
# Personality Profiling

Final Project
EECS 371: Knowledge Representation and Reasoning

Northwestern University
Winter 2018

Jeremy Midvidy, jam658
Ethan Park, jhp471

## Table of Contents

## Introduction

In the new season of the Netflix series *Black Mirror*, one episode begins with two people seemingly out on a dinner date. The audience is given zero background information on these two people or the world they live in.

The two people introduce themselves to each other; talking about their hobbies, interests, work lives, personal histories, and so on and so forth. After some initial talking, it is clear to the audience - though never explicitly said - that these two people have been set up on a blind date.

Couple on a blind date in *Black Mirror*

As the dinner progresses, it turns out these two people have a lot in common. They share all the same interests: they like the same food, they have the same favorite authors, movies, and TV shows, neither are morning people etc… As they continue to learn more about each other, they realize that they do not just have the same interests, but they share the same values. They

each want to live a fulfilling life, they each want to have kids one day, they each don't like using Facebook and such.

As the date continues, however, the couple begins to notice oddities in their environment. The waiter does not respond to their questions, and it seems like he is walking to the same tables over and over again, repeating the same line to each customer. Next, they eavesdrop the conversation of couple at the table beside them. Even more strangely, the neighboring couple keeps repeating the same five same lines to each other. These revelations indicate to the couple on a blind date, and to the audience, that something weird is afoot.

And then, the program freezes. A computer returns "99%" match, which is printed in large text on the screen. The twist is revealed: the couple were never on a blind date at all! It was all a computer simulation to gauge how these two users, based on their inputted personality settings, would interact with each other.

The point of this episode of *Black Mirror* is to highlight the inner workings of one of the the largest and most profitable fields of computer science today - personality profiling and matching.

Companies use personality profiling to match potential romantic interests, target individuals with advertisements, or recommend music, books, and movies users would like.

Regardless of application, the general process is the same: represent knowledge about the user to infer facts about his or her personality, and then use these inferences to build a "personality profile" than can then, in turn, be used to match people, products, and media.

This project will implements a version of this topical theme, utilizing representation and reasoning to build a personality profiler.

## Personality Types

To construct our personality profiler, we first had to answer two major questions about the topic of personalities. The first question revolved around the concept of *what is* a person's personality - what are its parts, how can it be divided, and how it can be broken down? The second question, in turn, revolved around the concept of *how to deduce* a person's personality - what elements about a person can tell you what information about that individual's personality?

These two fundamental questions formed the foundational logic and structure of our personality profiler. The first question gave us a way to represent a person's personality as a amalgamation of different quantifiable aspects. The second question gave us a way to infer these quantifiable parts from data.

Our research brought us to the OCEAN model of personalities. Out of every scholarly article or academic paper that we read, nearly all regarded the OCEAN model as a well-known and consensus way to understand and represent a person's personality. Therefore, we determined that using this model would be extremely useful and compatible for our needs.

OCEAN is an acronym, standing for: Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism. These are considered the "big five" elements of a human's psychological makeup. Our profiler would try and quantify a person's "big five" makeup depending on given data.

How to quantify the "big five" was the next major problem. Academic articles on the subject offered a variety of different solutions. Some recommended a survey of targeted questions designed to gauge a person's personality. Others recommended asking personal history questions about a person's age, education, employment, salary, credit score and so on and so

forth. Ultimately, we settled on research that analyzed a person's music preferences in order to infer personality characteristics.

Utilizing music has a couple clear advantages over other methods like someone's personal history or a targeted survey. First, music is an enriched form of data because it is collected from someone's personal use; they are not being asked questions (which they could falsify) or being annoyed into taking a survey (which they could fill out haphazardly.) Second, there is so much existing research that links a person's music preferences with different emotions, experiences, and personality types, that it seems like a natural choice to use for our personality engine. Third, there are so many different categories and genres of music that a person's musical tastes can tell a lot about someone, but also provide a degree of specificity and differentiation that is important when using the same type of data to analyze many different kinds of people. Fourth, and perhaps most importantly, almost everyone listens to music! Music is so popular and universal that there can be useful data from almost anyone, anywhere in the world, which seems like a good idea for this type of project.

