Untitled

June 1, 2021

# 1 Exploratory Data Analysis with Chocolate Rating¶

### Analysing the dataset of [Chocolate Bar Ratings](https://www.kaggle.com/rtatman/chocolate-bar-ratings) to unearth key insights hidden within the data¶

* + 1. **Context**

Chocolate is one of the most popular candies in the world. Each year, residents of the United States collectively eat more than 2.8 billions pounds. However, not all chocolate bars are created equal! This dataset contains expert ratings of over 1,700 individual chocolate bars, along with information on their regional origin, percentage of cocoa, the variety of chocolate bean used and where the beans were grown.

### Flavors of Cacao Rating System:

* + - * 5= Elite (Transcending beyond the ordinary limits)
      * 4= Premium (Superior flavor development, character and style)
      * 3= Satisfactory(3.0) to praiseworthy(3.75) (well made with special qualities)
      * 2= Disappointing (Passable but contains at least one significant flaw)
      * 1= Unpleasant (mostly unpalatable)

Each chocolate is evaluated from a combination of both objective qualities and subjective interpre- tation. A rating here only represents an experience with one bar from one batch. Batch numbers, vintages and review dates are included in the database when known.

The database is narrowly focused on plain dark chocolate with an aim of appreciating the flavors of the cacao when made into chocolate. The ratings do not reflect health benefits, social missions, or organic status. Flavor is the most important component of the Flavors of Cacao ratings. Diversity, balance, intensity and purity of flavors are all considered. It is possible for a straight forward single note chocolate to rate as high as a complex flavor profile that changes throughout. Genetics, terroir, post harvest techniques, processing and storage can all be discussed when considering the flavor component.

Texture has a great impact on the overall experience and it is also possible for texture related issues to impact flavor. It is a good way to evaluate the makers vision, attention to detail and level of proficiency. Aftermelt is the experience after the chocolate has melted. Higher quality chocolate will linger and be long lasting and enjoyable. Since the aftermelt is the last impression you get from the chocolate, it receives equal importance in the overall rating.

Overall Opinion is really where the ratings reflect a subjective opinion. Ideally it is my evaluation of whether or not the components above worked together and an opinion on the flavor development,

[1]:

character and style. It is also here where each chocolate can usually be summarized by the most prominent impressions that you would remember about each chocolate.

### Acknowledgements

These ratings were compiled by Brady Brelinski, Founding Member of the Manhattan Chocolate Society. Kindly visit[Flavors of Cacao](http://flavorsofcacao.com/index.html) for updated dataset.

### Hypothesis

Over time, there has been a correlation between cocoa ratings and the proportion of cocoa in a chocolate bar, as the quality of cocoa beans and chocolate bars has improved.

### Hypothetical Questions.

* + - * Which countries produces the best cocoa beans?
      * Which countires have the highest-rated chocolate bars?
      * Is there relationship between the proportion of cocoa in chocolate bar and the rating?
      * What are the top ten companies with the highest rating?
      * What is the pattern over the years with respect to rating?
      * Is there a correlation between a bean’s origin and the average rating of bars?
      * Which chocolate beans have the highest ratings?
  1. **1. Packages**

*# Import necessary libraries and packages*

**import pandas as pd** *# for data processing,*

**import matplotlib.pyplot as plt** *# for data-visualization*

%**matplotlib** inline

**import seaborn as sns** *# for data-visualization*

[29]:

## 2. Data preprocessing

### 2.1 Loading Data

*# Importing the dataset*

data = pd.read\_csv('flavors\_of\_cacao.csv') data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [29]: |  | CompanyÂ | \n(Maker-if known) | Specific | Bean | Origin\nor Bar Name | REF \ |
|  | 0 |  | A. Morin |  |  | Agua Grande | 1876 |
|  | 1 |  | A. Morin |  |  | Kpime | 1676 |
|  | 2 |  | A. Morin |  |  | Atsane | 1676 |
|  | 3 |  | A. Morin |  |  | Akata | 1680 |
|  | 4 |  | A. Morin |  |  | Quilla | 1704 |
|  | … |  | … |  |  | … … |  |
|  | 1790 |  | Zotter |  |  | Peru | 647 |
|  | 1791 |  | Zotter |  |  | Congo | 749 |
|  | 1792 |  | Zotter |  |  | Kerala State | 749 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1793 | Zotter | Kerala State | | | 781 | |
| 1794 | Zotter | Brazil, Mitzi Blue | | | 486 | |
|  | Review\nDate Cocoa\nPercent | Company\nLocation | Rating | Bean\nType | | \ |
| 0 | 2016 63% | France | 3.75 | Â | |  |
| 1 | 2015 70% | France | 2.75 | Â | |  |
| 2 | 2015 70% | France | 3.00 | Â | |  |
| 3 | 2015 70% | France | 3.50 | Â | |  |
| 4  …  1790 | 2015 70%  … …  2011 70% | France  … …  Austria | 3.50  …  3.75 | Â  Â | |  |
| 1791 | 2011 65% | Austria | 3.00 | Forastero | |  |
| 1792 | 2011 65% | Austria | 3.50 | Forastero | |  |
| 1793 | 2011 62% | Austria | 3.25 | Â | |  |
| 1794 | 2010 65% | Austria | 3.00 | Â | |  |
| 0 | Broad Bean\nOrigin  Sao Tome |  |  |  | |  |
| 1 | Togo |  |  |  | |  |
| 2 | Togo |  |  |  | |  |
| 3 | Togo |  |  |  | |  |
| 4  …  1790 | Peru  …  Peru |  |  |  | |  |
| 1791 | Congo |  |  |  | |  |
| 1792 | India |  |  |  | |  |
| 1793 | India |  |  |  | |  |
| 1794 | Brazil |  |  |  | |  |
| [1795 | rows x 9 columns] |  |  |  | |  |

The Flavors of Cacao dataset consists of 1795 rows and 9 columns. The Columns are:

Column Description

Company (Maker-if known) Name of the company manufacturing the bar. Specific Bean Originor Bar Name The specific geo-region of origin for the bar.

REF A value linked to when the review was entered in the database. Higher = more recent.

ReviewDate Date of publication of the review.

CocoaPercent Cocoa percentage (darkness) of the chocolate bar being reviewed.

CompanyLocation Manufacturer base country.

Rating Expert rating for the bar.

BeanType The variety (breed) of bean used, if provided.

Broad BeanOrigin The broad geo-region of origin for the bean.

### 2.2 Understanding basic information of the data

[25]:

*# function to for data information*

**def** data\_info(data):

print("Basic Information anout Data:**\n**") print('

*# Data Types of a DataFrame*

print("Data Types of all Columns",data.dtypes)

')

print('

*# Shape of the DataFrame*

print("**\n**Number of Rows:",data.shape[0]) print("Number of Columns:",data.shape[1])

print('

*# Summary Statistics* print("**\n**Summary Statistics:") print(data.describe())

')

')

print('

')

*# Getting basic information*

data\_info(data)

Basic Information anout Data:

Data Types of all Columns CompanyÂ \n(Maker-if known) object Specific Bean Origin\nor Bar Name object

REF int64

Review\nDate int64

Cocoa\nPercent object

Company\nLocation object

Rating float64

Bean\nType object

Broad Bean\nOrigin object dtype: object

Number of Rows: 1795 Number of Columns: 9

Summary Statistics:

REF Review\nDate Rating

|  |  |  |  |
| --- | --- | --- | --- |
| count | 1795.000000 | 1795.000000 | 1795.000000 |
| mean | 1035.904735 | 2012.325348 | 3.185933 |
| std | 552.886365 | 2.927210 | 0.478062 |
| min | 5.000000 | 2006.000000 | 1.000000 |
| 25% | 576.000000 | 2010.000000 | 2.875000 |
| 50% | 1069.000000 | 2013.000000 | 3.250000 |
| 75% | 1502.000000 | 2015.000000 | 3.500000 |
| max | 1952.000000 | 2017.000000 | 5.000000 |

[32]:

**The data type information reveals something about our data, and it’s not a good one!** The column names are a little messy because they contain the “n” or “newline” character, which will result in unidentifiable errors we have to go through each column name and rename it explicitly!.

