Exploratory Data Analysis - report

github.com/Macsok

October 2024

1 Introduction

1.1 What is EDA?

EDA stands for Exploratory Data Analysis. It's a crucial step in the data analysis process where you summarize the main characteristics of a dataset, often using visual methods. Here's the gist:

- Purpose: Understand the data better before jumping into modeling. It helps in uncovering patterns, spotting anomalies, testing hypotheses, and checking assumptions.
- Tools and Techniques: Descriptive statistics (mean, median, mode, standard deviation), visualizations (histograms, scatter plots, box plots), and data wrangling (handling missing values, scaling).
- Outcome: You get insights that guide your next steps in the data science workflow.

You can think of it as getting to know your dataset inside out, making sure you're fully prepped before the heavy lifting begins.

1.2 An aim

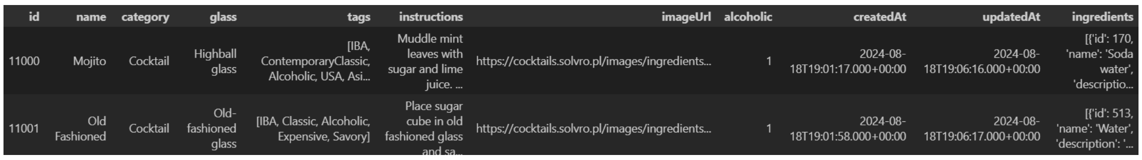
This analysis aims to understand the dataset of 134 cocktails. It aims to identify main characteristics of the cocktails, understand relations between ingredients and distinguish potential groups of similar drinks.

1.3 An approach

The approach is based on uncovering pure properties, relations and information from every datum that TheCocktailDB provides us with. It all started by subtle looking and each column and examining its content. The goal was to extrude as much information as we could (even exploring 'createdAt' and 'updatedAt' columns). After testing and making brief reconnaissance we moved to analysing cocktails alone, then ingredients dataframe, next we tried to merged them together and unravel relations between them, finally we analised whole set using scikit-learn library. In other words we started with data type manipulation and clearing data, moved through exploring single properties, and landed on clustering our set and suggesting similar drinks. **I highly suggest you to go file by file and follow thinking pattern that took us there.**

1.3.1 Brief overlook of the dataset

Main part of the data collection consists of **134** rows and **11** columns. Here is brief look at it:

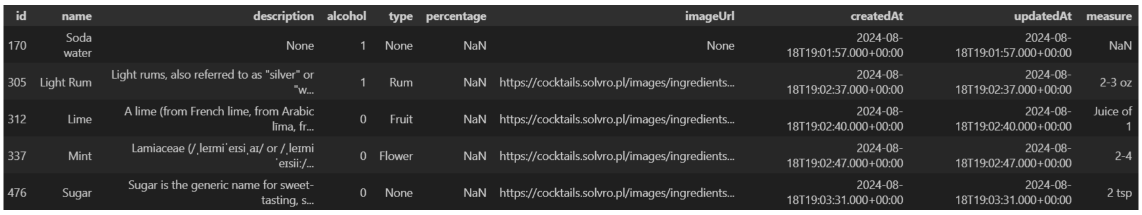


id	name	category	glass	tags	instructions	imageUrl	alcoholic	createdAt	updatedAt	ingredients
11000	Mojito	Cocktail	Highball glass	[IBA, ContemporaryClassic, Alcoholic, USA, Asi...	Muddle mint leaves with sugar and lime juice. ...	https://cocktails.solvro.pl/images/ingredients...	1	2024-08-18T19:01:17.000+00:00	2024-08-18T19:06:16.000+00:00	[{'id': 170, 'name': 'Soda water', 'descriptio...
11001	Old Fashioned	Cocktail	Old-fashioned glass	[IBA, Classic, Alcoholic, Expensive, Savory]	Place sugar cube in old fashioned glass and sa...	https://cocktails.solvro.pl/images/ingredients...	1	2024-08-18T19:01:58.000+00:00	2024-08-18T19:06:17.000+00:00	[{'id': 513, 'name': 'Water', 'description': '...

Figure 1: TheCocktailDB head

The columns are: id, name, category, glass, tags, instructions, imageUrl, alcoholic, createdAt, updatedAt, ingredients. Every column excpet 'id' and 'alcoholic' have object type. Other two are represented as int64.

In 'ingredients' column there is nested information about ingredients. Each drink have own piece of ingredients information there. If you unpack such block of information there will be data about ingredients needed to prepare such drink. Every record in 'ingredients' table has **10** attributes. They are as follows: id, name, alcohol, type, percentage, imageUrl, createdAt, updatedAt, measure. After unpacking ingredients data from every cocktail there will be **531** unique records in the 'ingredients' table.



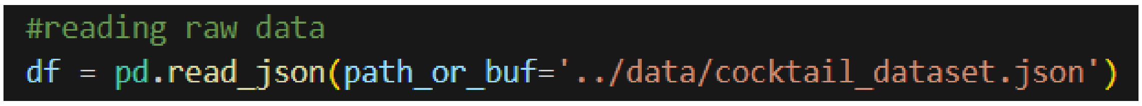
id	name	description	alcohol	type	percentage	imageUrl	createdAt	updatedAt	measure
170	Soda water	None	1	None	NaN	None	2024-08-18T19:01:57.000+00:00	2024-08-18T19:01:57.000+00:00	NaN
305	Light Rum	Light rums, also referred to as "silver" or "w...	1	Rum	NaN	https://cocktails.solvro.pl/images/ingredients...	2024-08-18T19:02:37.000+00:00	2024-08-18T19:02:37.000+00:00	2-3 oz
312	Lime	A lime (from French lime, from Arabic līma, fr...	0	Fruit	NaN	https://cocktails.solvro.pl/images/ingredients...	2024-08-18T19:02:40.000+00:00	2024-08-18T19:02:40.000+00:00	Juice of 1
337	Mint	Lamiaceae (/ˌleɪmiˈeɪsiˌaɪ/ or /ˌleɪmiˈeɪsii:/...	0	Flower	NaN	https://cocktails.solvro.pl/images/ingredients...	2024-08-18T19:02:47.000+00:00	2024-08-18T19:02:47.000+00:00	2-4
476	Sugar	Sugar is the generic name for sweet-tasting, s...	0	None	NaN	https://cocktails.solvro.pl/images/ingredients...	2024-08-18T19:03:31.000+00:00	2024-08-18T19:03:31.000+00:00	2 tsp

