Probability and Statistics

1. Probability
   - random variables in sympy
   - random variables in scipy
2. Statistics
   - bar plots
   - pie charts
   - analyzing a ratio with a histogram
   - time plots

MCS 507 Lecture 33
Mathematical, Statistical and Scientific Software
Jan Verschelde, 6 November 2023

Probability and Statistics

1. Probability
   - random variables in sympy
   - random variables in scipy
2. Statistics
   - bar plots
   - pie charts
   - analyzing a ratio with a histogram
   - time plots

random variables in sympy

Following pages 231 and 232 of *Mastering SciPy*,
we define three 6-sided dice and one 100-sided dice.

```
from sympy.stats import Die, sample_iter

D6_1 = Die('D6_1', 6)
D6_2 = Die('D6_2', 6)
D6_3 = Die('D6_3', 6)
D100 = Die('D100', 100)
X = D6_1 + D6_2 + D6_3 + D100
print('Ten samples : ', end='')
for item in sample_iter(X, numsamples=10):
    print(item, end=' ')
```

The output:

```
Ten samples : 71 86 91 60 78 76 39 99 81 103
```

computing probabilities, script continues

```
from sympy import Eq
from sympy.stats import P

print('Probability of sum of dice : ', end='')
print(P(D6_1 + D6_2 + D6_3 < D100))
print('Given that the first throw equals 5,')
print('probability that sum of first ', end='')
print('two throws is > 9 : ', end='')
print(P(D6_1 + D6_2 > 9, Eq(D6_1, 5)))
```

The output:

```
Probability of sum of dice : 179/200
Given that the first throw equals 5,
probability that sum of first two throws is > 9 : 1/3
```

variance, standard deviation, and expected value

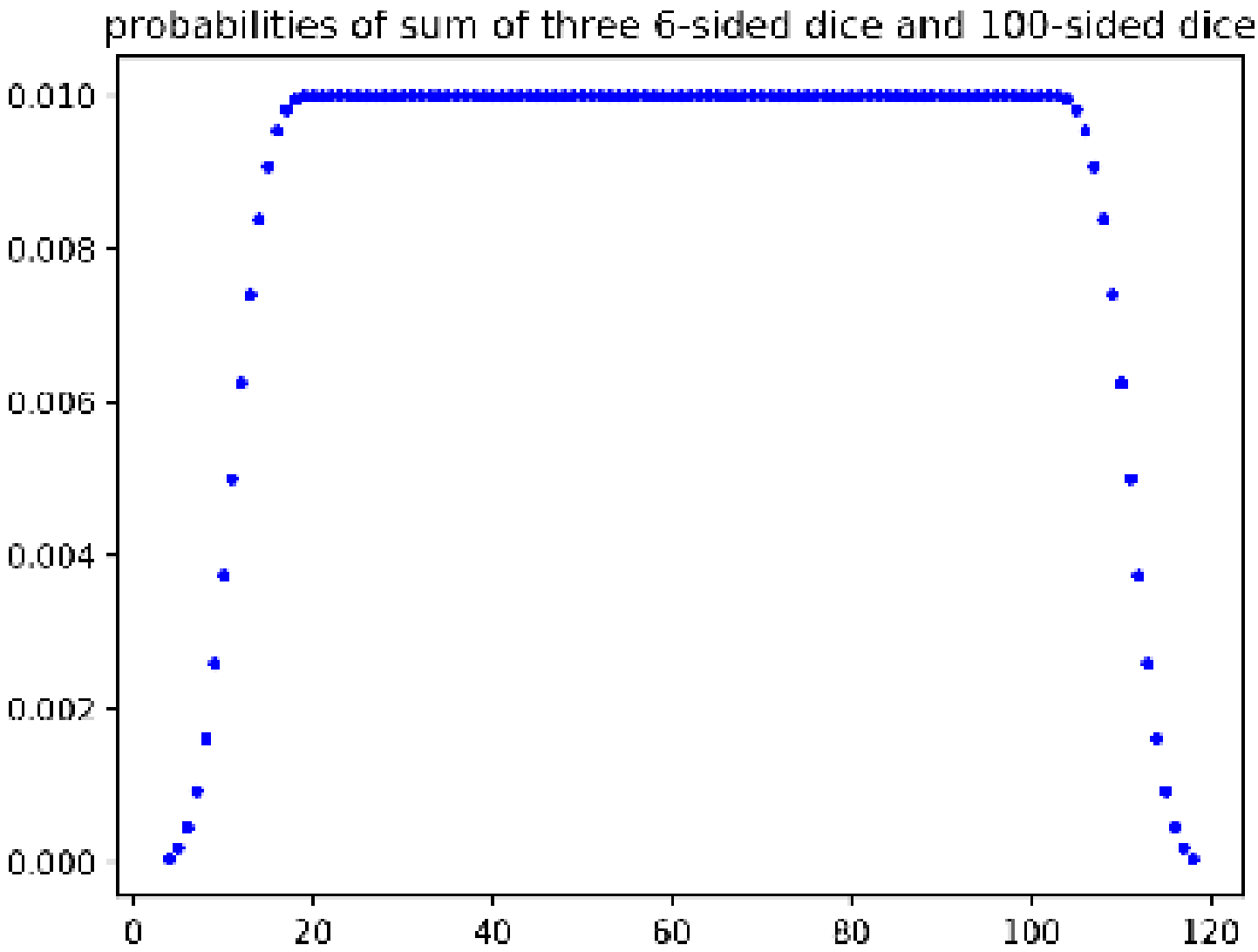
The script continues.

```
from sympy.stats import variance, std, E

print('variance of X : ', variance(X))
print('standard deviation of X :', std(X))
print('expected value of X :', E(X))
```