This research led us to an important article (Gomez) that details a study of people's music preferences and how it corresponds to each personality trait. The article researched and detailed how sixteen different genres of music correlate to each of the five personality elements described by the OCEAN model. The articles findings formed the basis of the way we categorized individuals personalities for further processing. All of the findings are in the appendix at the end of this writing. The results of this article are surmised in this table:

| **Genre/Trait** [OCEAN] | **Extraversion** [E] | **Intuition** [C] | **Thinking** [N] | **Judging** [O] | **Assertiveness** [A] |
|---|---|---|---|---|---|
| **Punk** | -4 | 10 | 2 | -11 | -5 |
| **Jazz** | 8 | 8 | 0 | -3 | 7 |
| **Classical** | -2 | 12 | 4 | 4 | 5 |
| **Rock** | -2 | 13 | 2 | -6 | 0 |
| **Alt Rock** | -1 | 7 | -5 | -6 | -6 |
| **Reggae** | 9 | 0 | -6 | -5 | -4 |
| **Ambient** | 3 | 6 | -6 | -6 | -2 |
| **World** | 3 | 9 | -10 | -2 | 1 |
| **Pop** | 10 | -8 | -16 | 0 | -9 |
| **Metal** | -9 | 11 | 9 | -8 | 3 |
| **Hip-Hop** | 14 | 2 | -2 | -5 | 2 |
| **Electronica** | 9 | 4 | -4 | -5 | -1 |
| **Religious** | 6 | -6 | -12 | 5 | 1 |
| **Blues** | 7 | 9 | -4 | -3 | 7 |
| **Country** | 7 | 9 | -10 | 1 | 3 |
| **Soul** | 13 | -1 | -13 | -3 | -1 |

*note: Every OCEAN parameter can have an opposite (i.e. Extroversion/Introversion, Assertiveness/Agreeableness etc…). We chose to represent all OCEAN parameters as just one side. Therefore, negative values indicate that someone is strongly not a certain characteristic (and thus strongly the opposite of that characteristic.) This method preserves all relevant data and also makes processing easier. A score of 0, likewise, means completely neutral given a personality trait.

The logic of this table is simple. For a given playlist of an individual's music, every time a song is an instance of a certain genre (i.e. “pop”, “metal”, “electronica” etc…), each item of the “big five” traits is adjusted according to the researched correlation. Then, all that is needed to represent someone’s personality is a large enough sample of his or her music preferences (like

that person's favorite playlist or top 50 most listened to songs). After that, we would have a way to represent that person's personality based off of significant data and meaningful psychological relationships.

So, with the psychological underpinnings of our profiler established, we were able to proceed to the actual implementation of our project.

## Writing Representations

Given the input data - a 50 song playlist of a user in this context - in order to write the representations, we first needed to piece together the knowledge. That is, using the associations between various musical genres and the OCEAN personality traits that were outlined in the previous section, we created a personality profile of the user. This was done via a Python script, which is included in the appendix of this report.

The file that is passed into the python script is a simple CSV file of 50 songs, delimited by line. Then, the songs are iterated over and that user's personality score is created. For each given song, that user's personality score for his or her "big five" traits is adjusted according to the table outlined in the previous section. For example, if the given row was of genre "Pop," that user's OCEAN fields would be updated as follows:

| **Pop** | 10 [E] | -8 [C] | -16 [N] | 0 [O] | -9 [A] |
|---|---|---|---|---|---|

Also, the sign of the score indicates whether the "side" of that personality trait - for example, a positive score for extraversion would mean the user is more extroverted, while negative more introverted. All possible genres and their corresponding personality scores are hardcoded into this script, but it is possible for these to instead be facts in a knowledge base or available in some external file and be accessed or queried as necessary. To build the overall personality profile for a particular user, the script averages the all the inputted personality scores, outputting 5 averaged scores which represents the five OCEAN traits.

Once these scores are computed, it becomes possible for this knowledge to be represented. This is done via forming Cyc statements and printing them into an output text file

for later addition to the OpenCyc knowledge base. Associating a numerical score with a certain personality type using a Cyc statement allows the reasoner to represent the user's tendencies for that specific trait, and these statements also allow for this knowledge to be easily linked to another entity - a person/user, for instance. The KB-specific details of these new Cyc statements are discussed in the next section.