Figure 2: A snippet of the 'ingredients' table

2 Data review

2.1 Data loading

In this section, we will demonstrate the steps required to load the dataset into our analysis environment. As we found out data is stored in .json format, so loading it into pandas DataFrame looks like this:

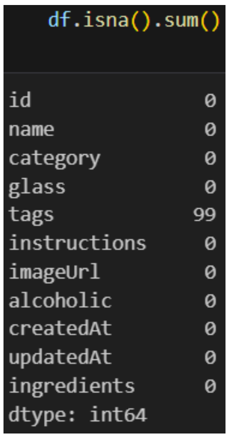


```
#reading raw data
df = pd.read_json(path_or_buf='../data/cocktail_dataset.json')
```

Figure 3: Loading data and storing it as pandas DataFrame

2.2 Missing values

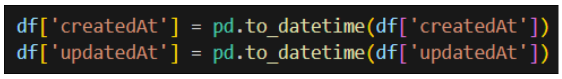
In this section we will examine missing values in our DataFrame and correct type of data in time columns. Figure 4 represents summed rows of missing values in each column of the dataset. We can see that most of the 'tags' column is missing (**99 out of 134**). Figure 5 shows code used to correct data type in 'createdAt' and 'updatedAt' columns.



```
df.isna().sum()

id              0
name            0
category        0
glass           0
tags           99
instructions    0
imageUrl        0
alcoholic       0
createdAt       0
updatedAt       0
ingredients     0
dtype: int64
```

Figure 4: Sums on missing values



```
df['createdAt'] = pd.to_datetime(df['createdAt'])
df['updatedAt'] = pd.to_datetime(df['updatedAt'])
```

Figure 5: Data type correction

2.3 Brief exploration

After short reconnaissance of the data in our DataFrame you can spot some important properties:

- every cocktail is alcoholic
- imageUrl won't be useful in the data analysis (at least for us)
- id's aren't consistent
- 'ingredients' column is stuffed with a .json data

In that case further explorations of the dataset will exclude 'imageUrl' and 'alcoholic' columns.

3 Exploring cocktails alone

In this section we pay attention only to cocktails - we won't unpack data from 'ingredients' column. In this case we drop four columns: 'ingredients', 'imageUrl', 'alcoholic', 'tags'.

3.1 Is there something hidden in those times?

Creating and updating times may seem harmless and boring but is there something hidden? Let's plot them and find out! Figure 6 shows the result, seems that nothing interesting was hidden there.

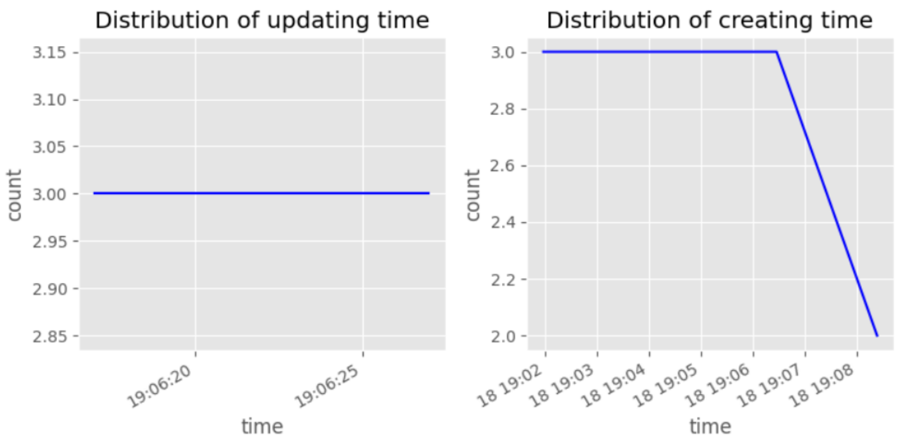
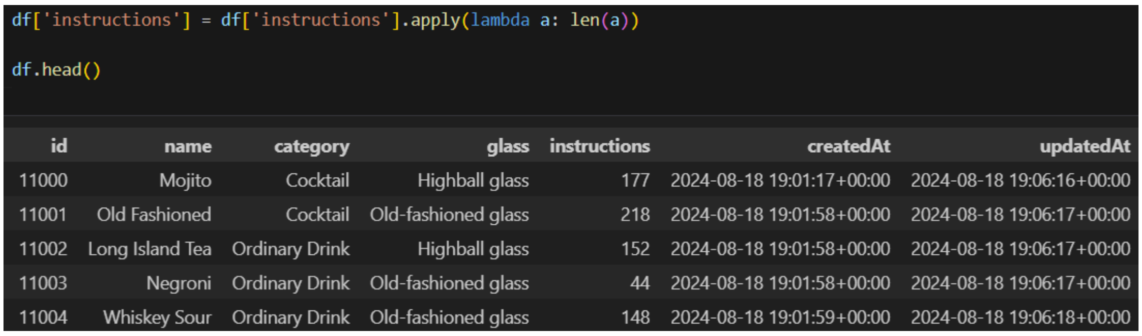


Figure 6: Times of creating and upgrading cocktails

3.2 Exploring other properties

Next we will simplify a little and assume that length of the instruction is somehow relevant to level of preparation complexity. We will replace textual value of instructions with the length of it. Below (on Figure 7) you can see brief code and the results of such operation. In this case we can, for example, find relations between type of the glass and the instruction length. Figure 8 presents lenghts of the instruction per every type of glass. Each red dot represents one cocktail. We can spot that Highball glasses tend to have a little longer instructions than Cocktail glasses. Old-fashioned glass drinks' instruction length vary more than Cocktails glass ones which are more regular.



```
df['instructions'] = df['instructions'].apply(lambda a: len(a))

df.head()
```

id	name	category	glass	instructions	createdAt	updatedAt
11000	Mojito	Cocktail	Highball glass	177	2024-08-18 19:01:17+00:00	2024-08-18 19:06:16+00:00
11001	Old Fashioned	Cocktail	Old-fashioned glass	218	2024-08-18 19:01:58+00:00	2024-08-18 19:06:17+00:00
11002	Long Island Tea	Ordinary Drink	Highball glass	152	2024-08-18 19:01:58+00:00	2024-08-18 19:06:17+00:00
11003	Negroni	Ordinary Drink	Old-fashioned glass	44	2024-08-18 19:01:58+00:00	2024-08-18 19:06:17+00:00
11004	Whiskey Sour	Ordinary Drink	Old-fashioned glass	148	2024-08-18 19:01:59+00:00	2024-08-18 19:06:18+00:00

