Computational Analysis of Flavonoid Pathways from Various Plant Species

Jordan R. Wilson

Kettering University Department of Chemistry, Biochemistry, and Chemical Engineering

Author Note

This senior thesis is submitted as partial fulfillment of the graduation requirements of Kettering University needed to obtain a Bachelor of Science in Chemistry. The conclusions and opinions expressed in this thesis are from myself and do not necessarily represent the position of Kettering University or anyone else affiliated with this culminating undergraduate experience.

Although this thesis represents the compilation of my own efforts, I would like to acknowledge and extend my sincere gratitude to the following individuals for their valuable time and assistance, without whom the completion of this thesis would not have been possible: Dr. Veronica Moorman (Assistant Professor of Chemistry and Biochemistry, Faculty Thesis Advisor), Dr. Montserrat Rabago-Smith (Associate Professor of Chemistry and Biochemistry, Faculty Thesis Co-Advisor), Kiah S. Lowe (Past Undergraduate Research Student), and Jinny L. Shows (Past Undergraduate Research Student).

Abstract

This project implemented the use of Python scripting in order to theoretically determine if certain plants can synthesize the flavonoids: catechin, epicatechin, naringenin, and eriodictyol. This data was then compared to the species of plants that were already experimentally known to produce these four flavonoids. The main objective of the project was to determine if the use of the script was a plausible method of predicting what plants can produce different types of flavonoids. Based on the data obtained from the script and through literature searching, the method was not plausible for predicting what plants could produce catechin and epicatechin, however, it was plausible for predicting which plants can produce naringenin and eriodictyol. This was based on comparisons between the two data sets obtained from the script and from literature searching. Overall, the method used was somewhat plausible form predicting flavonoid synthesis, however, further testing should be done in order to completely confirm the plausibility of the predicting flavonoid synthesis via Python scripting.

Keywords: Flavonoids, Pathways, Python, Coding

Table of Contents

Introduction 4

Conclusion and Recommendations.....11

Method.....19

Results 32

Discussion 43

References 49

Appendix 70

Computational Analysis of Flavonoid Pathways from Various Plant Species

Introduction

Nature and Purpose

The nature of this project is computational, involving the use of Python 2.7 scripts, in tandem with information obtained via literature searching. The purpose of this project was to predict if various plant species could produce a group of chemicals known as flavonoids and in which the predictions were determined using a Python script. The Python script specifically obtained the plant genes and the DNA sequences of these genes that encode for the enzymes that produce flavonoids of the flavonoid biochemical pathway that have been compiled in the Kyoto Encyclopedia of Genes and Genomes (KEGG). Using additional Python scripting, this information can be compiled into a variety of data files and formats that can be used to cross reference the genes between the various species of plants. The literature search was performed using Google Scholar and Kettering University's Library database and was used to obtain data to compare to the results obtained from the script and to provide insight on what was already known. Studying the genetics of the biochemical pathways related to flavonoids and flavonoid derivatives as well as their evolutionary origins will help future studies in determining which plants to use for future research based on the similarities and differences of the genes that encode enzymes that catalyze the biochemical reactions that produce flavonoids. If these methods prove plausible, they would help determine which plants use for future studies on different flavonoid content for a variety of plant species.

Background

Flavonoids. Flavonoids are a group of small organic molecules that have a variable phenolic structure and they are naturally occurring compounds that are found in almost every part of a plant and were first discovered in the 1930s (Kumar & Pandey, 2013). They all have a skeletal structure that generally contains three different rings connected to each other, two of which are generally phenolic by nature. Kumar and Pandey (2013) noted flavonoids to have strong antioxidant properties which prevent the formation of reactive oxygen species (ROS) and have been noted to have many other medicinal purposes; these properties make flavonoids a worthwhile chemical to study. They are broken up into six major subgroups depending on differences in functional groups and regioselectivity and their general structures can be seen in the figure below (Figure 1).

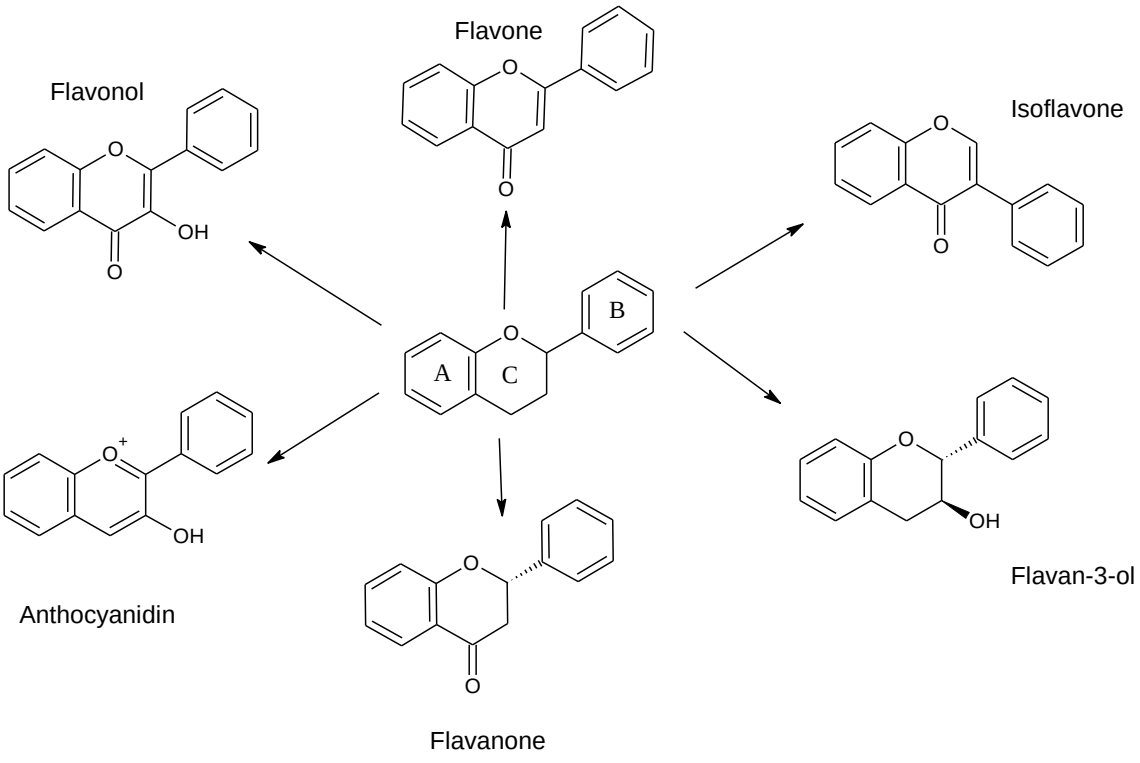

Figure 1. General structures of major groups of flavonoids.

These six major groups of flavonoids are flavonols, flavones, isoflavones, flavan-3-ols, flavanones, and anthocyanidins. They have been noted to be found in many different

species plants where they often appear as pigments in the leaves and flowers (Kumar, 2018). However, they are not always pigments and can have other properties such as helping protect plants against herbivory (Kumar, 2018). These chemicals have also been distinguished to have a wide variety of medicinal uses besides as antioxidants; they have been noted to be used as anti-mutagenics, anti-inflammatories, anti-carcinogenics, etc. (Panche, Diwan, & Chandra, 2016) Like all biological chemicals, flavonoids are made using biochemical pathways with each step catalyzed by an enzyme which in turn is encoded in the organism's genome.

Genes, Genomes, and Proteins. A genome is the complete set of genetic instructions in an organism which dictates (for the most part) how the organism will develop and grow throughout its life span. A gene is a unique sequence of nucleotides that acts as a functional unit of inheritance in an organism and typically code for proteins (Voet, Voet, & Pratt, 2016). Genes make up a portion of a genome but does not make it up entirely (Slack, 2014). Proteins are macromolecules that consist of one or more polypeptide chain (Voet, Voet, & Pratt, 2016). Proteins are encoded by genes and are synthesized based on them. In the case of this project, the proteins of interest are more specifically enzymes, meaning that the genes being examined encode specifically for enzymes that are involved in flavonoid biosynthesis.

Phylogenetic Trees. Phylogenetic trees are diagrams that display evolutionary relationships among individuals, species, or across multiple species (Saitou, 2013). They often can be made using information as simple as the taxonomy or scientific classification of organisms being compared, to things as complex as the DNA sequences of various genes that these organisms may or may not have in common with each other.

This will allow for a better understanding of how certain organisms evolved to perform certain biological functions or not. These trees display nodes in which speciation events have likely occurred and completely new species have been developed (Saitou, 2013). A simple phylogenetic tree can be seen in Figure 2, displaying several basic phylogenetic trees with several nodes, the nodes representing different speciation events that have occurred.

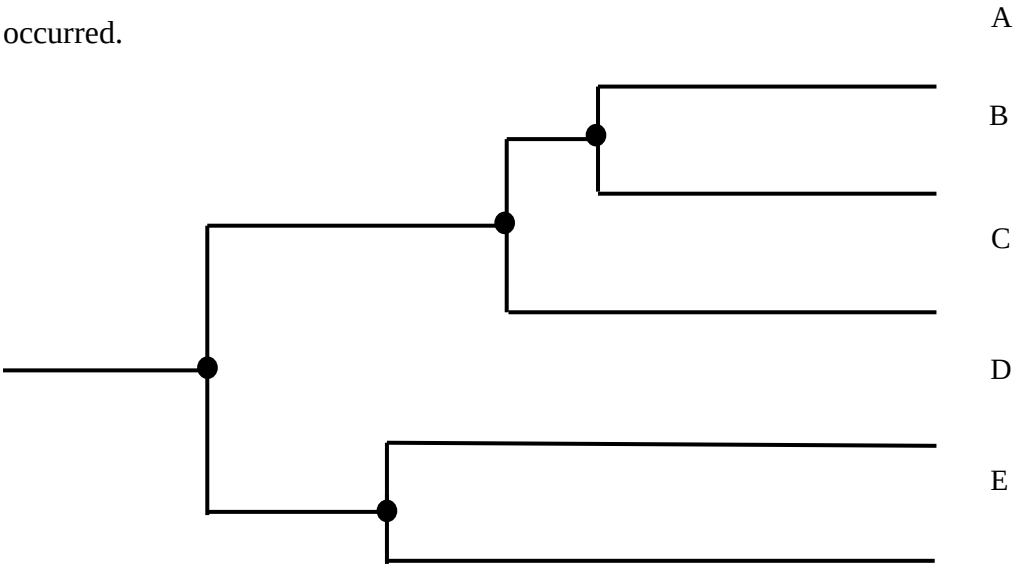

Figure 2. Simple phylogenetic tree with several nodes.

The main reason these trees were made for this project was to make comparison between different species of plants, to see the relationship between these various species of plants and if it directly relates to these plants' abilities to synthesize flavonoids.

Enzymes and EC Numbers. Enzymes are a group of protein-based biological catalysts that help catalyze various biochemical reactions within organisms. A catalyst is a molecule that promotes a chemical reaction to occur by lowering the activation energy required but does not undergo any permanent change itself on the completion of said reaction. They help reactions that generally would not occur fast enough and are essential for most biological functions of all organisms. Enzymes can be classified into seven

major classes based on what type of reaction they catalyze: oxidoreductases, transferases, hydrolases, lyases, isomerases, ligases, and translocases. Oxidoreductases are enzymes that catalyze the oxidation-reduction reactions. Transferases catalyze reactions that involve the transfer of functional groups between two different molecules. Hydrolases catalyze hydrolysis reactions, where hydrolysis reactions are the addition of a molecule of water in order to cleave a bond. Lyases catalyze reactions that involve group eliminations to form double bonds. Isomerases catalyze isomerization reaction, where isomerization is the transformation of one molecule to another in which the new molecule has the same molecular formula, but the groups are arranged in a different order. Ligases catalyze reactions that involve bond formation coupled with the hydrolysis of a molecule of adenosine triphosphate (ATP). Finally, translocases catalyze the transfer of an entire molecule across a membrane (Voet, Voet, & Pratt, 2016). These seven major classes help organize enzymes based on what reactions they help perform and are key features for enzyme commission numbers.

Enzyme commission (EC) numbers are a series of four numbers separated by periods that represent enzymes. The first number corresponds to one of the seven major classes of enzymes: oxidoreductases are number one, transferases are number two, hydrolases are number three, lyases are number four, isomerases are number five, ligases are number six, and finally translocases are number seven. The second and third number correspond to the subclass and the sub-subclass and help specify what functional groups or bonds are involved (Voet, Voet, & Pratt, 2016). The final number theoretically acts as the serial identifier which specifies what substrates and cofactors are involved (Dunn, M). EC numbers are helpful because they provide a classification of enzymes based on their

functions and what they react with as well as giving information on how different enzymes can perform the same functions.

Biochemical Pathways. A biochemical pathway is a series of enzyme-mediated reactions that produces a specific product (Voet, Voet, & Pratt, 2016). These functions can range from essential for the growth and development of an organism to nonessential functions. The chemicals involved in these reactions and functions are called metabolites (Voet, Voet, & Pratt, 2016). These metabolites are essential for the many biological processes that occur within organisms; at the same time, there are a group of metabolites that do not directly partake in the growth, development, and reproduction of organism. These metabolites are known as secondary metabolites, and they help organisms perform their various biological functions and they are generally derivatives of primary metabolites (Kumar, 2018). Flavonoids are one type of those secondary metabolites and have been noted to have a variety of uses.

Kyoto Encyclopedia of Genes and Genomes (KEGG). KEGG is a freely available online database that is sponsored by the Kanehisa Laboratories in the Institute of Chemical Research of Kyoto University and can be found at https://www.genome.jp/kegg/. The database contains information regarding the makeup of genes, proteins, and chemical substances in a set of well-known organisms. The database can also use this information in tandem with knowledge of various biological systems including biochemical pathways. This then allows for anyone to gain a basic understanding of the various complex biological systems, systems such as biochemical pathways, of many different organisms as well as giving practical applications that can be used for and in benefit for our societies (Kanehisa, 2019). KEGG contains information that includes various

metabolic pathways, genes and genomes, and relevant biological chemicals of many different animals, plants, and bacteria. Information regarding plant biosynthesis of phenylpropanoids, flavonoids, isoflavonoids, flavones and flavonols, and stilbenoids were investigated for this project, specifically every plant that had a flavonoid pathway entry was investigated, which in total was one hundred species of plants. These pathways were investigated since they synthesized the six major classes of flavonoids.

***Flavonoid Pathway*.** The flavonoid pathway, which is a biochemical pathway, is a complex series of reactions that occurs to create the flavonoid products. Flavonoids are often used as precursors to other reactions that occur such as leading into entirely different pathways. Some of these pathways include, isoflavonoid biosynthesis, flavone and flavonol biosynthesis, and anthocyanin biosynthesis (Kanehisa, 2019). The precursors that lead into flavonoid biosynthesis start from the phenylpropanoid pathway. (Kanehisa, 2019) The generalized reference pathway for flavonoid biosynthesis can be seen in Figure 3. This pathway shows the various chemicals that are formed and are used to synthesize a series of varying intermediates and products, these intermediates and products are represented by a circle with corresponding name above said circle. Some of these compounds connect to other biochemicals pathways such as the flavone and flavonol biochemical pathway. Each reaction is catalyzed by an enzyme represented by an EC number which is contained within a box that are between each intermediate in the pathway.

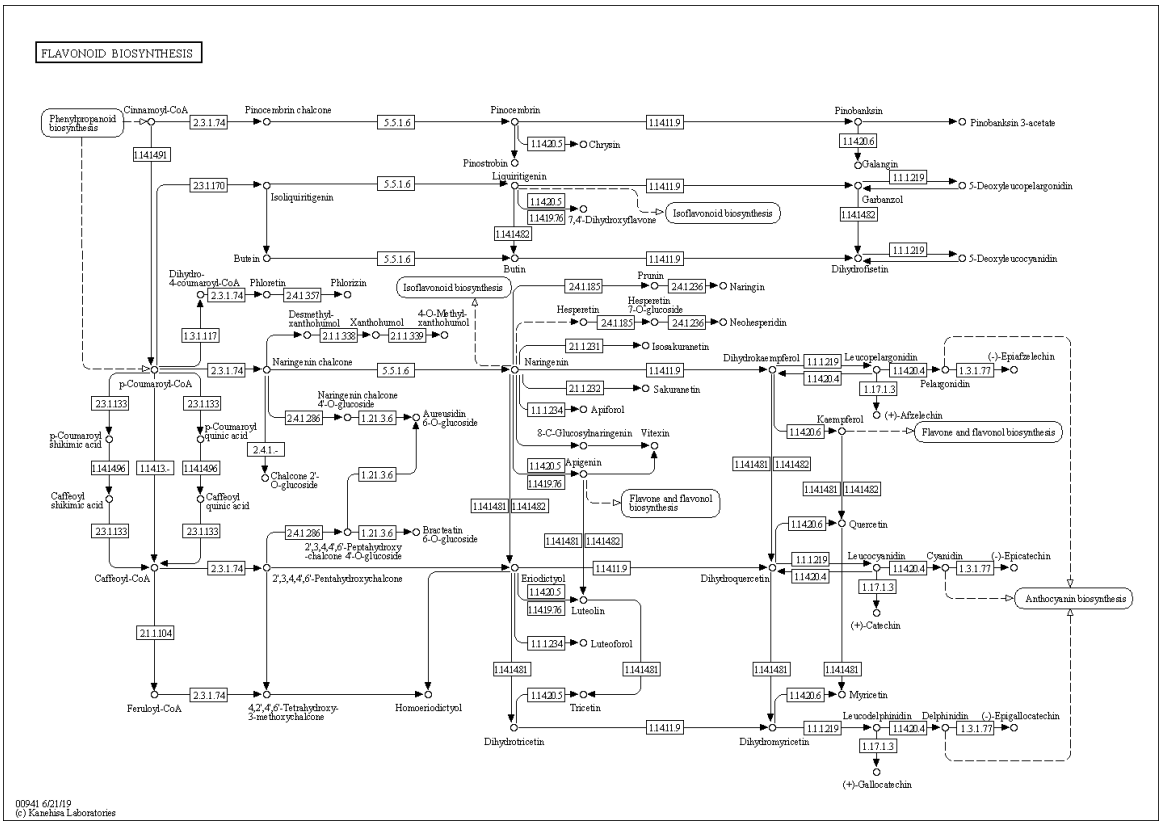

Figure 3. General pathway for flavonoid biosynthesis (Kanehisa, 2019). This pathway is a generalization and may not be the same for all species.

These pathways are all available to see in a freely available online database.

Python Scripting. Python is a high-level computer programming language that was used for this project and a script was written using Python 2.7 in a program known as Spyder (version 3.3.6), in order to access information from the KEGG database. A script is a series of commands executed by a program. A Python add-on package known as Bioservices was used in addition to the main script in order to access the KEGG database; the add-on developed by Thomas Cokelaer, Dennis Pultz, Lea M. Harder, Jordi Serra-Musach, and Julio Saez-Rodriguez.

Conclusions and Recommendations

Conclusions

KEGG Database. Many of the plant species that were analyzed from the KEGG database can theoretically synthesize at least two of the four major flavonoids that were studied for this project; the four major flavonoids that were studied included: catechin, epicatechin, naringenin, and eriodictyol. The plants that could synthesize only two of the four flavonoids generally could only synthesize naringenin and eriodictyol. The majority of plants could produce the two flavanones eriodictyol and naringenin which indicates that these two flavonoids are very important components of flavonoid biosynthesis. This can also be seen in the flavonoid pathway itself since naringenin and eriodictyol are key precursors to many different reactions in the KEGG pathway. Plants that could produce three of the flavonoids generally could synthesize naringenin and eriodictyol and one of the flavan-3-ols, catechin or epicatechin. Finally, many of the plant species that could not produce any of the flavonoids were generally species of plankton, which helps indicate that flavonoids are prominent in plants that have flowers or leaves that have some form of pigment since a majority of plants are from the Magnoliopsida class of plants which are all flowering plants.

Literature Search. The information obtained from the literature search largely revolved around the studies of catechin, epicatechin, naringenin, and eriodictyol. Of the one hundred plant species that were reviewed in this literature search, it was determined that only fifty-three of them were experimentally determined to produce at least one of the four flavonoids of interest, catechin, epicatechin, naringenin, or eriodictyol. Of those fifty-three, twelve of them were known to produce either of the flavan-3-ols, catechin and epicatechin, or both of them. Seven of the plant species could produce one of the flavanones, naringenin or eriodictyol. Nine of the plant species could produce one of the

flavan-3-ols and one of the flavanones. Sixteen plant species could produce both of the flavan-3-ols and either of the flavanones. Finally, eight species could produce all four of the flavonoids. None of the plants examined were experimentally known to produce only naringenin and eriodictyol, which should be noted since the majority of the data obtained from the KEGG database showed that many plants could synthesize these two flavonoids.

Comparison of Data Sets and Implications

Catechin. Comparisons between the two data sets from the plants that were theoretically determined to produce catechin as determined by KEGG and those that were already experimentally known to produce catechin as determined from literature searches can be seen in Figure 4.