## Reasoning

After the logical statements have been formatted, reasoning can then be done. Our reasoner sat upon the existing OpenCyc Knowledge Base. We also used parts of CogSketch to implement a functioning version of our reasoner.

The first step was to increase the robustness and capability of the OpenCyc KB so that it can handle processing personality scores. There is already a collection in Cyc called `PersonalityTypeByPersonalityTrait`. This is a useful collection because it is already incorporated into the Cyc knowledge hierarchy, so utilizing it is a good idea because we would not need to reorganize necessary subclasses and superclasses.

The collection `PersonalityTypeByPersonalityTrait` already has some personality types, like, Brutal, Fairness, Reasonability, and more, in the KB:

```
(isa Brutal PersonalityTypeByPersonalityTrait)
(isa Compassion PersonalityTypeByPersonalityTrait)
(isa Egalitarianism PersonalityTypeByPersonalityTrait)
(isa Fairness PersonalityTypeByPersonalityTrait)
(isa Impartiality PersonalityTypeByPersonalityTrait)
(isa Reasonability PersonalityTypeByPersonalityTrait)
(isa Unfairness PersonalityTypeByPersonalityTrait)
```

Yet, these instances do not match our needs to reason with the OCEAN model. Therefore, we added instances to this collection to make it useful for our processing. In addition, we added a new collection to the KB. This collection is called `PersonalityScore`. Every instance of a `PersonalityTypeByPersonalityTrait` can have a `PersonalityScore` (ergo every type of personality can have a score associated with it.)

This structure enabled us to do further reasoning on each personality trait described by the OCEAN model.

After our addition, the `PersonalityTypeByPersonalityTrait` collection was:

- (isa Brutal PersonalityTypeByPersonalityTrait)
- (isa Compassion PersonalityTypeByPersonalityTrait)
- (isa Egalitarianism PersonalityTypeByPersonalityTrait)
- (isa Fairness PersonalityTypeByPersonalityTrait)
- (isa Impartiality PersonalityTypeByPersonalityTrait)
- (isa Reasonability PersonalityTypeByPersonalityTrait)
- (isa Unfairness PersonalityTypeByPersonalityTrait)

**in EverythingPSC:**

- (has-a PersonalityScore PersonalityTypeByPersonalityTrait)
- (isa Assertive PersonalityTypeByPersonalityTrait)
- (isa Extroverted PersonalityTypeByPersonalityTrait)
- (isa Intuitive PersonalityTypeByPersonalityTrait)
- (isa Judging PersonalityTypeByPersonalityTrait)
- (isa Thinking PersonalityTypeByPersonalityTrait)

With the updated collection, we were then able to do significant reasoning on personality scores.

Every time an individual is processed by the reasoner, that individual's personality scores get added to the collection. For example, after one iteration on one individual's music, the collection `PersonalityScore` contained that current person's profile (according to the formatted representations passed to it.) The collection look like this:

```
(has (PersonalityScore Assertiveness 3))
(has (PersonalityScore Extroverted 11))
(has (PersonalityScore Intuition 1))
(has (PersonalityScore Judging -5))
(has (PersonalityScore Thinking -4))
(has-a PersonalityScore PersonalityTypeByPersonalityTrait)
(isa PersonalityScore Collection)
```

Now the OpenCyc reasoner is able to handle instances of `PersonalityScore` as well as link them to the collection `PersonalityTypeByPersonalityTrait`. These additions make it possible to link the Cyc reasoner to the statements previously outputted by the written representations portion of our implementation.