Information about the summary statistics on the dataset:

* The mean review date is 2012 which shows that the data is centered around 2012.
* The most current review date is 2017 and the oldest date is 2006.
* This shows the data was collected for the period 2006-2017(a 11-year period)
* The average rating of chacolate bar 3.18, which is satisfactory (represent a well made bar with special qualities.)
* The rating of the bars is within a range of 1 to 5

### 2.3 Data Cleaning

*# function to for identifying missing values and anomalies*

**def** data\_cleaning(data):

*# Missing Value Inspection*

print("Basic Information to check data:**\n**")

print('

print("**\n**Missing Values:") print(' print(data.isna().sum())

print("**\n**First 10 rows of the data frame.:") print('

print('head',data.head(10))

')

')

')

print("**\n**Last 10 rows of the data frame.:") print('

print(data.tail(10))

')

data\_cleaning(data)

Basic Information to check data:

Missing Values:

CompanyÂ \n(Maker-if known) 0

Specific Bean Origin\nor Bar Name 0

REF 0

Review\nDate 0

Cocoa\nPercent 0

Company\nLocation 0

Rating 0

Bean\nType 1

Broad Bean\nOrigin 1

dtype: int64

First 10 rows of the data frame.:

Last 10 rows of the data frame.:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | CompanyÂ | \n(Maker-if | known) | Specific | Bean | Origin\nor Bar Name | REF | \ |
| 1785 |  |  | Zotter |  |  | Huiwani Coop | 879 |  |
| 1786 |  |  | Zotter |  |  | El Ceibo Coop | 879 |  |
| 1787 |  |  | Zotter |  |  | Santo Domingo | 879 |  |
| 1788 |  |  | Zotter |  |  | Kongo, Highlands | 883 |  |
| 1789 |  |  | Zotter |  |  | Indianer, Raw | 883 |  |
| 1790 |  |  | Zotter |  |  | Peru | 647 |  |
| 1791 |  |  | Zotter |  |  | Congo | 749 |  |
| 1792 |  |  | Zotter |  |  | Kerala State | 749 |  |
| 1793 |  |  | Zotter |  |  | Kerala State | 781 |  |
| 1794 |  |  | Zotter |  |  | Brazil, Mitzi Blue | 486 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Review\nDate | Cocoa\nPercent | | Company\nLocation | Rating | \ |
| 1785 | 2012 | 75% | | Austria | 3.00 |  |
| 1786 | 2012 | 90% | | Austria | 3.25 |  |
| 1787 | 2012 | 70% | | Austria | 3.75 |  |
| 1788 | 2012 | 68% | | Austria | 3.25 |  |
| 1789 | 2012 | 58% | | Austria | 3.50 |  |
| 1790 | 2011 | 70% | | Austria | 3.75 |  |
| 1791 | 2011 | 65% | | Austria | 3.00 |  |
| 1792 | 2011 | 65% | | Austria | 3.50 |  |
| 1793 | 2011 | 62% | | Austria | 3.25 |  |
| 1794 | 2010 | 65% | | Austria | 3.00 |  |
|  | Bean\nType | | Broad Bean\nOrigin | | | |
| 1785 | Criollo, Trinitario | | Papua New Guinea | | | |
| 1786 | Â | | Bolivia | | | |
| 1787 | Â | | Dominican Republic | | | |

|  |  |  |
| --- | --- | --- |
| 1788 | Forastero | Congo |
| 1789 | Â | Â |
| 1790 | Â | Peru |
| 1791 | Forastero | Congo |
| 1792 | Forastero | India |
| 1793 | Â | India |
| 1794 | Â | Brazil |

[33]:

**The missing values information** BeanType and Broad BeanOrigin both have 1 missing value.

The first and last ten rows of the dataset shows a special character () within the BeanType and Broad BeanOrigin columns.