Figure 4. Comparison of data sets for catechin. The red circle contains species experimentally known to produce catechin and those in blue are for those determined to theoretically produce it.

This diagram showed that thirty-four of the plant species fell within both data sets. Ten of the species of plants fell within the experimentally known set but not within the data set that contained the plants that were theoretically determined to produce catechin. At the same time twenty-three species were theoretically determined to produce catechin but were not experimentally known to produce it. This implies that the script that was written to determine what plants could theoretically produce catechin is not a plausible method since the data set produced by the script is missing species that are already experimentally known to synthesize catechin.

Epicatechin. Comparisons of the two data sets for epicatechin can be seen in Figure 5.

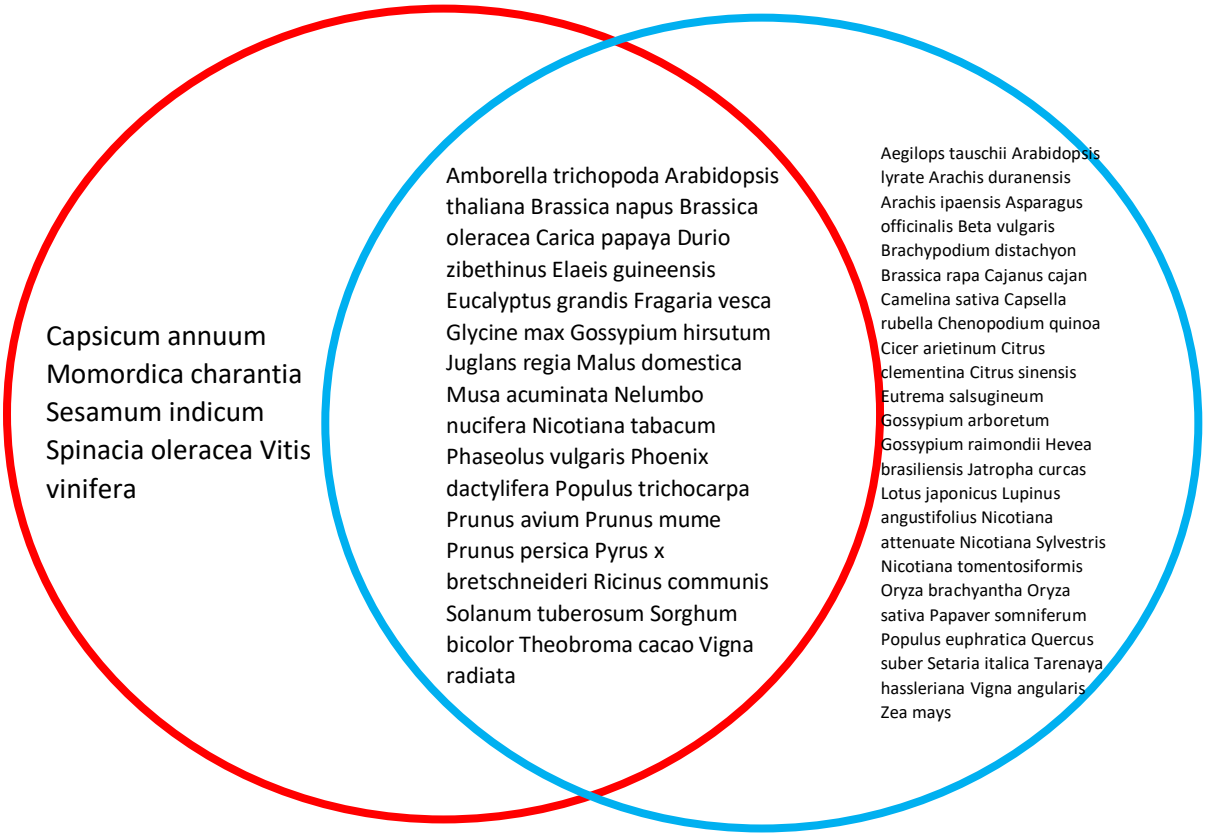

Figure 5. Comparison of data sets for epicatechin. The red circle contains species experimentally known to produce epicatechin and those in blue are for those determined to theoretically produce it.

Analyzing this diagram shows that twenty-eight species of plants fell within both data sets. However, there were five species that were experimentally known to produce epicatechin but did not appear in the data set of the species that were theoretically determined to synthesize epicatechin. At the same time, thirty-four species were theoretically determined to synthesize epicatechin but were not experimentally known to do so. This implies that the method that was incorporated in determining what plants could theoretically produce epicatechin is not a plausible method of doing so. Since there

are discrepancies between the two data sets, mainly that the theoretical data set not containing species experimentally known to produce epicatechin, means that the script is not a plausible method of predicting what plant species can produce epicatechin.

Naringenin. Comparisons of the two data sets obtained for the flavonoid naringenin can be seen in Figure 6.

Figure 6. Comparison of data sets for naringenin. The red circle contains species experimentally known to produce naringenin and those in blue are for those determined to theoretically produce it.

It displayed that forty of the species of plants were found in both sets of data and falling into being both experimentally known to produce naringenin and theoretically determined to do so as well. At the same time forty-five species of plants were found only in the data of the species theoretically determined to produce naringenin. This implies that method in

which was incorporated in determining what plants could theoretically produce naringenin is a plausible method of predicting what plants could produce naringenin and the large number of plants that were in the data set produced in the script indicates naringenin as a potential candidate for future study.

Eriodictyol. Comparisons of the data sets from eriodictyol can be seen in Figure 7.

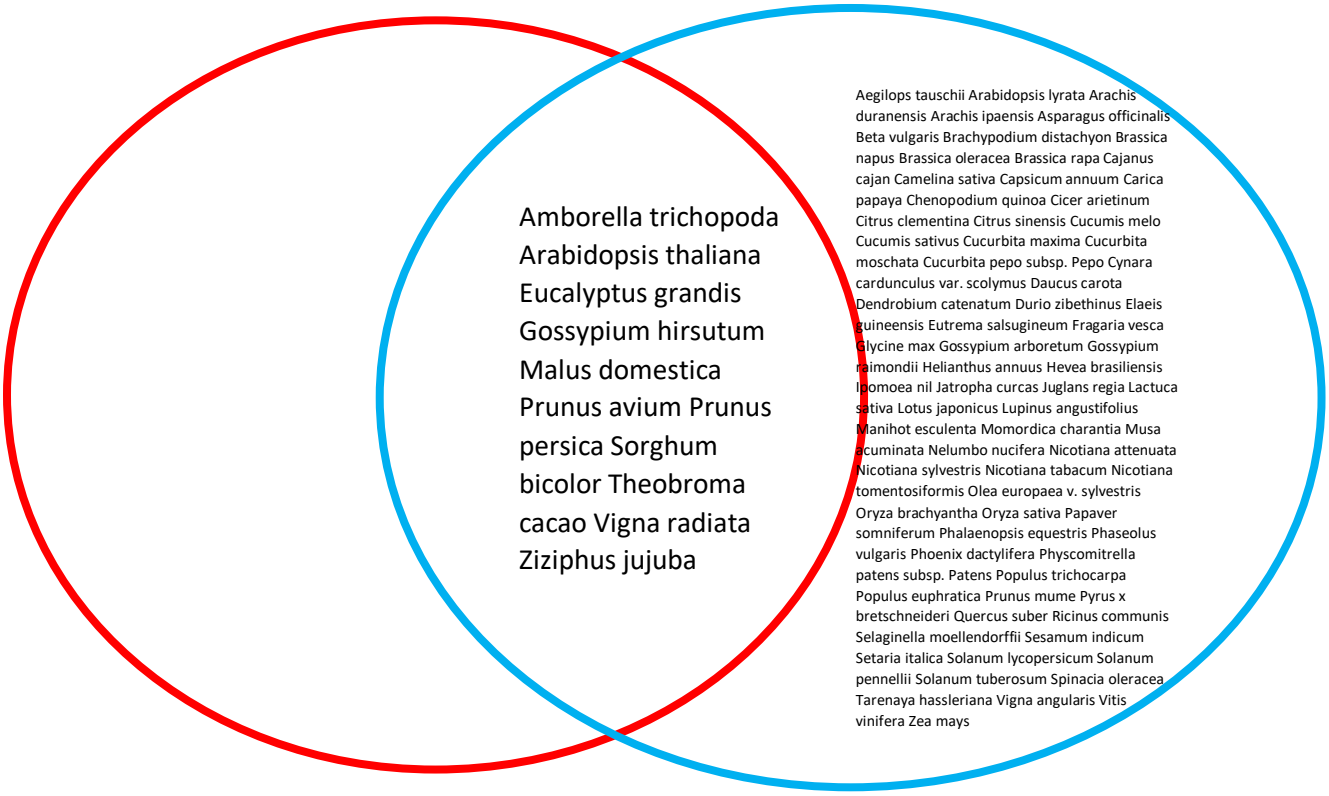

Figure 7. Comparison of data sets for eriodictyol. The red circle contains species experimentally known to produce eriodictyol and those in blue are for those determined to theoretically produce it.

This diagram showed that eleven species of plants were present in both data sets obtained. However, seventy-four species of plants were determined to theoretically produce eriodictyol, while none where present only in the experimentally known data set. This implies that the method used to predict what plants could produce eriodictyol is a

plausible method since no species appeared only in the experimentally known data set. Finally, this data implies that eriodictyol would be an exceptional candidate for future study since there are not many species experimentally known to synthesize and there are many species that were theoretically determined to produce it based off the KEGG database and the Python script.

It was possible that the reason that eriodictyol might not have been widely noted in plant species is that its primary use in flavonoid biosynthesis in most plant species might be an intermediate that is immediately used upon its formation. The reason naringenin may have observed more experimentally, may be because it is not immediately used upon its formation and could be separated from the plant tissue. Catechin and epicatechin may have had discrepancies between the two data sets because the data available on KEGG was incomplete at the time the script was ran.

Recommendations for Future Work

It is recommended for future research students to make comparisons between the DNA sequences of the genes that encode the enzymes which synthesize the flavonoids in order to better understand evolution of flavonoid biosynthesis across modern day plant species, evolution being the points in time in which there were mutations in the DNA sequences of the genes that caused speciation events to occur. This would help derive divergence points and show when and where different plants species began to utilize or stopped utilizing flavonoid biosynthesis. This is recommended since the data to do this has already been obtained.

It is also recommended other flavonoids such as quercetin and rutin were investigated using the same methods used in this study. It also should be done in order to

gain a better understanding of how well researched other flavonoids are and how they can be investigated further if there is very little information regarding them, if a flavonoid being used for further tests of the script is not well documented, this could skew the results of the script and make it difficult to determine if the use of the script is plausible. This would help further test the script used and determine if it is truly applicable to more flavonoids.

It should also be noted that if a species of plant was determined to theoretically produce a flavonoid, but was not experimentally known to do so, it is possible that some flavonoids are used as intermediates for other further downstream flavonoids in the biochemical pathway. It is also recommended that future research students analyze this possibility by looking at these plant species that had this occur in the data. Along with this, look at what further downstream flavonoids are produced and see if the reason why these flavonoids are predicted to be produced by a plant species, but are not experimentally known to do so is because those flavonoids are intermediates and are used in subsequent reactions.

Finally, the flavonoid eriodictyol, should be investigated further by the research group. Many plant species were predicted to be able to synthesize, but there is not a lot of information in the literature that shows that this flavonoid is very well studied. Testing the species that were determined to theoretically produce eriodictyol from the script would be a good place to start for future research and investigations to expand to knowledgebase of eriodictyol.

Method

KEGG Data Collection

It should be noted that this section of this writing goes over the code that was written for this project. The lists of plants species that theoretically could produce catechin, epicatechin, naringenin, and eriodictyol were obtained from the KEGG database. This was done using a Python 2.7 script that was written using Spyder, a program available from the Anaconda Navigator software. This script required the installation of an outside add-on package known as Bioservices which allows the user to access the KEGG database through the script. The initial lines of code can be seen in Figure 8 and contain the required dependencies for the entire script to work.

```
'''This are required for the script to work - it has all of the required dependencies'''
#bioservices must be installed on your computer
from bioservices.kegg import KEGG
import os
import sys
import datetime
k = KEGG()
now=datetime.datetime.now()
```

Figure 8. Dependencies required for the script.

The next group of commands creates the folders where all the subsequent files will be saved at and asks the user to input a name for the main folder. These commands can be seen in Figure 9.

```

'''Make the necessary folders'''
foldername = os.getcwd()+ "\\" + raw_input("Input a save folder name: ")#gets current working directory and asks for user input for a new foldername
if os.path.exists(foldername): #determines if the new foldername already exists
    decision = raw_input("Warning, this folder already exists. Press the return key to continue anyway, or type anything to try again: ")
    if decision == "": #if return key hit will continue running th code, will overwrite anything in the prexisting folder
        pass
    else:
        sys.exit("Try an unused folder name next time") #stops code completely and the code will need to be restarted with a different foldername
genefoldername = foldername+"\Gene_Data" #variable used to create the folder for the gene data
fastafoldername = foldername+"\FASTA_Data" #variable used to create the folder for the fasta data
chemicalfoldername=foldername+"\Chemical_Data"
try: os.mkdir(foldername) #creates the main working folder for the code
except WindowsError: pass #stops code from making folder if this error occurs
try: os.mkdir(genefoldername) #creates folder were gene data files will be inserted
except WindowsError: pass
try: os.mkdir(fastafoldername) #creates folder where fasta files will be inserted
except WindowsError: pass
try: os.mkdir(chemicalfoldername) #creates folder where fasta files will be inserted
except WindowsError: pass
```

Figure 9. Lines of code that generate folders for outputs.

The next lines of code contain a function that allows the removal of duplicate items from a list, which is required for later functions of the entire script. This function can be seen in Figure 10.

```
'''Python code to remove duplicate elements --- needs to be up here because of testing code '''
def Remove(listwithduplicates):
    listwithoutduplicates = [] #creates an empty list
    for item in listwithduplicates:
        if item not in listwithoutduplicates:
            listwithoutduplicates.append(item) #adds item to empty list if it's not already in the list
    return listwithoutduplicates
```

Figure 10. Function defined to remove duplicate items from a list.

Following this function are two different sets of a list and a dictionary corresponding to the items in the list: one containing the KEGG codes for the species of interest along with a dictionary that defines what species of plant is represented by the code and the other contain the KEGG codes for the pathways of interest along with the dictionary to define what each pathway is. Along with this is a line of code that combines the items in the two different lists into a separate list that is used for later functions of the script in order to search entries in the KEGG database. These species were used because they all had entries in KEGG that pertained to the flavonoid pathway. This can be seen in Figure 11.

```
'''These are the list of the codes that you need to iterate over.
 If you are needing just one code, just have one item in the list, but keep it as a list.
 You must have both species codes and pathway codes for this script to work.'''
speciescode_list = ["ats", "atr", "aly", "ath", "adu", "aip", "aof", "apro", "bpg", "bvg", "bdi", "bna",
                    "boe", "brp", "ccaj", "csat", "crb", "cann", "cpap", "cqi", "cre", "cvr", "ccp", "cam", "cic", "cit",
                    "csl", "cmo", "csv", "cmax", "cmos", "cpep", "cme", "ccav", "dcr", "dct", "dzi", "egu", "egr",
                    "eus", "fve", "gsl", "gmx", "gab", "ghi", "gra", "han", "hbr", "ini", "jcu", "jre",
                    "lsv", "lja", "lang", "mdm", "mesc", "mis", "mpp", "mcha", "mng", "mus", "nnu", "nau", "nsy", "nta",
                    "nto", "oeu", "obr", "dosa", "osa", "olu", "ota", "psom", "peq", "pvu", "pda", "ppp", "pop", "peu", "pavi", "pmum",
                    "pper", "pxb", "qsu", "rcu", "smo", "sind", "sita", "sly", "spen", "sot", "sbi",
                    "soe", "thj", "tcc", "var", "vra", "vvi", "vcn", "zma", "zju"]#species codes for plants of interest,  "mtr" cut due to code errors
speciescode_dictionary = {"ats": "Aegilops tauschii", "atr": "Amborella trichopoda", "aly": "Arabidopsis lyrata",
                          "ath": "Arabidopsis thaliana", "adu": "Arachis duranensis", "aip": "Arachis ipaensis",
                          "aof": "Asparagus officinalis", "apro": "Auxenochlorella protothecoides",
                          "bpg": "Bathycoccus prasinos", "bvg": "Beta vulgaris", "bdi": "Brachypodium distachyon",
                          "bna": "Brassica napus", "boe": "Brassica oleracea", "brp": "Brassica rapa",
                          "ccaj": "Cajanus cajan", "csat": "Camelina sativa", "crb": "Capsella rubella",
                          "cann": "Capsicum annuum", "cpap": "Carica papaya", "cqi": "Chenopodium quinoa",
                          "cre": "Chlamydomonas reinhardtii", "cvr": "Chlorella variabilis", "ccp": "Chondrus crispus", "cam": "Cicer arietinum",
                          "cic": "Citrus clementina", "cit": "Citrus sinensis", "csl": "Coccomyxa subellipsoidea", "cmo": "Cucumis melo",
                          "csv": "Cucumis sativus", "cmax": "Cucurbita maxima", "cmos": "Cucurbita moschata",
                          "cpep": "Cucurbita pepo subsp. pepo", "cme": "Cyanidioschyzon merolae",  "ccav": "Cynara cardunculus var. scolymus",
                          "dcr": "Daucus carota", "dct": "Dendrobium catenatum", "dzi": "Durio zibethinus",
                          "egu": "Elaeis guineensis", "egr": "Eucalyptus grandis", "eus": "Eutrema salsugineum",
                          "fve": "Fragaria vesca", "gsl": "Galdieria sulphuraria", "gmx": "Glycine max", "gab": "Gossypium arboreum", "ghi": "Gossypium hirsutum",
                          "gra": "Gossypium raimondii", "han": "Helianthus annuus", "hbr": "Hevea brasiliensis",
                          "ini": "Ipomoea nil", "jcu": "Jatropha curcas", "jre": "Juglans regia",
                          "lsv": "Lactuca sativa", "lja": "Lotus japonicus", "lang": "Lupinus angustifolius", "mdm": "Malus domestica",
                          "mesc": "Manihot esculenta", "mis": "Micromonas commoda",
                          "mpp": "Micromonas pusilla", "mcha": "Momordica charantia", "mng": "Monoraphidium neglectum", "mus": "Musa acuminata",
                          "nnu": "Nelumbo nucifera", "nau": "Nicotiana attenuata", "nsy": "Nicotiana sylvestris", "nta": "Nicotiana tabacum",
                          "nto": "Nicotiana tomentosiformis", "oeu": "Olea europaea v. sylvestris", "obr": "Oryza brachyantha",
                          "dosa": "Oryza sativa japonica (RAPDB)", "osa": "Oryza sativa japonica (RefSeq)",
                          "olu": "Ostreococcus lucimarinus", "ota": "Ostreococcus tauri", "psom": "Papaver somniferum",
                          "peq": "Phalaenopsis equestris", "pvu": "Phaseolus vulgaris", "pda": "Phoenix dactylifera",
                          "ppp": "Physcomitrella patens subsp. Patens", "pop": "Populus trichocarpa", "peu": "Populus euphratica",
                          "pavi": "Prunus avium", "pmum": "Prunus mume", "pper": "Prunus persica",
                          "pxb": "Pyrus x bretschneideri", "qsu": "Quercus suber", "rcu": "Ricinus communis",
                          "smo": "Selaginella moellendorffii", "sind": "Sesamum indicum", "sita": "Setaria italica",
                          "sly": "Solanum lycopersicum", "spen": "Solanum pennellii", "sot": "Solanum tuberosum",
                          "sbi": "Sorghum bicolor", "soe": "Spinacia oleracea", "thj": "Tarenaya hassleriana",
                          "tcc": "Theobroma cacao", "var": "Vigna angularis", "vra": "Vigna radiata",
                          "vvi": "Vitis vinifera", "vcn": "Volvox carteri f. nagariensis", "zma": "Zea mays", "zju": "Ziziphus jujuba"}#Dictionary that defines the ap
pathwaycode_list = ["00940", "00941", "00942", "00943", "00944"]#list of pathways of interest
pathwaycode_dictionary = {"00940": "phenylpropanoids", "00941": "flavonoids", "00942": "anthocyanins",
                          "00943": "isoflavonoids", "00944": "flavones/flavonols", "00945": "stilbenoids"}#Dictionary defining each pathway by each chemical they're r

pathwayID_list = [i+j for i in speciescode_list for j in pathwaycode_list] #this is the full list of every pathway and species from the above lists
```

Figure 11. List of species codes and pathway codes used along with dictionaries to define them.