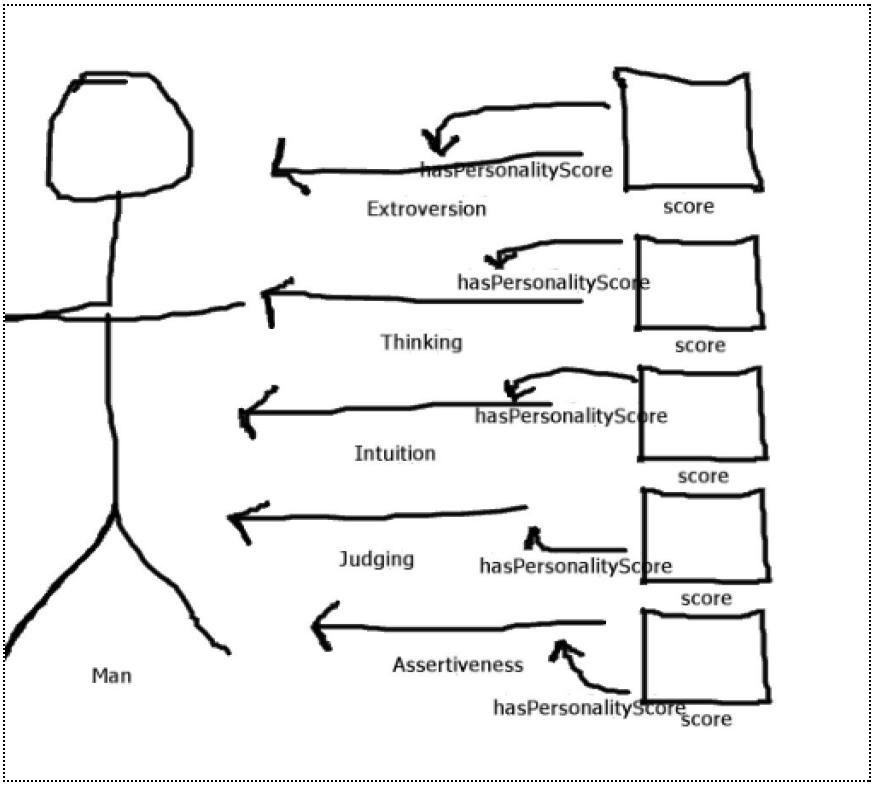
**Proof of Concept and Performance Evaluation**

---

The ultimate goal of our project is to make a more robust form of reasoning for personality profiling and matching. Now that we have added written representations to our reasoner, it should be more agile when trying to utilize personality knowledge in its reasoning, especially involving any sort of personality-based matching.

OpenCyc (or any reasoner that can be retrofitted) can know **A LOT** about someone's personality by having access to that person's OCEAN personality profile. This increase in knowledge about someone can be represented in CogSketch, where the MAN has different personality elements, and each personality element has a SCORE.

This illustrates how we have increased the robustness of the OpenCyc ability to reason:



Without our project, the reasoner would have a very difficult time accurately and meaningfully attempting to understand a person's personality and its significance to personality matching. Now with our project implemented, all any reasoning system needs is access a person's music preferences (iTunes 25 most played songs playlist would more than suffice) to determine a personality profile and proceed with the appropriate reasoning.

The evaluation of our project, therefore, boils down to this simple question: does our profiler significantly improve the accuracy of the reasoner's ability to match people and things based on personality? If so, then we have implemented an effective form of personality profiling whereby the profiler meaningfully represents important elements of the user's personality, and then uses these representations to evaluate whether that person is a match for a given target object or person. If not, then the reasoner does not accomplish this task.

We believe that our reasoner accomplishes this goal. Our profiler, with just a sample of a user's music playlists, can construct a significant (according to psychological consensus) model of that user's personality. This model is useful because it quantifies that individual's "big five" personality characteristics. From this, we write OpenCyc representations to implement this quantifiable data. With the representation added to the OpenCyc KB, we alter and add the necessary collections to the reasoner to ensure that it can process as needed. After the reasoner is equipped with our added collections and written statements, the reasoner can then evaluate a user's personality. This is a clear increase in the robustness of the reasoner since it can now consider personality traits when matching a person to a target object, and therefore we conclude that our profiler improves the reasoner's matching capability.

## Applications and Further Improvements

Previously, we mentioned personality-based matching to target objects. These target objects can be many different things: other people, products, services, and media. The ability to accurately and automatically determine someone's personality and further draw conclusions or connections based on this knowledge has powerful applications, many of which are in widespread use today in one form or another. This includes matchmaking in online dating sites and services such as eHarmony or Tinder, individually-targeted advertising and commercials on Facebook or Google, or media recommendations based on viewing and listening history on Spotify, YouTube, Amazon, Steam, Netflix, and the like.