The output:

```
variance of X :  842
standard deviation of X : sqrt(842)
expected value of X : 61
```

the probability density function



matplotlib code for the probability density

```
import numpy as np, matplotlib.pyplot as plt
from sympy.stats import Die, density

D6_1 = Die('D6_1', 6)
D6_2 = Die('D6_2', 6)
D6_3 = Die('D6_3', 6)
D100 = Die('D100', 100)
X = D6_1 + D6_2 + D6_3 + D100
pdfX = density(X)

outcomes = list(pdfX.keys())
probabilities = list(pdfX.values())
xnp = np.array(outcomes)
ynp = np.array(probabilities)

plt.title('probabilities of sum of three \
    6-sided dice and 100-sided dice')
plt.plot(xnp, ynp, 'b.')
plt.show()
```

defining a continuous random variable

```
from sympy import var
from sympy.stats import Exponential, density

print('defining a continuous random variable Y')
mu = var('mu', positive=True)
t = var('t')
Y = Exponential('Y', mu)
print('cumulative distribution function of Y :')
print(density(Y)(t))
```

The output:

```
cumulative distribution function of Y :
mu*exp(-mu*t)
```

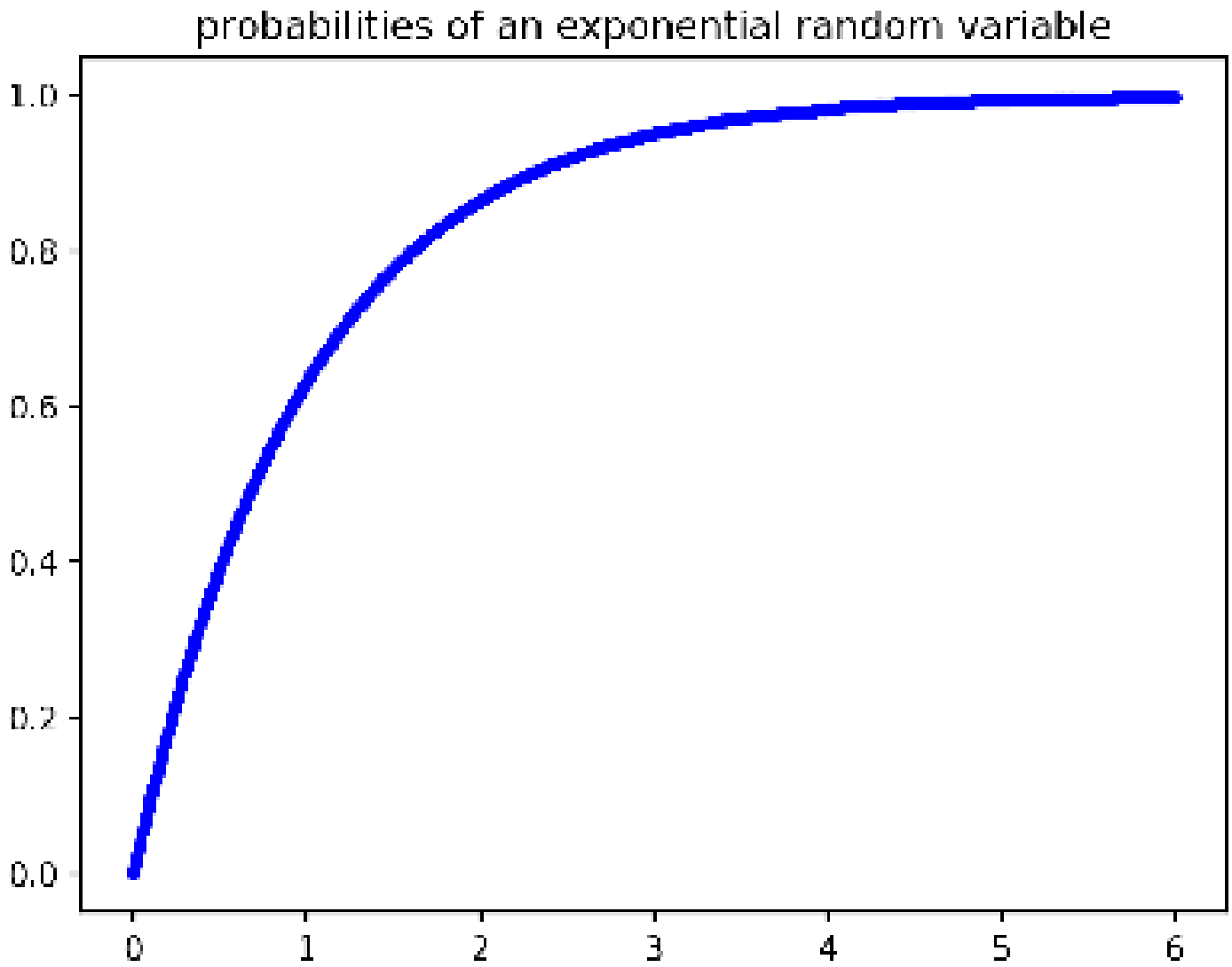
the probability density function

```
from sympy.stats import cdf

print('probability density function of Y :')
pdfY = cdf(Y)(t)
print(pdfY)
pdfYmu1 = pdfY.subs(mu, 1)
print(pdfYmu1)
```

The output:

```
probability density function of Y :
Piecewise((1 - exp(-mu*t), t >= 0), (0, True))
Piecewise((1 - exp(-t), t >= 0), (0, True))
```

plot of the probability density function



code to make the plot

```
import numpy as np, matplotlib.pyplot as plt
from sympy import lambdify

fun = lambdify(t, pdfYmu1, 'numpy')
xnp = np.linspace(0, 6, 1000)
ynp = fun(xnp)

plt.title('probabilities of an exponential \
    random variable')
plt.plot(xnp, ynp, 'b.')
plt.show()
```