Figure 7: Replacement of the instruction and current look at head of our cocktails data

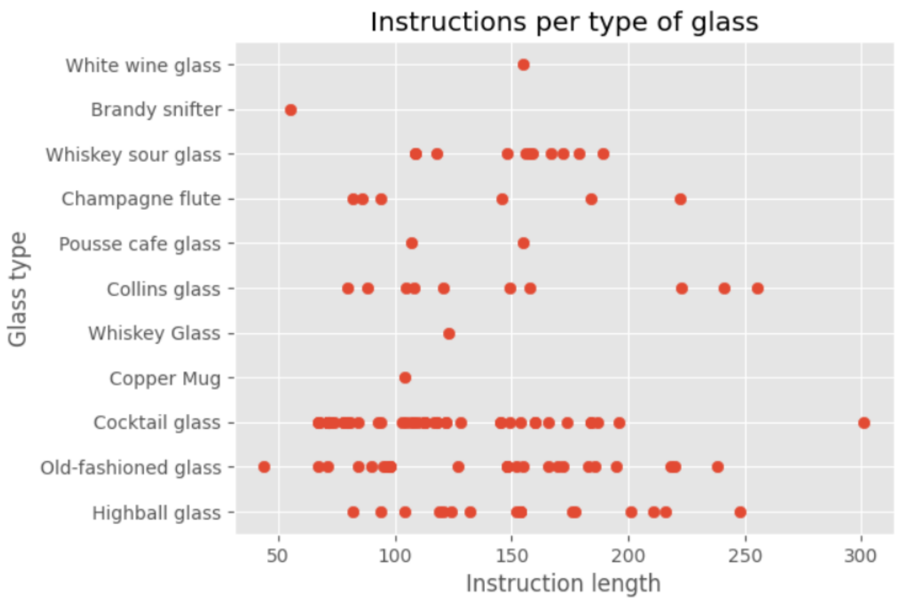


Figure 8: Length of the instruction per type of the glass

3.3 Dividing into groups

Let's take a look at popularity of each drink category. There are three such categories: Punch / Party Drink, Cocktail, Ordinary Drink. To count how many drinks are in each category we will use function: value_counts(), then we will simply plot it as horizontal bar plot. Here is the result (Figure 9):

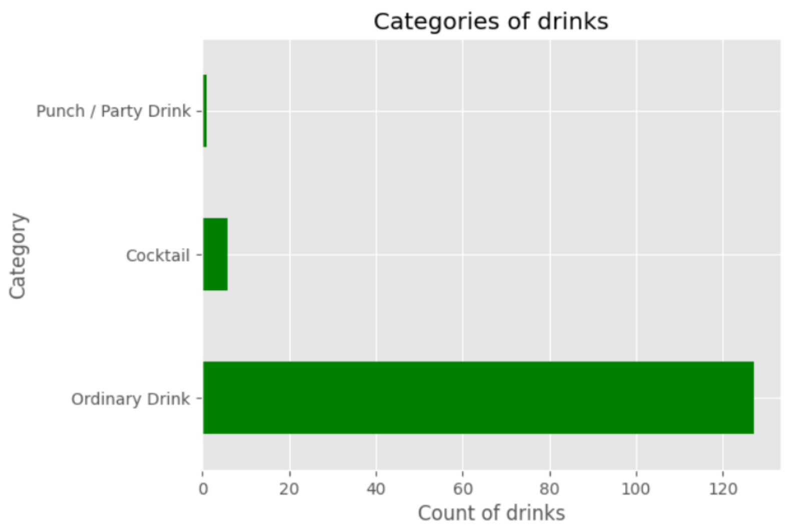


Figure 9: Count of drinks in each category

Similarly we can count the frequency of using each glass type in drinks and present the mon bar plot. This time we will focus on top 10 used glass types.

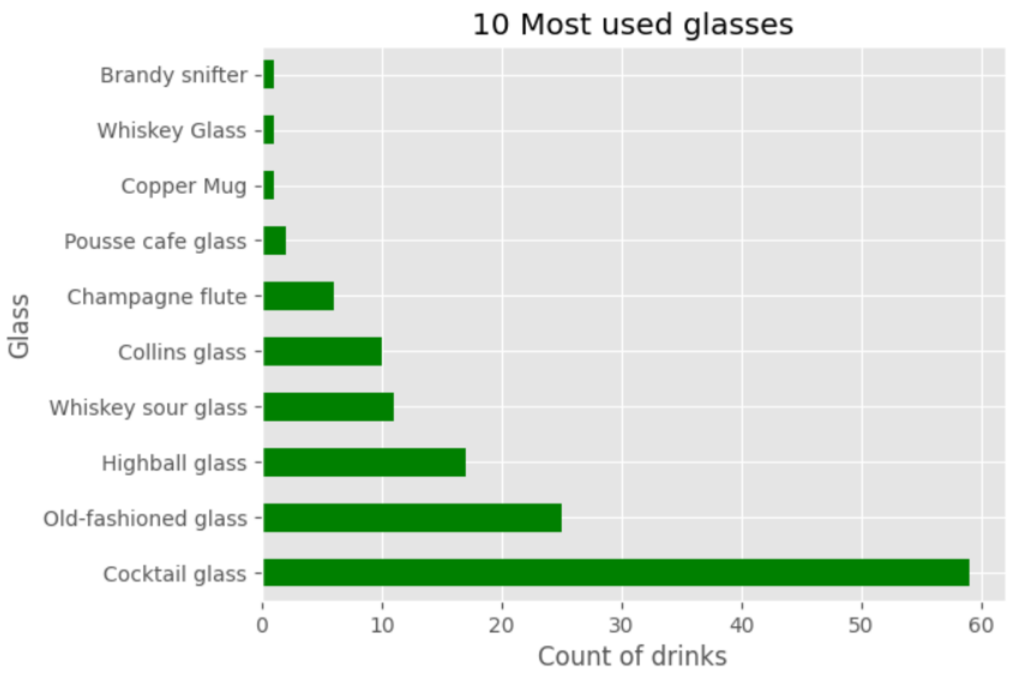
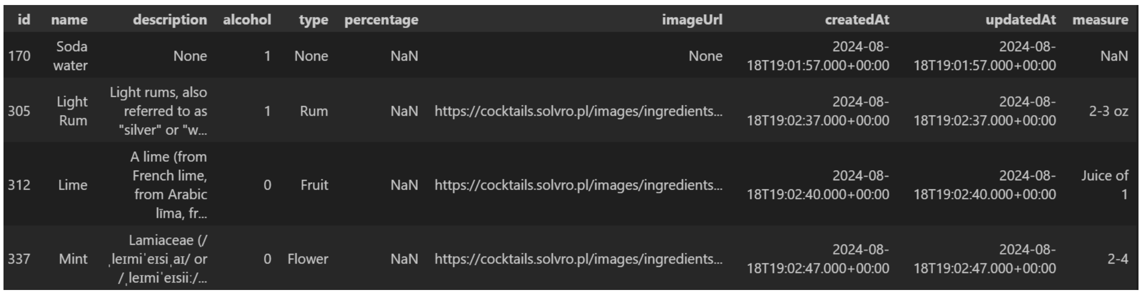


