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
```

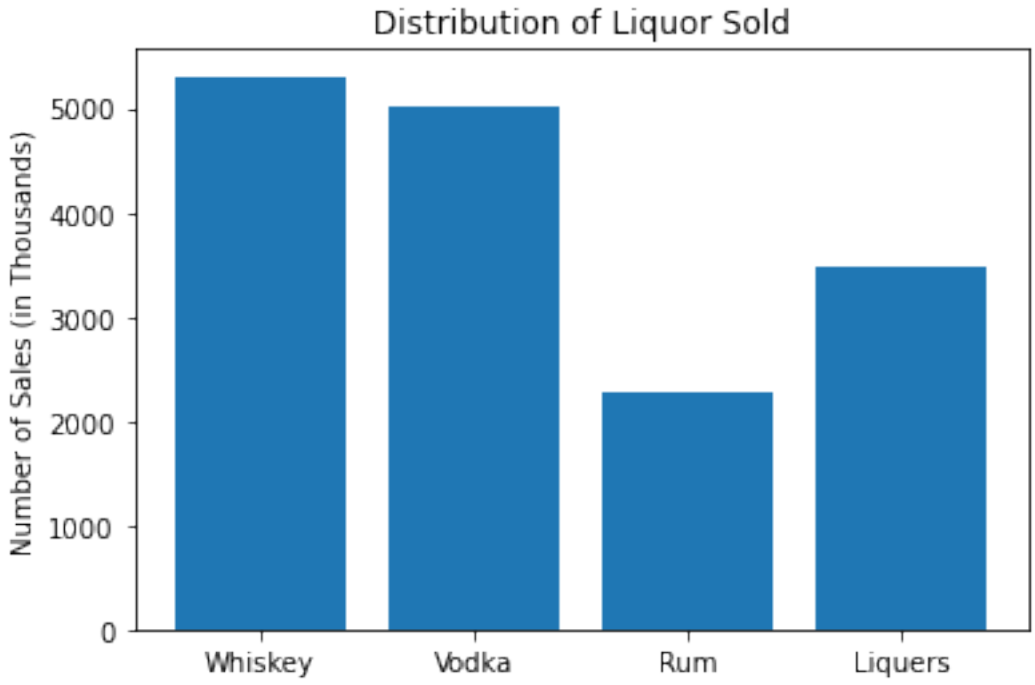
[13]:
```
# We see that about 42% of sales are of imported liquor, and this is consistent␣
 ↪throughout the years:
print(df.groupby(df.Date.dt.year)['is_imported'].mean())
```

```
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()
```

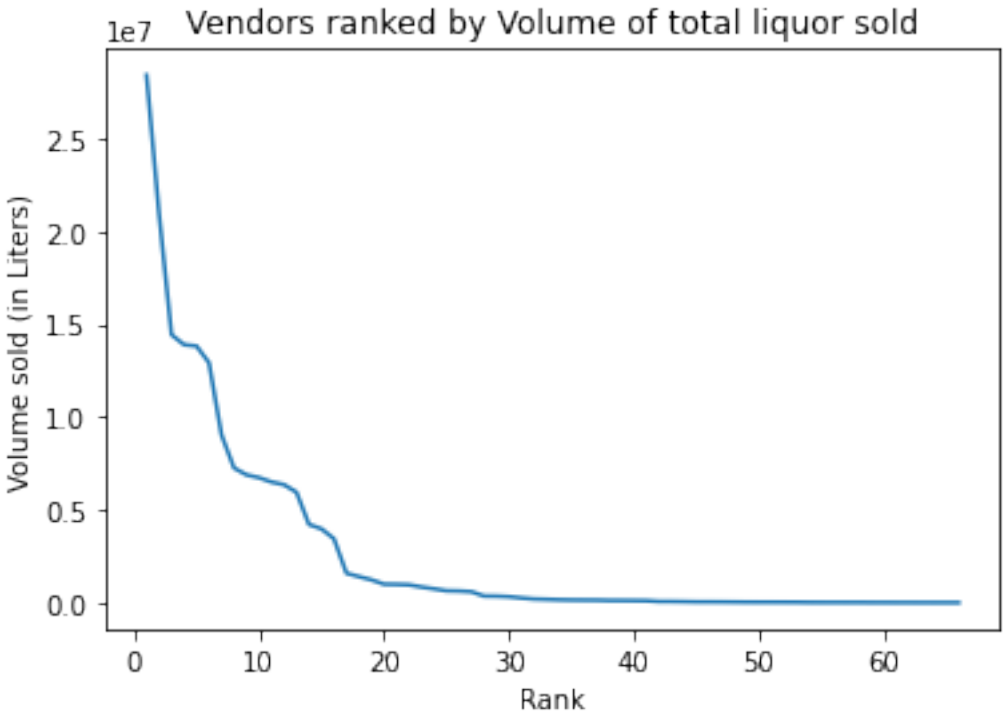
[17]:
```
# Let's plot four of the main liquor types and see how they compare
d = bytype.loc[[101,103,106,108]]
y = [xx/1000. for xx in d]
plt.figure()
x = range(1,5)
plt.bar(x,y)
plt.title("Distribution of Liquor Sold")
labels = [lcodes[i] for i in [101,103,106,108]]
plt.xticks(x,labels)
plt.ylabel("Number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