A further probe will give us more information on how to handle it.

*#cleaning column names*

cols\_names = list(data.columns)

*# Function to replace newline characters and spaces in the feature names*

**def** replace\_columnNames(column\_names): replace\_names = []

**for** f **in** column\_names: replace\_names.append(((f.casefold()).replace("**\n**","\_")).replace("␣

*‹→*","\_"))

**return** replace\_names

print("Column Names before Cleaning:") print(cols\_names)

print("**\n**Column Names after Cleaning:") print(replace\_columnNames(cols\_names))

[36]:

Column Names before Cleaning:

['CompanyÂ\xa0\n(Maker-if known)', 'Specific Bean Origin\nor Bar Name', 'REF', 'Review\nDate', 'Cocoa\nPercent', 'Company\nLocation', 'Rating', 'Bean\nType', 'Broad Bean\nOrigin']

Column Names after Cleaning:

['companyâ\xa0\_(maker-if\_known)', 'specific\_bean\_origin\_or\_bar\_name', 'ref', 'review\_date', 'cocoa\_percent', 'company\_location', 'rating', 'bean\_type', 'broad\_bean\_origin']

The columns now look much better than before. However, the **“company”** column still looks very weird.

We will manually edit and rename the column.

Finally, we shall re-assign the new columns names to our dataframe.

*# Manually renaming Company name*

new\_ColumnNames = replace\_columnNames(cols\_names)

new\_ColumnNames[0] = 'Company'

*# Re-assigning column names back to data* data=data.rename(columns=dict(zip(data.columns,new\_ColumnNames))) data.dtypes

[36]: Company object

specific\_bean\_origin\_or\_bar\_name object ref int64

review\_date int64

cocoa\_percent object

company\_location object

rating float64

bean\_type object

broad\_bean\_origin object dtype: object

The columns names looks better and easily readable.

[45]:

*# Probing the Â character further*

print('Checking special character in bean\_type data') print(' ') print(data['bean\_type'].value\_counts().head()) print("Missing Spaces encoded as:") list(data['bean\_type'][0:10])

Checking special character in bean\_type data

Â 887

Trinitario 419

Criollo 153

Forastero 87

Forastero (Nacional) 52

Name: bean\_type, dtype: int64 Missing Spaces encoded as:

[45]: ['Â\xa0',

'Â\xa0',

'Â\xa0',

'Â\xa0',

'Â\xa0', 'Criollo', 'Â\xa0', 'Criollo', 'Criollo', 'Â\xa0']

The information shows that we actually have 887 instances in which “bean\_type” is encoded as a

special character and space or .

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [54]: | *# Replace the Â character spaces with None (Symbolizes no data)*  *# Creating a dictionary for the replace character*  replace\_character = {'Â**\xa0**':'None'}  *# Replacing Â character spaces with None*  data.replace(replace\_character, regex=**True**, inplace=**True**)  *# Replacing missing value with None*  data['bean\_type'] = data['bean\_type'].fillna('None') data['broad\_bean\_origin'] = data['broad\_bean\_origin'].fillna('None')  data.head(10) | | | | | |
|  | *#* |  |  |  |  |  |
| [54]: | Company | specific\_bean\_origin\_or\_bar\_name | ref | review\_date | cocoa\_percent | \ |
|  | 0 A. Morin | Agua Grande | 1876 | 2016 | 63% |  |
|  | 1 A. Morin | Kpime | 1676 | 2015 | 70% |  |
|  | 2 A. Morin | Atsane | 1676 | 2015 | 70% |  |
|  | 3 A. Morin | Akata | 1680 | 2015 | 70% |  |
|  | 4 A. Morin | Quilla | 1704 | 2015 | 70% |  |
|  | 5 A. Morin | Carenero | 1315 | 2014 | 70% |  |
|  | 6 A. Morin | Cuba | 1315 | 2014 | 70% |  |
|  | 7 A. Morin | Sur del Lago | 1315 | 2014 | 70% |  |
|  | 8 A. Morin | Puerto Cabello | 1319 | 2014 | 70% |  |
|  | 9 A. Morin | Pablino | 1319 | 2014 | 70% |  |