The next lines of code defines two functions: one that pulls the data pertaining to the genes that encode for the enzymes involved in the pathways of interest for each plant species and another that creates the CSV and TXT files that contain the data that was obtained from the entire script and saves them in the folder indicated by the user. These functions can be seen in Figure 12.

```
'''defining the function that will actually get all of the data that is required'''
def gene_pathway_data(pathwayID):
    #print "Looking up data from: "+pathwayID #good for testing but takes up a lot of time in actuality
    entrylines_list = k.get(pathwayID).split("\n") #gets all of the data and splits it by line
    #print genes
    linecount=0
    for line in entrylines_list: #finds the places that have the genes listed
        entrylines_list[linecount]=line.strip()#removes extra unicode/spaces and replaces the original entry
        #print str(linecount) + ": " + line
        if line.startswith('GENE'): #finds where GENE is at in each entry
            #print line
            entrylines_list[linecount]=line.replace("GENE","").strip()#replaces GENE with a blank and removes extra spaces at the beginning and end
            GENElocator=linecount #gene locator is now the item in the list that has GENE
        if line.startswith('COMPOUND'): #finds where COMPOUND is in the entry
            #print line
            COMPOUNDlocator=linecount#compoundLocator is now the item in the list that begins with compound
        linecount+=1
    geneline_list= entrylines_list[GENElocator:COMPOUNDlocator] #makes a list that is just the gene entry lines
    linecount=0
    for i in geneline_list: #this section makes a list of lists that are appreciately separated
        i= i.replace("  ","^*^").replace(";","^*^").replace("[","^*^").replace("]","")#makes ^*^ the signifier for splitting
        i=i.split("^*^")
        i.insert(0,speciescode_dictionary[filter(str.isalpha, pathwayID)])#pathwayID[0:3]])#replaces the species code with the genus species from the dictionary value
        jcount=0
        for j in i: #cleans up list of lists of extra spaces at the beginning and end of each list
            i[jcount]=j.strip()
            jcount+=1
        geneline_list[linecount]= i #replaces the list with the new cleaned list of lists
        linecount+=1#iterates through each list in the entry
    return geneline_list

'''function that saves each file with a name that includes pathwayID  using the data from genes_listoflists'''
def get_lists(whatlist, outname, outfolder=os.getcwd()): #whatlist should be a list of lists, outname should include an appropriate extension
    os.chdir(outfolder)#changes directory to current working directory
    writedoc = file(outname,"w") #gives write privilages for the function to write to the file
    for line in whatlist:
        for item in line:
            item=str(item).replace("\n","") #removes the new lines in each list of list
            if item == "":
                writedoc.write("-") #if the list in the list of list is empty writes a dash
            else:
                writedoc.write(item) #writes the entry in the list of lists to the file
            writedoc.write(",")#tab deliniated; use "," for csv files
        writedoc.write("\n") #creates a new line after the above portion of teh function
    writedoc.write("\n")
    writedoc.close() #closes file, required
    print  outname, "saved in: ", outfolder
```

Figure 12. Defined functions that obtain data from KEGG and generates the output files.

The next lines of code utilize the previous functions defined in order to collect the data from the KEGG database and create a list of lists from the data. Then, it uses the function to remove duplicate items in the lists and remove any false values that may have appeared in the lists of lists. The code can be seen in Figure 13.

```
'''iterate over pathwayID_list and use the fuction defined above'''
masterlist=[]
for pathwayID in pathwayID_list:
    notpresent=0
    try: currentlist=gene_pathway_data(pathwayID)  #need to ignore everything if there is no pathway for that species
    except AttributeError:
        #print " No data in "+ pathwayID #good for testing but takes up a lot of time in actuality
        notpresent=1
    if notpresent==0:
        masterlist.extend(currentlist)

    #run the function that saves each file
    notpresent2=0
    try: get_gene_lists = get_lists(gene_pathway_data(pathwayID), "Gene_data_"+pathwayID+".csv", genefoldername) #try actually does it if it works
    except AttributeError:
        #print " No data in "+ pathwayID #good for testing but takes up a lot of time in actuality
        notpresent2=1

'''remove duplicates'''
#print masterlist
masterlist_nodup=Remove(masterlist)
masterlist_nodup = list(filter(None,masterlist_nodup)) #removes false values from masterlist_nodup and turns it into a list
count=0
for i in masterlist_nodup: #removes false values and iterates through the list of lists
    masterlist_nodup[count]=list(filter(None,masterlist_nodup[count]))
    count+=1
```

Figure 13. Code that utilizes function to obtain data from KEGG and remove duplicates.

Following this, another function is defined that finds unique items in a list and inputs the unique items into a separate list. The function is then used to obtain the unique EC numbers from the data set obtained from KEGG. This can be seen in Figure 14.

```
'''find unique EC numbers so have a generic function and run it'''
def UniqueElementList(listname, locationnumber): #be careful as there are cases of one less item - use "last" to fix that problem here
    originallocationnumber=locationnumber
    ElementList=[]
    for i in listname:
        #print
        if originallocationnumber == "last": locationnumber=len(i)-1 #assigns the string "last" to the very last list in the list of lists
        #print locationnumber
        if i[int(locationnumber)] not in ElementList: #finds unique EC number not in the list
            ElementList.append(i[int(locationnumber)]) #adds it to the list
    return ElementList
EC_list=UniqueElementList(masterlist_nodup, "last")
```

Figure 14. Defined function to obtain unique elements from a list.

Using the unique EC list and the previous list of lists, a matrix was made using a list of lists that counted the number of times each EC number occurs for each plant species. This can be seen in Figure 15.

```
'''creating the matrix and adding up the counts'''
mastercount_list = [["Species"]]
mastercount_list[0].extend(EC_list) #adds the Unique EC numbers to the end of the mastercount list of lists
for i in speciescode_list: #@#
    mastercount_list.append([i]) #first item in each row (but first) is the speciescode
#print EC_list
#print mastercount_list

icount=0
for i in mastercount_list[0]: #for each EC number
    #print "i: "+ i
    if icount != 0: #first column isn't actually an EC#,
        EC=i
        #print EC
        jcount=0
        for j in mastercount_list: #for each species (using the speciecode)
            #print j
            if jcount != 0: #first row isn't actually a species
                #print j
                species=speciescode_dictionary[j[0]]
                #print species
                lcount=0
                for l in masterlist_nodup: #iterate over the culled masterlist to check for matching sets
                    if l[0]==species and l[len(l)-1]==EC:
                        lcount+=1
                mastercount_list[jcount].append(lcount)
            #else: print "help"
            jcount+=1 ####
    icount+=1
#print mastercount_list#[0:3] #print this to test, but hide when actually running

#change mastercount_list to be actual species:
icount=0
for i in mastercount_list:
    if icount != 0: mastercount_list[icount][0]=speciescode_dictionary[i[0]] #replaces species code with genus specie names
    icount+=1
```

Figure 15. Generation of matrix and counting number of genes for each EC of each species

Following these lines of code, the next few lines of code generate two master CSV files, one that contains the designated numbers for the genes related to the enzymes involved in

the desired pathways for every plant species that was looked at, and the other contains the previously mentioned matrix. This can be seen in Figure 16.

```
'''Make Master Files'''
get_masterlist = get_lists(masterlist_nodup, "Master_List.csv", foldername)
get_mastercount = get_lists(mastercount_list, "Master_Count.csv", foldername)
```

Figure 16. Generation of master files from previous list of lists.

After these master files were made, another master file was made which contained all the DNA sequences of every gene that were present in the master list that was previously created from the KEGG database. This was done by defining another function and applying it to the master list and can be seen in Figure 17.

```
'''make a master fasta file saved in foldername'''
rev_dict={v:k for k,v in speciescode_dictionary.iteritems()}#reverses dictionary keys and values
genelist_frommaster=[]
for i in masterlist_nodup:
    #print rev_dict[i[0]]+":"+i[1]
    genelist_frommaster.append(rev_dict[i[0]]+":"+i[1]) #combines species codes and gene numbers in a list to be used for the master fasta function

def get_master_fasta(gene):
    DNA_info_list=[]
    for gene in genelist_frommaster:
        #print gene
        gene_fasta_data = k.get(gene).split("\n") #calls the entry from KEGG and splits it into new lines
        linecount = 0
        for line in gene_fasta_data:
            gene_fasta_data[linecount]=line.strip() #removes blank spaces at the beginning and end of each line
            if line.startswith('ORGANISM'): #finds where the entry that begins with organism is
                gene_fasta_data[linecount]=line.replace("ORGANISM","").strip() #removes organism and removes blank spaces to beginning and end
                find_blankspace=gene_fasta_data[linecount].find(" ") #finds where thedouble blank space is in the organism line
                organism_name= gene_fasta_data[linecount][find_blankspace:] #the genus specie name is left from the orginal entry
            linecount+=1
        linecount=0
        for line in gene_fasta_data:
            gene_fasta_data[linecount]=line.strip()
            if line.startswith('ORTHOLOGY'):
                gene_fasta_data[linecount]=line.strip
                find_EC=line.find("EC:") #finds within the line where the EC number is
                gene_fasta_data[linecount]=line[find_EC:-1].replace("[", "").replace("EC:", "") #removes the beginning bracket and EC: leaving just the number
                EC_number="EC "+ gene_fasta_data[linecount] #adds EC back
                joined_organism_EC=[">"+str(organism_name).strip()+"%"+EC_number+"%"+gene.split(":")[1]]#adds > to beginning to find the beginning of each entry mor
                DNA_info_list.append(joined_organism_EC) #adds the entry to the blank list
            linecount+=1
        linecount=0
        for line in gene_fasta_data:
            gene_fasta_data[linecount]=line.strip()
            if line.startswith('NTSEQ'):
                gene_fasta_data[linecount]=line.replace("NTSEQ","").strip()
                NTSEQlocator=linecount
            linecount+=1
        DNA_data_list=gene_fasta_data[NTSEQlocator:]
        DNA_Seq=DNA_data_list[1:len(DNA_data_list)-2] #Takes just the DNA sequence
        separator=""
        joined_DNA_seq=[separator.join(DNA_Seq)] #combines the separate DNA sequence lines into one string and turns that into a single entry list
        DNA_info_list.append(joined_DNA_seq) #adds single entry list to the list of lists
    return DNA_info_list

get_lists(get_master_fasta(genelist_frommaster), "Master_FASTA.csv", fastafoldername)

masterfasta=get_master_fasta(genelist_frommaster)#should actually go above the first time it gets called
```

Figure 17. Generation of master FASTA file.

Following the creation of the master file that contains the DNA sequences of every single gene, was the lines of code that generate similar files to the previous master file of DNA sequences but rather creates separate files for each EC number. This is seen in Figure 18.

```
'''creates the FASTA files by EC numbers'''
icount=0
for i in ECorderlist:
    name=i.replace(".","p").replace("EC ","").replace(" " ,"")
    FASTAbyEC= get_lists(fasta_byEC[icount], name+".csv", fastafoldername)
    print name+".csv" + " saved in: "+fastafoldername
    icount+=1
```

Figure 18. Generation of FASTA files by EC number

The lines of code create a ReadMe file that provides the name of the code, the date in which the script was ran and, a description of the outputted files. These lines of code can be seen in Figure 19.

```
'''Creates ReadMe file'''
with open(foldername+"\ReadMe.txt", "w") as ReadMe:
    ReadMe.write("KEGG_v1p1.py\n")
    ReadMe.write(now.strftime("%m-%d-%Y")+"\n")
    ReadMe.write(foldername+"\n")
    ReadMe.write("This script creates a series of files related to the genes associated with plant flavonoids from various species of plants. This script first create
    ReadMe.write(". The script also creates files that only contains the genes of a single plant species biochemical pathway which are located in "+genefoldername+". "
    ReadMe.close
```

Figure 19. Generation of the ReadMe

Following the creation of the ReadMe, another function is defined that is used to determine if the plants have the correct genes to produce one of the four specified flavonoids. This function can be seen in Figure 20.

```
'''Function that determines if the plant species has the correct enzymes to produce specified chemicals'''
def ECandor(listname):
    icount=0
    codestring=''
    for i in listname:
        if icount>0:
            codestring+='"'+i+'"'
            try:
                listname[icount+1]
            except IndexError:
                break
            codestring+=' '+listname[0]+' '
        icount+=1
    print codestring
    return codestring
```

Figure 20. Defined function to determine if plants have correct genes

The following lines of code create a list that counts the number of genes that each plant species has that relate to the specified EC number in the KEGG database. This is seen in Figure 21.

```
'''Creates a list of the number of times the specified enzymes appear for each plant specie'''
masterEC_list = [["species","EC#s"]]
icount=0
for i in mastercount_list: #for each species
    #print "species :", i[0]
    speciesEClist=[]
    if icount==0:
        pass

    else:
        jcount=0
        for j in i: #for EC in species
            if jcount==0:
                speciesEClist.append(j)
                print j
            else:
                if str(j)=="0":
                    print " "+ str(j)+ " = no enzyme"
                    pass
                else:
                    print mastercount_list[0][jcount] + " = " + str(j)
                    speciesEClist.append(mastercount_list[0][jcount])
            jcount+=1
    masterEC_list.append(speciesEClist)
    icount+=1

#print masterEC_list
masterEC_list=masterEC_list[1:]
```

Figure 21. List generated that counts the number of genes related to each EC number indicated

Some of the last lines of code determines which plant species have the necessary genes to produce catechin, epicatechin, naringenin, and eriodictyol based on the previous list that counted the number of genes related to each EC number for each plant species, this can be seen in Figure 22. The EC numbers required for each flavonoid were predetermined in lists and were incorporated into the commands that determine if the plant species can produce the specific flavonoid. These lines of code can also be seen in Figure 22.

```
phenylalanineTOcinnamicacid=["or","EC:4.3.1.24","EC:4.3.1.25"]
cinnamicacidTOpcoumaroyllcoa=["and","EC:6.2.1.12","EC:1.14.14.91"]
pcoumaroyllcoaTOcaffeoylcoa1=["and","EC:1.14.13.-"]
pcoumaroyllcoaTOcaffeoylcoa2=["and","EC:2.3.1.133","EC:1.14.14.96"]
pcoumaroyllcoaTOnaringenin=["and","EC:2.3.1.74","EC:5.5.1.6"]
naringeninTOeriodictyol=["or","EC:1.14.14.81","EC:1.14.14.82"]
caffeoylcoaTOeriodictyol=["and","EC:2.3.1.74"]
eriodictyolTOleucocyanidin=["and","EC:1.14.11.9","EC:1.1.12.19"]
leucocyanidinTOcatechin=["and","EC:1.17.1.3"]
leucocyanidinTOcyanidin=["and","EC:1.14.20.4"]
cyanidinTOepicatechin=["and","EC:1.3.1.77"]
pcoumaroyllcoaTOnaringenin=["and","EC:2.3.1.74","EC:5.5.1.6"]
eriodictyolTOluteolin=["or","EC:1.14.20.5", "EC:1.14.19.76"]
naringeninTOapigenin=["or","EC:1.14.20.5","EC:1.14.19.76"]
apigeninTOluteolin=["or","EC:1.14.14.81","EC:1.14.14.82"]


epicatechinlist=[]
catechinlist=[]
eriodictyollist=[]
naringeninlist=[]
luteolinlist=[]
#Be careful in making these of parenthases
for i in masterEC_list:
    #print i

    epicatechin= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and " + ECandor(cinnamicacidTOpcoumaroyllcoa)+ " and (" + ECandor(pcoumaroyllcoaTOcaffeoylcoa1)+" or (
        pcoumaroyllcoaTOcaffeoylcoa2)+"))"+" and "+ECandor(caffeoylcoaTOeriodictyol)+" and "+ECandor(eriodictyolTOleucocyanidin)+" and "+ECandor(
        leucocyanidinTOcyanidin)+" and "+ECandor(cyanidinTOepicatechin)+") in i: epicatechinlist.append([i[0]])"
    exec epicatechin

    catechin= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and " + ECandor(cinnamicacidTOpcoumaroyllcoa)+ " and (" + ECandor(pcoumaroyllcoaTOcaffeoylcoa1)+" or (
        pcoumaroyllcoaTOcaffeoylcoa2)+"))"+" and "+ECandor(caffeoylcoaTOeriodictyol)+" and "+ECandor(eriodictyolTOleucocyanidin)+" and "+ECandor(
        leucocyanidinTOcatechin)+") in i: catechinlist.append([i[0]])"
    exec catechin

    eriodictyol= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and " + ECandor(cinnamicacidTOpcoumaroyllcoa)+ " and (" + ECandor(pcoumaroyllcoaTOcaffeoylcoa1)+" or (
        pcoumaroyllcoaTOcaffeoylcoa2)+"))"+" and "+ECandor(caffeoylcoaTOeriodictyol)+") in i: eriodictyollist.append([i[0]])"
    exec eriodictyol

    luteolin= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and " + ECandor(cinnamicacidTOpcoumaroyllcoa)+ " and (" + ECandor(pcoumaroyllcoaTOcaffeoylcoa1)+" or (
        pcoumaroyllcoaTOcaffeoylcoa2)+"))"+" and "+ECandor(caffeoylcoaTOeriodictyol)+" and ("+ECandor(eriodictyolTOluteolin)+")) in i: luteolinlist.append([i[0]])"
    exec luteolin
    '''luteolin= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and " + ECandor(cinnamicacidTOpcoumaroyllcoa)+ " and (" + ECandor(pcoumaroyllcoaTOcaffeoylcoa1)+" or (
        pcoumaroyllcoaTOcaffeoylcoa2)+"))"+" and "+ECandor(caffeoylcoaTOeriodictyol)+" and "+ECandor(eriodictyolTOluteolin)+") in i: luteolinlist.append([i[0]])"
    exec luteolin'''#didn't work

    naringenin= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and " + ECandor(cinnamicacidTOpcoumaroyllcoa) + " and "+ ECandor(pcoumaroyllcoaTOnaringenin) +") in
    exec naringenin
```

Figure 22. Generation of list that specifies what plants can produce the specified flavonoids

Data pertaining to luteolin was originally supposed to be compiled in this script as seen in Figure 22, however, when the script was executed it returned a blank list, so the analysis was not performed for luteolin. The last lines of code generate the TXT files that contain the list of plants that can produce each of the flavonoids. This is seen in Figure 23.

```
get_lists(epicatechinlist, "epicatechinspecies.txt", outfolder=foldername+"\Chemical_Data")
get_lists(catechinlist, "catechinspecies.txt", outfolder=foldername+"\Chemical_Data")
get_lists(eriodictyollist, "eriodictyolspecies.txt", outfolder=foldername+"\Chemical_Data")
get_lists(luteolinlist, "luteolinspecies.txt", outfolder=foldername+"\Chemical_Data")
get_lists(naringeninlist, "naringeninspecies.txt", outfolder=foldername+"\Chemical_Data")
```

Figure 23. Generation of TXT files containing what species can produce specified flavonoids

In short, a list of species codes and pathway codes were defined and then used to create a list that was iterated using a function that searched KEGG for the corresponding species for each pathway and gathered thee relevant gene data and inputted that information into

CSV files. The script also created a matrix and that counted the number of times each EC number appeared for each species of plant and inputted that information into another CSV. Following that, the DNA sequences for each gene for each plant species from each pathway were obtained from the KEGG entries and placed into CSV files. Finally, the enzymes and intermediates were defined to reach catechin, epicatechin, naringenin, and eriodictyol and the plant species that had the genes for the specified enzymes were inputted into a list that was then written into TXT files for each flavonoid. See the Appendix for the full code.