Determining personality through music is an effective and accessible component of user profiling that is used in these manners. As mentioned previously, music is intuitive, based more on the user's natural instincts. Each song, every genre, represents a distinct facet of a user's personality. Furthermore, with the advent of large-scale music streaming and download services like Amazon Prime Music, Apple Music, Spotify, Google Play, etc..., a user's musical tendencies has become more accessible than ever for large-scale aggregation and analysis.

The accuracy and usefulness of music-based personality profiling is enhanced, furthermore, when used in tandem with profiling done with other forms of media and representations of user tendencies, such as favorite Movies, TV Shows, most watched Netflix program, and online purchases. A way to improve musical profiling specifically would be increasing the complexity of the musical classification and the personality breakdown - using more genres and more than just the OCEAN personality traits - with the tradeoff of computational cost and decreasing likelihood of exact matches, in that the more choices and

profile possibilities, the more difficult and time-consuming it is for the reasoner to perform satisfactory matching. These improvements would only add to the significance of the person's matching ability. There is a reason why dating websites like eHarmony or Match.com ask users hundreds of targeted questions about their preferences - it leads to more accurate matches!

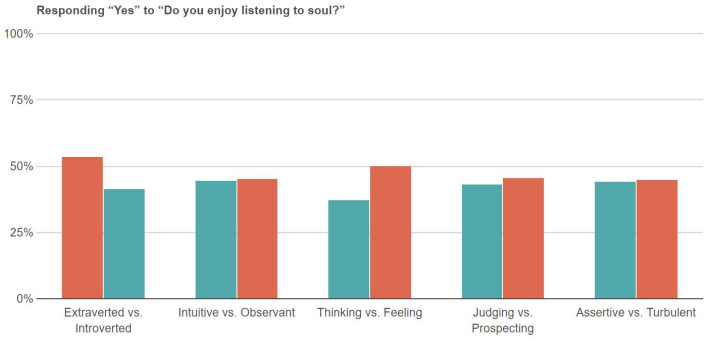
There is so much room for improvement, like incorporating more data or more personality traits, but we assess our profiler to be a successful implementation of a personality profiler and a meaningful addition to OpenCyc's reasoning abilities.

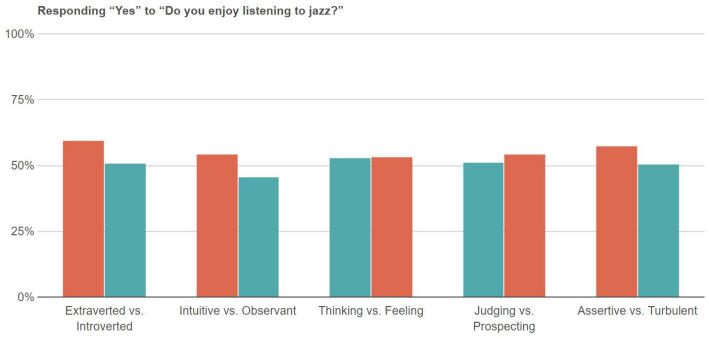
## References

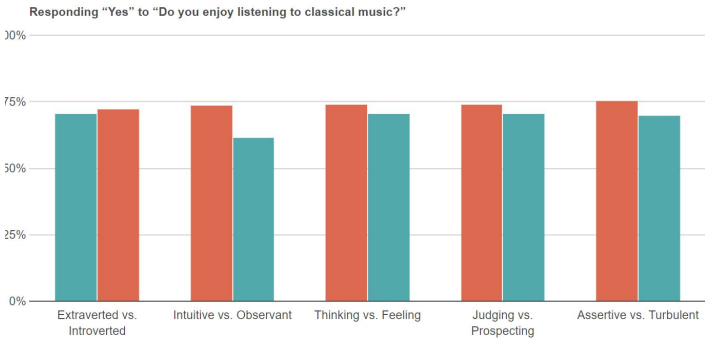
Gomez, Nathaniel. “Music Preferences by Personality Type.” 16Personalities, 2017, www.16personalities.com/articles/music-preferences-by-personality-type.