Figure 10: Top 10 used glasses

As we can see the most commonly used glass type is Cocktail glass, second one is Old-fashioned glass and on third position there is Highball glass. The top one leaves opponents far behind.

4 Exploring ingredients alone

In this section we will take closer look on data containing only ingredients of the cocktails. Similarly to the previous section we won't pay attention to the connections between drinks and their ingredients - at least not yet. In this part data about every cocktail ingredients is stored in separate table.



id	name	description	alcohol	type	percentage	imageUrl	createdAt	updatedAt	measure
170	Soda water	None	1	None	NaN	None	2024-08-18T19:01:57.000+00:00	2024-08-18T19:01:57.000+00:00	NaN
305	Light Rum	Light rums, also referred to as "silver" or "w...	1	Rum	NaN	https://cocktails.solvro.pl/images/ingredients...	2024-08-18T19:02:37.000+00:00	2024-08-18T19:02:37.000+00:00	2-3 oz
312	Lime	A lime (from French lime, from Arabic līma, fr...	0	Fruit	NaN	https://cocktails.solvro.pl/images/ingredients...	2024-08-18T19:02:40.000+00:00	2024-08-18T19:02:40.000+00:00	Juice of 1
337	Mint	Lamiaceae (/ˌleɪmiˈeɪsiˌaɪ/ or /ˌleɪmiˈeɪsii:/...	0	Flower	NaN	https://cocktails.solvro.pl/images/ingredients...	2024-08-18T19:02:47.000+00:00	2024-08-18T19:02:47.000+00:00	2-4

Figure 11: Head of the ingredients dataframe

4.1 Missing values and duplicates

Checking each column shows that many of them have missing values:

- 431 records from 'percentage'
- 224 records from 'description'
- 121 records from 'type'
- 35 records from 'measure'

Note that a shape of ingredients dataframe is **11 x 531**.

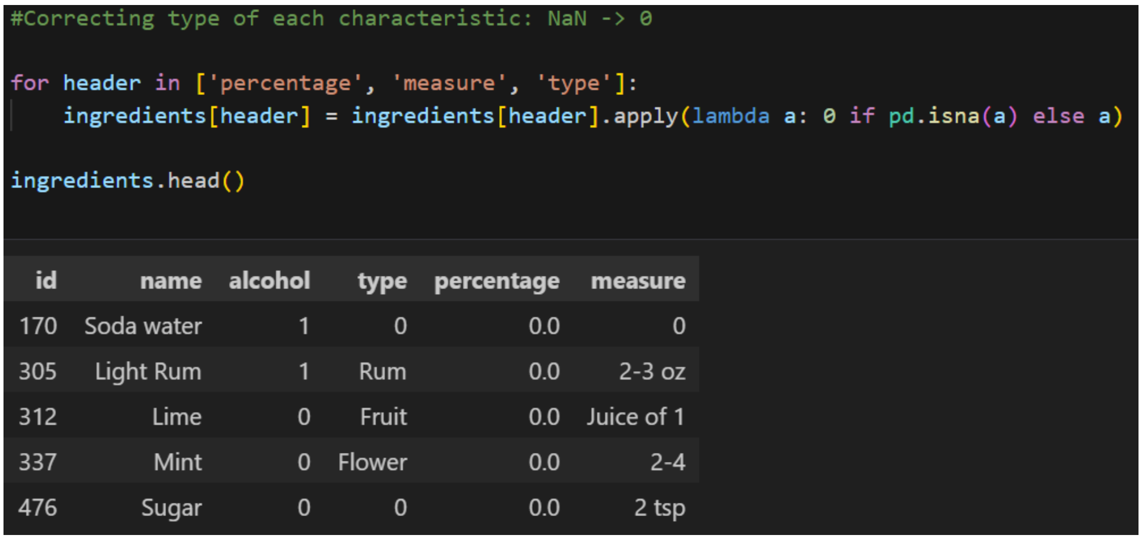
Each column have it individual explanation of shortage. 'percentage' column is missing values in every record that has 0 percent. To repair this issue we can simply map every NaN value to zero. 'description' column is missing information for simple ingredients such as water. 'type' and 'measure' are just not specified in some cases of the ingredients (missing measure of added water, water and sugar have none type). Due to lack of the information in the 'description' column and not normalized data in 'measure' column we won't consider those in further analysis (normalising measure column could be a fun thing to do...).

There are many duplicates in the 'name' column cause of the usage of similar ingredients in different drinks. For example there are many records of 'Gin' with various types of the measurement.

4.2 Exploring ingredients

Before diving into world of relations and connections we can make some corrections in data types of our set. As before we will updated data type in columns which store date information. Furthermore we will map every missing value to 0 (not recommended in every case).

