Iowa Liquor Sales

October 26, 2020

[1]:
```
import pandas as pd
import pickle
import matplotlib.pyplot as plt
```

[2]:
```
cd Desktop/Iowa_Liquor_Sales/
```

```
/Users/adrienpeltzer/Desktop/Iowa_Liquor_Sales
```

[4]:
```
# Since our dataset has ~19M rows, let's start by loading just the first 100␣
↪rows and see what columns there are
df = pd.read_csv("Iowa_Liquor_Sales.csv",nrows=100)
```

[5]:
```
[i for i in zip(range(len(df.columns)),df.columns.tolist())]
```

[5]:
```
[(0, 'Invoice/Item Number'),
 (1, 'Date'),
 (2, 'Store Number'),
 (3, 'Store Name'),
 (4, 'Address'),
 (5, 'City'),
 (6, 'Zip Code'),
 (7, 'Store Location'),
 (8, 'County Number'),
 (9, 'County'),
 (10, 'Category'),
 (11, 'Category Name'),
 (12, 'Vendor Number'),
 (13, 'Vendor Name'),
 (14, 'Item Number'),
 (15, 'Item Description'),
 (16, 'Pack'),
 (17, 'Bottle Volume (ml)'),
 (18, 'State Bottle Cost'),
 (19, 'State Bottle Retail'),
 (20, 'Bottles Sold'),
 (21, 'Sale (Dollars)'),
 (22, 'Volume Sold (Liters)'),
```

```
 (23, 'Volume Sold (Gallons)')]
```

[6]:
```
# There are 24 columns, but some of them can be inferred from the rest. For␣
 ↪example,
# the 'Volume Sold (Liters)' column is just "Bottles Sold" multiplied by "Bottle␣
 ↪Volume (mL)"
# Clearly then, we shouldn't load the whole dataset
```

[7]:
```
# Let's find out what Categories of Liquor Sales there are:
C = pd.read_csv("Iowa_Liquor_Sales.csv",usecols=[10,11])
C=C.dropna().drop_duplicates().reset_index(drop=True)
```

[8]:
```
C.head()
```

[8]:

	Category	Category Name
0	1032200.0	Imported Flavored Vodka
1	1012100.0	Canadian Whiskies
2	1012200.0	Scotch Whiskies
3	1032100.0	Imported Vodkas
4	1011400.0	Tennessee Whiskies

[9]:
```
# We notice that the category codes are neatly grouped. If the code starts with␣
 ↪"103", for example, then it is a Vodka. We use modular arithmetic to slice␣
 ↪the frame:
lcodes={}

lcodes[101] = 'Whiskey'
lcodes[102] = 'Tequila'
lcodes[103] = 'Vodka'
lcodes[104] = 'Gin'
lcodes[105] = 'Brandies'
lcodes[106] = 'Rum'
lcodes[107] = "Cocktails"
lcodes[108] = "Liquers"
lcodes[109] = "Distilled Spirits"
lcodes[110] = ""
lcodes[150] = "High Proof Beer"
lcodes[170] = "Temporary and Specialty Packages"
lcodes[190] = "Special Order Items"

Vodka=C[C['Category'].apply(lambda x: x//10000)==103]
Whiskies=C[C['Category'].apply(lambda x: x//10000)==101]
```

[10]:
```
Vodka.head(20)
```

[10]:

	Category	Category Name
0	1032200.0	Imported Flavored Vodka

```
3    1032100.0          Imported Vodkas
6    1031100.0          American Vodkas
22   1031200.0  American Flavored Vodka
45   1031080.0           VODKA 80 PROOF
47   1031200.0           VODKA FLAVORED
55   1032080.0           IMPORTED VODKA
58   1031000.0           American Vodka
66   1032200.0    IMPORTED VODKA - MISC
67   1032000.0           Imported Vodka
75   1031100.0          100 PROOF VODKA
121  1031090.0        OTHER PROOF VODKA
128  1031110.0          LOW PROOF VODKA
133  1032230.0  IMPORTED VODKA - CHERRY
```

[11]:
```
# Even more, '1032' is imported vodka, '1031' is American vodka:
# We notice the fourth digit is 1 if its imported, 2 if its domestic, and 0 if␣
 ↪its a special order item
# Let's load some more columns and do more exploratory analysis...

df = pd.read_csv("Iowa_Liquor_Sales.
 ↪csv",usecols=[1,6,10,22],parse_dates=['Date'],date_parser=pd.
 ↪to_datetime,infer_datetime_format=True)
df=df.dropna()
df['is_imported'] = (df.Category.apply(lambda x: str(x)[3] == '2')).astype(int)
```

```
/Users/adrienpeltzer/anaconda3/lib/python3.8/site-
packages/IPython/core/interactiveshell.py:3071: DtypeWarning: Columns (6) have
mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

[12]:
```
print(df['is_imported'].mean())
```