Literature Search

The literature search was performed using the Kettering Library database and Google Scholar. The information was search by searching the genus species name of each plant of interest followed by the flavonoid of interest. In between every word searched, the "and" delineator was used to help specifically search for the plant species of interest and the flavonoid of interest. Most of the information was found in the Kettering Library database with a few scientific papers found on Google Scholar. The first ten results for each search were examined for each plant species of each flavonoid, if no relevant information was found for the plant species or if the scientific name was not specified, then it was assumed that the plant could not produce the specified flavonoid or no research has been conducted on that plant for that specific flavonoid. This information was then compiled into a spreadsheet along with the references in which the information came from. (Akomolafe, S., Oboh, G., Oyeleye, S., Molehin, O., & Ogunsuyi, O., 2016; Akyol, H., Riciputi, Y., Capanoglu, E., Caboni, M. F., & Verardo, V., 2016; Arts, I. C. W., van de Putte, B., & Hollman, P. C. H., 2000; Auger et al., 2010; Babu, M. A.,

Suriyakala, M. A., & Gothandam, K. M., 2012; Baljeet, S. Y., Roshanlal, Y., & Ritika, B. Y., 2016; Bhagwat, S., Haytowitz, D.B. Holden, J.M. (Ret.)., 2014; Bhandari, S. R., & Lee, J. G., 2016; Boches, P., Peterschmidt, B., & Myers, J. R., 2011; Buschmann, H., Reilly, K., Rodriguez, M. X., Tohme, J., & Beeching, J. R., 2000; Charoenkiatkul, S., Thiyajai, P., & Judprasong, K., 2016; Chen, H. et al., 2019; Chen, Y. N. et al., 2014; Chen, Y. et al., 2013; Chung, I., Oh, J., & Kim, S., 2017; Ciric, A. R. et al., 2014; Cuong, D. M. et al., 2018; Davey, M. P., Burrell, M. M., Woodward, F. I., & Quick, W. P., 2008;2007; de Pascual-Teresa, S., Santos-Buelga, C., & Rivas-Gonzalo, J. C., 2000; Di Sotto, A. et al., 2018; DongDong, Z., Chen, J., Li, T., Chen, F., & Qun Cui, D., 2015; Dueñas, M., Hernández, T., Estrella, I., & Fernández, D., 2009; El-Kholy, W. M., Soliman, T. N., & Darwish, A. M. G., 2019; Frary, A. et al., 2010; Gallego, A. M. et al., 2019; Ganesan, K., & Xu, B., 2017; Ganesan, K., & Xu, B., 2018; Gao, Q., Wu, C., Wang, M., Xu, B., & Du, L., 2012; Gasperotti, M., Masuero, D., Mattivi, F., & Vrhovsek, U., 2015; Ghasemnezhad, M., Sherafati, M., & Payvast, G. A., 2011; Günç Ergönül, P., & Aksoylu Özbek, Z., 2018; Hamid, M. G., & Mohamed Nour, Abdel Azim Ahmed., 2018; Hammouda, H., Chérif, J. K., Trabelsi-Ayadi, M., Baron, A., & Guyot, S., 2013; Hashmi, M. A., Khan, A., Hanif, M., Farooq, U., & Perveen, S., 2015; Hemalatha, P., Bomzan, D. P., Sathyendra Rao, B. V., & Sreerama, Y. N., 2016; Hou, Z., Luo, J., & Kong, L., 2009; Huang, C. F. et al., 2011; Igbinosa, O. O. et al., 2011; Jacobo-Valenzuela, N. et al., 2011; Jaffri, J. M. et al., 2011; Jiang, C., Schommer, C. K., Kim, S. Y., & Suh, D., 2006; Kečkeš, S. et al., 2013; Khallouki, F., Ricarte, I., Breuer, A., & Owen, R. W., 2018; Kim, E. et al., 2013; Kim, M. Y. et al., 2017; Kim, Y. J. et al., 2015; Kitamura, S. et al., 2010; Klejdus, B., Lojková, L., Plaza, M., Šnóblová, M., & Štěrbová,

D., 2010; Kong, C. et al., 2018; Koopman, F. W. et al., 2012; Koprivica, M. R. et al., 2018; Kovinich, N., Saleem, A., Arnason, J. T., & Miki, B., 2012; Lei, Z. et al., 2018; Li, X. et al., 2011; Liew, S. S., Ho, W. Y., Yeap, S. K., & Sharifudin, S. A. B., 2018; Lim, T. K., 2012; Liu, R., Cai, Z., & Xu, B., 2017; Liu, Y. et al., 2013; Luo, X., Cui, J., Zhang, H., & Duan, Y., 2018; Mallek-Ayadi, S., Bahloul, N., & Kechaou, N., 2017; McNulty, J. et al., 2009; Mrázová, A. et al., 2017; Nix, A., Paull, C. A., & Colgrave, M., 2015; Nix, A., Paull, C., & Colgrave, M., 2017; Nurraihana, H., Wan Rosli, W. I., Sabreena, S., & Norfarizan-Hanoon, N. A., 2018; Oboh, G., Olabiyi, A. A., & Akinyemi, A. J., 2013; Oertel, A. et al., 2017; Ortuño, A. et al., 2011; Peng, M. et al., 2017; Perestrelo, R. et al., 2012; Preuß, A. et al., 2009; Quintero-Soto, M. F. et al., 2018; Roda, A. L., Oldham, N. J., Svatos, A., & Baldwin, I. T., 2003; Rodrigues, C., Nicácio, A., Jardim, I., Visentainer, J., & Maldaner, L., 2019; San, B., & Yildirim, A. N., 2010; Sánchez-Rabaneda, F. et al., 2003; Santos, S. A. O., Vilela, C., Freire, C. S. R., Neto, C. P., & Silvestre, A. J. D., 2013; Sanz, M. et al., 2010; Serra, A. T. et al., 2010; Sharma, N. et al., 2019; Sharma, S., Saxena, D. C., & Riar, C. S., 2015; Shin, E. J. et al., 2013; Singh, S., 2016; Sumczynski, D., Kotásková, E., Družbíková, H., & Mlček, J., 2016; Svensson, L., Sekwati-Monang, B., Lutz, D. L., Schieber, A., & Gänzle, M. G., 2010; Tamasi, G. et al., 2019; Tang, X., Tang, P., & Liu, L., 2017; Terpinc, P., Polak, T., Makuc, D., Ulrih, N. P., & Abramovič, H., 2012; Thiruvengadam, M., & Chung, I., 2015; Thiruvengadam, M. et al., 2014; Tsanova-Savova, S., Ribarova, F., & Gerova, M., 2005; Valiñas, M. A., Lanteri, M. L., ten Have, A., & Andreu, A. B., 2017; Vinha, A. F., Barreira, J. C. M., Costa, A. S. G., & Oliveira, M. Beatriz P. P., 2016; Vu, D. C., Vo, P. H., Coggeshall, M. V., & Lin, C., 2018; Wang, L. et al., 2013; Woo, K. S. et al., 2018; Wu, S., Wilson, A. E., Chang, L., &

Tian, L., 2019; Xu, X. et al., 2019; Yan, M. et al., 2019; Yi, T. G., Yeoung, Y. R., Choi, I., & Park, N., 2019; Yıldırım, F. et al., 2016; Yobi, A. et al., 2012; Zahir, A. A. et al., 2012; Zdunic, G. et al., 2016; Zeb, A., 2016; Zeb, A., Muhammad, B., & Ullah, F., 2017; Zhai, R. et al., 2014; Zhang, X. et al., 2015; Zhang, X., Sun, X., Zhao, H., Xue, M., & Wang, D., 2017; Zhou, J. et al., 2019)

Phylogenetic Trees

The phylogenetic trees were made using the lists of plants species obtained from the script and from the literature search for catechin, epicatechin, naringenin, and eriodictyol. These lists were formatted into a fashion that was readable by the National Center for Biotechnology Information (NCBI) tree viewer. The website known as PhyloT was used to format these lists into TXT files based on taxological information. These TXT files then were uploaded to the NCBI tree viewer and generated the phylogenetic trees.

Results

Kyoto Encyclopedia of Genes and Genomes

These results were obtained by the use of the script that was written. The script predicts which plant species can produce each flavonoid based on if the plant species had the predetermined enzymes defined in the code and adds those species that do have the corresponding enzymes into a list.

Catechin. The plants that were determined to theoretically synthesize catechin can be seen in Table 1. There are fifty-seven plant species that were predicted to produce catechin based on the KEGG script. All fifty-seven of these species fell within the Magnoliopsida class, which is the class for all flowering plants (Thorne & Reveal, 2007).

Table 1. Plants that Can Theoretically Synthesize Catechin

Plant Species				
Aegilops tauschii	Elaeis guineensis	Manihot esculenta	Populus euphratica	Ziziphus jujuba
Amborella trichopoda	Eucalyptus grandis	Musa acuminata	Prunus avium	
Arachis duranensis	Fragaria vesca	Nelumbo nucifera	Prunus mume	
Arachis ipaensis	Glycine max	Nicotiana attenuata	Prunus persica	
Asparagus officinalis	Gossypium arboreum	Nicotiana sylvestris	Pyrus x bretschneideri	
Beta vulgaris	Gossypium hirsutum	Nicotiana tabacum	Quercus suber	
Brachypodium distachyon	Gossypium raimondii	Nicotiana tomentosiformis	Ricinus communis	
Cajanus cajan	Helianthus annuus	Olea europaea v. sylvestris	Solanum pennellii	
Carica papaya	Hevea brasiliensis	Oryza brachyantha	Solanum tuberosum	
Chenopodium quinoa	Jatropha curcas	Oryza sativa japonica	Spinacia oleracea	
Cicer arietinum	Juglans regia	Papaver somniferum	Theobroma cacao	
Citrus clementina	Lotus japonicus	Phaseolus vulgaris	Vigna angularis	
Citrus sinensis	Lupinus angustifolius	Phoenix dactylifera	Vigna radiata	
Durio zibethinus	Malus domestica	Populus trichocarpa	Vitis vinifera	

Epicatechin. The plants that were determined to theoretically synthesize epicatechin can be seen in Table 2. There was a total of sixty-six species of plants that were predicted to produce epicatechin. Similar to catechin, all of the species fell within the Magnoliopsida class, meaning that all of these plant species were flowering plants.

Table 2. Plants that Can Theoretically Synthesize Epicatechin

Plant Species				
Aegilops tauschii	Capsella rubella	Gossypium raimondii	Oryza brachyantha	Solanum lycopersicum
Amborella trichopoda	Carica papaya	Hevea brasiliensis	Oryza sativa japonica	Solanum pennellii
Arabidopsis lyrata	Chenopodium quinoa	Jatropha curcas	Papaver somniferum	Solanum tuberosum
Arabidopsis thaliana	Cicer arietinum	Juglans regia	Phaseolus vulgaris	Sorghum bicolor
Arachis duranensis	Citrus clementina	Lotus japonicus	Phoenix dactylifera	Tarenaya hassleriana
Arachis ipaensis	Citrus sinensis	Lupinus angustifolius	Populus trichocarpa	Theobroma cacao
Asparagus officinalis	Durio zibethinus	Malus domestica	Populus euphratica	Vigna angularis
Beta vulgaris	Elaeis guineensis	Manihot esculenta	Prunus avium	Vigna radiata

Brachypodium distachyon	Eucalyptus grandis	Musa acuminata	Prunus mume	Zea mays
Brassica napus	Eutrema salsugineum	Nelumbo nucifera	Prunus persica	Ziziphus jujuba
Brassica oleracea	Fragaria vesca	Nicotiana attenuata	Pyrus x bretschneideri	
Brassica rapa	Glycine max	Nicotiana sylvestris	Quercus suber	
Cajanus cajan	Gossypium arboreum	Nicotiana tabacum	Ricinus communis	
Camelina sativa	Gossypium hirsutum	Nicotiana tomentosiformis	Setaria italica	

Naringenin. The plants that were determined to theoretically synthesize naringenin can be seen in Table 3. In total there were eighty-six plant species that were determined to theoretically produce naringenin. While most of the plant species fell within the Magnoliopsida class, there were two species that were not part of this class. The species *Selaginella moellendorffii* and *Physcomitrella patens* subsp. *patens* did not fall within the Magnoliopsida class and were determined to theoretically produce naringenin. Both of these species were noted to be different class of mosses rather than flowering plants.

Table 3. Plants that Can Theoretically Synthesize Naringenin

Plants Species						
Aegilops tauschii	Capsella rubella	Dendrobium catenatum	Juglans regia	Oryza brachyantha	Ricinus communis	Zea mays
Amborella trichopoda	Capsicum annuum	Durio zibethinus	Lactuca sativa	Oryza sativa japonica	Selaginella moellendorffii	Ziziphus jujuba
Arabidopsis lyrata	Carica papaya	Elaeis guineensis	Lotus japonicus	Papaver somniferum	Sesamum indicum	
Arabidopsis thaliana	Chenopodium quinoa	Eucalyptus grandis	Lupinus angustifolius	Phalaenopsis equestris	Setaria italica	
Arachis duranensis	Cicer arietinum	Eutrema salsugineum	Malus domestica	Phaseolus vulgaris	Solanum lycopersicum	

Arachis ipaensis	Citrus clementina	Fragaria vesca	Manihot esculenta	Phoenix dactylifera	Solanum pennellii	
Asparagus officinalis	Citrus sinensis	Glycine max	Momordica charantia	Physcomitrella patens subsp. patens	Solanum tuberosum	
Beta vulgaris	Cucumis melo	Gossypium arboreum	Musa acuminata	Populus trichocarpa	Sorghum bicolor	
Brachypodium distachyon	Cucumis sativus	Gossypium hirsutum	Nelumbo nucifera	Populus euphratica	Spinacia oleracea	
Brassica napus	Cucurbita maxima	Gossypium raimondii	Nicotiana attenuata	Prunus avium	Tarenaya hassleriana	
Brassica oleracea	Cucurbita moschata	Helianthus annuus	Nicotiana sylvestris	Prunus mume	Theobroma cacao	
Brassica rapa	Cucurbita pepo subsp. pepo	Hevea brasiliensis	Nicotiana tabacum	Prunus persica	Vigna angularis	
Cajanus cajan	Cynara cardunculus var. scolymus	Ipomoea nil	Nicotiana tomentosiformis	Pyrus x bretschneideri	Vigna radiata	
Camelina sativa	Daucus carota	Jatropha curcas	Olea europaea v. sylvestris	Quercus suber	Vitis vinifera	

Eriodictyol. The plants that were determined to theoretically synthesize eriodictyol can be seen in Table 4. The number of species that were determined to theoretically produce eriodictyol is eighty-five. Similar to naringenin, the majority of the species fell within the class of flowering plants, but there were two species, *Selaginella moellendorffii* and *Physcomitrella patens* subsp. *patens*, that didn't fall within that class of plant, rather they were two different classes of mosses, Lycopodiopsida and Bryopsida respectively. This is similar to the results obtained for naringenin as these two plant species also appeared in that data set. The main difference between the data sets of eriodictyol and naringenin is that naringenin has one more species predicted to produce naringenin and that species was *Capsella rubella*.

Table 4. Plants that Can Theoretically Synthesize Eriodictyol

Plant Species						
Aegilops tauschii	Capsicum annuum	Durio zibethinus	Lactuca sativa	Oryza sativa japonica	Selaginella moellendorffii	Ziziphus jujuba
Amborella trichopoda	Carica papaya	Elaeis guineensis	Lotus japonicus	Papaver somniferum	Sesamum indicum	
Arabidopsis lyrata	Chenopodium quinoa	Eucalyptus grandis	Lupinus angustifolius	Phalaenopsis equestris	Setaria italica	
Arabidopsis thaliana	Cicer arietinum	Eutrema salsugineum	Malus domestica	Phaseolus vulgaris	Solanum lycopersicum	
Arachis duranensis	Citrus clementina	Fragaria vesca	Manihot esculenta	Phoenix dactylifera	Solanum pennellii	
Arachis ipaensis	Citrus sinensis	Glycine max	Momordica charantia	Physcomitrella patens subsp. patens	Solanum tuberosum	
Asparagus officinalis	Cucumis melo	Gossypium arboreum	Musa acuminata	Populus trichocarpa	Sorghum bicolor	
Beta vulgaris	Cucumis sativus	Gossypium hirsutum	Nelumbo nucifera	Populus euphratica	Spinacia oleracea	
Brachypodium distachyon	Cucurbita maxima	Gossypium raimondii	Nicotiana attenuata	Prunus avium	Tarenaya hassleriana	
Brassica napus	Cucurbita moschata	Helianthus annuus	Nicotiana sylvestris	Prunus mume	Theobroma cacao	
Brassica oleracea	Cucurbita pepo subsp. pepo	Hevea brasiliensis	Nicotiana tabacum	Prunus persica	Vigna angularis	
Brassica rapa	Cynara cardunculus var. scolymus	Ipomoea nil	Nicotiana tomentosiformis	Pyrus x bretschneideri	Vigna radiata	
Cajanus cajan	Daucus carota	Jatropha curcas	Olea europaea v. sylvestris	Quercus suber	Vitis vinifera	
Camelina sativa	Dendrobium catenatum	Juglans regia	Oryza brachyantha	Ricinus communis	Zea mays	

Literature Search

The literature search was performed using the Kettering University's Library database and Google Scholar and by examining the first ten results for each plant species and while using the "and" delineator to narrow the searches.

Catechin. The plants that were experimentally known to produce catechin can be found in Table 5. There was a total of forty-four species of plants that were experimentally known to produce catechin. Similar to catechin obtained from the KEGG script, all of the species were part of the Magnoliopsida class, therefore they were all types of flowering plants.

Table 5. Plants Experimentally Known to Produce Catechin

Plant Species			
Aegilops tauschii	Durio zibethinus	Oryza sativa	Vitis vinifera
Amborella trichopoda	Elaeis guineensis	Phaseolus vulgaris	Ziziphus jujuba
Brassica oleracea	Eucalyptus grandis	Phoenix dactylifera	
Brassica rapa	Fragaria vesca	Populus trichocarpa	
Beta vulgaris	Glycine max	Prunus avium	
Camelina sativa	Gossypium hirsutum	Prunus persica	
Carica papaya	Juglans regia	Pyrus x bretschneideri	
Capsicum annuum	Malus domestica	Sesamum indicum	
Chenopodium quinoa	Manihot esculenta	Solanum tuberosum	
Cicer arietinum	Momordica charantia	Sorghum bicolor	
Citrus sinensis	Musa acuminata	Spinacia oleracea	
Cucumis melo	Nelumbo nucifera	Theobroma cacao	
Cucumis sativus	Nicotiana tabacum	Vigna angularis	
Cucurbita pepo subsp. pepo	Olea europaea v. sylvestris	Vigna radiata	

Epicatechin. The plants that were experimentally known to produce epicatechin can be found in Table 6. Based off the data obtained for epicatechin, there were thirty-four species of plants that were experimentally known to produce epicatechin. All of the species fell in the Magnoliopsida class, similar to the catechin data.

Table 6. Plants Experimentally Known to Produce Epicatechin

Plant Species		
Amborella trichopoda	Momordica charantia	Sorghum bicolor
Arabidopsis thaliana	Musa acuminata	Spinacia oleracea
Brassica napus	Nelumbo nucifera	Theobroma cacao
Brassica oleracea	Nicotiana tabacum	Vigna radiata
Carica papaya	Phaseolus vulgaris	Vitis vinifera
Capsicum annuum	Phoenix dactylifera	Ziziphus jujuba
Durio zibethinus	Populus trichocarpa	
Elaeis guineensis	Prunus avium	
Eucalyptus grandis	Prunus mume	
Fragaria vesca	Prunus persica	
Glycine max	Pyrus x bretschneideri	
Gossypium hirsutum	Ricinus communis	
Juglans regia	Sesamum indicum	
Malus domestica	Solanum tuberosum	

Naringenin. The plants that were experimentally known to produce naringenin can be found in Table 7. For naringenin, forty species were experimentally known to produce it. The majority of species that were experimentally known to produce are part of the Magnoliopsida class, however, similar to the KEGG data there were two species, *Selaginella moellendorffii* and *Physcomitrella patens* subsp. *patens*, that were not part of this class. Rather, they were two different classes of mosses, Lycopodiopsida and Bryopsida.

Table 7. Plants Experimentally Known to Produce Naringenin

Plant Species		
Amborella trichopoda	Durio zibethinus	Prunus avium
Arabidopsis thaliana	Eucalyptus grandis	Prunus mume
Asparagus officinalis	Fragaria vesca	Prunus persica
Brachypodium distachyon	Glycine max	Selaginella moellendorffii
Brassica napus	Gossypium hirsutum	Solanum lycopersicum

Brassica oleracea	Juglans regia	Solanum tuberosum
Brassica rapa	Momordica charantia	Sorghum bicolor
Cajanus cajan	Musa acuminata	Spinacia oleracea
Carica papaya	Nelumbo nucifera	Theobroma cacao
Capsicum annuum	Olea europaea v. sylvestris	Vigna angularis
Chenopodium quinoa	Oryza sativa	Vigna radiata
Citrus sinensis	Phaseolus vulgaris	Vitis vinifera
Cucumis melo	Phoenix dactylifera	
Cynara cardunculus var. scolymus	Physcomitrella patens subsp. patens	

Eriodictyol. The plants that were experimentally known to produce eriodictyol can be found in Table 8. Only eleven species of plants were experimentally known to produce eriodictyol and all of them were part of the Magnoliopsida class of plants.

Table 8. Plants Experimentally Known to Produce Epicatechin

Plant Species
Amborella trichopoda
Arabidopsis thaliana
Eucalyptus grandis
Gossypium hirsutum
Malus domestica
Prunus avium
Prunus persica
Sorghum bicolor
Theobroma cacao
Vigna radiata
Ziziphus jujuba

Phylogenetic Trees

Catechin. The phylogenetic tree obtained for catechin can be seen in Figure 24. There were thirty-four species of plants that were in common between the literature search data of catechin and that obtained from the KEGG script and there was a total of forty-four

species being examined in Figure 24. All of the species did still fall within the Magnoliopsida class of plants.