Having prepared data we can start playing with it. Again we will try to extract information from every bit of the data - we will analyze updatig times again (Figure 13). This time it looks more promising as it is not a straight boring line. Most of the updates were made at the beginning - around 19.01 - then it lapsed down and there was another peak of updates around 19.02.30. After it, it went down again. Was there something useful? Well... Maybe not but again - we are here to learn and try to find hidden relations.



```
#Correcting type of each characteristic: NaN -> 0

for header in ['percentage', 'measure', 'type']:
    ingredients[header] = ingredients[header].apply(lambda a: 0 if pd.isna(a) else a)

ingredients.head()
```

id	name	alcohol	type	percentage	measure
170	Soda water	1	0	0.0	0
305	Light Rum	1	Rum	0.0	2-3 oz
312	Lime	0	Fruit	0.0	Juice of 1
337	Mint	0	Flower	0.0	2-4
476	Sugar	0	0	0.0	2 tsp

Figure 12: Correcting data types

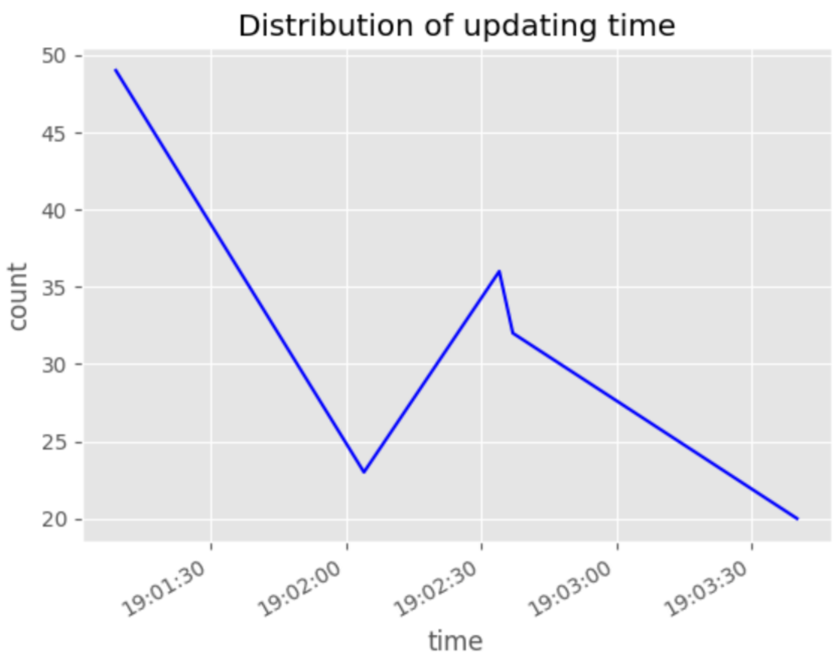


Figure 13: Update times of the ingredients plotted on time axis

4.3 Questions and answers

This part will focus on answering some questions about data in 'ingredients' table.

- What is top 10 most used types of the ingredients?

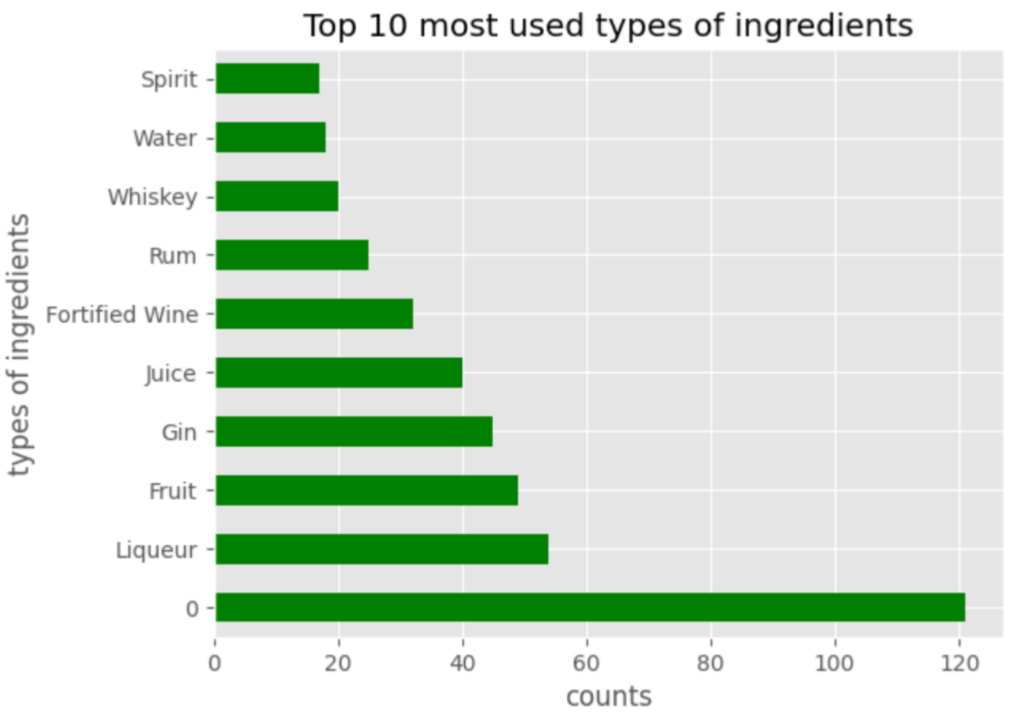


Figure 14: Top 10 most used types of the ingredients

Most used ingredient is labeled as a ’0’. Knowing that we have replaced every missing values before it means that this category covers ingredients that couldn’t have been labeled (ex. water and sugar).

- What are the most used ingredients?

