925	1-Octadecanol
13360617	1-Pentadecene
1454859	n-Heptadecanol-1
68797955	(Z)6-Pentadecen-1-ol
1454848	n-Nonadecanol-1
56797401	7-Hexadecenal, (Z)-
103822	Benzeneacetic acid
2613890	Propanedioic acid, phenyl-
143077	Dodecanoic acid
EPA-351744	Pentadecyl trifluoroacetate
629583	1-Pentadecanol acetate
53800025	Trifluoroacetic acid,n-tridecyl ester
57-11-4	octadecanoic acid
7782798	Hydrogen azide
71238	1-Propanol
109604	n-Propyl acetate
96548	1H-Pyrrole, 1-methyl-
66251	Hexanal
141060	Pentanoic acid, propyl ester
636419	1H-Pyrrole, 2-methyl-

551939	Ethanone, 1-(2-aminophenyl)-
42972463	5-Acetyl-2-methylpyridine
99923	Acetophenone, 4'-amino-
96764	Phenol, 2,4-bis(1,1-dimethylethyl)-
1075065	Ethanone, 2,2-dihydroxy-1-phenyl-
119619	Benzophenone
71432	Benzene
502692	2-Pentadecanone, 6,10,14-trimethyl-
110861	Pyridine
5399304	Phenacyl thiocyanate
109999	tetrahydrofuran

REFERENCES

(1) Brown, M.; Dunn, W. B.; Ellis, D. I.; Goodacre, R.; Handl, J.; Knowles, J. D.; O'Hagan, S.; Spasić, I.; Kell, D. B. A Metabolome Pipeline: From Concept to Data to Knowledge. *Metabolomics* **2005**, *1*, 39–51.

(2) Abubucker, S.; Segata, N.; Goll, J.; Schubert, A. M.; Izard, J.; Cantarel, B.; Rodriguez-Mueller, B.; Zucker, J.; Thiagarajan, M.; Henrissat, B.; et al. Metabolic Reconstruction for Metagenomic Data and Its Application to the Human Microbiome. *PLoS Comput Biol* **2012**, *8*.

(3) Wishart, D. S. Computational Approaches to Metabolomics. In *Bioinfomatics Methods in Clinical Researh*; Humana Press, 2010; Vol. 593, pp. 283–313.

(4) Wishart, D. S. Metabolomics: Applications to Food Science and Nutrition Research. *Trends Food Sci. Technol.* **2008**, *19*, 482–493.

(5) Ghannoum, M. A.; Jurevic, R. J.; Mukherjee, P. K.; Cui, F.; Sikaroodi, M.; Naqvi, A.; Gillevet, P. M. Characterization of the Oral Fungal Microbiome (Mycobiome) in Healthy Individuals. *PLoS Pathog.* **2010**, *6*, e1000713.

(6) Vinayavekhin, N.; Homan, E. A.; Saghatelian, A. Exploring Disease through Metabolomics. *ACS Chem. Biol.* **2010**, *5*, 91–103.

(7) Mamas, M.; Dunn, W. B.; Neyses, L.; Goodacre, R. The Role of Metabolites and Metabolomics in Clinically Applicable Biomarkers of Disease. *Arch. Toxicol.* **2010**, *85*, 5–17.

(8) Dixon, E.; Clubb, C.; Pittman, S.; Ammann, L.; Rasheed, Z.; Kazmi, N.; Keshavarzian, A.; Gillevet, P.; Rangwala, H.; Couch, R. D. Solid-Phase Microextraction and the Human Fecal VOC Metabolome. *PLoS ONE* **2011**, *6*, e18471.

(9) Yang, Z.; Kobayashi, E.; Katsuno, T.; Asanuma, T.; Fujimori, T.; Ishikawa, T.; Tomomura, M.; Mochizuki, K.; Watase, T.; Nakamura, Y.; et al. Characterisation of Volatile and Non-Volatile Metabolites in Etiolated Leaves of Tea (Camellia Sinensis) Plants in the Dark. *Food Chem.* **2012**, *135*, 2268–2276.