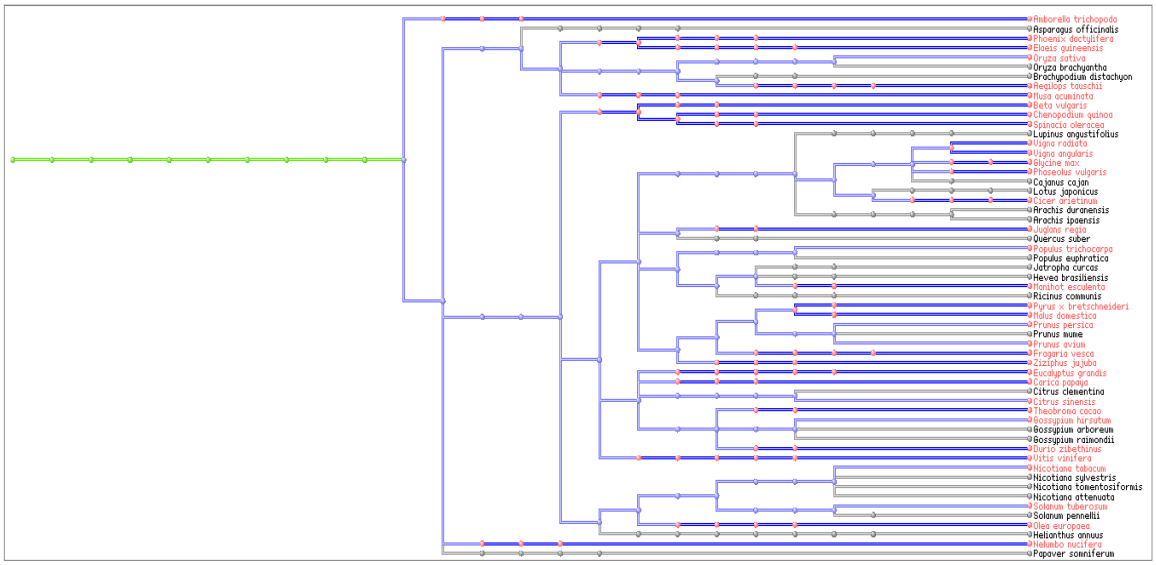

Figure 24. Phylogenetic tree for catechin.

The names highlighted in red were the species experimentally known to produce catechin, while the rest only appeared in the data obtained from the KEGG script. While all of the plants did fall within the Magnoliopsida class, they could be broken into two primary orders, Mesangiospermae and Amborellales. There were species that appeared in the experimentally know data set, but not within the data set obtained through the KEGG script.

Epicatechin. The phylogenetic tree obtained for epicatechin can be seen in Figure 25. There were sixty-six species being examined in this phylogenetic tree, and all of them fell within the Magnoliopsida class of plants.

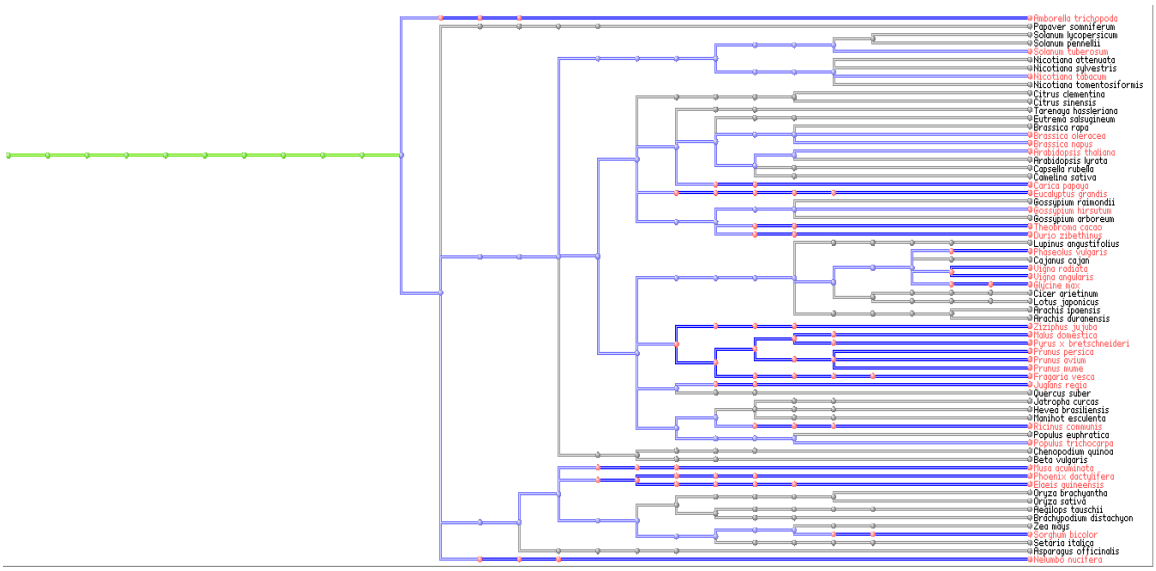

Figure 25. Phylogenetic tree for epicatechin.

The names highlighted in red were the species experimentally known to produce epicatechin, while rest only appeared in the data obtained from the KEGG database via the script. All of them were from the same class of plants, Magnoliopsida, and similar to catechin, they were from two different orders, Mesangiospermae and Amborellales. However, there were species that appeared in the experimentally known data set, but not the species determined from the KEGG script.

Naringenin. The phylogenetic tree for naringenin can be seen in Figure 26. In total, there were eighty-six species of plants examined in Figure 26, where all forty of the species experimentally known to produce naringenin appeared in the KEGG script data set. However, not all of the species were part of the Magnoliopsida class, there were two species of mosses, *Selaginella moellendorffii* and *Physcomitrella patens* subsp. *patens*, that were part of different classes.

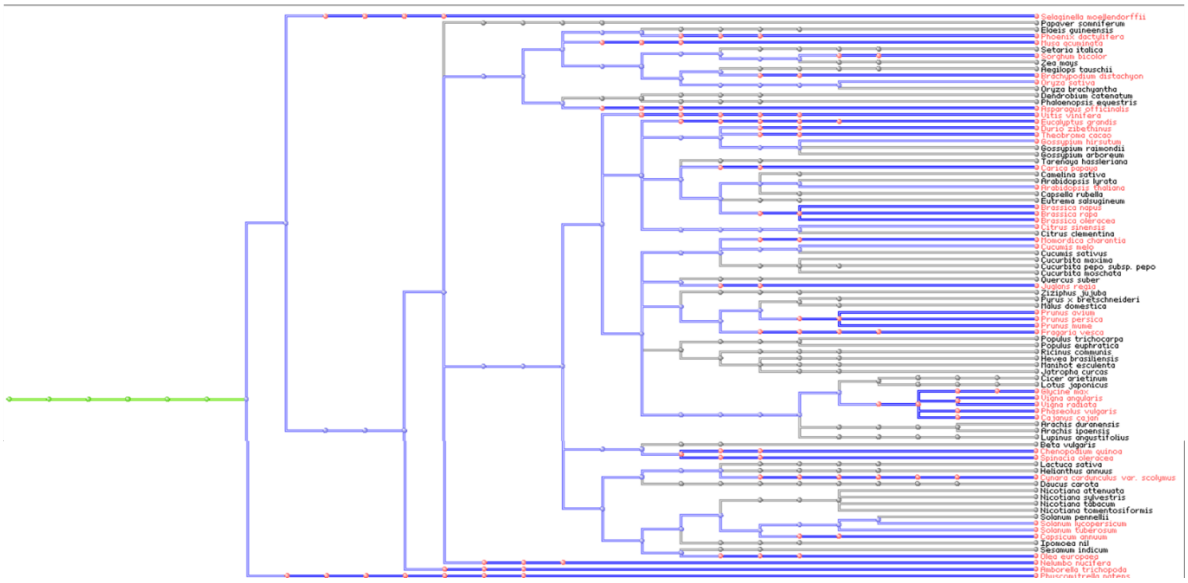

Figure 26. Phylogenetic tree for naringenin.

The names highlighted in red were the species experimentally known to produce naringenin, while rest only appeared in the data obtained from the KEGG database via the script. All of the species of plants experimentally known to produce naringenin appeared in the list of plants that were determined to theoretically produce it. The two species, *Selaginella moellendorffii* and *Physcomitrella patens* subsp. *patens*, were part of nonflowering plants that were experimentally known and theoretically determined to produce naringenin. *Selaginella moellendorffii* is part of the Lycopodiopsida class of plants and *Physcomitrella patens* subsp. *patens* is part of the Bryopsida class of plants.

Eriodictyol. The phylogenetic tree obtained for eriodictyol can be seen in Figure 27. There were eighty-five species of plants examined in Figure 27, eleven of which were experimentally known to produce eriodictyol. Similar to naringenin, not all of the plant species were part of the Magnoliopsida class.

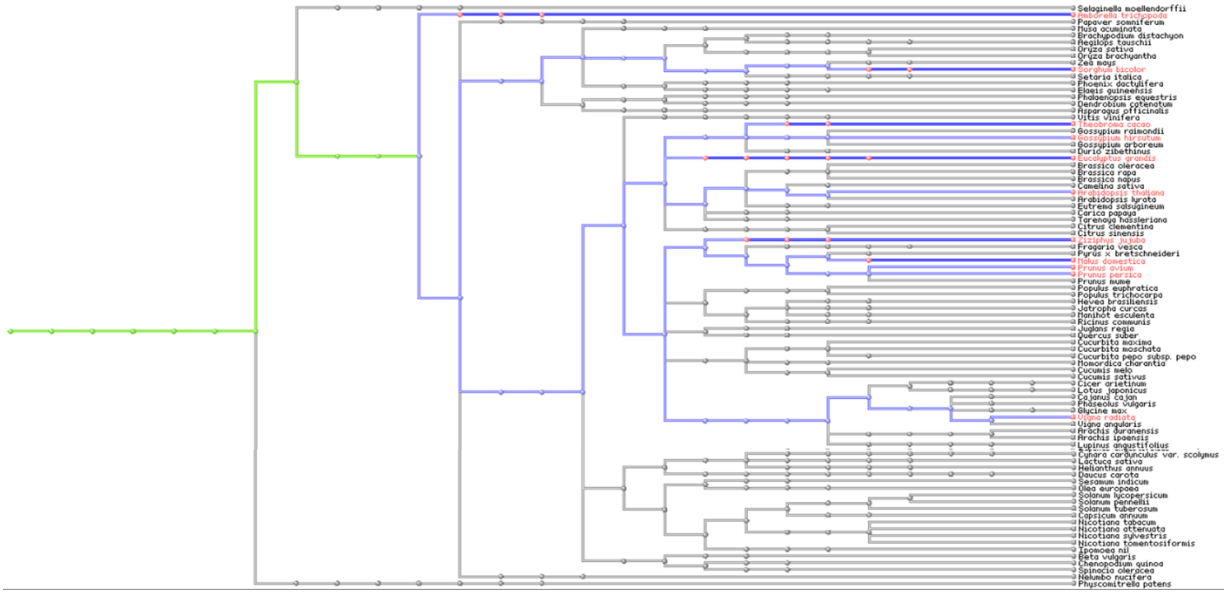

Figure 27. Phylogenetic tree for eriodictyol.

The names highlighted in red were the species experimentally known to produce eriodictyol, while rest only appeared in the data obtained from the KEGG database via the script. All of the plants that were experimentally known to produce eriodictyol appeared in the list of plants that were theoretically determined to produce from the KEGG script. Similar to the naringenin phylogenetic tree, there were two species, *Selaginella moellendorffii* and *Physcomitrella patens* subsp. *patens*, that were not part of the majority class, Magnoliopsida. Rather they fell within the two classes of Lycopodiopsida and Bryopsida, respectively.

Discussion

Catechin

Determining if using Python scripts along with data from KEGG as a plausible method of predicting if certain species of plants can produce various flavonoids was the primary goal of this project. To determine this, comparisons between what is already known and what was determined need to be made. A comparison of the two different data

sets can be seen in Figure 4 for catechin. In the Venn diagram, the red circle contains the plant species that only appeared in the data set of plants experimentally known to produce catechin, while the blue circle contains the plant species that were only present in the data set obtained from the script. Based on the comparison, ten plant species only appeared in the experimentally known data set while twenty-three appeared only in the data set from the script. At the same time only thirty-four of the one hundred plant species examined for this project appeared in both data sets. There were plant species only found in the experimentally known data set, this meant that the script could potentially miss plant species if it was used as a method of predicting what plants could synthesize catechin. This then would artificially narrow the potential candidates for any future work involving these plant species and catechin. This artificial narrowing of candidates is why this was not a plausible method of predicting what plants can synthesize catechin and the script that was used may need to be tested more or reevaluated in order to prevent this.

Epicatechin

The same comparison done for catechin was also done for epicatechin. The data sets obtained through the script and through literature searching were compared to each other in order to determine which species did and did not appear in each data set. This comparison can be seen in Figure 5. From the comparisons, five plant species were only found in the experimentally known data set that was obtained through literature searching. Thirty-four plant species were only found in the data set obtained from the script. Finally, twenty-eight species were found in both data sets. This comparison displayed that the method used to predict what plants can synthesize epicatechin came across the same issue as catechin. The data set obtained for plants theoretically

determined to produce epicatechin were missing species that were experimentally known and tested to produce epicatechin. This again, artificially narrows potential candidates for future studies involving epicatechin synthesis in plants. Thus, this indicated that the method was not plausible for the predicting what species of plants could synthesize epicatechin.

Naringenin

A comparison of the two data sets obtained for naringenin was performed in the same way the data sets of catechin and epicatechin were performed. These comparisons can be seen in Figure 6. Based on the comparisons, there were no plant species that were only present in the data set obtained from the literature searching, there were forty species that were present in both data sets, and forty-five species were only present in the data obtained from the KEGG script; this indicates that the method implemented for predicting what plants can synthesize naringenin is plausible since there were no plant species present only in the data set of plants experimentally known to produce naringenin. This along with there being many species only appearing in the script data set indicates naringenin as a possible flavonoid for future study. Finally, the presence of non-flowering plants in the data is a key piece of data to note, this is because many flavonoids were detected in plants that are part of the flowering-plant class, generally as pigments in the plants. However, if non-flowering plants have the potential of producing naringenin it could open up many research opportunities revolving around non-flowering plants as possible sources for naringenin when they were never previously considered.

Eriodictyol

Finally, the comparison performed on the other three flavonoids was performed on eriodictyol. The comparison of the two obtained data sets can be seen in Figure 7. Based on the diagram, there are no plant species that only appeared in the data set of plant species experimentally known to produce eriodictyol. At the same time, eleven plant species were found in both data sets, while seventy-four were found only in the data set obtained from the script. This indicated that method used to predict if different plant species can synthesize eriodictyol is a plausible method. This is because there are no plants being experimentally known to produce eriodictyol but was not theoretically determined to produce it. This also indicated that eriodictyol is not a very widely studied flavonoid with little literature regarding it being produced in plants. This project could beneficial to studies trying to the determine the existence of eriodictyol in plants as it gives many possible candidates that can be studied and because the data was obtained based on the presence of genes related to the enzymes involved in the synthesis of eriodictyol. Also, similar to naringenin, the presence of non-flowering plants is key to note as it can possibly lead to studying more non-flowering plants as potential sources of eriodictyol.

Comparison Discussions

Naringenin and Eriodictyol. There were no plant species that were experimentally known to produce naringenin and that were not determined to theoretically synthesize naringenin. This means that the use of the KEGG script to predict what plants can synthesize naringenin is plausible since there are no discrepancies between what is already known and what was predicted. Also, unlike catechin and epicatechin, the species of plants that were in both data sets were not all from the Magnoliopsida class, rather

there were a few species that were from different classes, specifically Lycopodiopsida and Bryopsida. This meaning that not only can flowering plants can produce naringenin, but also some mosses can as well. Similar to naringenin, there were no plant species that were experimentally known to produce eriodictyol and not found on the list of species that were determined to theoretically synthesize eriodictyol. With that being true, the use of the KEGG script to predict if varying species of plants can synthesize eriodictyol is plausible since there are no discrepancies between what is already known and what was predicted by the KEGG script. Another point to note, is that, in a similar fashion to naringenin, not all the species in the data set obtained from the KEGG script data sets were part of the Magnoliopsida class rather there were mosses from the Lycopodiopsida and Bryopsida classes that were predicted to produce eriodictyol. The data obtained for both of these flavonoids is similar mostly due to how similar the biosynthesis of these two flavonoids are. Starting from the phenylpropanoid biosynthesis pathway, the synthesis of these two flavonoids follow a very similar path with the synthesis of eriodictyol requiring a few different enzymes to go forward when compared to naringenin.

Catechin and Epicatechin. The diagrams seen in Figure 4 show that were species of plants that were not present in the list of plant species determined to theoretically synthesize catechin but were already experimentally known to synthesize it. These missing plant species could have meant that there is missing information on KEGG that has not been added to the database or that there was something wrong with the script when it was used to obtain the data. This alone would make the KEGG script an implausible method in regard to predicting if plants can synthesize catechin. However, it

should still be noted that the all of the plant species, from both data sets, were all part of the Magnoliopsida class of plants, which makes them all types of flowering plants, mean they produce flowers in one form or fashion. This is important because flavonoids as a whole are often found as pigments in plants and the flowers of plants often contain a large amount of those pigments. The data obtained for epicatechin displayed a similar trend to that of catechin. There was species of plants that were experimentally known to produce epicatechin that did not appear in the plants that were determined to theoretically produce it. This once again could have been due to missing information or errors in the script and similarly to catechin, makes the use of the KEGG script for predicting what plant species can synthesize epicatechin implausible. On a similar note to catechin, the plants in both data sets were all from the Magnoliopsida class, thus making all of them flowering plants that would have pigmented flowers of some form. The differences between the biosynthesis of catechin and epicatechin are very little, which is why the data between these two flavonoids is relatively similar. The synthesis of both catechin and epicatechin follows a very similar path with the only differences between the need of a couple different enzymes nearing the end of the reaction.

References

Akomolafe, S., Oboh, G., Oyeleye, S., Molehin, O., & Ogunsuyi, O. (2016). Phenolic composition and inhibitory ability of methanolic extract from pumpkin (cucurbita pepo L) seeds on fe-induced thiobarbituric acid reactive species in albino rat's testicular tissue in-vitro. *Journal of Applied Pharmaceutical Scie*nce, *6*(9), 115-20. doi:10.7324/JAPS.2016.60917

Akyol, H., Riciputi, Y., Capanoglu, E., Caboni, M. F., & Verardo, V. (2016). Phenolic compounds in the potato and its byproducts: An overview. *International Journal of Molecular Sciences*, *17*(6), 835. doi:10.3390/ijms17060835

Arts, I. C. W., van de Putte, B., & Hollman, P. C. H. (2000). Catechin contents of foods commonly consumed in the netherlands. 1. fruits, vegetables, staple foods, and processed foods. *Journal of Agricultural and Food Chemistry*, *48*(5), 1746-51. doi:10.1021/jf000025h

Auger, B., Marnet, N., Gautier, V., Maia-Grondard, A., Leprince, F., Renard, M., . . . Routaboul, J. (2010). A detailed survey of seed coat flavonoids in developing seeds of brassica napus L. *Journal of Agricultural and Food Chemistry*, *58*(10), 6246-56. doi:10.1021/jf903619v

Babu, M. A., Suriyakala, M. A., & Gothandam, K. M. (2012). Varietal impact on phytochemical contents and antioxidant properties of musa acuminata (banana). *Journal of Pharmaceutical Sciences and Research*, *4*(10), 1950.

Baljeet, S. Y., Roshanlal, Y., & Ritika, B. Y. (2016). Effect of cooking methods and extraction solvents on the antioxidant activity of summer squash (cucurbita pepo) vegetable extracts. *International Food Research Journal*, *23*(4), 1531.