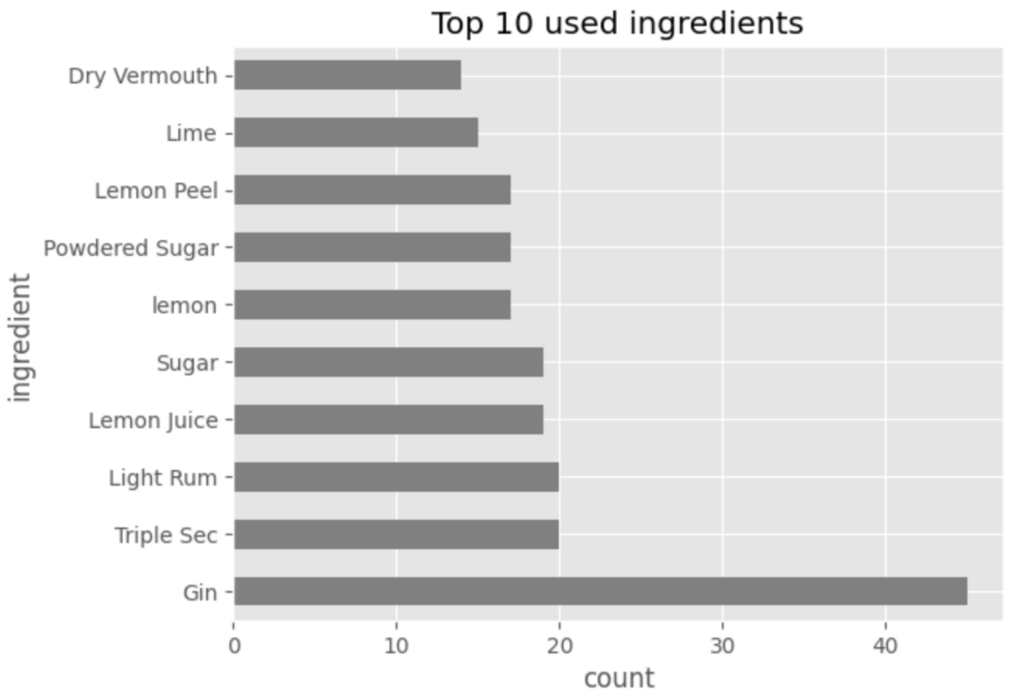


Figure 15: Most used components

The most used ingredient is Gin. Next two components in the list are alcohols as well - the main ingredient of the cocktail is the alcohol!

- Which ingredients are more populated along all the ingredients: alcoholic or non-alcoholic?

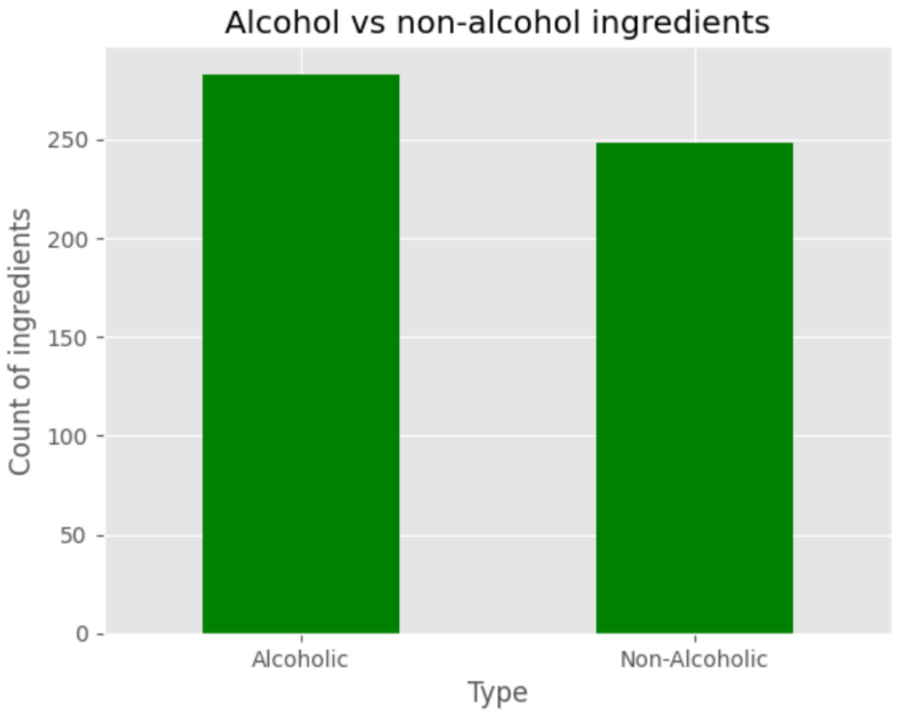


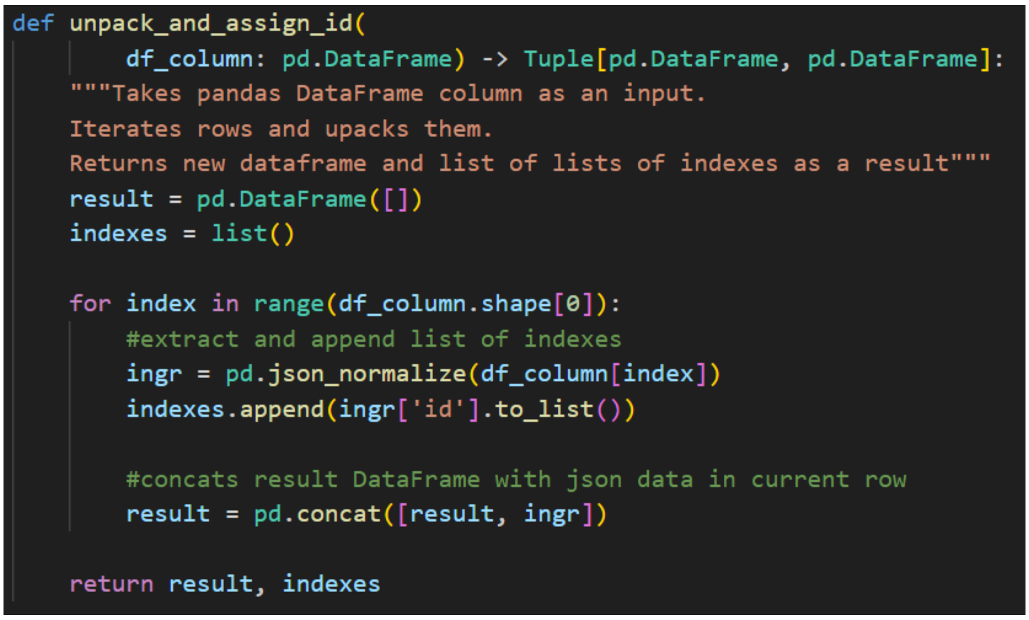
Figure 16: Alcoholic vs non-alcoholic ingredients

Both types of the ingredients are on similar level with little domination on the alcoholic side. Have in mind that this comparison one cube of an ice is worth as much as one liter of a gin.

- What is an average percentage among all of the ingredients? Answer: Average percentage per ingredient is: 6.79%. Every non-alcoholic component of drinks was labeled with 0%.