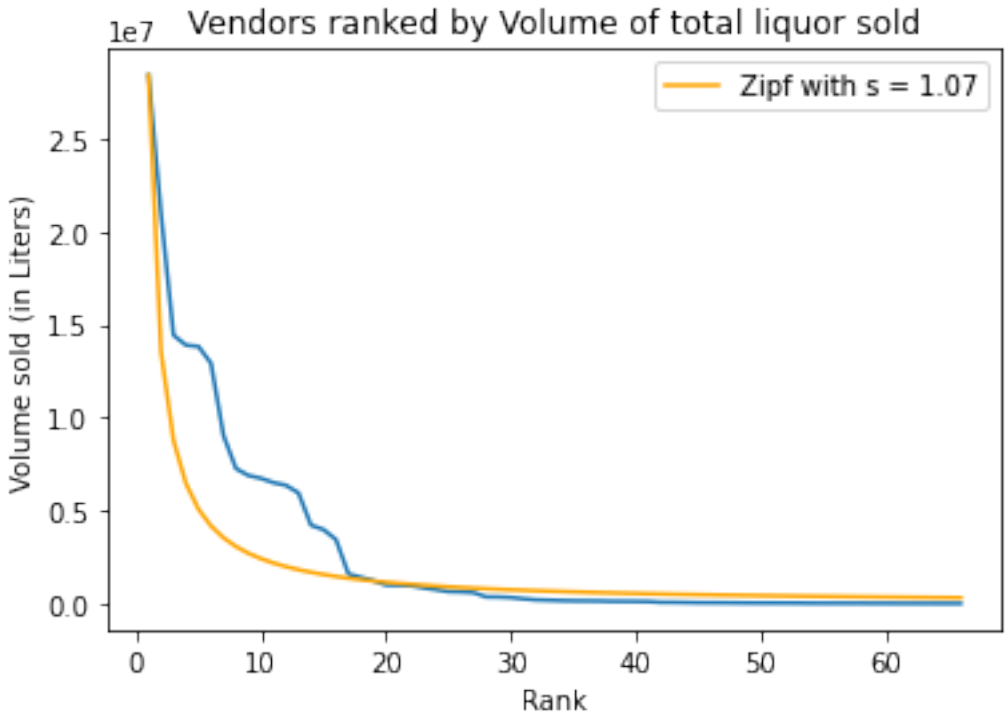
[19]:
```
# Whiskey and Vodka are leading in the sales department, followed by Rum and
 ↪Liquers.

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
```