Bhagwat, S., Haytowitz, D.B. Holden, J.M. (Ret.). (2014). USDA Database for the Flavonoid Content of Selected Foods, Release 3.1. U.S. Department of Agriculture, Agricultural Research Service. Nutrient Data Laboratory Home Page: http://www.ars.usda.gov/nutrientdata/flav

Bhandari, S. R., & Lee, J. G. (2016). Ripening-dependent changes in antioxidants, color attributes, and antioxidant activity of seven tomato (solanum lycopersicum L.) cultivars. *Journal of Analytical Methods in Chemistry*, *2016*, 5498618-13. doi:10.1155/2016/5498618

Boches, P., Peterschmidt, B., & Myers, J. R. (2011). Evaluation of a subset of the solanum lycopersicum var. cerasiforme core collection for horticultural quality and fruit phenolic content. *Hortscience*, *46*(11), 1450-5. doi:10.21273/HORTSCI.46.11.1450

Buschmann, H., Reilly, K., Rodriguez, M. X., Tohme, J., & Beeching, J. R. (2000). Hydrogen peroxide and flavan-3-ols in storage roots of cassava (manihot esculenta crantz) during postharvest deterioration. *Journal of Agricultural and Food Chemistry*, *48*(11), 5522-9. doi:10.1021/jf000513p

Charoenkiatkul, S., Thiyajai, P., & Judprasong, K. (2016). Nutrients and bioactive compounds in popular and indigenous durian (durio zibethinus murr.). *Food Chemistry*, *193*, 181-6. doi:10.1016/j.foodchem.2015.02.107

Chen, H., Chen, H., Sun, K., Sun, K., Yang, Z., Yang, Z., . . . Wei, S. (2019). Identification of antioxidant and anti-α-amylase components in lotus (nelumbo nucifera, gaertn.) seed epicarp. *Applied Biochemistry and Biotechnology*, *187*(3), 677-90. doi:10.1007/s12010-018-2844-x

Chen, Y. N., Ren, X. P., Zhou, X. J., Huang, L., Huang, J. Q., Yan, L. Y., . . . Jiang, H. F. (2014). Alteration of gene expression profile in the roots of wild diploid arachis duranensis inoculated with ralstonia solanacearum. *Plant Pathology, 63*(4), 803-11. doi:10.1111/ppa.12158

Chen, Y., Chen, J., Leng, H., Duan, Y., Zhao, W., Yang, G., . . . Hu, Q. (2013). Three new flavonoids from the leaves of oriental tobacco and their cytotoxicity. *Phytochemistry Letters*, *6*(1), 144-7. doi:10.1016/j.phytol.2012.12.001

Chung, I., Oh, J., & Kim, S. (2017). Comparative study of phenolic compounds, vitamin E, and fatty acids compositional profiles in black seed-coated soybeans (glycine max (L.) merrill) depending on pickling period in brewed vinegar. *Chemistry Central Journal*, *11*(1), 1-11. doi:10.1186/s13065-017-0298-9

Ciric, A. R., Ivanovic, N., Cvijovic, M. S., Jelikic-Stankov, M., Joksovic, L., & Djurdjevic, P. T. (2014). Chemometric-assisted optimization of RP-HPLC method for determination of some bioflavonoids in brassica oleracea species and their antioxidative activity. *Food Analytical Methods, 7*(7), 1387-99. doi:10.1007/s12161-013-9761-y

Cuong, D. M., Kwon, S., Jeon, J., Park, Y. J., Park, J. S., & Park, S. U. (2018). Identification and characterization of phenylpropanoid biosynthetic genes and their accumulation in bitter melon (momordica charantia). *Molecules (Basel, Switzerland)*, *23*(2), 469. doi:10.3390/molecules23020469

Davey, M. P., Burrell, M. M., Woodward, F. I., & Quick, W. P. (2008;2007;). Population-specific metabolic phenotypes of arabidopsis lyrata ssp. petraea. *The New Phytologist*, *177*(2), 380-8. doi:10.1111/j.1469-8137.2007.02282.x

de Pascual-Teresa, S., Santos-Buelga, C., & Rivas-Gonzalo, J. C. (2000). Quantitative analysis of flavan-3-ols in spanish foodstuffs and beverages. *Journal of Agricultural and Food Chemistry*, *48*(11), 5331-7. doi:10.1021/jf000549h

Di Sotto, A., Vecchiato, M., Abete, L., Toniolo, C., Giusti, A. M., Mannina, L., . . . Di Giacomo, S. (2018). Capsicum annuum L. var. cornetto di pontecorvo PDO: Polyphenolic profile and in vitro biological activities. *Journal of Functional Foods*, *40*, 679-91. doi:10.1016/j.jff.2017.11.041

DongDong, Z., Chen, J., Li, T., Chen, F., & Qun Cui, D. (2015). Molecular survey of Tamyb10-1 genes and their association with grain colour and germinability in chinese wheat and aegilops tauschii. *Journal of Genetics*, *94*(3), 453-9. doi:10.1007/s12041-015-0559-0

Dueñas, M., Hernández, T., Estrella, I., & Fernández, D. (2009). Germination as a process to increase the polyphenol content and antioxidant activity of lupin seeds (lupinus angustifolius L.). *Food Chemistry*, *117*(4), 599-607. doi:10.1016/j.foodchem.2009.04.051

Dunn, M. (n.d.). Understand EC numbers in 5 minutes Part I: How EC numbers work. Retrieved December 2, 2019, from https://bitesizebio.com/10683/understand-ec-numbers-in-5-minutes-part-i-how-ec-numbers-work/.

El-Kholy, W. M., Soliman, T. N., & Darwish, A. M. G. (2019). Evaluation of date palm pollen (phoenix dactylifera L.) encapsulation, impact on the nutritional and functional properties of fortified yoghurt. *Plos One*, *14*(10), e0222789. doi:10.1371/journal.pone.0222789

Frary, A., Göl, D., Keleş, D., Okmen, B., Pinar, H., Siğva, H. O., . . . Doğanlar, S. (2010). Salt tolerance in solanum pennellii: Antioxidant response and related QTL. *BMC Plant Biology, 10*(1), 58. doi:10.1186/1471-2229-10-58

Gallego, A. M., Rojas, L. F., Rodriguez, H. A., Mora, C., Atehortúa, L., Urrea, A. I., . . . Pabón-Mora, N. (2019). Metabolomic profile of cacao cell suspensions growing in blue light/dark conditions with potential in food biotechnology. *Plant Cell, Tissue and Organ Culture (PCTOC), 139*(2), 275-94. doi:10.1007/s11240-019-01679-3

Ganesan, K., & Xu, B. (2017). Polyphenol-rich dry common beans (phaseolus vulgaris L.) and their health benefits. *International Journal of Molecular Sciences, 18*(11), 2331. doi:10.3390/ijms18112331

Ganesan, K., & Xu, B. (2018). A critical review on phytochemical profile and health promoting effects of mung bean (vigna radiata). *Food Science and Human Wellness, 7*(1), 11-33. doi:10.1016/j.fshw.2017.11.002

Gao, Q., Wu, C., Wang, M., Xu, B., & Du, L. (2012). Effect of drying of jujubes (ziziphus jujuba mill.) on the contents of sugars, organic acids, α-Tocopherol, β-Carotene, and phenolic compounds. *Journal of Agricultural and Food Chemistry, 60*(38), 9642-8. doi:10.1021/jf3026524

Gasperotti, M., Masuero, D., Mattivi, F., & Vrhovsek, U. (2015). Overall dietary polyphenol intake in a bowl of strawberries: The influence of fragaria spp. in nutritional studies. *Journal of Functional Foods, 18,* 1057-69. doi:10.1016/j.jff.2014.08.013

Ghasemnezhad, M., Sherafati, M., & Payvast, G. A. (2011). Variation in phenolic compounds, ascorbic acid and antioxidant activity of five coloured bell pepper (capsicum annum) fruits at two different harvest times. *Journal of Functional Foods*, *3*(1), 44-9. doi:10.1016/j.jff.2011.02.002

Günç Ergönül, P., & Aksoylu Özbek, Z. (2018). Identification of bioactive compounds and total phenol contents of cold pressed oils from safflower and camelina seeds. *Journal of Food Measurement and Characterization*, *12*(4), 2313-23. doi:10.1007/s11694-018-9848-7

Hamid, M. G., & Mohamed Nour, Abdel Azim Ahmed. (2018). Effect of different drying methods on quality attributes of beetroot (beta vulgaris) slices. *World Journal of Science, Technology and Sustainable Development*, *15*(3), 287-98. doi:10.1108/WJSTSD-11-2017-0043

Hammouda, H., Chérif, J. K., Trabelsi-Ayadi, M., Baron, A., & Guyot, S. (2013). Detailed polyphenol and tannin composition and its variability in tunisian dates (phoenix dactylifera L.) at different maturity stages. *Journal of Agricultural and Food Chemistry*, *61*(13), 3252-63. doi:10.1021/jf304614j

Hashmi, M. A., Khan, A., Hanif, M., Farooq, U., & Perveen, S. (2015). Traditional uses, phytochemistry, and pharmacology of olea europaea (olive). *Evidence-Based Complementary and Alternative Medicine : ECAM*, *2015*, 541591. doi:10.1155/2015/541591

Hemalatha, P., Bomzan, D. P., Sathyendra Rao, B. V., & Sreerama, Y. N. (2016). Distribution of phenolic antioxidants in whole and milled fractions of quinoa and

their inhibitory effects on α-amylase and α-glucosidase activities. *Food Chemistry*, *199*, 330-8. doi:10.1016/j.foodchem.2015.12.025

Hou, Z., Luo, J., & Kong, L. (2009). Medium-pressure liquid chromatography coupled to electrospray ionization mass spectrometry for separation and on-line characterization of flavonoids from asparagus officinalis. *Chromatographia*, *70*(9-10), 1447-50. doi:10.1365/s10337-009-1329-z

Huang, C. F., Chen, Y. W., Yang, C. Y., Lin, H. Y., Way, T. D., Chiang, W., & Liu, S. H. (2011). Extract of lotus leaf (nelumbo nucifera) and its active constituent catechin with insulin secretagogue activity. *Journal of Agricultural and Food Chemistry*, *59*(4), 1087-94. doi:10.1021/jf103382h

Igbinosa, O. O., Igbinosa, I. H., Chigor, V. N., Uzunuigbe, O. E., Oyedemi, S. O., Odjadjare, E. E., . . . Igbinosa, E. O. (2011). Polyphenolic contents and antioxidant potential of stem bark extracts from jatropha curcas (linn). *International Journal of Molecular Sciences*, *12*(5), 2958-71. doi:10.3390/ijms12052958

Jacobo-Valenzuela, N., Zazueta-Morales, J. D. J., Gallegos-Infante, J. A., Aguilar-Gutierrez, F., Camacho-Hernandez, I. L., Rocha-Guzman, N. E., & Gonzalez-Laredo, R. F. (2011). Chemical and physicochemical characterization of winter squash (cucurbita moschata D.). *Notulae Botanicae Horti Agrobotanici Cluj-Napoca*, *39*(1), 34. doi:10.15835/nbha3915848

Jaffri, J. M., Mohamed, S., Ahmad, I. N., Mustapha, N. M., Manap, Y. A., & Rohimi, N. (2011). Effects of catechin-rich oil palm leaf extract on normal and hypertensive

rats' kidney and liver. *Food Chemistry*, *128*(2), 433-41. doi:10.1016/j.foodchem.2011.03.050

Jiang, C., Schommer, C. K., Kim, S. Y., & Suh, D. (2006). Cloning and characterization of chalcone synthase from the moss, physcomitrella patens. *Phytochemistry*, *67*(23), 2531-40. doi:10.1016/j.phytochem.2006.09.030

Kanehisa, M. (2019). Toward understanding the origin and evolution of cellular organisms. *Protein Sci*. *28*, 1947-51

Kečkeš, S., Gašić, U., Veličković, T. Ć., Milojković-Opsenica, D., Natić, M., & Tešić, Ž. (2013). The determination of phenolic profiles of serbian unifloral honeys using ultra-high-performance liquid chromatography/high resolution accurate mass spectrometry. *Food Chemistry*, *138*(1), 32-40. doi:10.1016/j.foodchem.2012.10.025

Khallouki, F., Ricarte, I., Breuer, A., & Owen, R. W. (2018). Characterization of phenolic compounds in mature moroccan medjool date palm fruits (phoenix dactylifera) by HPLC-DAD-ESI-MS. *Journal of Food Composition and Analysis*, *70*, 63-71. doi:10.1016/j.jfca.2018.03.005

Kim, E., Kim, S., Kim, J., Lee, B., Lee, O., Park, S., . . . Chung, I. (2013). Variation and correlation analysis of phenolic compounds in mungbean (vigna radiata L.) varieties. *Food Chemistry*, *141*(3), 2988-97. doi:10.1016/j.foodchem.2013.05.060

Kim, M. Y., Lee, S. H., Lee, Y. R., Lee, J., Jang, G. Y., Li, M., & Jeong, H. S. (2017). Changes of phenolic-acids and vitamin E profiles on germinated rough rice (oryza sativa L.) treated by high hydrostatic pressure. *Food Chemistry*, *217*, 106-11. doi:10.1016/j.foodchem.2016.08.069

Kim, Y. J., Kim, Y. B., Li, X., Choi, S. R., Park, S., Park, J. S., . . . Park, S. U. (2015). Accumulation of phenylpropanoids by white, blue, and red light irradiation and their organ-specific distribution in chinese cabbage (brassica rapa ssp. pekinensis). *Journal of Agricultural and Food Chemistry*, *63*(30), 6774-80. doi:10.1021/acs.jafc.5b02086

Kitamura, S., Matsuda, F., Tohge, T., Yonekura-Sakakibara, K., Yamazaki, M., Saito, K., & Narumi, I. (2010). Metabolic profiling and cytological analysis of proanthocyanidins in immature seeds of arabidopsis thaliana flavonoid accumulation mutants. *The Plant Journal, 62*(4), 549-59. doi:10.1111/j.1365-313X.2010.04174.x

Klejdus, B., Lojková, L., Plaza, M., Šnóblová, M., & Štěrbová, D. (2010). Hyphenated technique for the extraction and determination of isoflavones in algae: Ultrasound-assisted supercritical fluid extraction followed by fast chromatography with tandem mass spectrometry. *Journal of Chromatography A*, *1217*(51), 7956-65. doi:10.1016/j.chroma.2010.07.020

Kong, C., Zhang, S., Li, Y., Xia, Z., Yang, X., Meiners, S. J., & Wang, P. (2018). Plant neighbor detection and allelochemical response are driven by root-secreted signaling chemicals. *Nature Communications*, *9*(1), 3867-9. doi:10.1038/s41467-018-06429-1

Koopman, F. W., Beekwilder, J., Crimi, B., Van Houwelingen, A., Hall, R. D., Bosch, D., . . . Daran, J. M. (2012). De novo production of the flavonoid naringenin in engineered saccharomyces cerevisiae. *Microbial Cell Factories*, *11*(1), 155. doi:10.1186/1475-2859-11-155

Koprivica, M. R., Trifković, J. Đ., Dramićanin, A. M., Gašić, U. M., Akšić, M. M. F., & Milojković-Opsenica, D. M. (2018). Determination of the phenolic profile of peach (prunus persica L.) kernels using UHPLC–LTQ OrbiTrap MS/MS technique. *European Food Research and Technology*, *244*(11), 2051-64. doi:10.1007/s00217-018-3116-2

Kovinich, N., Saleem, A., Arnason, J. T., & Miki, B. (2012). Identification of two anthocyanidin reductase genes and three red-brown soybean accessions with reduced anthocyanidin reductase 1 mRNA, activity, and seed coat proanthocyanidin amounts. *Journal of Agricultural and Food Chemistry*, *60*(2), 574-84. doi:10.1021/jf2033939

Kumar, S. (2018). *Secondary metabolite and functional food components: Role in health and disease*. New York: Nova Biomedical.

Kumar, S., & Pandey, A. K. (2013). Chemistry and biological activities of flavonoids: An overview. *The Scientific World Journal*, *2013*, 162750-16. doi:10.1155/2013/162750

Lei, Z., Zhou, C., Ji, X., Wei, G., Huang, Y., Yu, W., . . . Qiu, Y. (2018). Transcriptome analysis reveals genes involved in flavonoid biosynthesis and accumulation in dendrobium catenatum from different locations. *Scientific Reports*, *8*(1), 6373-16. doi:10.1038/s41598-018-24751-y

Li, X., Qin, J., Wang, Q., Wu, X., Lang, C., Pan, H., . . . Gao, M. (2011). Metabolic engineering of isoflavone genistein in brassica napus with soybean isoflavone synthase. *Plant Cell Reports*, *30*(8), 1435-42. doi:10.1007/s00299-011-1052-8

Liew, S. S., Ho, W. Y., Yeap, S. K., & Sharifudin, S. A. B. (2018). Phytochemical composition and in vitro antioxidant activities of citrus sinensis peel extracts. *Peerj*, *6*, e5331. doi:10.7717/peerj.5331

Lim T.K. (2012) Durio zibethinus. *Edible Medicinal and Non-Medicinal Plants*. Springer, Dordrecht

Liu, R., Cai, Z., & Xu, B. (2017). Characterization and quantification of flavonoids and saponins in adzuki bean (vigna angularis L.) by HPLC–DAD–ESI–MSn analysis. *Chemistry Central Journal*, *11*(1), 1-17. doi:10.1186/s13065-017-0317-x

Liu, Y., Feng, S., Song, L., He, G., Chen, M., & Huang, D. (2013). Secondary metabolites in durian seeds: Oligomeric proanthocyanidins. *Molecules (Basel, Switzerland)*, *18*(11), 14172-85. doi:10.3390/molecules181114172

Luo, X., Cui, J., Zhang, H., & Duan, Y. (2018). Subcritical water extraction of polyphenolic compounds from sorghum (sorghum bicolor L.) bran and their biological activities. *Food Chemistry*, *262*, 14-20. doi:10.1016/j.foodchem.2018.04.073

Mallek-Ayadi, S., Bahloul, N., & Kechaou, N. (2017). Characterization, phenolic compounds and functional properties of cucumis melo L. peels. *Food Chemistry*, *221*, 1691-7. doi:10.1016/j.foodchem.2016.10.117

McNulty, J., Nair, J. J., Bollareddy, E., Keskar, K., Thorat, A., Crankshaw, D. J., . . . Ejim, L. (2009). Isolation of flavonoids from the heartwood and resin of prunus avium and some preliminary biological investigations. *Phytochemistry*, *70*(17), 2040-6. doi:10.1016/j.phytochem.2009.08.018

Mrázová, A., Belay, S. A., Eliášová, A., Perez-Delgado, C., Kaducová, M., Betti, M., . . . Paľove-Balang, P. (2017). Expression, activity of phenylalanine-ammonia-lyase and accumulation of phenolic compounds in lotus japonicus under salt stress. *Biologia, 72*(1), 36-42. doi:10.1515/biolog-2017-0001

Nix, A., Paull, C. A., & Colgrave, M. (2015). The flavonoid profile of pigeonpea, cajanus cajan: A review. *Springerplus, 4*(1), 125. doi:10.1186/s40064-015-0906-x

Nix, A., Paull, C., & Colgrave, M. (2017). Flavonoid profile of the cotton plant, gossypium hirsutum: A review. *Plants (Basel, Switzerland), 6*(4), 43. doi:10.3390/plants6040043

Nurraihana, H., Wan Rosli, W. I., Sabreena, S., & Norfarizan-Hanoon, N. A. (2018). Optimisation extraction procedure and identification of phenolic compounds from fractional extract of corn silk (zea mays hair) using LC-TOF/MS system. *Journal of Food Measurement and Characterization, 12*(3), 1852-62. doi:10.1007/s11694-018-9799-z

Oboh, G., Olabiyi, A. A., & Akinyemi, A. J. (2013). Inhibitory effect of aqueous extract of different parts of unripe pawpaw (carica papaya) fruit on Fe2+-induced oxidative stress in rat pancreas in vitro. *Pharmaceutical Biology, 51*(9), 1165-74. doi:10.3109/13880209.2013.782321

Oertel, A., Matros, A., Hartmann, A., Arapitsas, P., Dehmer, K. J., Martens, S., & Mock, H. (2017). Metabolite profiling of red and blue potatoes revealed cultivar and tissue specific patterns for anthocyanins and other polyphenols. *Planta, 246*(2), 281-97. doi:10.1007/s00425-017-2718-4

Ortuño, A., Díaz, L., Alvarez, N., Porras, I., García-Lidón, A., & Del Río, J. A. (2011). Comparative study of flavonoid and scoparone accumulation in different citrus species and their susceptibility to penicillium digitatum. *Food Chemistry*, *125*(1), 232-9. doi:10.1016/j.foodchem.2010.09.012

Panche, A. N., Diwan, A. D., & Chandra, S. R. (2016). Flavonoids: an overview. *Journal of Nutritional Science,* 5. doi: 10.1017/jns.2016.41

Peng, M., Shahzad, R., Gul, A., Subthain, H., Shen, S., Lei, L., . . . Luo, J. (2017). Differentially evolved glucosyltransferases determine natural variation of rice flavone accumulation and UV-tolerance. *Nature Communications, 8*(1), 1975-12. doi:10.1038/s41467-017-02168-x

Perestrelo, R., Lu, Y., Santos, S. A. O., Silvestre, A. J. D., Neto, C. P., Câmara, J. S., & Rocha, S. M. (2012). Phenolic profile of sercial and tinta negra vitis vinifera L. grape skins by HPLC–DAD–ESI-MSn. *Food Chemistry*, *135*(1), 94-104. doi:10.1016/j.foodchem.2012.04.102

Preuß, A., Stracke, R., Weisshaar, B., Hillebrecht, A., Matern, U., & Martens, S. (2009). Arabidopsis thaliana expresses a second functional flavonol synthase. *FEBS Letters*, *583*(12), 1981-6. doi:10.1016/j.febslet.2009.05.006

Quintero-Soto, M. F., Saracho-Peña, A. G., Chavez-Ontiveros, J., Garzon-Tiznado, J. A., Pineda-Hidalgo, K. V., Delgado-Vargas, F., & Lopez-Valenzuela, J. A. (2018). Phenolic profiles and their contribution to the antioxidant activity of selected chickpea genotypes from mexico and ICRISAT collections. *Plant Foods for Human Nutrition*, *73*(2), 122-9. doi:10.1007/s11130-018-0661-6

Roda, A. L., Oldham, N. J., Svatos, A., & Baldwin, I. T. (2003). Allometric analysis of the induced flavonols on the leaf surface of wild tobacco (nicotiana attenuata). *Phytochemistry*, *62*(3), 527-36. doi:10.1016/S0031-9422(02)00608-8