(10) Couch, R. D.; Navarro, K.; Sikaroodi, M.; Gillevet, P.; Forsyth, C. B.; Mutlu, E.; Engen, P. A.; Keshavarzian, A. The Approach to Sample Acquisition and Its Impact on the Derived Human Fecal Microbiome and VOC Metabolome. *PLoS ONE* **2013**, *8*, e81163.

(11) Yin, P.; Peter, A.; Franken, H.; Zhao, X.; Neukamm, S. S.; Rosenbaum, L.; Lucio, M.; Zell, A.; Haring, H.-U.; Xu, G.; et al. Preanalytical Aspects and Sample Quality Assessment in Metabolomics Studies of Human Blood. *Clin. Chem.* **2013**, *59*, 833–845.

(12) Liu, X.; Wu, H.; Ji, C.; Wei, L.; Zhao, J.; Yu, J. An Integrated Proteomic and Metabolomic Study on the Chronic Effects of Mercury in Suaeda Salsa under an Environmentally Relevant Salinity. *PLoS ONE* **2013**, *8*, e64041.

(13) Arthur, C. L.; Pawliszyn, J. Solid Phase Microextraction with Thermal Desorption Using Fused Silica Optical Fibers. *Anal. Chem.* **1990**, *62*, 2145–2148.

(14) Beleggia, R.; Platani, C.; Papa, R.; Di Chio, A.; Barros, E.; Mashaba, C.; Wirth, J.; Fammartino, A.; Sautter, C.; Conner, S.; et al. Metabolomics and Food Processing: From Semolina to Pasta. *J. Agric. Food Chem.* **2011**, *59*, 9366–9377.

(15) Lee, J.-E.; Hwang, G.-S.; Lee, C.-H.; Hong, Y.-S. Metabolomics Reveals Alterations in Both Primary and Secondary Metabolites by Wine Bacteria. *J. Agric. Food Chem.* **2009**, *57*, 10772–10783.

(16) Chong, S. L.; Wang, D.; Hayes, J. D.; Wilhite, B. W.; Malik, A. Sol-Gel Coating Technology for the Preparation of Solid-Phase Microextraction Fibers of Enhanced Thermal Stability. *Anal. Chem.* **1997**, *69*, 3889–3898.

(17) Van de Kant, K. D.; van der Sande, L. J.; Jöbsis, Q.; van Schayck, O. C.; Dompeling, E. Clinical Use of Exhaled Volatile Organic Compounds in Pulmonary Diseases: A Systematic. **2012**.

(18) Basanta, M.; Ibrahim, B.; Dockry, R.; Douce, D.; Morris, M.; Singh, D.; Woodcock, A.; Fowler, S. J. Exhaled Volatile Organic Compounds for Phenotyping Chronic Obstructive Pulmonary Disease: A Cross-Sectional Study. *Respir. Res.* **2012**, *13*, 72.

(19) Font, X.; Artola, A.; Sánchez, A. Detection, Composition and Treatment of Volatile Organic Compounds from Waste Treatment Plants. *Sensors* **2011**, *11*, 4043–4059.

(20) Dam, N. M.; Qiu, B.-L.; Hordijk, C. A.; Vet, L. E. M.; Jansen, J. J. Identification of Biologically Relevant Compounds in Aboveground and Belowground Induced Volatile Blends. *J. Chem. Ecol.* **2010**, *36*, 1006–1016.

(21) Lozano, J.; Santos, J. P.; Gutiérrez, J.; Horrillo, M. C. Comparison among Several Fiber Coating for a Solid Phase Microextraction (SPME) Based Electronic Nose Applied to Wine Discrimination. In *Sensors, 2005 IEEE*; 2005; pp. 1339–1342.

(22) Li, R. W.; Wu, S.; Li, W.; Navarro, K.; Couch, R. D.; Hill, D.; Urban, J. F. Alterations in the Porcine Colon Microbiota Induced by the Gastrointestinal Nematode Trichuris Suis. *Infect. Immun.* **2012**, *80*, 2150–2157.