Probability and Statistics

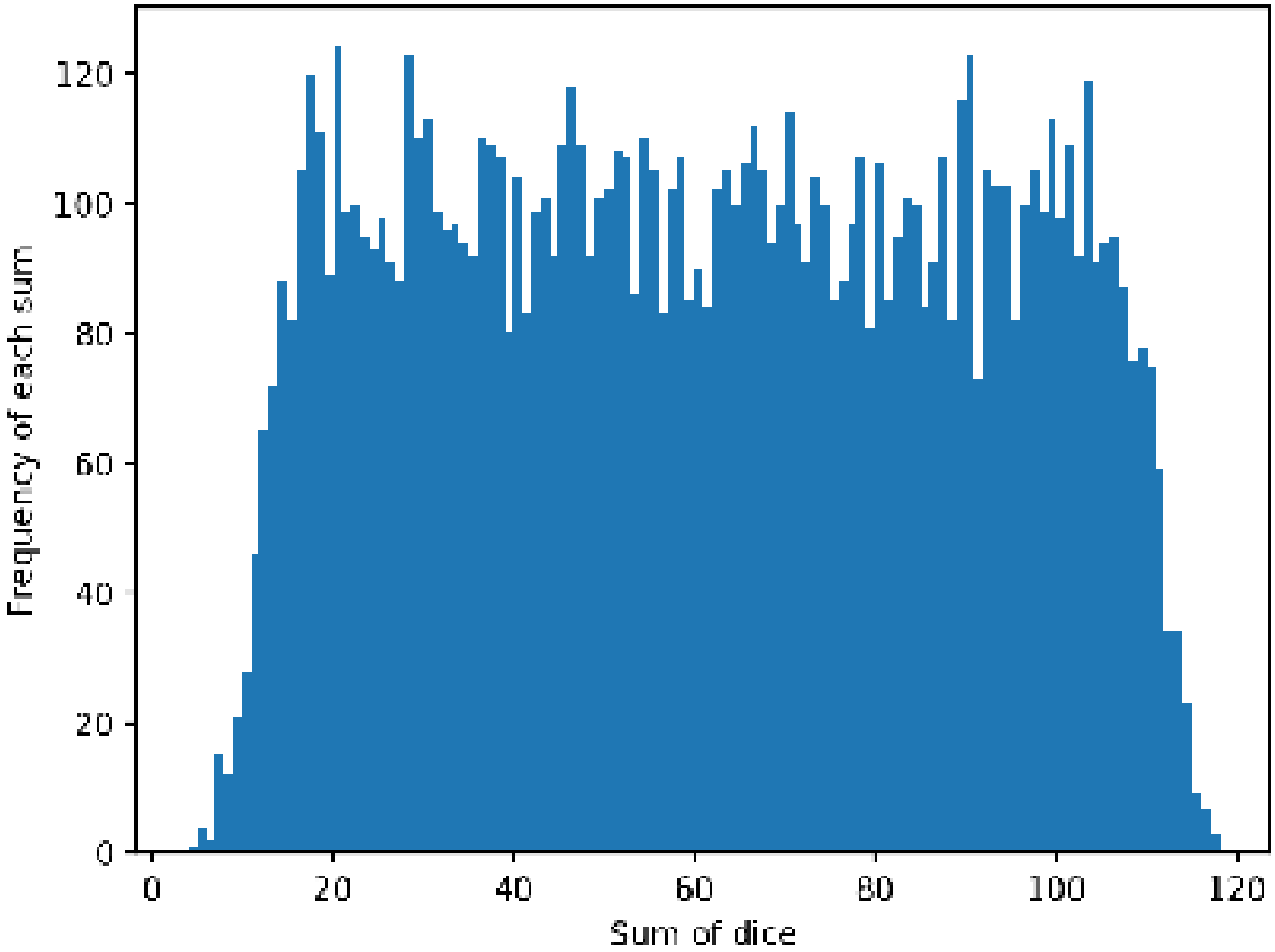
1. Probability
   - random variables in sympy
   - random variables in scipy
2. Statistics
   - bar plots
   - pie charts
   - analyzing a ratio with a histogram
   - time plots