Rodrigues, C., Nicácio, A., Jardim, I., Visentainer, J., & Maldaner, L. (2019). Determination of phenolic compounds in red sweet pepper (capsicum annuum L.) using a modified QuEChERS method and UHPLC-MS/MS analysis and its relation to antioxidant activity. *Journal of the Brazilian Chemical Society*, doi:10.21577/0103-5053.20190018

Saitou, N., & SpringerLink (Online service). (2013;2014;). *Introduction to evolutionary genomics* (1st 2013.;2013; ed.). London: Springer London. doi:10.1007/978-1-4471-5304-7

San, B., & Yildirim, A. N. (2010). Phenolic, alpha-tocopherol, beta-carotene and fatty acid composition of four promising jujube (ziziphus jujuba miller) selections. *Journal of Food Composition and Analysis*, *23*(7), 706-10. doi:10.1016/j.jfca.2010.02.008

Sánchez-Rabaneda F, Jáuregui O, Casals I, Andrés-Lacueva C, Izquierdo-Pulido M & Lamuela-Raventós RM (2003) Liquid chromatographic/electrospray ionization tandem mass spectrometric study of the phenolic composition of cocoa (Theobroma cacao). *J Mass Spectrom*, *38*, 35–42. doi: 10.1002/jms.395

Santos, S. A. O., Vilela, C., Freire, C. S. R., Neto, C. P., & Silvestre, A. J. D. (2013). Ultra-high performance liquid chromatography coupled to mass spectrometry applied to the identification of valuable phenolic compounds from eucalyptus

wood. *Journal of Chromatography B, 938*, 65-74. doi:10.1016/j.jchromb.2013.08.034

Sanz, M., Cadahía, E., Esteruelas, E., Muñoz, A. M., Fernández De Simón, B., Hernández, T., & Estrella, I. (2010). Phenolic compounds in cherry (prunus avium) heartwood with a view to their use in cooperage. *Journal of Agricultural and Food Chemistry*, *58*(8), 4907-14. doi:10.1021/jf100236v

Serra, A. T., Seabra, I. J., Braga, M. E. M., Bronze, M. R., de Sousa, H. C., & Duarte, C. M. M. (2010). Processing cherries (prunus avium) using supercritical fluid technology. part 1: Recovery of extract fractions rich in bioactive compounds. *The Journal of Supercritical Fluids*, *55*(1), 184-91. doi:10.1016/j.supflu.2010.06.005

Sharma, N., Mishra, K. P., Chanda, S., Bhardwaj, V., Tanwar, H., Ganju, L., . . . Singh, S. B. (2019). Evaluation of anti-dengue activity of carica papaya aqueous leaf extract and its role in platelet augmentation. *Archives of Virology*, *164*(4), 1095-110. doi:10.1007/s00705-019-04179-z

Sharma, S., Saxena, D. C., & Riar, C. S. (2015). Antioxidant activity, total phenolics, flavonoids and antinutritional characteristics of germinated foxtail millet (setaria italica). *Cogent Food & Agriculture*, *1*(1) doi:10.1080/23311932.2015.1081728

Shin, E. J., Hur, H. J., Sung, M. J., Park, J. H., Yang, H. J., Kim, M. S., . . . Hwang, J. (2013). Ethanol extract of the prunus mume fruits stimulates glucose uptake by regulating PPAR-γ in C2C12 myotubes and ameliorates glucose intolerance and fat accumulation in mice fed a high-fat diet. *Food Chemistry*, *141*(4), 4115-21. doi:10.1016/j.foodchem.2013.06.059

Singh, S. (2016). Enhancing phytochemical levels, enzymatic and antioxidant activity of spinach leaves by chitosan treatment and an insight into the metabolic pathway using DART-MS technique. *Food Chemistry, 199*, 176-84. doi:10.1016/j.foodchem.2015.11.127

Slack, J. M. W. (2014). *Genes: A very short introduction* (First ed.). London: Oxford University Press.

Sumczynski, D., Kotásková, E., Družbíková, H., & Mlček, J. (2016). Determination of contents and antioxidant activity of free and bound phenolics compounds and in vitro digestibility of commercial black and red rice (oryza sativa L.) varieties. *Food Chemistry, 211*, 339-46. doi:10.1016/j.foodchem.2016.05.081

Svensson, L., Sekwati-Monang, B., Lutz, D. L., Schieber, A., & Gänzle, M. G. (2010). Phenolic acids and flavonoids in nonfermented and fermented red sorghum (sorghum bicolor (L.) moench). *Journal of Agricultural and Food Chemistry, 58*(16), 9214-20. doi:10.1021/jf101504v

Tamasi, G., Baratto, M. C., Bonechi, C., Byelyakova, A., Pardini, A., Donati, A., . . . Rossi, C. (2019). Chemical characterization and antioxidant properties of products and by-products from olea europaea L. *Food Science & Nutrition, 7*(9), 2907-20. doi:10.1002/fsn3.1142

Tang, X., Tang, P., & Liu, L. (2017). Molecular structure-affinity relationship of flavonoids in lotus leaf (nelumbo nucifera gaertn.) on binding to human serum albumin and bovine serum albumin by spectroscopic method. *Molecules (Basel, Switzerland), 22*(7), 1036. doi:10.3390/molecules22071036

Terpinc, P., Polak, T., Makuc, D., Ulrih, N. P., & Abramovič, H. (2012). The occurrence and characterisation of phenolic compounds in camelina sativa seed, cake and oil. *Food Chemistry, 131*(2), 580-9. doi:10.1016/j.foodchem.2011.09.033

Thiruvengadam, M., & Chung, I. (2015). Selenium, putrescine, and cadmium influence health-promoting phytochemicals and molecular-level effects on turnip (brassica rapa ssp. rapa). *Food Chemistry, 173*, 185-93. doi:10.1016/j.foodchem.2014.10.012

Thiruvengadam, M., Praveen, N., Maria John, K. M., Yang, Y., Kim, S., & Chung, I. (2014). Establishment of momordica charantia hairy root cultures for the production of phenolic compounds and determination of their biological activities. *Plant Cell, Tissue and Organ Culture (PCTOC), 118*(3), 545-57. doi:10.1007/s11240-014-0506-4

Thorne, R. F., & Reveal, J. L. (2007). An updated classification of the class magnoliopsida ("Angiospermae"). *The Botanical Review, 73*(2), 67-181. doi:10.1663/0006-8101(2007)73[67:AUCOTC]2.0.CO;2

Tsanova-Savova, S., Ribarova, F., & Gerova, M. (2005). (+)-catechin and (−)-epicatechin in bulgarian fruits. *Journal of Food Composition and Analysis, 18*(7), 691-8. doi:10.1016/j.jfca.2004.06.008

Valiñas, M. A., Lanteri, M. L., ten Have, A., & Andreu, A. B. (2017). Chlorogenic acid, anthocyanin and flavan-3-ol biosynthesis in flesh and skin of andean potato tubers (solanum tuberosum subsp. andigena). *Food Chemistry, 229*, 837-46. doi:10.1016/j.foodchem.2017.02.150

Vinha, A. F., Barreira, J. C. M., Costa, A. S. G., & Oliveira, M. Beatriz P. P. (2016). A new age for quercus spp. fruits: Review on nutritional and phytochemical composition and related biological activities of acorns. *Comprehensive Reviews in Food Science and Food Safety*, *15*(6), 947-81. doi:10.1111/1541-4337.12220

Voet, D., Voet, J. G., & Pratt, C. W. (2016). *Fundamentals of Biochemistry: Life at the Molecular Level*. United States: Wiley & Sons, Inc.

Vu, D. C., Vo, P. H., Coggeshall, M. V., & Lin, C. (2018). Identification and characterization of phenolic compounds in black walnut kernels. *Journal of Agricultural and Food Chemistry*, *66*(17), 4503-11. doi:10.1021/acs.jafc.8b01181

Wang, L., Jiang, Y., Yuan, L., Lu, W., Yang, L., Karim, A., & Luo, K. (2013). Isolation and characterization of cDNAs encoding leucoanthocyanidin reductase and anthocyanidin reductase from populus trichocarpa. *PloS One*, *8*(5), e64664. doi:10.1371/journal.pone.0064664

Woo, K. S., Kim, H., Lee, J. H., Lee, B. W., Lee, Y. Y., Lee, B. K., & Ko, J. Y. (2018). Quality characteristics and antioxidant activities of Rice/Adzuki bean mixtures cooked using two different methods. *Journal of Food Quality*, *2018*, 1-9. doi:10.1155/2018/4874795

Wu, S., Wilson, A. E., Chang, L., & Tian, L. (2019). Exploring the phytochemical landscape of the early-diverging flowering plant amborella trichopoda baill. *Molecules*, *24*(21), 3814. doi:10.3390/molecules24213814

Xu, X., Tian, H., He, M., Gebretsadik, K., Qi, X., Xu, Q., & Chen, X. (2019). Changes in catechin contents and expression of catechin biosynthesis-associated genes during

early cucumber fruit development. *Acta Physiologiae Plantarum, 41*(8), 1-9. doi:10.1007/s11738-019-2925-7

Yan, M., Chen, M., Zhou, F., Cai, D., Bai, H., Wang, P., . . . Ma, Q. (2019). Separation and analysis of flavonoid chemical constituents in flowers of juglans regia L. by ultra-high-performance liquid chromatography-hybrid quadrupole time-of-flight mass spectrometry. *Journal of Pharmaceutical and Biomedical Analysis, 164*, 734-41. doi:10.1016/j.jpba.2018.11.029

Yi, T. G., Yeoung, Y. R., Choi, I., & Park, N. (2019). Transcriptome analysis of asparagus officinalis reveals genes involved in the biosynthesis of rutin and protodioscin. *PloS One, 14*(7), e0219973. doi:10.1371/journal.pone.0219973

Yıldırım, F., Yıldırım, F., Yıldırım, A. N., Yıldırım, A. N., San, B., San, B., . . . Ercişli, S. (2016). The relationship between growth vigour of rootstock and phenolic contents in apple (malus × domestica). *Erwerbs-Obstbau, 58*(1), 25-9. doi:10.1007/s10341-015-0253-7

Yobi, A., Wone, B. W. M., Xu, W., Alexander, D. C., Guo, L., Ryals, J. A., . . . Cushman, J. C. (2012). Comparative metabolic profiling between desiccation-sensitive and desiccation-tolerant species of selaginella reveals insights into the resurrection trait. *The Plant Journal, 72*(6), 983-99. doi:10.1111/tpj.12008

Zahir, A. A., Zahir, A. A., Rahuman, A. A., Rahuman, A. A., Bagavan, A., Bagavan, A., . . . Elango, G. (2012). Evaluation of medicinal plant extracts and isolated compound epicatechin from ricinus communis against paramphistomum cervi. *Parasitology Research, 111*(4), 1629-35. doi:10.1007/s00436-011-2589-8

Zdunic, G., Menkovic, N., Jadranin, M., Novakovic, M., Savikin, K., & Zivkovic, J. (2016). Phenolic compounds and carotenoids in pumpkin fruit and related traditional products. *Hemijska Industrija*, *70*(4), 429-33. doi:10.2298/HEMIND150219049Z

Zeb, A. (2016). Phenolic Profile and antioxidant activity of melon (cucumis melo L.) seeds from pakistan. *Foods (Basel, Switzerland)*, *5*(4), 67. doi:10.3390/foods5040067

Zeb, A., Muhammad, B., & Ullah, F. (2017). Characterization of sesame (sesamum indicum L.) seed oil from pakistan for phenolic composition, quality characteristics and potential beneficial properties. *Journal of Food Measurement and Characterization*, *11*(3), 1362-9. doi:10.1007/s11694-017-9514-5

Zhai, R., Liu, X., Feng, W., Chen, S., Xu, L., Wang, Z., . . . Ma, F. (2014). Different biosynthesis patterns among flavonoid 3-glycosides with distinct effects on accumulation of other flavonoid metabolites in pears (pyrus bretschneideri rehd.). *PloS One*, *9*(3), e91945. doi:10.1371/journal.pone.0091945

Zhang, X., Lin, Z., Fang, J., Liu, M., Niu, Y., Chen, S., & Wang, H. (2015). An on-line high-performance liquid chromatography–diode-array detector–electrospray ionization–ion-trap–time-of-flight–mass spectrometry–total antioxidant capacity detection system applying two antioxidant methods for activity evaluation of the edible flowers from prunus mume. *Journal of Chromatography A*, *1414*, 88-102. doi:10.1016/j.chroma.2015.08.033

Zhang, X., Sun, X., Zhao, H., Xue, M., & Wang, D. (2017). Phenolic compounds induced by bemisia tabaci and trialeurodes vaporariorum in nicotiana tabacum L. and their

relationship with the salicylic acid signaling pathway. *Arthropod-Plant Interactions, 11*(5), 659-67. doi:10.1007/s11829-017-9508-6

Zhou, J., Ma, Y., Jia, Y., Pang, M., Cheng, G., & Cai, S. (2019). Phenolic profiles, antioxidant activities and cytoprotective effects of different phenolic fractions from oil palm (elaeis guineensis jacq.) fruits treated by ultra-high pressure. *Food Chemistry*, *288*, 68-77. doi:10.1016/j.foodchem.2019.03.002

Appendix

KEGG Script

```
# -*- coding: utf-8 -*-
"""
Originally Created on Wed Apr 17 2019

@author: vmoorman and Jordan Wilson
Made using Python 2.7
KEGG.py - JW started the code to get information from KEGG about the species we were interested in - April 2019
v0p1 - VRM getting the iteration and parsing part of the code to work - April 2019
v0p2 - VRM making the count code work - April 2019
v0p3 - JW getting the file output to work, revising the species codes and adding a dictionary - April 2019
v0p4 - JW creating directory change function for Gene_data and adding additional plant species - multiple things still broken - May 2019
v0p5 - VRM cleaned up some duplicate issues, folder locations, and ensured that species are written out in full  - May 2019
v0p6 - JW creating function for master FASTA file (error occurs when running every entry) - May 2019
v0p7 - JW editing lists for master fasta function and updating fasta function - May 2019
v0p8 - VRM fixing and creating gene list for master fasta file - June 2019
v0p8p2 - JW made a fasta file for each EC#- June 2019
v1p0 - VRM cleaned up code
v1p1p1 - JW added ReadMe
v1p2 - JW added function to get plants that make Epicatechin - incorrectly
v1p3 - VRM coded in epicatechin, catechin, eriodictyol, luteolin, naringenin listed in Chemical_Data --- leutolin actually wrong
v1p3p1 - VRM tried to fix leutolin


"""

'''This are required for the script to work - it has all of the required dependencies'''
#bioservices must be installed on your computer
from bioservices.kegg import KEGG
import os
import sys
import datetime
k = KEGG()
now=datetime.datetime.now()