(23) Weingart, G.; Kluger, B.; Forneck, A.; Krska, R.; Schuhmacher, R. Establishment and Application of a Metabolomics Workflow for Identification and Profiling of Volatiles from Leaves of Vitis Vinifera by HS-SPME-GC-MS. *Phytochem. Anal.* **2012**, *23*, 345–358.

(24) Raman, M.; Ahmed, I.; Gillevet, P. M.; Probert, C. S.; Ratcliffe, N. M.; Smith, S.; Greenwood, R.; Sikaroodi, M.; Lam, V.; Crotty, P.; et al. Fecal Microbiome and Volatile Organic Compound Metabolome in Obese Humans With Nonalcoholic Fatty Liver Disease. *Clin. Gastroenterol. Hepatol.* **2013**, *11*, 868–875.e3.

(25) Alcohol Alert #64. http://pubs.niaaa.nih.gov/publications/aa64/aa64.htm (accessed Jul 17, 2011).

(26) Hudak, C. M.; Gallo, B. M.; Morton, P. G. *Critical Care Nursing: A Holistic Approach*; 7th ed.; Lippincott: Philadelphia, 1998.

(27) Keshavarzian, A.; Holmes, E. W.; Patel, M.; Iber, F.; Fields, J. Z.; Pethkar, S. Leaky Gut in Alcoholic Cirrhosis: A Possible Mechanism for Alcohol-Induced Liver Damage. *Am. J. Gastroenterol.* **1999**, *94*, 200–207.

(28) FASTSTATS - Alcohol Use. http://www.cdc.gov/nchs/fastats/alcohol.htm (accessed Jul 22, 2011).

(29) Statistics on Alcoholics. http://www.alcoholics-info.com/Statistics_on_Alcoholics.html (accessed Feb 3, 2012).

(30) Alcoholism and alcohol abuse. http://www.nlm.nih.gov/medlineplus/ency/article/000944.htm (accessed Jul 19, 2011).

(31) Garner, C. E.; Smith, S.; de Lacy Costello, B.; White, P.; Spencer, R.; Probert, C. S. J.; Ratcliffe, N. M. Volatile Organic Compounds from Feces and Their Potential for Diagnosis of Gastrointestinal Disease. *FASEB J.* **2007**, *21*, 1675–1688.

(32) Gramenzi, A.; Caputo, F.; Biselli, M.; Kuria, F.; Loggi, E.; Andreone, P.; Bernardi, M. Review Article: Alcoholic Liver Disease - Pathophysiological Aspects and Risk Factors. *Aliment. Pharmacol. Ther.* **2006**, *24*, 1151–1161.

(33) Seth, D.; Haber, P. S.; Syn, W.-K.; Diehl, A. M.; Day, C. P. Pathogenesis of Alcohol-Induced Liver Disease: Classical Concepts and Recent Advances. *J. Gastroenterol. Hepatol.* **2011**, *26*, 1089–1105.

(34) Marsano, L. S.; Mendez, C.; Hill, D.; Barve, S.; McClain, C. J. Diagnosis and Treatment of Alcoholic Liver Disease and Its Complications. *Alcohol Res. Health* **2003**, *27*, 247–256.

(35) Alcoholic liver disease - Penn State Hershey Medical Center. http://pennstatehershey.adam.com/content.aspx?productId=117&pid=1&gid=000281 (accessed Aug 6, 2013).

(36) Maher, J. J. Exploring Alcohol's Effects on Liver Function. *Alcohol Health Res. World* **1997**, *21*, 5–12.

(37) Cirrhosis - National Digestive Diseases Information Clearinghouse. http://digestive.niddk.nih.gov/ddiseases/pubs/cirrhosis/#what (accessed Jul 19, 2011).

(38) Steinhoff, U. Who Controls the Crowd? New Findings and Old Questions about the Intestinal Microflora. *Immunol. Lett.* **2005**, *99*, 12–16.

(39) Sears, C. L. A Dynamic Partnership: Celebrating Our Gut Flora. *Anaerobe* **2005**, *11*, 247–251.