[38]:
```
df.groupby(df.Date.dt.year)['Volume Sold (Liters)'].sum()
```

[38]:
```
Date
2012    1.875211e+07
```

```
2013    1.857239e+07
2014    1.915247e+07
2015    1.961214e+07
2016    1.999624e+07
2017    2.064086e+07
2018    2.184513e+07
2019    2.224774e+07
2020    1.784637e+07
Name: Volume Sold (Liters), dtype: float64
```

[39]:
```
twentytwenty  = df[df.Date.dt.year == 2020]
twentytwenty = twentytwenty.sort_values('Date')
```

[40]:
```
twentytwenty.head()
```

[40]:

	Date	Zip Code	Category	Volume Sold (Liters)	is_imported	\
876394	2020-01-02	50010	1031100.0	12.0	0	
878602	2020-01-02	51537	1011200.0	0.8	0	
878603	2020-01-02	51301	1081600.0	1.5	0	
878604	2020-01-02	50311	1082200.0	3.0	1	
878605	2020-01-02	51360	1011600.0	1.5	0	

	Liquor Type
876394	103.0
878602	101.0
878603	108.0
878604	108.0
878605	101.0

[41]:
```
twentytwenty.tail()
```

[41]:

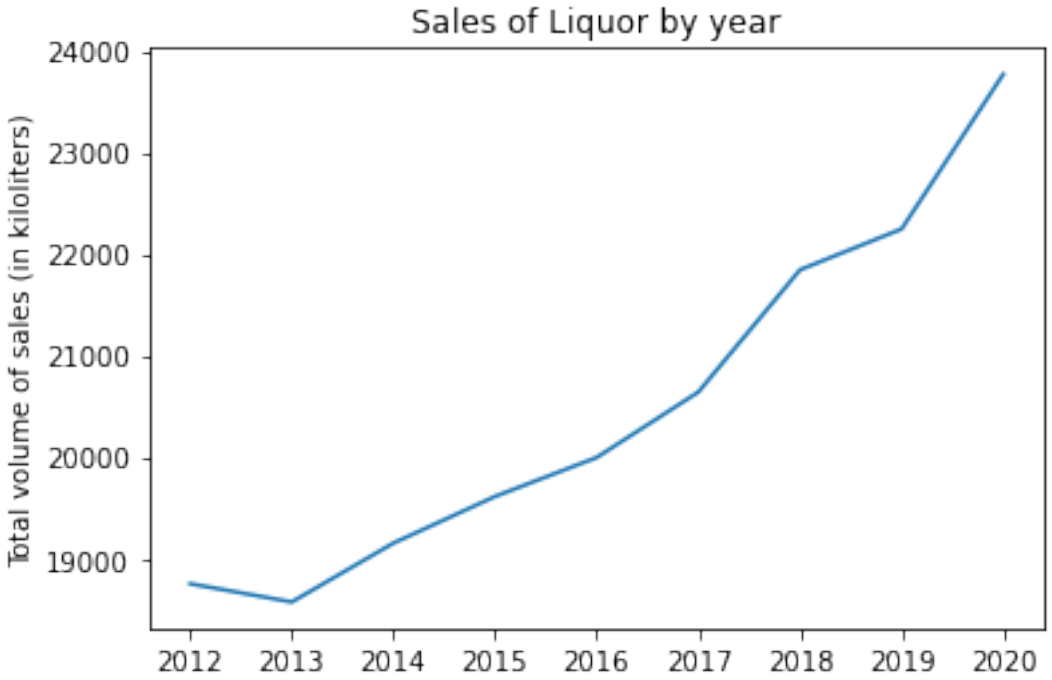
	Date	Zip Code	Category	Volume Sold (Liters)	is_imported	\
1796708	2020-09-30	50701	1012100.0	9.00	1	
1796709	2020-09-30	50311	1012400.0	9.00	1	
1796710	2020-09-30	51351	1031100.0	4.50	0	
1796699	2020-09-30	50010	1062400.0	10.50	1	
1792445	2020-09-30	52402	1011200.0	0.15	0	

	Liquor Type
1796708	101.0
1796709	101.0
1796710	103.0
1796699	106.0
1792445	101.0