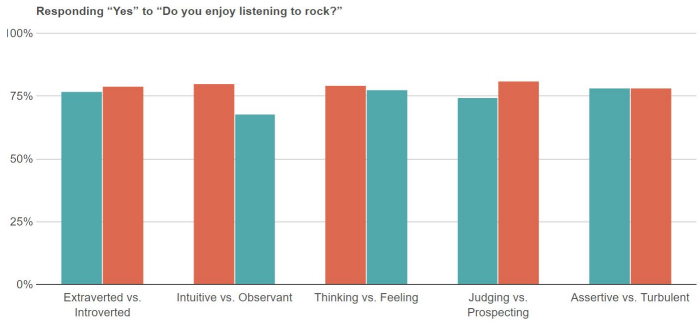
# Appendix

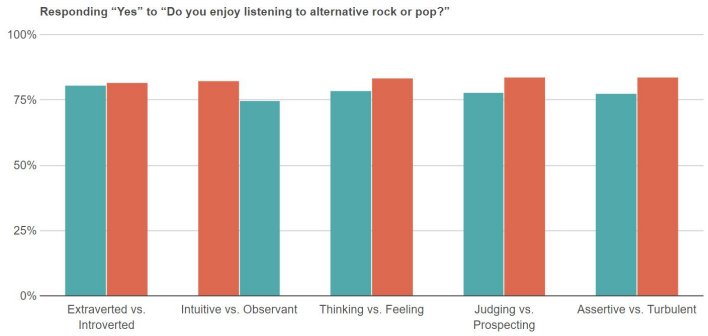
## Results per Genre from Article:

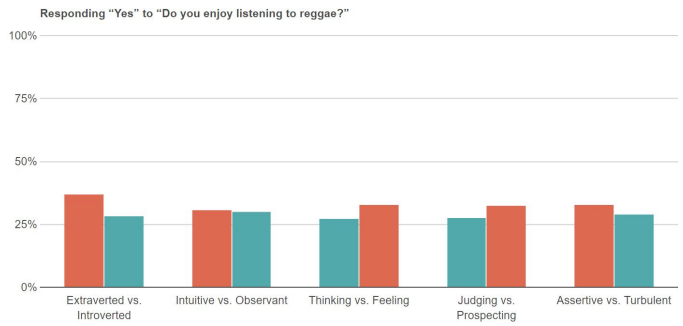


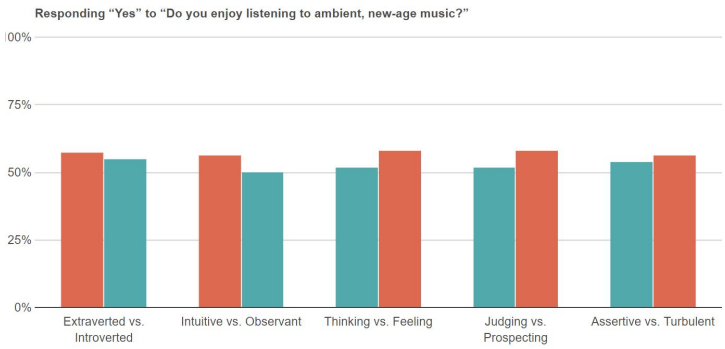



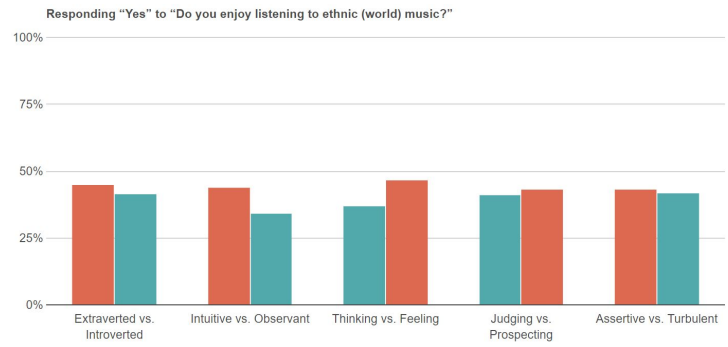



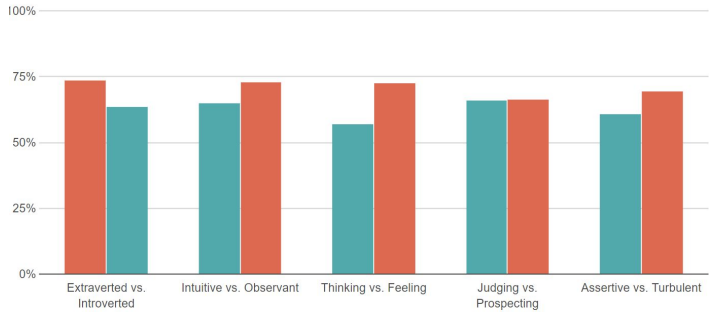



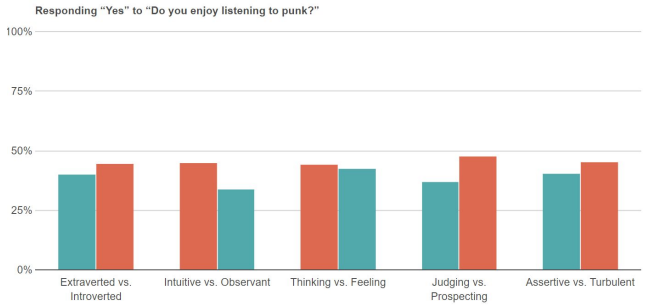



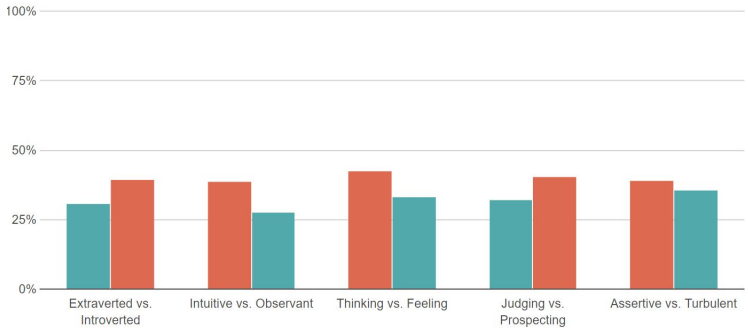
Responding "Yes" to "Do you enjoy listening to pop songs?"



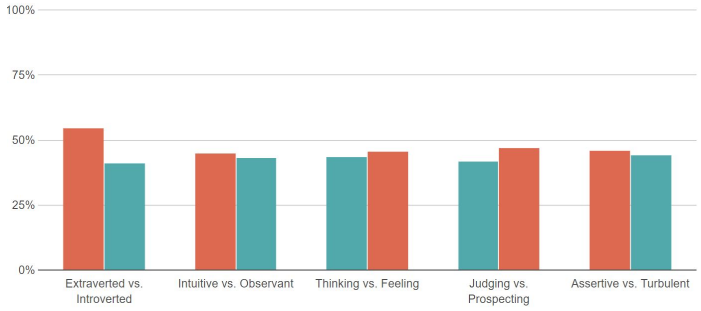

Responding "Yes" to "Do you enjoy listening to punk?"



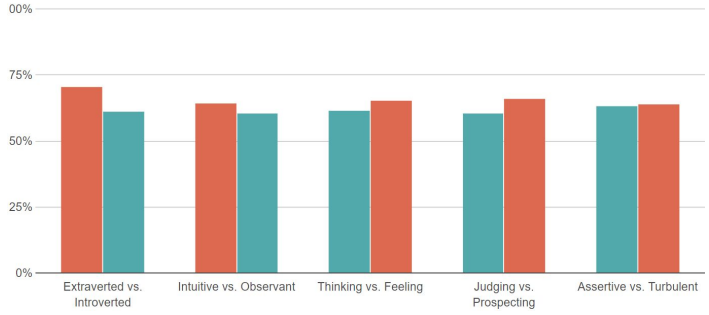

Responding "Yes" to "Do you enjoy listening to metal?"



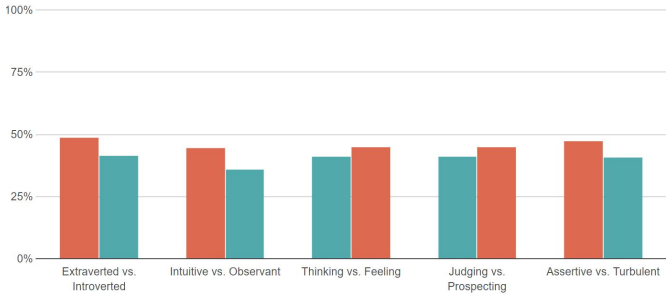

Responding "Yes" to "Do you enjoy listening to rap or hip-hop?"