(40) Wheeler, M. Endotoxin and Kupffer Cell Activation in Alcoholic Liver Disease. http://pubs.niaaa.nih.gov/publications/arh27-4/300-306.htm (accessed Jul 20, 2011).

(41) Su, G. L.; Klein, R. D.; Aminlari, A.; Zhang, H. Y.; Steinstraesser, L.; Alarcon, W. H.; Remick, D. G.; Wang, S. C. Kupffer Cell Activation by Lipopolysaccharide in Rats: Role for Lipopolysaccharide Binding Protein and Toll-like Receptor 4. *Hepatology* **2000**, *31*, 932–936.

(42) Haber, P. S.; Warner, R.; Seth, D.; Gorrell, M. D.; Mccaughan, G. W. Pathogenesis and Management of Alcoholic Hepatitis. *J. Gastroenterol. Hepatol.* **2003**, *18*, 1332–1344.

(43) Dinarello, C. A. Proinflammatory Cytokines. *CHEST J.* **2000**, *118*, 503–508.

(44) Gutsmann, T.; Muller, M.; Carroll, S. F.; MacKenzie, R. C.; Wiese, A.; Seydel, U. Dual Role of Lipopolysaccharide (LPS)-Binding Protein in Neutralization of LPS and Enhancement of LPS-Induced Activation of Mononuclear Cells. *Infect. Immun.* **2001**, *69*, 6942–6950.

(45) Levine, B. *Principles of Forensic Toxicology*; AACCPress: Washington, D.C., 2010.

(46) Assimakopoulos, S. F. Pentoxifylline: A First Line Treatment Option for Severe Alcoholic Hepatitis and Hepatorenal Syndrome? *World J. Gastroenterol.* **2009**, *15*, 3194.

(47) Nishimura, M.; Teschke, R. Alcohol and Gamma-Glutamyltransferase. *Klin. Wochenschr.* **1983**, *61*, 265–275.

(48) Devaraj, S.; Chu, V. H. Aspartate Aminotransferase http://emedicine.medscape.com/article/2087224-overview#a30 (accessed Feb 16, 2014).

(49) Orlewicz, M. S.; Vovchuk, E. Alanine Aminotransferase http://emedicine.medscape.com/article/2087247-overview#a30 (accessed Feb 16, 2014).

(50) French, T.; Blue, J.; Stokol, T. Alanine amiotransferase - ALT https://ahdc.vet.cornell.edu/clinpath/modules/chem/alt.htm (accessed Feb 16, 2014).

(51) French, T.; Blue, J.; Stokol, T. Aspartate aminotransferase - AST https://ahdc.vet.cornell.edu/clinpath/modules/chem/ast.htm (accessed Feb 16, 2014).

(52) Nathwani, R. A.; Pais, S.; Reynolds, T. B.; Kaplowitz, N. Serum Alanine Aminotransferase in Skeletal Muscle Diseases. *Hepatology* **2005**, *41*, 380–382.

(53) Ghadban, R. Gamma-Glutamyltransferase http://emedicine.medscape.com/article/2087891-overview#a30 (accessed Feb 14, 2014).

(54) Shirey, R. E.; Mindrup, R. F. A Systematic Approach for Selecting the Appropriate SPME Fiber., 1999.

(55) Tang, B.; Isacsson, U. Analysis of Mono- and Polycyclic Aromatic Hydrocarbons Using Solid-Phase Microextraction: State-of-the-Art. *Energy Fuels* **2008**, *22*, 1425–1438.

(56) Skoog, D. A.; Holler, F. J.; Crouch, S. R. *Principles of Intrumental Analysis*; 6th ed.; Thomson Brooks/Cole: Belmont, Ca, 2007.

(57) D'Arcy, P.; Mallard, W. G. AMDIS—user Guide. *Autom. Mass Spectr. Deconvolution Identif. Syst. NIST Gaithersburg* **2004**.

(58) Mandrake, L.; Lee, S.; Bornstein, B.; Bue, B. Adapting AMDIS for Autonomous Spectral Identification of Hazardous Compounds for ISS Monitoring. In *Aerospace conference, 2009 IEEE*; 2009; pp. 1–12.