```

```
'''Make the necessary folders'''
foldername = os.getcwd()+ "\\" + raw_input("Input a save folder name: ")#gets current working directory and asks for user input for a new foldername
if os.path.exists(foldername): #determines if the new foldername already exists
    decision = raw_input("Warning, this folder already exists. Press the return key to continue anyway, or type anything to try again: ")
    if decision == "": #if return key hit will continue running th code, will overwrite anything in the prexisting folder
        pass
    else:
        sys.exit("Try an unused folder name next time") #stops code completely and the code will need to be restarted with a different foldername
genefoldername = foldername+"\Gene_Data" #variable used to create the folder for the gene data
fastafoldername = foldername+"\FASTA_Data" #variable used to create the folder for the fasta data
chemicalfoldername=foldername+"\Chemical_Data"
try: os.mkdir(foldername) #creates the main working folder for the code
except WindowsError: pass #stops code from making folder if this error occurs
try: os.mkdir(genefoldername) #creates folder were gene data files will be inserted
except WindowsError: pass
try: os.mkdir(fastafoldername) #creates folder where fasta files will be inserted
except WindowsError: pass
try: os.mkdir(chemicalfoldername) #creates folder where fasta files will be inserted
except WindowsError: pass


'''Python code to remove duplicate elements --- needs to be up here because of testing code '''
def Remove(listwithduplicates):
    listwithoutduplicates = [] #creates an empty list
    for item in listwithduplicates:
        if item not in listwithoutduplicates:
            listwithoutduplicates.append(item) #adds item to empty list if it's not already in the list
    return listwithoutduplicates

'''These are the list of the codes that you need to iterate over.
 If you are needing just one code, just have one item in the list, but keep it as a list.
 You must have both species codes and pathway codes for this script to work.'''
speciescode_list = ["ats", "atr", "aly", "ath", "adu", "aip", "aof", "apro", "bpg", "bvg", "bdi", "bna",
```

```
                "boe", "brp", "ccaj", "csat", "crb", "cann", "cpap", "cqi", "cre", "cvr",
"ccp", "cam", "cic", "cit",
                "csl", "cmo", "csv", "cmax", "cmos", "cpep", "cme", "ccav", "dcr",
"dct", "dzi", "egu", "egr",
                "eus", "fve", "gsl", "gmx", "gab", "ghi", "gra", "han", "hbr", "ini",
"jcu", "jre",
                "lsv", "lja", "lang", "mdm", "mesc", "mis", "mpp", "mcha", "mng",
"mus", "nnu", "nau", "nsy", "nta",
                "nto", "oeu", "obr", "dosa", "osa", "olu", "ota", "psom", "peq", "pvu",
"pda", "ppp", "pop", "peu", "pavi", "pmum",
                "pper", "pxb", "qsu", "rcu", "smo", "sind", "sita", "sly", "spen", "sot",
"sbi",
                "soe", "thj", "tcc", "var", "vra", "vvi", "vcn", "zma", "zju"]#species
codes for plants of interest,  "mtr" cut due to code errors
speciescode_dictionary = {"ats": "Aegilops tauschii", "atr": "Amborella
trichopoda", "aly": "Arabidopsis lyrata",
                    "ath": "Arabidopsis thaliana", "adu": "Arachis duranensis", "aip":
"Arachis ipaensis",
                    "aof": "Asparagus officinalis", "apro": "Auxenochlorella
protothecoides",
                    "bpg": "Bathycoccus prasinos", "bvg": "Beta vulgaris", "bdi":
"Brachypodium distachyon",
                    "bna": "Brassica napus", "boe": "Brassica oleracea", "brp":
"Brassica rapa",
                    "ccaj": "Cajanus cajan", "csat": "Camelina sativa", "crb":
"Capsella rubella",
                    "cann": "Capsicum annuum", "cpap": "Carica papaya", "cqi":
"Chenopodium quinoa",
                    "cre": "Chlamydomonas reinhardtii", "cvr": "Chlorella variabilis",
"ccp": "Chondrus crispus", "cam": "Cicer arietinum",
                    "cic": "Citrus clementina", "cit": "Citrus sinensis", "csl":
"Coccomyxa subellipsoidea", "cmo": "Cucumis melo",
                    "csv": "Cucumis sativus", "cmax": "Cucurbita maxima", "cmos":
"Cucurbita moschata",
                    "cpep": "Cucurbita pepo subsp. pepo", "cme": "Cyanidioschyzon
merolae",  "ccav": "Cynara cardunculus var. scolymus",
                    "dcr": "Daucus carota", "dct": "Dendrobium catenatum", "dzi":
"Durio zibethinus",
                    "egu": "Elaeis guineensis", "egr": "Eucalyptus grandis", "eus":
"Eutrema salsugineum",
                    "fve": "Fragaria vesca", "gsl": "Galdieria sulphuraria", "gmx":
"Glycine max", "gab": "Gossypium arboreum", "ghi": "Gossypium hirsutum",
                    "gra": "Gossypium raimondii", "han": "Helianthus annuus",
"hbr": "Hevea brasiliensis",
                    "ini": "Ipomoea nil", "jcu": "Jatropha curcas", "jre": "Juglans
regia",
```

```
                "lsv": "Lactuca sativa", "lja": "Lotus japonicus", "lang": "Lupinus angustifolius", "mdm": "Malus domestica",
                "mesc": "Manihot esculenta", "mis": "Micromonas commoda",
                "mpp": "Micromonas pusilla", "mcha": "Momordica charantia", "mng": "Monoraphidium neglectum", "mus": "Musa acuminata",
                "nnu": "Nelumbo nucifera", "nau": "Nicotiana attenuata", "nsy": "Nicotiana sylvestris", "nta": "Nicotiana tabacum",
                "nto": "Nicotiana tomentosiformis", "oeu": "Olea europaea v. sylvestris", "obr": "Oryza brachyantha",
                "dosa": "Oryza sativa japonica (RAPDB)", "osa": "Oryza sativa japonica (RefSeq)",
                "olu": "Ostreococcus lucimarinus", "ota": "Ostreococcus tauri", "psom": "Papaver somniferum",
                "peq": "Phalaenopsis equestris", "pvu": "Phaseolus vulgaris", "pda": "Phoenix dactylifera",
                "ppp": "Physcomitrella patens subsp. Patens", "pop": "Populus trichocarpa", "peu": "Populus euphratica",
                "pavi": "Prunus avium", "pmum": "Prunus mume", "pper": "Prunus persica",
                "pxb": "Pyrus x bretschneideri", "qsu": "Quercus suber", "rcu": "Ricinus communis",
                "smo": "Selaginella moellendorffii", "sind": "Sesamum indicum", "sita": "Setaria italica",
                "sly": "Solanum lycopersicum", "spen": "Solanum pennellii", "sot": "Solanum tuberosum",
                "sbi": "Sorghum bicolor", "soe": "Spinacia oleracea", "thj": "Tarenaya hassleriana",
                "tcc": "Theobroma cacao", "var": "Vigna angularis", "vra": "Vigna radiata",
                "vvi": "Vitis vinifera", "vcn": "Volvox carteri f. nagariensis", "zma": "Zea mays", "zju": "Ziziphus jujuba"}#Dictionary that defines the appropriate genus species for each code
        pathwaycode_list = ["00940", "00941", "00942", "00943", "00944"]#list of pathways of interest
        pathwaycode_dictionary = {"00940": "phenylpropanoids", "00941": "flavonoids", "00942": "anthocyanins",
                "00943": "isoflavonoids", "00944": "flavones/flavonols", "00945": "stilbenoids"}#Dictionary defining each pathway by each chemical they're responisble for, "mtr": "Medicago truncatula" cut due to code error

        pathwayID_list = [i+j for i in speciescode_list for j in pathwaycode_list] #this is the full list of every pathway and species from the above lists
        '"pathwayID_list = ["ats00940", "ats00941", "brp00940", "brp00941", "aip00940", "aip00941"] #use for testing only; comment out when actually running the script
```

```
    speciescode_list=[] #use for testing only; comment out when actually running the script
    for i in pathwayID_list: speciescode_list.append(filter(str.isalpha, i)) #use for testing only; comment out when actually running the script
    speciescode_list = Remove(speciescode_list) #use for testing only; comment out when actually running the script'''



    '''defining the function that will actually get all of the data that is required'''
    def gene_pathway_data(pathwayID):
        #print "Looking up data from: "+pathwayID #good for testing but takes up a lot of time in actuality
        entrylines_list = k.get(pathwayID).split("\n") #gets all of the data and splits it by line
        #print genes
        linecount=0
        for line in entrylines_list: #finds the places that have the genes listed
            entrylines_list[linecount]=line.strip()#removes extra unicode/spaces and replaces the original entry
            #print str(linecount) + ": " + line
            if line.startswith('GENE'): #finds where GENE is at in each entry
                #print line
                entrylines_list[linecount]=line.replace("GENE","").strip()#replaces GENE with a blank and removes extra spaces at the beginning and end
                GENElocator=linecount #gene locator is now the item in the list that has GENE
            if line.startswith('COMPOUND'): #finds where COMPOUND is in the entry
                #print line
                COMPOUNDlocator=linecount#compoundlocator is now the item in the list that begins with compound
            linecount+=1
        geneline_list= entrylines_list[GENElocator:COMPOUNDlocator] #makes a list that is just the gene entry lines
        linecount=0
        for i in geneline_list: #this section makes a list of lists that are appreciately separated
            i= i.replace("
","^*^").replace(";","^*^").replace("[","^*^").replace("]","")#makes ^*^ the signifier for splitting
            i=i.split("^*^")
            i.insert(0,speciescode_dictionary[filter(str.isalpha, pathwayID)])#pathwayID[0:3]])#replaces the species code with the genus species from the dictionary value for each key
            jcount=0
```

```
143.                for j in i: #cleans up list of lists of extra spaces at the beginning and
    end of each list
144.                    i[jcount]=j.strip()
145.                    jcount+=1
146.                geneline_list[linecount]= i #replaces the list with the new cleaned list
    of lists
147.                linecount+=1#iterates through each list in the entry
148.            return geneline_list
149.
150.        '''function that saves each file with a name that includes pathwayID  using
    the data from genes_listoflists'''
151.        def get_lists(whatlist, outname, outfolder=os.getcwd()): #whatlist should
    be a list of lists, outname should include an appropriate extension
152.            os.chdir(outfolder)#changes directory to current working directory
153.            writedoc = file(outname,"w") #gives write privilages for the function to
    write to the file
154.            for line in whatlist:
155.                for item in line:
156.                    item=str(item).replace("\n","") #removes the new lines in each list
    of list
157.                    if item == "":
158.                        writedoc.write("-") #if the list in the list of list is empty writes a
    dash
159.                    else:
160.                        writedoc.write(item) #writes the entry in the list of lists to the
    file
161.                    writedoc.write(",")#tab deliniated; use "," for csv files
162.                writedoc.write("\n") #creates a new line after the above portion of teh
    function
163.            writedoc.write("\n")
164.            writedoc.close() #closes file, required
165.            print  outname, "saved in: ", outfolder
166.
167.        '''iterate over pathwayID_list and use the fuction defined above'''
168.        masterlist=[]
169.        for pathwayID in pathwayID_list:
170.            notpresent=0
171.            try: currentlist=gene_pathway_data(pathwayID)  #need to ignore
    everything if there is no pathway for that species
172.            except AttributeError:
173.                #print " No data in "+ pathwayID #good for testing but takes up a lot
    of time in actuality
174.                notpresent=1
175.            if notpresent==0:
176.                masterlist.extend(currentlist)
177.
```

```
178.            #run the function that saves each file
179.            notpresent2=0
180.            try: get_gene_lists = get_lists(gene_pathway_data(pathwayID),
    "Gene_data_"+pathwayID+".csv", genefoldername) #try actually does it if it
    works
181.            except AttributeError:
182.                #print " No data in "+ pathwayID #good for testing but takes up a lot
    of time in actuality
183.                notpresent2=1
184.
185.        '''remove duplicates'''
186.        #print masterlist
187.        masterlist_nodup=Remove(masterlist)
188.        masterlist_nodup = list(filter(None,masterlist_nodup)) #removes false
    values from masterlist_nodup and turns it into a list
189.        count=0
190.        for i in masterlist_nodup: #removes false values and iterates through the
    list of lists
191.            masterlist_nodup[count]=list(filter(None,masterlist_nodup[count]))
192.            count+=1
193.
194.
195.        '''find unique EC numbers so have a generic function and run it'''
196.        def UniqueElementList(listname, locationnumber): #be careful as there are
    cases of one less item - use "last" to fix that problem here
197.            originallocationnumber=locationnumber
198.            ElementList=[]
199.            for i in listname:
200.                #print
201.                if originallocationnumber == "last": locationnumber=len(i)-1
    #assigns the string "last" to the very last list in the list of lists
202.                #print locationnumber
203.                if i[int(locationnumber)] not in ElementList: #finds unique EC
    number not in the list
204.                    ElementList.append(i[int(locationnumber)]) #adds it to the list
205.            return ElementList
206.        EC_list=UniqueElementList(masterlist_nodup, "last")
207.        #print EC_list
208.        #print speciescode_list
209.
210.        '''creating the matrix and adding up the counts'''
211.        mastercount_list = [["Species"]]
212.        mastercount_list[0].extend(EC_list) #adds the Unique EC numbers to the
    end of the mastercount list of lists
213.        for i in speciescode_list: #@#
```

```
214.            mastercount_list.append([i]) #first item in each row (but first) is the
    speciescode
215.        #print EC_list
216.        #print mastercount_list
217.
218.        icount=0
219.        for i in mastercount_list[0]: #for each EC number
220.            #print "i: "+ i
221.            if icount != 0: #first column isn't actually an EC#,
222.                EC=i
223.                #print EC
224.                jcount=0
225.                for j in mastercount_list: #for each species (using the speciecode)
226.                    #print j
227.                    if jcount != 0: #first row isn't actually a species
228.                        #print j
229.                        species=speciescode_dictionary[j[0]]
230.                        #print species
231.                        lcount=0
232.                        for l in masterlist_nodup: #iterate over the culled masterlist to
    check for matching sets
233.                            if l[0]==species and l[len(l)-1]==EC:
234.                                lcount+=1
235.                        mastercount_list[jcount].append(lcount)
236.                    #else: print "help"
237.                    jcount+=1 ####
238.            icount+=1
239.        #print mastercount_list#[0:3] #print this to test, but hide when actually
    running
240.
241.        #change mastercount_list to be actual species:
242.        icount=0
243.        for i in mastercount_list:
244.            if icount != 0: mastercount_list[icount][0]=speciescode_dictionary[i[0]]
    #replaces species code with genus specie names
245.            icount+=1
246.
247.
248.
249.        '"Make Master Files"'
250.        get_masterlist = get_lists(masterlist_nodup, "Master_List.csv",
    foldername)
251.        get_mastercount = get_lists(mastercount_list, "Master_Count.csv",
    foldername)
252.
253.
```

```
254.
255.        '''make a master fasta file saved in foldername'''
256.        rev_dict={v:k for k,v in speciescode_dictionary.iteritems()}#reverses
    dictionary keys and values
257.        genelist_frommaster=[]
258.        for i in masterlist_nodup:
259.            #print rev_dict[i[0]]+":"+i[1]
260.            genelist_frommaster.append(rev_dict[i[0]]+":"+i[1]) #combines species
    codes and gene numbers in a list to be used for the master fasta function
261.
262.        def get_master_fasta(gene):
263.            DNA_info_list=[]
264.            for gene in genelist_frommaster:
265.                #print gene
266.                gene_fasta_data = k.get(gene).split("\n") #calls the entry from KEGG
    and splits it into new lines
267.                linecount = 0
268.                for line in gene_fasta_data:
269.                    gene_fasta_data[linecount]=line.strip() #removes blank spaces at
    the beginning and end of each line
270.                    if line.startswith('ORGANISM'): #finds where the entry that begins
    with organism is
271.
    gene_fasta_data[linecount]=line.replace("ORGANISM","").strip() #remoes
    organism and removes blank spaces ta beginning and end
272.                        find_blankspace=gene_fasta_data[linecount].find(" ") #finds
    where thedouble blank space is in the organism line
273.                        organism_name= gene_fasta_data[linecount][find_blankspace:]
    #the genus specie name is left from the orginal entry
274.                    linecount+=1
275.                linecount=0
276.                for line in gene_fasta_data:
277.                    gene_fasta_data[linecount]=line.strip()
278.                    if line.startswith('ORTHOLOGY'):
279.                        gene_fasta_data[linecount]=line.strip
280.                        find_EC=line.find("EC:") #finds within the line where the EC
    number is
281.                        gene_fasta_data[linecount]=line[find_EC:-1].replace("[",
    "").replace("EC:", "") #removes the beginning bracket and EC: leaving just the
    number
282.                        EC_number="EC "+ gene_fasta_data[linecount] #adds EC back
283.
    joined_organism_EC=[">"+str(organism_name).strip()+"%"+EC_number+"%"+g
    ene.split(":")[1]]#adds > to beginning to find the beginning of each entry more
    easily, adds % between EC number and the rest of the entry to help separate the
```

```
    EC number for later and removes semicolon from the gene entry and adds the
    gene number
284.                    DNA_info_list.append(joined_organism_EC) #adds the entry to
    the blank list
285.                linecount+=1
286.            linecount=0
287.            for line in gene_fasta_data:
288.                gene_fasta_data[linecount]=line.strip()
289.                if line.startswith('NTSEQ'):
290.                    gene_fasta_data[linecount]=line.replace("NTSEQ","").strip()
291.                    NTSEQlocator=linecount
292.                linecount+=1
293.            DNA_data_list=gene_fasta_data[NTSEQlocator:]
294.            DNA_Seq=DNA_data_list[1:len(DNA_data_list)-2] #Takes just the
    DNA sequence
295.            separator=""
296.            joined_DNA_seq=[separator.join(DNA_Seq)] #combines the
    separate DNA sequence lines into one string and turns that into a single entry list
297.            DNA_info_list.append(joined_DNA_seq) #adds single entry list to
    the list of lists
298.        return DNA_info_list
299.
300.    get_lists(get_master_fasta(genelist_frommaster), "Master_FASTA.csv",
    fastafoldername)
301.
302.    masterfasta=get_master_fasta(genelist_frommaster)#should actually go
    above the first time it gets called
303.
304.    '''make a fasta file for each EC number saved in fastafoldername'''
305.    ECorderlist=[]
306.    fasta_byEC=[]
307.    icount=0
308.    for i in masterfasta:
309.        #print i
310.        if i[0].startswith(">"):
311.            isplit=i[0].split("%") #finds the EC numbers using the % added
    previously
312.            #print isplit
313.            EC=isplit[1]
314.            EC_tf="false"
315.            jcount=0
316.            for j in ECorderlist: #if the EC number is already present sets it to
    true and continues
317.                if EC==j:
318.                    ECcount=jcount
319.                    EC_tf="true"
```

```
                    jcount+=1
                if EC_tf =="false": #if the EC number is not there, it will be added to the list
                    ECorderlist.append(EC)
                    fasta_byEC.append(masterfasta[icount:icount+2])
                else:
                    fasta_byEC[ECcount].extend(masterfasta[icount:icount+2])

            icount+=1

        #print  ECorderlist
        #print fasta_byEC

        '''creates the FASTA files by EC numbers'''
        icount=0
        for i in ECorderlist:
            name=i.replace(".","p").replace("EC ","").replace(" " ,"")
            FASTAbyEC= get_lists(fasta_byEC[icount], name+".csv", fastafoldername)
            print name+".csv" + " saved in: "+fastafoldername
            icount+=1

        '''Creates ReadMe file'''
        with open(foldername+"\ReadMe.txt", "w") as ReadMe:
            ReadMe.write("KEGG_v1p1.py\n")
            ReadMe.write(now.strftime("%m-%d-%Y")+"\n")
            ReadMe.write(foldername+"\n")
            ReadMe.write("This script creates a series of files related to the genes associated with plant flavonoids from various species of plants. This script first creates the MasterCount and MasterList files; the MasterCount counts the number genes each plant species have that correspond with each EC number; while the MasterList lists every gene with number for each plant specie. These are located in "+os.getcwd())
            ReadMe.write(". The script also creates files that only contains the genes of a single plant species biochemical pathway which are located in "+genefoldername+". The script also creates a Master FASTA files which contains the DNA sequence of each gene and FASTA files organized by EC number, these are located in "+fastafoldername)
            ReadMe.close

        '''Function that determines if the plant species has the correct enzymes to produce specified chemicals'''
        def ECandor(listname):
            icount=0
            codestring="
            for i in listname:
```

```
354.                if icount>0:
355.                    codestring+=""+i+""
356.                    try:
357.                        listname[icount+1]
358.                    except IndexError:
359.                        break
360.                    codestring+=' '+listname[0]+' '
361.                icount+=1
362.            print codestring
363.            return codestring
364.
365.        '''Creates a list of the number of times the speecified enzymes appear for
        each plant specie'''
366.        masterEC_list = [["species","EC#s"]]
367.        icount=0
368.        for i in mastercount_list: #for each species
369.            #print "species :", i[0]
370.            speciesEClist=[]
371.            if icount==0:
372.                pass
373.
374.            else:
375.                jcount=0
376.                for j in i: #for EC in species
377.                    if jcount==0:
378.                        speciesEClist.append(j)
379.                        print j
380.                    else:
381.                        if str(j)=="0":
382.                            print " "+ str(j)+ " = no enzyme"
383.                            pass
384.                        else:
385.                            print mastercount_list[0][jcount] + " = " + str(j)
386.                            speciesEClist.append(mastercount_list[0][jcount])
387.                    jcount+=1
388.            masterEC_list.append(speciesEClist)
389.            icount+=1
390.
391.        #print masterEC_list
392.        masterEC_list=masterEC_list[1:]
393.
394.        phenylalanineTOcinnamicacid=["or","EC:4.3.1.24","EC:4.3.1.25"]
395.        cinnamicacidTOpcoumaroyllcoa=["and","EC:6.2.1.12","EC:1.14.14.91"]
396.        pcoumaroyllcoaTOcaffeoylcoa1=["and","EC:1.14.13.-"]
397.        pcoumaroyllcoaTOcaffeoylcoa2=["and","EC:2.3.1.133","EC:1.14.14.96"]
398.        pcoumaroyllcoaTOnaringenin=["and","EC:2.3.1.74","EC:5.5.1.6"]
```

```
399.        naringeninTOeriodictyol=["or","EC:1.14.14.81","EC:1.14.14.82"]
400.        caffeoylcoaTOeriodictyol=["and","EC:2.3.1.74"]
401.        eriodictyolTOleucocyanidin=["and","EC:1.14.11.9","EC:1.1.12.19"]
402.        leucocyanidinTOcatechin=["and","EC:1.17.1.3"]
403.        leucocyanidinTOcyanidin=["and","EC:1.14.20.4"]
404.        cyanidinTOepicatechin=["and","EC:1.3.1.77"]
405.        pcoumaroyllcoaTOnaringenin=["and","EC:2.3.1.74","EC:5.5.1.6"]
406.        eriodictyolTOluteolin=["or","EC:1.14.20.5", "EC:1.14.19.76"]
407.        naringeninTOapigenin=["or","EC:1.14.20.5","EC:1.14.19.76"]
408.        apigeninTOluteolin=["or","EC:1.14.14.81","EC:1.14.14.82"]
409.
410.
411.        epicatechinlist=[]
412.        catechinlist=[]
413.        eriodictyollist=[]
414.        naringeninlist=[]
415.        luteolinlist=[]
416.        #Be careful in making these of parenthases
417.        for i in masterEC_list:
418.            #print i
419.
420.            epicatechin= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and "
    + ECandor(cinnamicacidTOpcoumaroyllcoa)+ " and (" +
    ECandor(pcoumaroyllcoaTOcaffeoylcoa1)+" or ("+ECandor(
421.                pcoumaroyllcoaTOcaffeoylcoa2)+"))"+" and
    "+ECandor(caffeoylcoaTOeriodictyol)+" and
    "+ECandor(eriodictyolTOleucocyanidin)+" and "+ECandor(
422.                leucocyanidinTOcyanidin)+" and
    "+ECandor(cyanidinTOepicatechin)+") in i: epicatechinlist.append([i[0]])"
423.            exec epicatechin
424.
425.            catechin= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and " +
    ECandor(cinnamicacidTOpcoumaroyllcoa)+ " and (" +
    ECandor(pcoumaroyllcoaTOcaffeoylcoa1)+" or ("+ECandor(
426.                pcoumaroyllcoaTOcaffeoylcoa2)+"))"+" and
    "+ECandor(caffeoylcoaTOeriodictyol)+" and
    "+ECandor(eriodictyolTOleucocyanidin)+" and "+ECandor(
427.                leucocyanidinTOcatechin)+") in i: catechinlist.append([i[0]])"
428.            exec catechin
429.
430.            eriodictyol= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and "
    + ECandor(cinnamicacidTOpcoumaroyllcoa)+ " and (" +
    ECandor(pcoumaroyllcoaTOcaffeoylcoa1)+" or ("+ECandor(
431.                pcoumaroyllcoaTOcaffeoylcoa2)+"))"+" and
    "+ECandor(caffeoylcoaTOeriodictyol)+") in i: eriodictyollist.append([i[0]])"
432.            exec eriodictyol
```

```
433.
434.            luteolin= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and " +
    ECandor(cinnamicacidTOpcoumaroyllcoa)+ " and (" +
    ECandor(pcoumaroyllcoaTOcaffeoylcoa1)+" or ("+ECandor(
435.                pcoumaroyllcoaTOcaffeoylcoa2)+"))"+" and
    "+ECandor(caffeoylcoaTOeriodictyol)+" and
    ("+ECandor(eriodictyolTOluteolin)+")) in i: luteolinlist.append([i[0]])"
436.            exec luteolin
437.            '"luteolin= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and " +
    ECandor(cinnamicacidTOpcoumaroyllcoa)+ " and (" +
    ECandor(pcoumaroyllcoaTOcaffeoylcoa1)+" or ("+ECandor(
438.                pcoumaroyllcoaTOcaffeoylcoa2)+"))"+" and
    "+ECandor(caffeoylcoaTOeriodictyol)+" and
    "+ECandor(eriodictyolTOluteolin)+") in i: luteolinlist.append([i[0]])"
439.            exec luteolin'"#didn't work
440.
441.            naringenin= "if (("+ECandor(phenylalanineTOcinnamicacid) + ") and "
    + ECandor(cinnamicacidTOpcoumaroyllcoa) + " and "+
    ECandor(pcoumaroyllcoaTOnaringenin) +") in i: naringeninlist.append([i[0]])"
442.            exec naringenin
443.
444.        get_lists(epicatechinlist, "epicatechinspecies.txt",
    outfolder=foldername+"\Chemical_Data")
445.        get_lists(catechinlist, "catechinspecies.txt",
    outfolder=foldername+"\Chemical_Data")
446.        get_lists(eriodictyollist, "eriodictyolspecies.txt",
    outfolder=foldername+"\Chemical_Data")
447.        get_lists(luteolinlist, "luteolinspecies.txt",
    outfolder=foldername+"\Chemical_Data")
448.        get_lists(naringeninlist, "naringeninspecies.txt",
    outfolder=foldername+"\Chemical_Data")
```