|  |  |  |  |
| --- | --- | --- | --- |
| company\_location | rating | bean\_type | broad\_bean\_origin |
| 0 France | 3.75 | None | Sao Tome |
| 1 France | 2.75 | None | Togo |
| 2 France | 3.00 | None | Togo |
| 3 France | 3.50 | None | Togo |
| 4 France | 3.50 | None | Peru |
| 5 France | 2.75 | Criollo | Venezuela |
| 6 France | 3.50 | None | Cuba |
| 7 France | 3.50 | Criollo | Venezuela |
| 8 France | 3.75 | Criollo | Venezuela |
| 9 France | 4.00 | None | Peru |

Thus, we have filled those special characters and missing values with None.

NOTE : Imputing the missing values with None does not offer great advantage from the viewpoint of analysis. However, it helps us maintain a much cleaner dataset which I feel is as important as

[55]:

[56]:

keeping the visualizations clean.

data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 1795 entries, 0 to 1794 Data columns (total 9 columns):

# Column Non-Null Count Dtype

1. Company 1795 non-null object
2. specific\_bean\_origin\_or\_bar\_name 1795 non-null object
3. ref 1795 non-null int64
4. review\_date 1795 non-null int64
5. cocoa\_percent 1795 non-null object
6. company\_location 1795 non-null object
7. rating 1795 non-null float64
8. bean\_type 1795 non-null object
9. broad\_bean\_origin 1795 non-null object dtypes: float64(1), int64(2), object(6)

memory usage: 126.3+ KB

### From the data information, data has no missing values.

**Convert Cocoa\_percent to numerical values** \*\* We change the % notation in ‘co- coa\_percent’ to make it a numerical column for manipulation.\*\*

*# Converting % sign to percentage value*

data['cocoa\_percent']=data['cocoa\_percent'].str.replace('%','').astype(float)/

*‹→*100

data.head()

1. : Company specific\_bean\_origin\_or\_bar\_name ref review\_date \

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 A. Morin | Agua Grande | | 1876 | 2016 |
| 1 A. Morin | Kpime | | 1676 | 2015 |
| 2 A. Morin | Atsane | | 1676 | 2015 |
| 3 A. Morin | Akata | | 1680 | 2015 |
| 4 A. Morin | Quilla | | 1704 | 2015 |
| cocoa\_percent company\_location rating bean\_type broad\_bean\_origin | | | | |
| 0 | 0.63 | France 3.75 | None | Sao Tome |
| 1 | 0.70 | France 2.75 | None | Togo |
| 2 | 0.70 | France 3.00 | None | Togo |
| 3 | 0.70 | France 3.50 | None | Togo |
| 4 | 0.70 | France 3.50 | None | Peru |

The cocoa percent is now a numerical value.

[57]:

*# checking if null values have been filled* null\_bool = pd.isnull(data['bean\_type']) data[null\_bool]

1. : Empty DataFrame

Columns: [Company, specific\_bean\_origin\_or\_bar\_name, ref, review\_date, cocoa\_percent, company\_location, rating, bean\_type, broad\_bean\_origin] Index: []

[61]:

*# Droping the ref column*

data.drop(['ref'], axis = 1,inplace = **True**) data.head(10)