[42]:
```
X=df.groupby(df.Date.dt.year)['Volume Sold (Liters)'].sum()
```

[46]:
```
# Let's plot the sales by year. Let's adjust 2020 based on how many days have␣
 ↪passed so far in the dataset.
# From the tail above, we see the last datapoint was on 2020-09-30, the 274th␣
 ↪day of the year. So let's multiply
# the last  value in X by 365/274
X[2020] = X[2020]*365/274
```

[52]:
```
plt.figure()
plt.title("Sales of Liquor by year")
plt.ylabel("Total volume of sales (in kiloliters)")
plt.plot(X.index,X.values/1000)
plt.show()
```

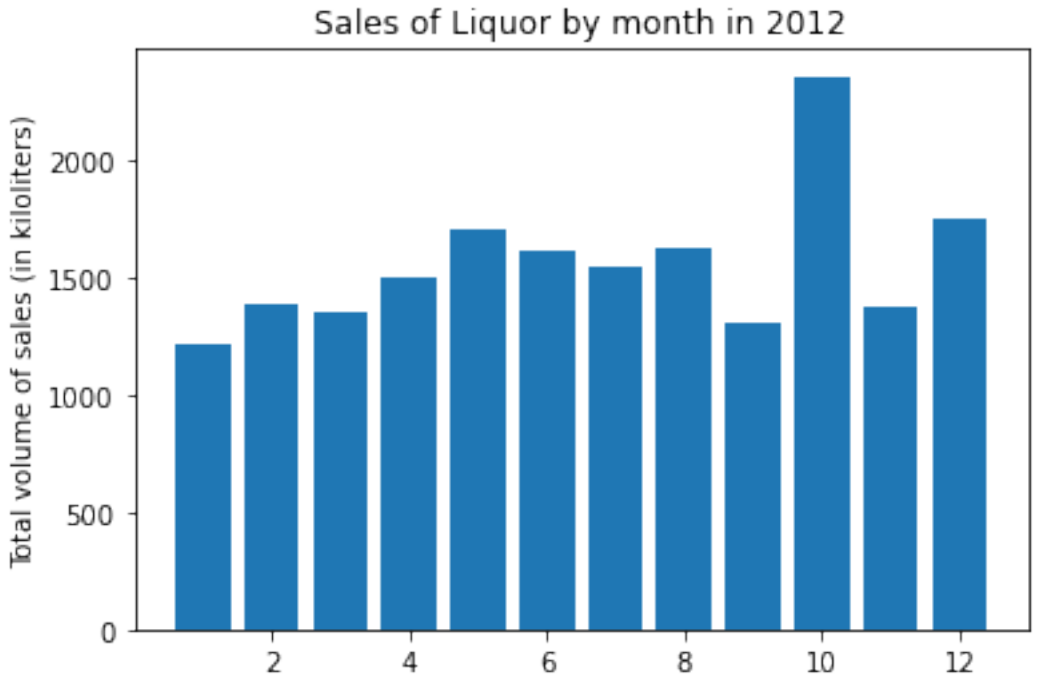


[53]:
```
# We can adjust this plot in a better way. Here, we assumed the total volume in␣
 ↪sales up to a certain day D
# grows linearly in the number of days before D.
# This is probably not the case, as we observe here:

T19 = df[df.Date.dt.year == 2019]

X = T19.groupby(T19.Date.dt.month)['Volume Sold (Liters)'].sum()
```