random variables in scipy

We can import `randint` from `scipy.stats`
to redo the sum of the dice experiment we did with `sympy`.

```
import numpy as np, matplotlib.pyplot as plt
from scipy.stats import randint

D6 = randint(1, 7)
D100 = randint(1, 101)

samples = D6.rvs(10000) + D6.rvs(10000) \
        + D6.rvs(10000) + D100.rvs(10000)
plt.hist(samples, bins=118-4)
plt.xlabel('Sum of dice')
plt.ylabel('Frequency of each sum')
plt.show()
```

the histogram of the samples



compute a kernel density estimate

Using Gaussian kernels, we compute a kernel density estimate:

```
from scipy.stats import gaussian_kde, rv_discrete

kernel = gaussian_kde(samples)

print(kernel(50))
# The actual answer is 1/100
print(kernel.integrate_box_1d(0,100))
# The actual answer is 177/200
```

The output:

```
[0.01008687]
0.8908240691739442
```

applying convolution on the samples

```
probs_6dice = D6.pmf(np.linspace(1,6,6))
probs_100dice = D100.pmf(np.linspace(1,100,100))
probs_sum = np.convolve(np.convolve(probs_6dice,probs_6dice),
                        np.convolve(probs_6dice,probs_100dice))
space_sum = np.linspace(4, 118, 115)
sum_of_dice = rv_discrete(name="sod",
                          values=(space_sum, probs_sum))
print(sum_of_dice.pmf(50))
print(sum_of_dice.cdf(100))
```

The output:

```
0.01
0.8950000000000006
```

Probability and Statistics

1. Probability
   - random variables in sympy
   - random variables in scipy

2. Statistics
   - bar plots
   - pie charts
   - analyzing a ratio with a histogram
   - time plots

the Consumer Complaints Database

The Consumer Complaints Database is available at `http://catalog.data.gov/dataset/consumer-complaint-database` as one large `.csv` file.

With `pandas` we load the database into a `Dataframe`.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("Consumer_Complaints.csv")
print(data.head())
```

output of `data.head()`

```
  Date received  ... Complaint ID
0    08/09/2015  ...       1509954
1    01/29/2019  ...       3136759
2    08/19/2015  ...       1527601
3    03/04/2016  ...       1816726
4    03/18/2013  ...        358304

[5 rows x 18 columns]
```

What is the range of dates of received complaints?

examining the columns

To answer the question about the range of dates,
we examine the type of the columns first.

```
C = data.columns
print(C)
```

The output:

```
Index(['Date received', 'Product', 'Sub-product', 'Issue',
       'Sub-issue', 'Consumer complaint narrative',
       'Company public response', 'Company', 'State',
       'ZIP code', 'Tags', 'Consumer consent provided?',
       'Submitted via', 'Date sent to company',
       'Company response to consumer', 'Timely response?',
       'Consumer disputed?', 'Complaint ID'],
      dtype='object')
```

computing the range of the dates

To compare dates, we must convert to the `Timestamp` type, just doing `min()` and `max()` gives wrong results.

The script continues:

```
received = C[0]
dates = data[received]
dates = pd.to_datetime(dates)
print(dates.min())
print(dates.max())
```

The output:

```
Timestamp('2011-12-01 00:00:00')
Timestamp('2019-11-01 00:00:00')
```

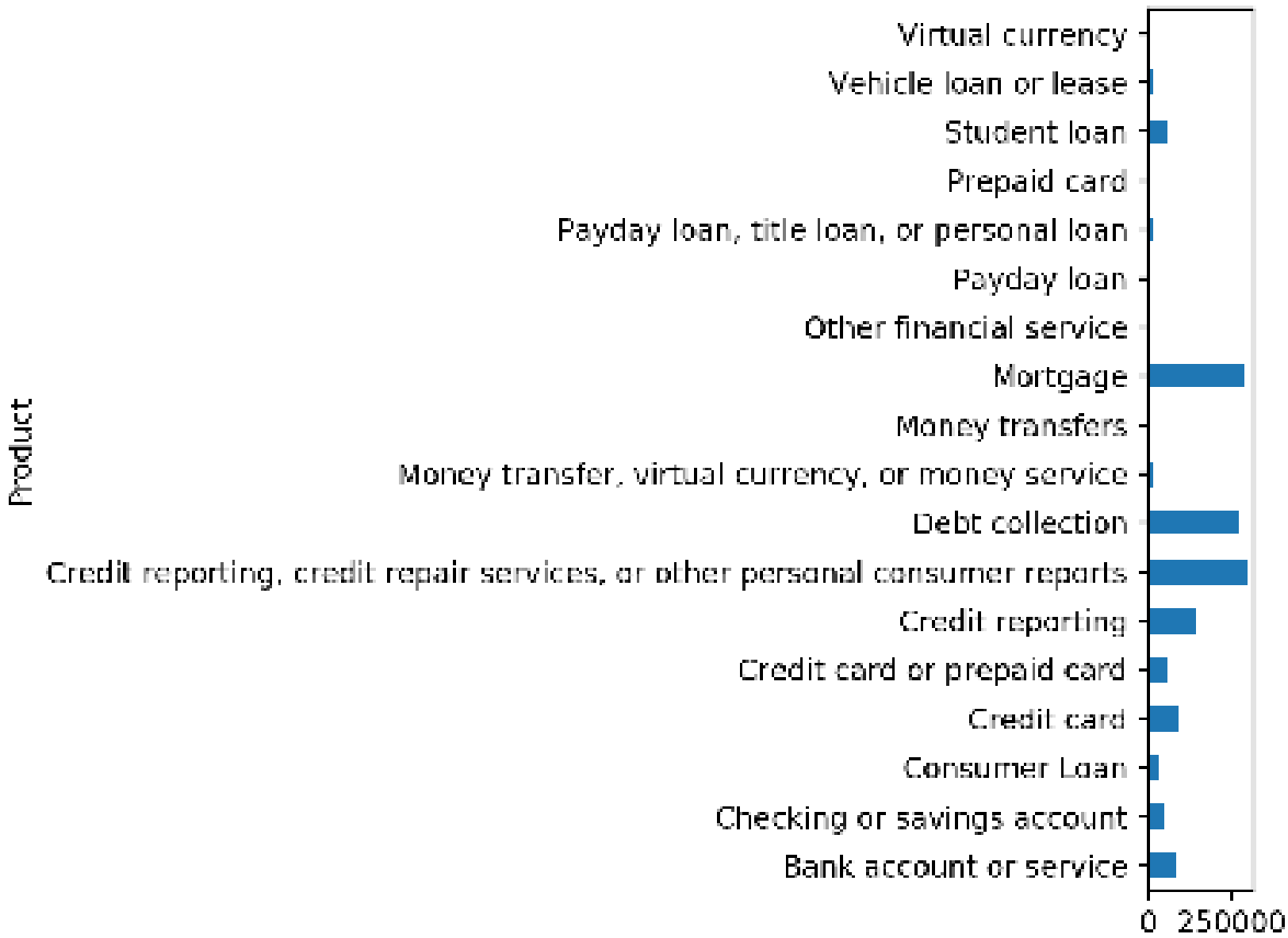
We have almost nine years of consumer complaints.

making a bar plot

We make a horizontal bar plot to show the size of complaints for each different product.