(59) Meng, C.-K.; Szelewski, M. Can "Deconvolution" Improve GC/MS Detectability?, 2010.

(60) Du, X.; Zeisel, S. H. Spectral Deconvolution for Gas Chromatography Mass Spectrometry-Based Metabolomics: Current Status and Future Perspectives. *Comput. Struct. Biotechnol. J.* **2013**, *4*.

(61) MS Solutions #22: AMDIS – Setting Up and Running a Deconvolution and Target Analysis – Part 2 - Separation Science :: Premier Learning for Analytical Chemists. http://www.sepscience.com/Information/Archive/MS-Solutions/252-/MS-Solutions-22-AMDIS--Setting-Up-and-Running-a-Deconvolution-and-Target-Analysis--Part-2 (accessed Jul 1, 2013).

(62) Steuer, R.; Morgenthal, K.; Weckwerth, W.; Selbig, J. A Gentle Guide to the Analysis of Metabolomic Data. In; Humana Press: Totowa, N.J., 2007; pp. 105–126.

(63) Bhattacharyya, G. K.; Johnson, R. A. *Statistical Concepts and Methods*; Jogn Wiley & Sons: New York, 1977.

(64) Smith, L. I. A Tutorial on Principal Components Analysis. *Cornell Univ. USA* **2002**, *51*, 52.

(65) PCA graphical explanation. http://www.cmbi.kun.nl/edu/bioinf4/prac-microarray/stats/PCA%20graphical%20explanation.htm (accessed Feb 11, 2013).

(66) Sheets, H. D.; Swiderski, D. L.; Zeldich, M. L. Principal Components Generation Utility. http://www3.canisius.edu/~sheets/PCAGenManual.pdf (accessed Feb 11, 2013).

(67) Schmidt, G.; Amman, C. Dummies Guide to the Latest" Hockey Stick" Controversy. *Online Post.* **2005**, *18*.

(68) Xia, J.; Mandal, R.; Sinelnikov, I. V.; Broadhurst, D.; Wishart, D. S. MetaboAnalyst 2.0--a Comprehensive Server for Metabolomic Data Analysis. *Nucleic Acids Res.* **2012**, *40*, W127–W133.

(69) Tobias, R. D. An Introduction to Partial Least Squares Regression. In *Proc. Ann. SAS Users Group Int. Conf., 20th, Orlando, FL*; 1995; pp. 2–5.

(70) Fattah, M. Multicollinearity. http://www.chsbs.cmich.edu/fattah/courses/empirical/multicollinearity.html (accessed Oct 30, 2013).

(71) Martínez, A. M.; Kak, A. C. PCA versus LDA. *Pattern Anal. Mach. Intell. IEEE Trans. On* **2001**, *23*, 228–233.

(72) Worley, B.; Halouska, S.; Powers, R. Utilities for Quantifying Separation in PCA/PLS-DA Scores Plots. *Anal. Biochem.* **2013**, *433*, 102–104.

(73) Tan, Y.; Shi, L.; Tong, W.; Hwang, G. T. G.; Wang, C. Multi-Class Tumor Classification by Discriminant Partial Least Squares Using Microarray Gene Expression Data and Assessment of Classification Models. *Comput. Biol. Chem.* **2004**, *28*, 235–244.

(74) PLS-DA intro. http://www.camo.com/resources/pls-da.html (accessed Oct 29, 2013).

(75) Wheelock, Å.; Wheelock, C. Multivariate Data Analysis and Modelling, 2009.

(76) Kjeldahl, K.; Bro, R. Some Common Misunderstandings in Chemometrics. *J. Chemom.* **2010**, *24*, 558–564.

(77) Cassotti, M.; Grisoni, F. *Variable Selection Methods: An Introduction.*; Theory of Statistics; Tutorial 9; University of Milano-Bicocca: Milano, Italy, 2013.

(78) SAS/STAT(R) 9.2 User's Guide, Second Edition. http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_pls_sect021.htm (accessed Oct 29, 2013).