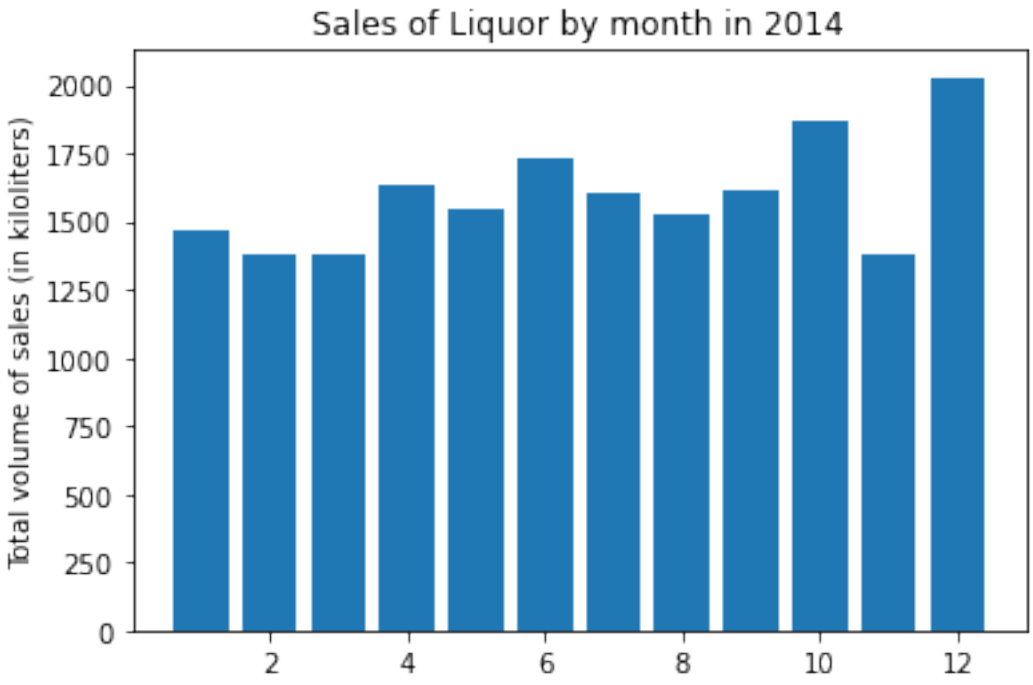
[54]:
```
X
```

[54]:
```
Date
1     1.560848e+06
2     1.679582e+06
3     1.606563e+06
4     1.850576e+06
5     2.171601e+06
6     1.861377e+06
7     2.032186e+06
8     1.805305e+06
9     1.746780e+06
10    2.096479e+06
11    1.799774e+06
12    2.036671e+06
Name: Volume Sold (Liters), dtype: float64
```

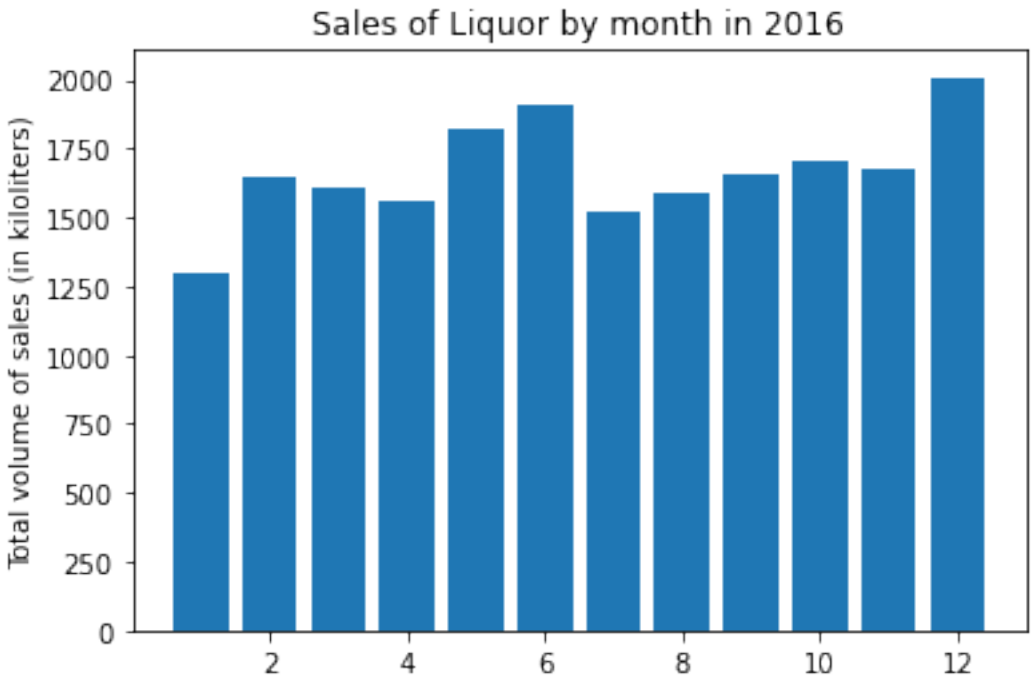
[59]:
```
for group, frame in df.groupby(df.Date.dt.year):
    X = frame.groupby(frame.Date.dt.month)['Volume Sold (Liters)'].sum()
    plt.figure()
    plt.title("Sales of Liquor by month in %s"% group)
    plt.ylabel("Total volume of sales (in kiloliters)")
    plt.bar(X.index,X.values/1000)
    plt.show()
```