[61]:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Company | specific\_bean\_origin\_or\_bar\_name | review\_date | cocoa\_percent | \ |
| 0 A. Morin | Agua Grande | 2016 | 0.63 |  |
| 1 A. Morin | Kpime | 2015 | 0.70 |  |
| 2 A. Morin | Atsane | 2015 | 0.70 |  |
| 3 A. Morin | Akata | 2015 | 0.70 |  |
| 4 A. Morin | Quilla | 2015 | 0.70 |  |
| 5 A. Morin | Carenero | 2014 | 0.70 |  |
| 6 A. Morin | Cuba | 2014 | 0.70 |  |
| 7 A. Morin | Sur del Lago | 2014 | 0.70 |  |
| 8 A. Morin | Puerto Cabello | 2014 | 0.70 |  |
| 9 A. Morin | Pablino | 2014 | 0.70 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| company\_location | rating | bean\_type | broad\_bean\_origin |
| 0 France | 3.75 | None | Sao Tome |
| 1 France | 2.75 | None | Togo |
| 2 France | 3.00 | None | Togo |
| 3 France | 3.50 | None | Togo |
| 4 France | 3.50 | None | Peru |
| 5 France | 2.75 | Criollo | Venezuela |
| 6 France | 3.50 | None | Cuba |
| 7 France | 3.50 | Criollo | Venezuela |
| 8 France | 3.75 | Criollo | Venezuela |
| 9 France | 4.00 | None | Peru |

[62]:

*## Checking for the relationship between the variables # using the correlation matrix and heatmap*

*# calculate correlation matrix*

corr\_matrix = data.corr()

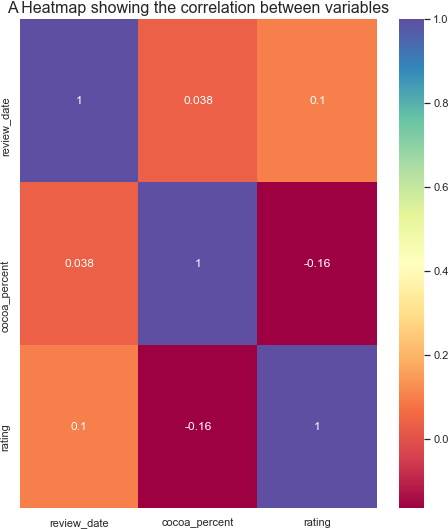
*# plot the heatmap*

sns.heatmap(corr\_matrix, xticklabels=corr\_matrix.

*‹→*columns,yticklabels=corr\_matrix.columns, annot=**True**, cmap="Spectral") sns.set(rc={'figure.figsize':(8, 9)})

plt.title("A Heatmap showing the correlation between variables", fontdict =␣

*‹→*{'fontsize' : 16});



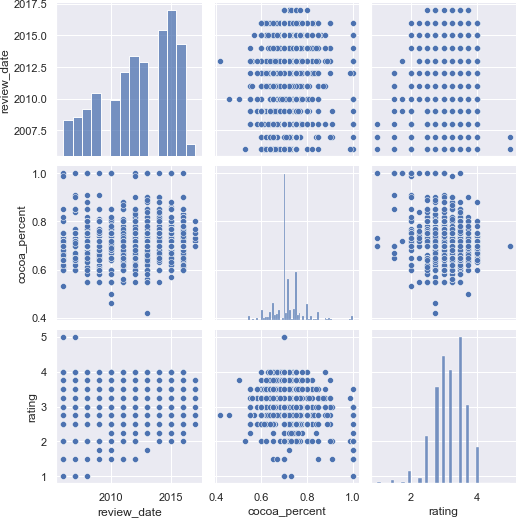
[64]:

**Observations:** It is clear from the heatmap and the seaborn(sns) correlation graphs that:

* + There is a No correlation amongst the variables.

*# pair plot to further visualize the relationship between the variables*

sns.pairplot(data);



[ ]:

The pair plot further indicate a NO correlation between rating, percentage of cocoa and review date.

# 2 Analyzing Hypothetical Questions