(79) Example 51.1: Examining Model Details. http://v8doc.sas.com/sashtml/stat/chap51/sect19.htm (accessed Oct 29, 2013).

(80) Stockton, R. PERL -- Practical Extraction and Report Language. http://www.cs.cmu.edu/htbin/perl-man (accessed Jul 30, 2013).

(81) Principal Component Analysis in Excel - XLSTAT. http://www.xlstat.com/en/products-solutions/feature/principal-component-analysis-pca.html (accessed Aug 28, 2013).

(82) Kohler, U.; Luniak, M. Data Inspection Using Biplots. *Strata J.* **2005**, *5*, 208–223.

(83) Fold Change - Workbench. http://wiki.c2b2.columbia.edu/workbench/index.php/Fold_Change (accessed Oct 24, 2013).

(84) Huxtable, R. Physiological Actions of Taurine. *Physiol. Rev.* **1992**, *72*.

(85) Bouckenooghe, T.; Remacle, C.; Reusens, B. Is Taurine a Functional Nutrient? *Curr. Opin. Clin. Nutr. Metab. Care* **2006**, *9*, 728–733.

(86) Brosnan, J. T.; Brosnan, M. E. The Sulfur-Containing Amino Acids: An Overview. *J. Nutr.* **2006**, *136*, 1636S–1640S.

(87) PLS discriminant analysis with XLSTAT-PLS. http://www.xlstat.com/en/learning-center/tutorials/running-a-partial-least-square-pls-discriminant-analysis-with-xlstat-pls.html (accessed Oct 29, 2013).

(88) Azizan, K. A.; Baharum, S. N.; Ressom, H. W.; Noor, N. M. GC-MS Analysis and PLS-DA Validation of the Trimethyl Silyl-Derivatization Techniques. *Am. J. Appl. Sci.* **2012**, *9*, 1124–1136.

(89) Gould, G.; Scott, R. *The Practitioner's Medical Dictionary*; 3rd ed.; P. Blakiston's Son and Company: Philadelphia, 1919.

(90) Weinstock, J. V.; Elliott, D. E. Helminths and the IBD Hygiene Hypothesis: *Inflamm. Bowel Dis.* **2009**, *15*, 128–133.

(91) Koloski, N. A. Hygiene Hypothesis in Inflammatory Bowel Disease: A Critical Review of the Literature. *World J. Gastroenterol.* **2008**, *14*, 165.

(92) Flohr, C.; Tuyen, L.; Lewis, S.; Quinnell, R.; Minh, T.; Liem, H.; Campbell, J.; Pritchard, D.; Hien, T.; Farrar, J. Poor Sanitation and Helminth Infection Protect

against Skin Sensitization in Vietnamese Children: A Cross-Sectional Study. *J. Allergy Clin. Immunol.* **2006**, *118*, 1305–1311.

(93) Summers, R. W. Trichuris Suis Therapy in Crohn's Disease. *Gut* **2005**, *54*, 87–90.

(94) Romagnani, S. Th1/Th2 Cells. *Inflamm. Bowel Dis.* **1999**, *5*, 285–294.

(95) Campylobacter | FoodSafety.gov. http://www.foodsafety.gov/poisoning/causes/bacteriaviruses/campylobacter/ (accessed Aug 22, 2013).

(96) Wedge, D. C.; Allwood, J. W.; Dunn, W.; Vaughan, A. A.; Simpson, K.; Brown, M.; Priest, L.; Blackhall, F. H.; Whetton, A. D.; Dive, C.; et al. Is Serum or Plasma More Appropriate for Intersubject Comparisons in Metabolomic Studies? An Assessment in Patients with Small-Cell Lung Cancer. *Anal. Chem.* **2011**, *83*, 6689–6697.

(97) Liu, X.; Zhang, L.; You, L.; Wu, H.; Zhao, J.; Cong, M.; Li, F.; Wang, Q.; Li, L.; Li, C.; et al. Metabolomic Study on the Halophyte Suaeda Salsa in the Yellow River Delta. *CLEAN - Soil Air Water* **2011**, *39*, 720–727.