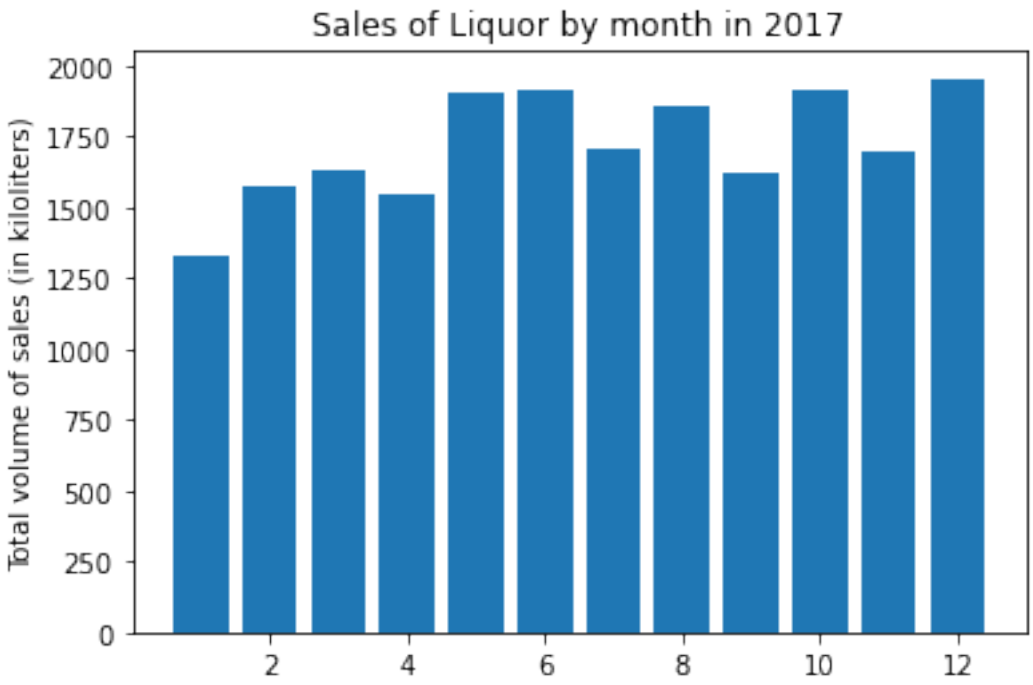


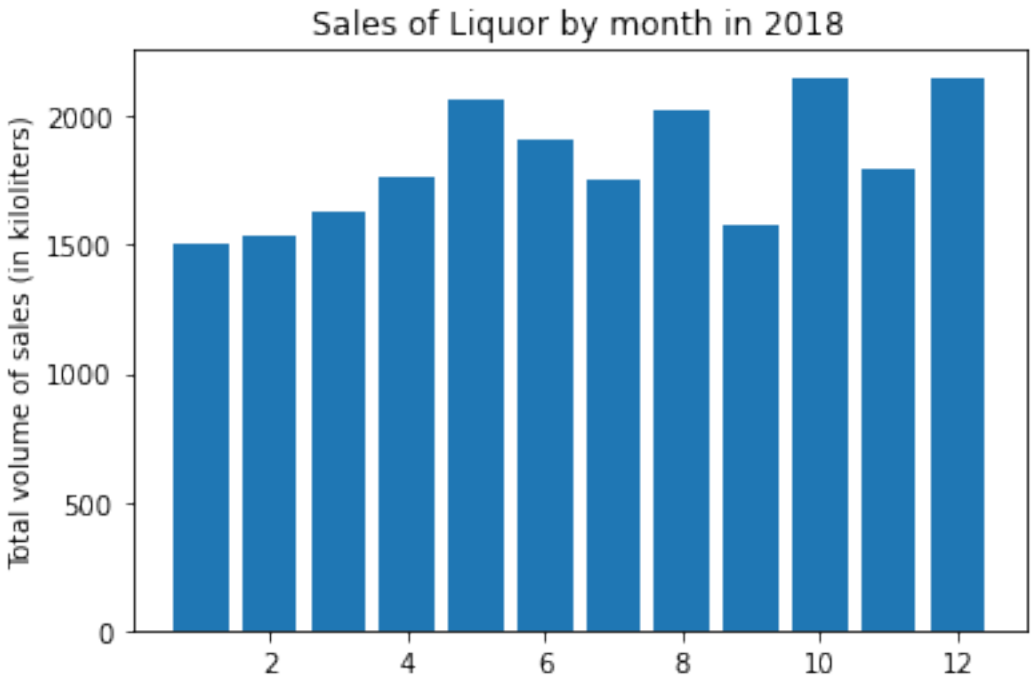

Sales of Liquor by month in 2013
Total volume of sales (in kiloliters)
0
500
1000
1500
2000
2