5 Analysis of whole dataset

In this approach we will try to represents each cocktail ingredients as list of indexes. Cocktails will be stored in one DataFrame object and all ingredients will be stored in another DataFrame. We will try to find connections and relations between them. Here is fragment of a code that is responsible for unpacking and storing lists of indexes of ingredients:



```python
def unpack_and_assign_id(
        df_column: pd.DataFrame) -> Tuple[pd.DataFrame, pd.DataFrame]:
    """Takes pandas DataFrame column as an input.
    Iterates rows and upacks them.
    Returns new dataframe and list of lists of indexes as a result"""
    result = pd.DataFrame([])
    indexes = list()

    for index in range(df_column.shape[0]):
        #extract and append list of indexes
        ingr = pd.json_normalize(df_column[index])
        indexes.append(ingr['id'].to_list())

        #concats result DataFrame with json data in current row
        result = pd.concat([result, ingr])

    return result, indexes
```

Figure 17: Fragment of a code representing function which unpacks each list of indexes

Based on previous analyses we drop some of columns:

- imageUrl - unnecessary in our analysis
- ingredients - we store those in another DataFrame object
- tags - missing values
- alcoholic - every drink is alcoholic
- measure - not normalized

5.1 Questions and answers

- Is there any relationship between instruction length and number of used ingredients?

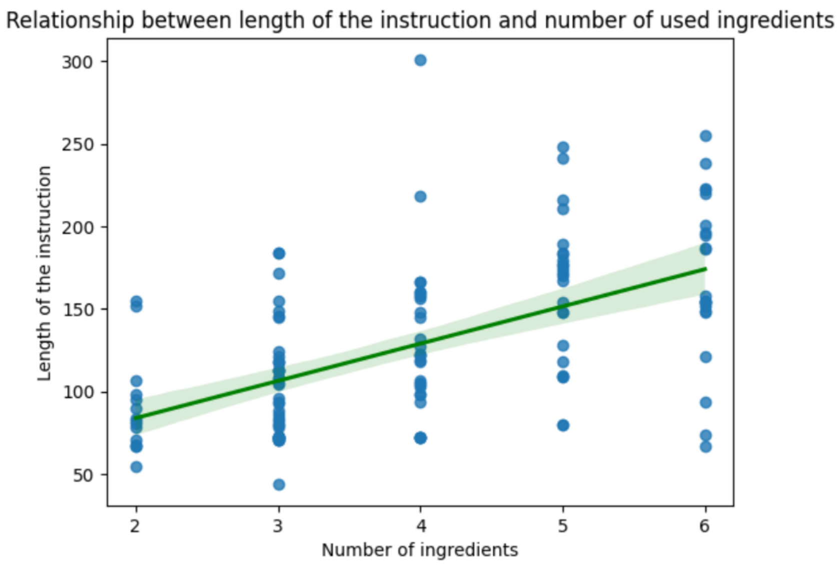


Figure 18: Regression of a instruction length

We can clearly see that as number of ingredients grows, length of an instruction is growing. Approximately every additional ingredient adds 20 symbols to length of an instruction.

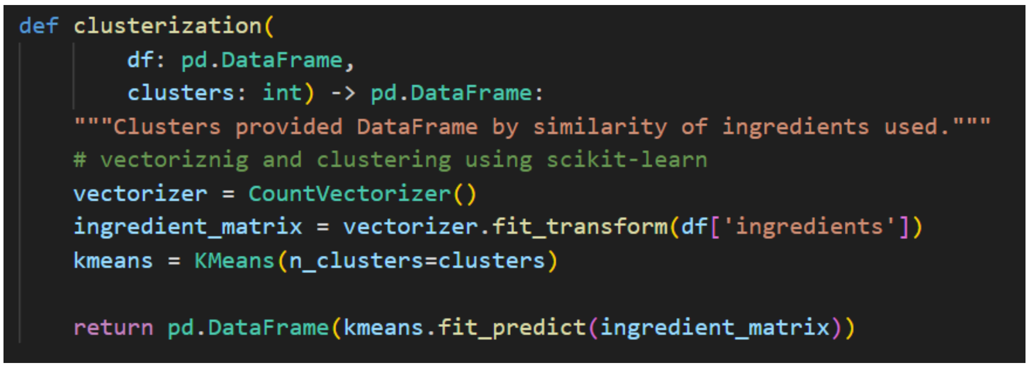
- According to the data, there are around 25 drinks served in an Old-fashioned glass. Which ingredient is the most used in these drinks? To answer this question, we need to filter the DataFrame to include only drinks served in an Old-fashioned glass. Then, for each list of ingredients, we need to find the names of the components and increase the counter for each one. A result shows that most used ingredient in such glass type is powdered sugar which appeared 15 times.

6 Analysis using scikit-learn library

This sections focuses on analysing our dataset using library that provides a wide range of tools for machine learning, pre-processing, cross-validation, and visualization. We will store data in two tables: drinks and ingredients. This time each drink will have its own record of ingredients listed by name. This will help with vectorizing them.

6.1 Clustering

Here we divide our drinks in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups. It's a key technique in data mining and machine learning, often used to find hidden patterns or natural groupings in data. Our separation will be based on similarity in used ingredients.

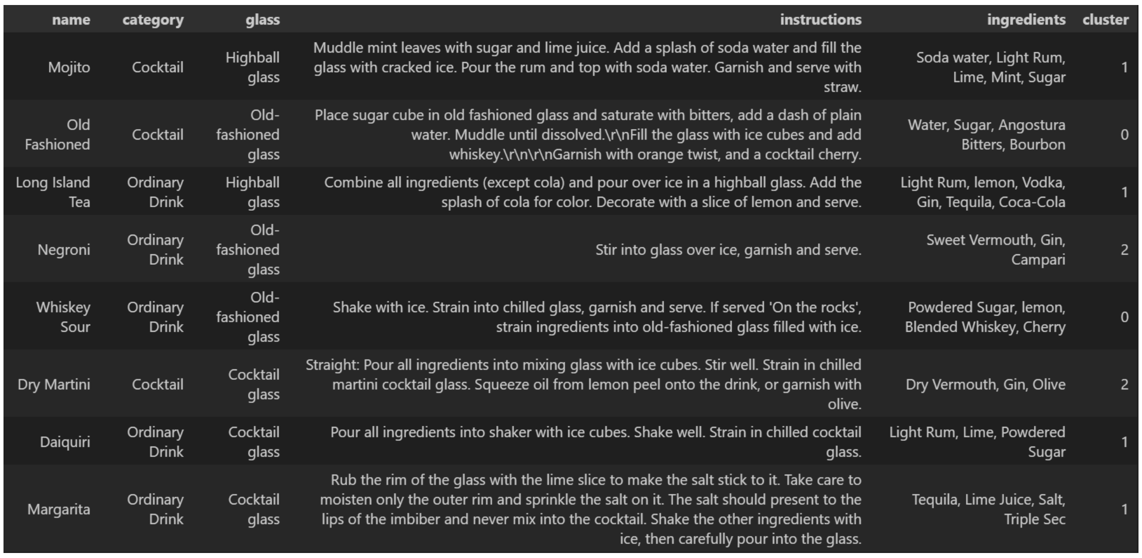


```python
def clusterization(
        df: pd.DataFrame,
        clusters: int) -> pd.DataFrame:
    """Clusters provided DataFrame by similarity of ingredients used."""
    # vectoriznig and clustering using scikit-learn
    vectorizer = CountVectorizer()
    ingredient_matrix = vectorizer.fit_transform(df['ingredients'])
    kmeans = KMeans(n_clusters=clusters)

    return pd.DataFrame(kmeans.fit_predict(ingredient_matrix))
```