(98) León, Z.; García-Cañaveras, J. C.; Donato, M. T.; Lahoz, A. Mammalian Cell Metabolomics: Experimental Design and Sample Preparation: General. *ELECTROPHORESIS* **2013**, 2762–2775.

(99) Oikawa, A.; Fujita, N.; Horie, R.; Saito, K.; Tawaraya, K. Solid-Phase Extraction for Metabolomic Analysis of High-Salinity Samples by Capillary Electrophoresis-Mass Spectrometry. *J. Sep. Sci.* **2011**, *34*, 1063–1068.

(100) Dettmer, K.; Aronov, P. A.; Hammock, B. D. Mass Spectrometry-Based Metabolomics. *Mass Spectrom. Rev.* **2007**, *26*, 51–78.

(101) Shen, Q.; Dong, W.; Yang, M.; Li, L.; Cheung, H.-Y.; Zhang, Z. Lipidomic Fingerprint of Almonds (Prunus Dulcis L. Cv Nonpareil) Using TiO2 Nanoparticle Based Matrix Solid-Phase Dispersion and MALDI-TOF/MS and Its Potential in Geographical Origin Verification. *J. Agric. Food Chem.* **2013**, *61*, 7739–7748.

(102) Mu, G.; Liu, H.; Xu, L.; Tian, L.; Luan, F. Matrix Solid-Phase Dispersion Extraction and Capillary Electrophoresis Determination of Tetracycline Residues in Milk. *Food Anal. Methods* **2012**, *5*, 148–153.

(103) Bogialli, S.; Curini, R.; Di Corcia, A.; Nazzari, M.; Tamburro, D. A Simple and Rapid Assay for Analyzing Residues of Carbamate Insecticides in Vegetables and

Fruits: Hot Water Extraction Followed by Liquid Chromatography-Mass Spectrometry. *J. Agric. Food Chem.* **2004**, *52*, 665–671.

(104) Patti, G. J.; Yanes, O.; Siuzdak, G. Innovation: Metabolomics: The Apogee of the Omics Trilogy. *Nat. Rev. Mol. Cell Biol.* **2012**, *13*, 263–269.

(105) Yoshida, H.; Yamazaki, J.; Ozawa, S.; Mizukoshi, T.; Miyano, H. Advantage of LC-MS Metabolomics Methodology Targeting Hydrophilic Compounds in the Studies of Fermented Food Samples. *J. Agric. Food Chem.* **2009**, *57*, 1119–1126.

(106) Tredoux, A.; de Villiers, A.; Májek, P.; Lynen, F.; Crouch, A.; Sandra, P. Stir Bar Sorptive Extraction Combined with GC-MS Analysis and Chemometric Methods for the Classification of South African Wines According to the Volatile Composition. *J. Agric. Food Chem.* **2008**, *56*, 4286–4296.

(107) Pérez-Enciso, M.; Tenenhaus, M. Prediction of Clinical Outcome with Microarray Data: A Partial Least Squares Discriminant Analysis (PLS-DA) Approach. *Hum. Genet.* **2003**, *112*, 581–592.

(108) Szymańska, E.; Saccenti, E.; Smilde, A. K.; Westerhuis, J. A. Double-Check: Validation of Diagnostic Statistics for PLS-DA Models in Metabolomics Studies. *Metabolomics* **2011**, *8*, 3–16.

(109) Principal Component Analysis Tutorial | How to easily interpret a chart obtained from a PCA | Statistics package for Excel - XLSTAT. http://www.xlstat.com/en/learning-center/tutorials/customizing-a-pca-chart-with-xlstat-to-make-it-easier-to-interpret.html (accessed Aug 27, 2013).

(110) Abdi, H.; Williams, L. J. Principal Component Analysis. *Wiley Interdiscip. Rev. Comput. Stat.* **2010**, *2*, 433–459.

BIOGRAPHY

Karl Anthony Navarro graduated from Paul VI Catholic High School, Fairfax, Virginia, in 2006. He received his Bachelor of Science from Christopher Newport University in 2010. He was employed as a GTA at George Mason University for three years.