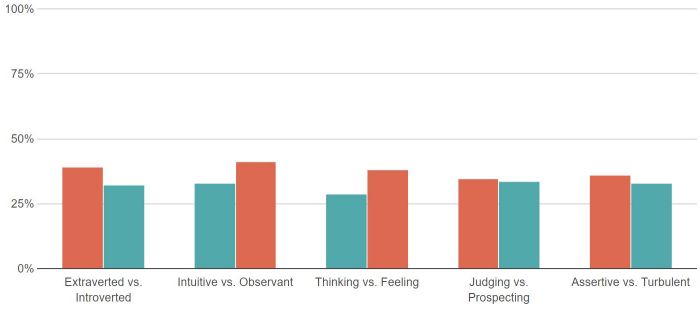

Responding "Yes" to "Do you enjoy listening to electronic or dance music?"




Responding "Yes" to "Do you enjoy listening to blues or bluegrass?"




Responding "Yes" to "Do you enjoy listening to country?"



## Python Code for Processing Personalities:

```
# -*- coding: utf-8 -*-
import numpy as np
import sys
import csv

#############################################################

# ------- INITIALIZE VARIABLES AND CONSTANTS ----------#

#############################################################

OUT = []
GENRE_SORES = {}
INPUT = []
personalities = ["extraversion", "intuition", "thinking", "judging", "assertiveness"]
genres = ["punk", "jazz", "classical", "rock", "alternative rock",
        "reggae", "ambient", "world", "pop", "metal", "hip-hop",
        "electronica", "religious", "blues", "country", "soul"]
personality_scores =  [[-4, 10,2,-11,-5],
                    [8, 8, 0, -3, 7],
                    [-2, 12, 4, 4, 5],
                    [-2, 12, 2,-6,0],
                    [-1, 7, -5, -6, -6],
                    [9,0,-6,-5,4],
                    [3,6,-6,-6,-2],
                    [3,9,-10,-2,1],
                    [10,-8,-16,0,-9],
                    [-9,11,9,-8,3],
                    [14,2,-2,-5,2],
                    [9,4,-4,-5,-1],
                    [6, -6, -12, 5, 1],
                    [7, 9, -4, -3, 7],
                    [7, -9, -10, 1, 3],
                    [13, -1, -13, -3, -1]]

#############################################################

# -------------------------- HANDLE INPUT ----------------------------#

#############################################################

#inputs a CSV file to be read and then processed
filename = sys.argv[1]

#read the file
with open(filename, 'r') as p:
    filenameReader = csv.reader(p, delimiter=" ")
    for row in filenameReader:
        INPUT.append(row[0].lower())


#############################################################

# ------------------------------- PROCESS -------------------------------#

#############################################################

#gets the number of elements in the inputted CSV file
NUM_ELEMENTS = len(INPUT)

#initalizes the output list
for row in personalities:
    OUT.append(0)
```

```
#go through each element in genre and create the
#personality profile for the current user
i = 0
for row in genres:
   GENRE_SORES[row] = personality_scores[i]
   i = i + 1

#update personaltiy scores depending on num
#occurences of each genre
for row in INPUT:
   scores = GENRE_SORES[row]
   a = np.add(scores, OUT)
   b = np.ndarray.tolist(a)
   OUT = b

#get the averages of the scores
#in order to complete personality profiler
averages = []
for row in OUT:
   avg =  row / NUM_ELEMENTS
   averages.append(avg)

############################################################

# ---------------- CYC STATEMENTS TO OUTPUT ------------------#

############################################################


#create cyc statements for the CYC language
#corresponding to personality profiler

zero = "(has (PersonalityScore Extroverted " + str(averages[0]) + "))"
one = "(has (PersonalityScore Intuition " + str(averages[1]) + "))"
two = "(has (PersonalityScore Thinking " + str(averages[2]) + "))"
three = "(has (PersonalityScore Judging " + str(averages[3]) + "))"
four = "(has (PersonalityScore Assertiveness " + str(averages[4]) + "))"

f = open("output.txt", "w")
print >> f, zero
print >> f, one
print >> f, two
print >> f, three
print >> f, four

f.close()
```