```
0.4211277705865148
```

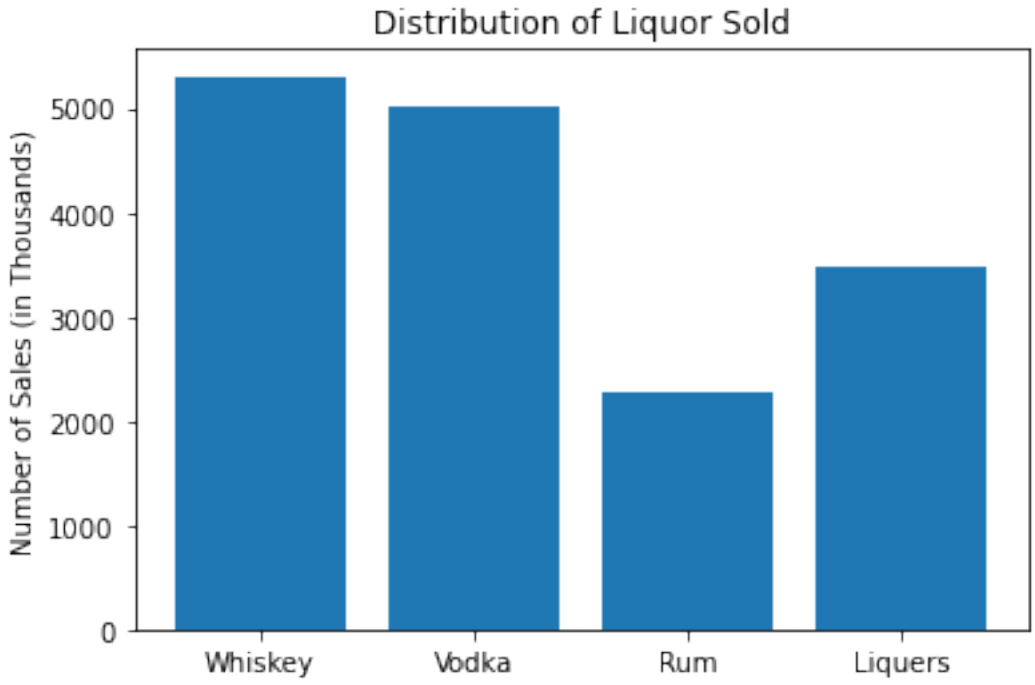
[13]:
```
# We see that about 42% of sales are of imported liquor, and this is consistent␣
 ↪throughout the years:
print(df.groupby(df.Date.dt.year)['is_imported'].mean())
```

```
Date
2012    0.419857
2013    0.418968
2014    0.420526
2015    0.420487
2016    0.425664
2017    0.429249
2018    0.423080
2019    0.418872
```