Figure 19: Fragment of code representing function used to assign a cluster to each record

Observations shows that dividing data into 3 clusters separates cocktails by ingredients in which indeed drinks have common ingredients. Figure 20 shows that drinks in cluster '1' tends to be made of light rum and some kind of lemon/lime. Cocktails in cluster '0' most likely will consider some kind of sugar (regular, powdered). Drinks in cluster '2' tends to consist of gin and vermouth.



name	category	glass	instructions	ingredients	cluster
Mojito	Cocktail	Highball glass	Muddle mint leaves with sugar and lime juice. Add a splash of soda water and fill the glass with cracked ice. Pour the rum and top with soda water. Garnish and serve with straw.	Soda water, Light Rum, Lime, Mint, Sugar	1
Old Fashioned	Cocktail	Old-fashioned glass	Place sugar cube in old fashioned glass and saturate with bitters, add a dash of plain water. Muddle until dissolved.\r\nFill the glass with ice cubes and add whiskey.\r\n\r\nGarnish with orange twist, and a cocktail cherry.	Water, Sugar, Angostura Bitters, Bourbon	0
Long Island Tea	Ordinary Drink	Highball glass	Combine all ingredients (except cola) and pour over ice in a highball glass. Add the splash of cola for color. Decorate with a slice of lemon and serve.	Light Rum, lemon, Vodka, Gin, Tequila, Coca-Cola	1
Negroni	Ordinary Drink	Old-fashioned glass	Stir into glass over ice, garnish and serve.	Sweet Vermouth, Gin, Campari	2
Whiskey Sour	Ordinary Drink	Old-fashioned glass	Shake with ice. Strain into chilled glass, garnish and serve. If served 'On the rocks', strain ingredients into old-fashioned glass filled with ice.	Powdered Sugar, lemon, Blended Whiskey, Cherry	0
Dry Martini	Cocktail	Cocktail glass	Straight: Pour all ingredients into mixing glass with ice cubes. Stir well. Strain in chilled martini cocktail glass. Squeeze oil from lemon peel onto the drink, or garnish with olive.	Dry Vermouth, Gin, Olive	2
Daiquiri	Ordinary Drink	Cocktail glass	Pour all ingredients into shaker with ice cubes. Shake well. Strain in chilled cocktail glass.	Light Rum, Lime, Powdered Sugar	1
Margarita	Ordinary Drink	Cocktail glass	Rub the rim of the glass with the lime slice to make the salt stick to it. Take care to moisten only the outer rim and sprinkle the salt on it. The salt should present to the lips of the imbiber and never mix into the cocktail. Shake the other ingredients with ice, then carefully pour into the glass.	Tequila, Lime Juice, Salt, Triple Sec	1

Figure 20: Fragment of a clustered DataFrame

Having each cocktail allotted to a cluster we can transform all to them to a 2D plane which will let us show it on a grid.

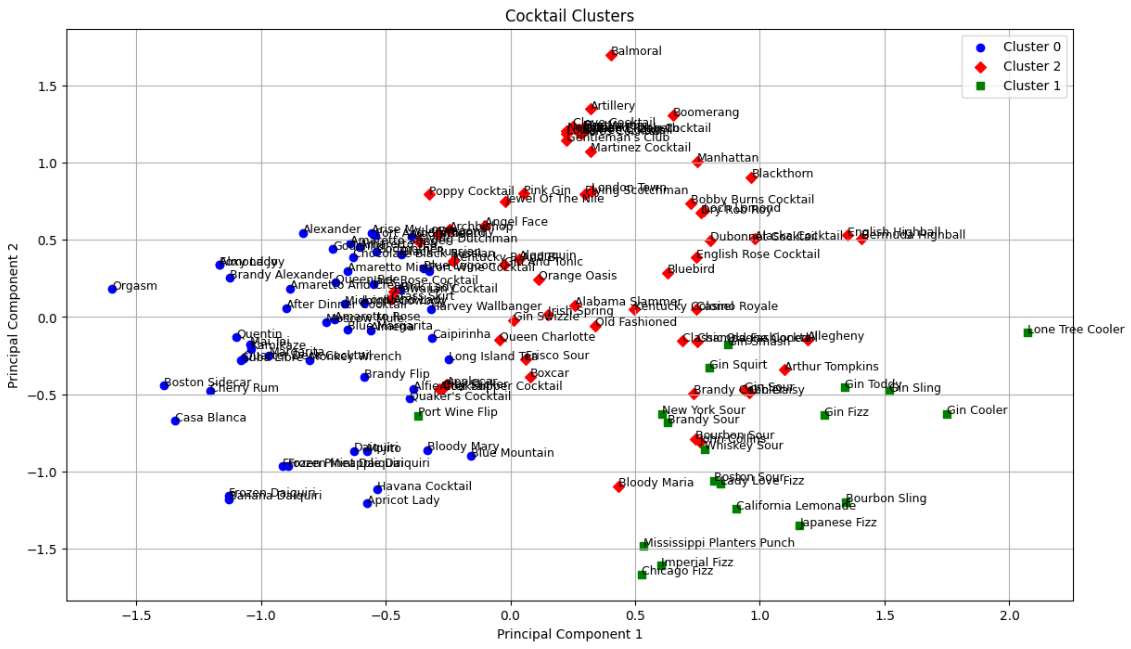


Figure 21: Clustered data on 2D plane (3 clusters)

6.2 Cocktail recommendation

Vectorizing list of ingredients give us plenty other applications. For example for given drink we can find similar one by finding closest point in a space of ingredients. To do such you need to count cosine similarity between given drink and every other, then simply sort list of them. This way you can find most similar (or lest similar) cocktail to provided one.

for example...