4
6
8
10
12

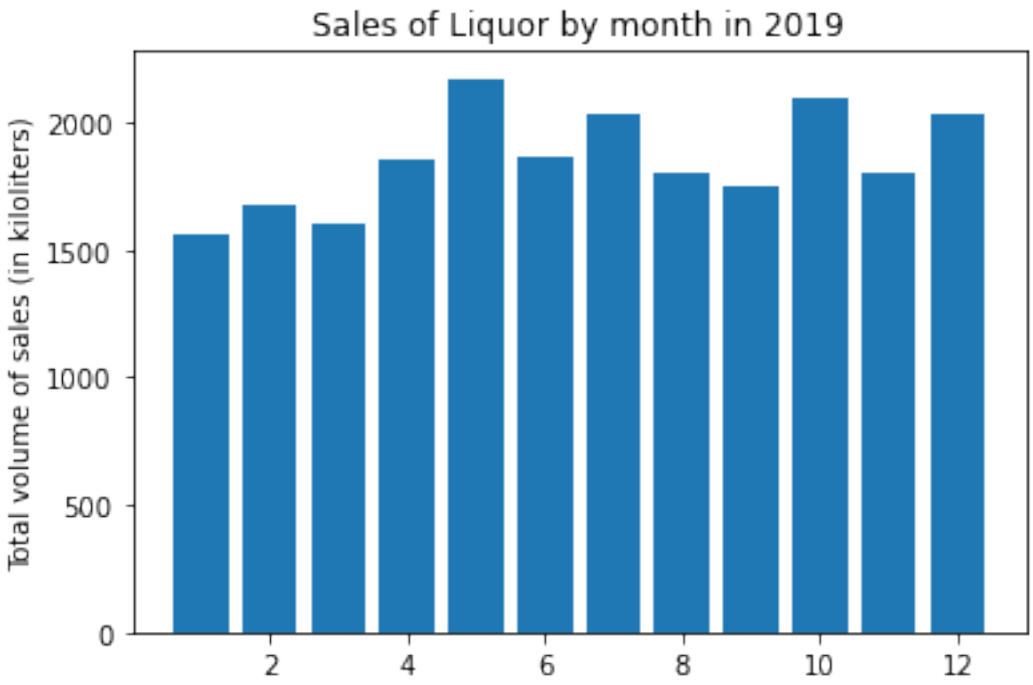

Sales of Liquor by month in 2014
Total volume of sales (in kiloliters)
0
250
500
750
1000
1250
1500
1750
2000
2
4
6
8
10
12