```
2020    0.411866
Name: is_imported, dtype: float64
```

[16]:
```
# Let's group the liquors by our more refined liquor type (e.g. Whiskey, Vodka,
 ↪Tequila, etc), and see how they sell
df['Liquor Type'] = df['Category'].apply(lambda x: x//10000)
bytype=df.groupby("Liquor Type").size()
```

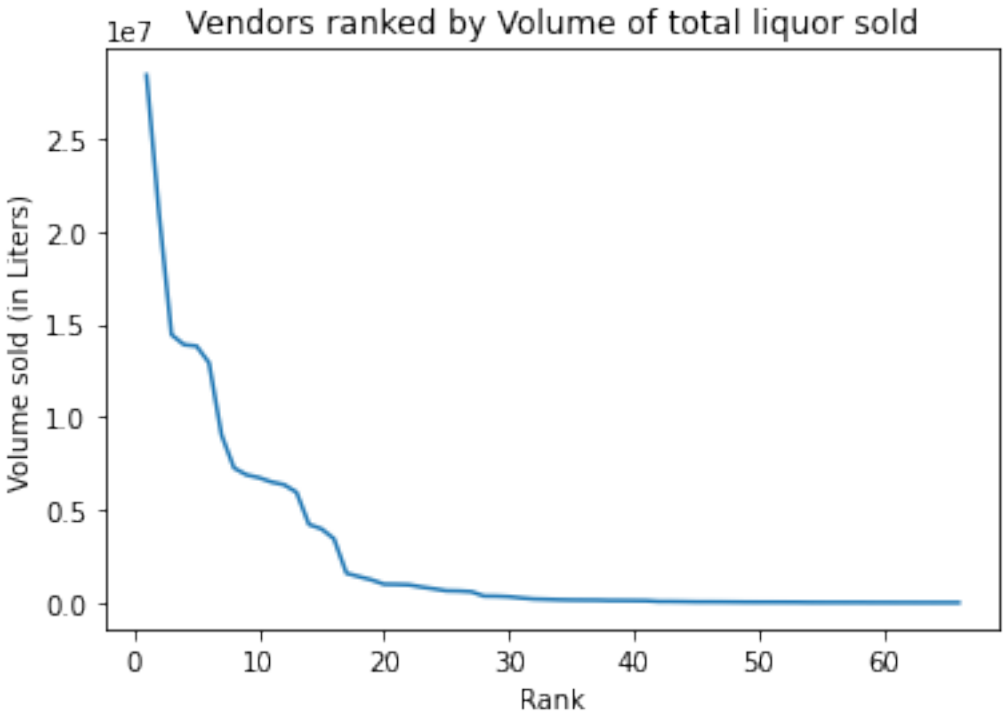
[17]:
```
# Let's plot four of the main liquor types and see how they compare
d = bytype.loc[[101,103,106,108]]
y = [xx/1000. for xx in d]
plt.figure()
x = range(1,5)
plt.bar(x,y)
plt.title("Distribution of Liquor Sold")
labels = [lcodes[i] for i in [101,103,106,108]]
plt.xticks(x,labels)
plt.ylabel("Number of Sales (in Thousands)")
plt.show()
```



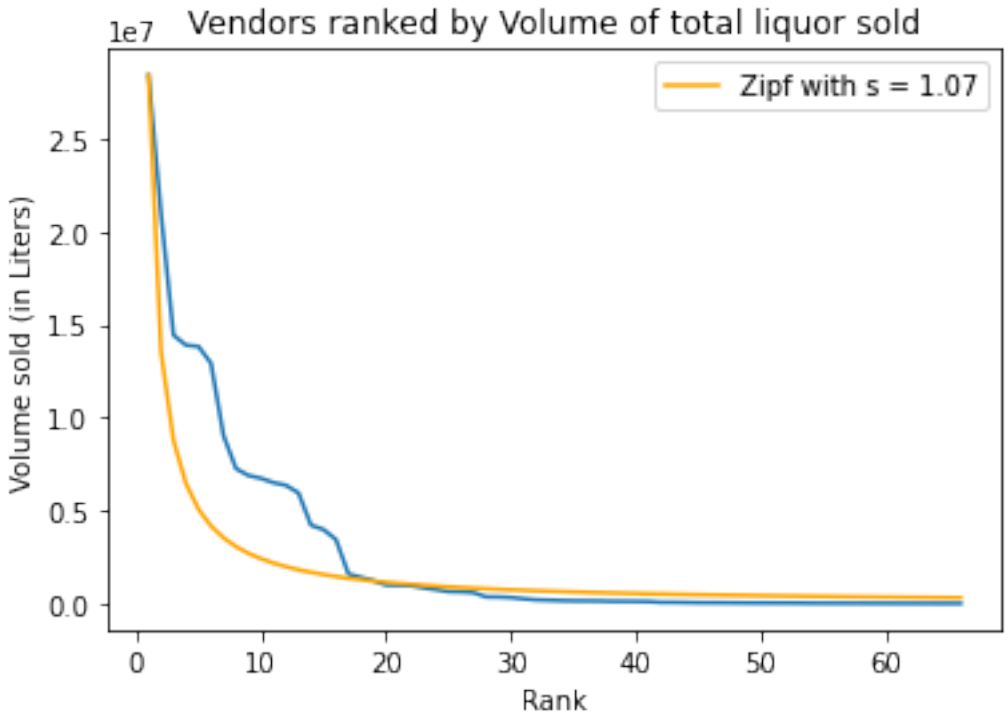
[19]:
```
# Whiskey and Vodka are leading in the sales department, followed by Rum and
 ↪Liquers.

plt.close()
```

[20]:
```
# Plot a graph of the stores ranked by total sales in volume of liquor sold. The
 ↪x is the rank, y is the volume. What type of distribution does this follow?


S = pd.read_csv("Iowa_Liquor_Sales.csv",usecols = [2,12,22])

#StoresByVolume = S.groupby("Store Number")['Volume Sold (Liters)'].sum().
 ↪sort_values(ascending=False)

VendorsByVolume = S.groupby("Vendor Number")['Volume Sold (Liters)'].sum().
 ↪sort_values(ascending=False)

# Remove the smallest Vendors
VV=VendorsByVolume[VendorsByVolume>10000]
```

[25]:
```
plt.figure()
plt.title("Vendors ranked by Volume of total liquor sold")
plt.xlabel("Rank")
plt.ylabel("Volume sold (in Liters)")
x=range(1,len(VV)+1)
plt.plot(x,VV)
plt.show()
```



[36]:
```
# Can we fit a line through this?
from math import factorial
import numpy as np
plt.figure()
plt.title("Vendors ranked by Volume of total liquor sold")
plt.xlabel("Rank")
plt.ylabel("Volume sold (in Liters)")
x=range(1,len(VV)+1)
plt.plot(x,VV)
# Plot a zipf distribution
s1=1.07
y1 = [VV.iloc[0]/n**s1 for n in x]
plt.plot(x,y1, label = "Zipf with s = 1.07",color="orange")

plt.legend()
plt.show()
```



[33]:
```
# That's it for Exploratoy Analysis. We will continue later
```

[ ]:
```

```