Sales of Liquor by month in 2015
Total volume of sales (in kiloliters)
2000
1750
1500
1250
1000
750
500
250
0
2
4
6
8
10
12


Sales of Liquor by month in 2016
Total volume of sales (in kiloliters)
2000
1750
1500
1250
1000
750
500
250
0
2
4
6
8
10
12


Sales of Liquor by month in 2017
Total volume of sales (in kiloliters)
2000
1750
1500
1250
1000
750
500
250
0
2
4
6
8
10
12

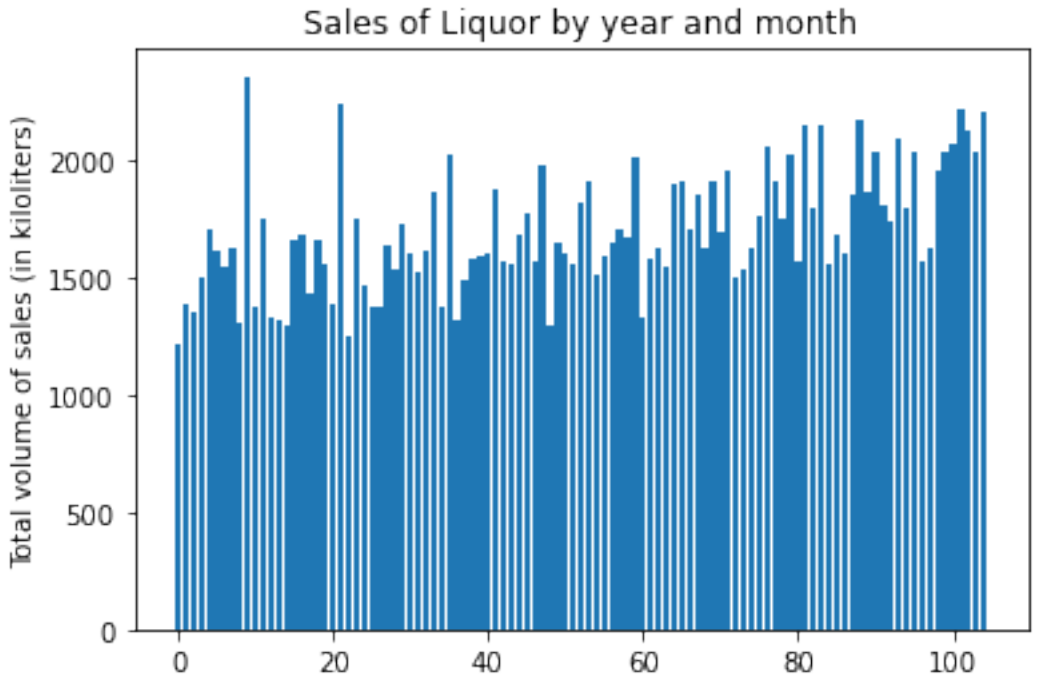

Sales of Liquor by month in 2018
Total volume of sales (in kiloliters)
2000
1500
1000
500
0
2
4
6
8
10
12



[58]:
```
# We see a general trend that sales grow by month every year.
```

[58]:
```
Series([], Name: Volume Sold (Liters), dtype: float64)
```

[72]:
```
# Let's group it all on one plot
D=df.groupby([df.Date.dt.year,df.Date.dt.month])['Volume Sold (Liters)'].sum()
plt.figure()
plt.title("Sales of Liquor by year and month")
plt.ylabel("Total volume of sales (in kiloliters)")
plt.bar(range(D.shape[0]),D.values/1000)
plt.show()
```



[73]:
```
# Here we see the cyclic pattern a little bit better
```

[ ]:
```

```

[ ]:
```

```