```
groups = data.groupby('Product').size()
print(groups)
groups.plot(kind='barh')
plt.tight_layout()
plt.show()
```

the histogram of the complaints



mortgage complaints in the midwest

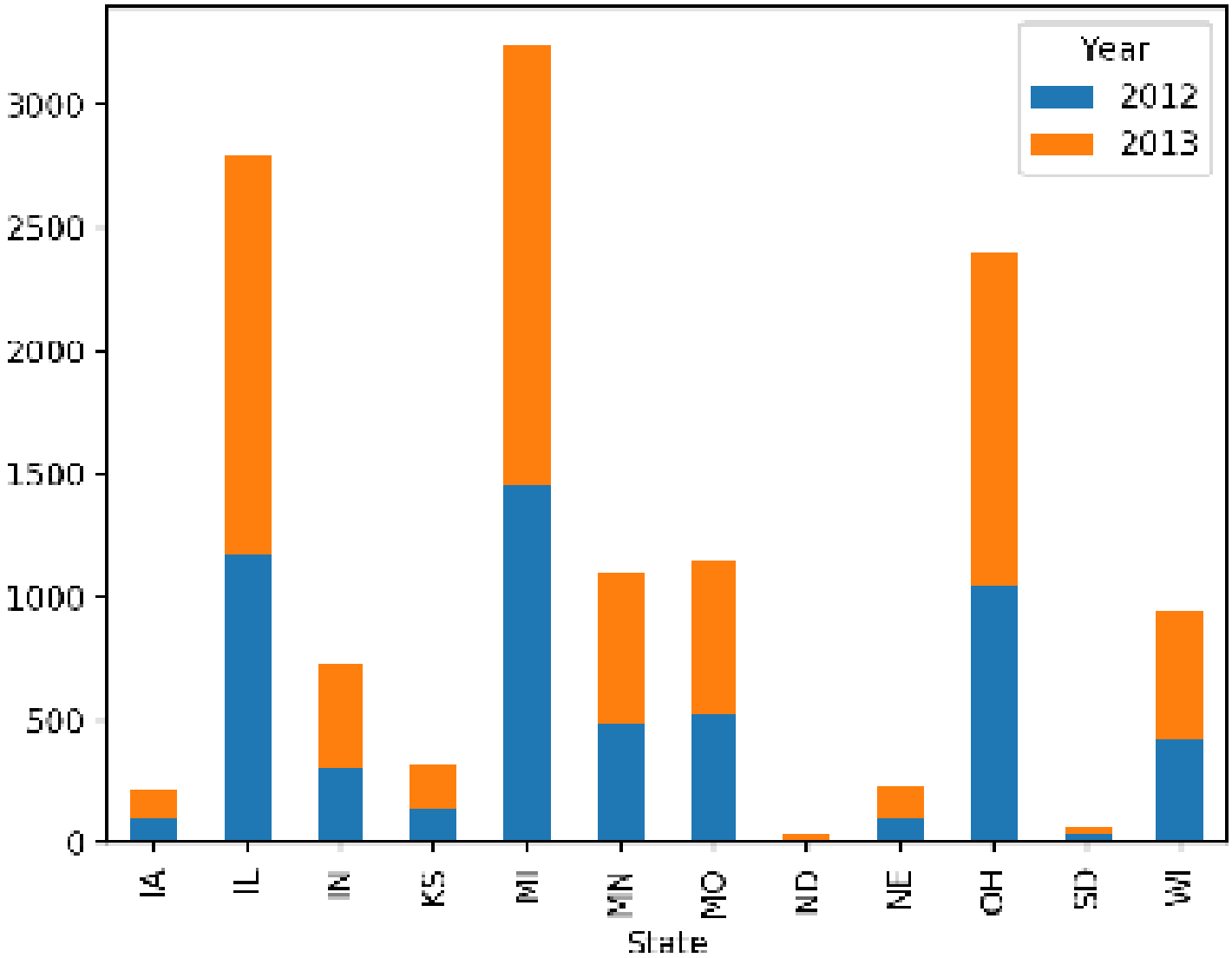
Let us focus on mortgage cmoplaints in the midwest,
during the years 2012 and 2013.
The `data` contains the DataFrame of the database.

```
midwest = ['ND', 'SD', 'NE', 'KS', 'MN', 'IA', \
           'MO', 'IL', 'IN', 'OH', 'WI', 'MI']
df = data[data.Product == 'Mortgage'];
yrdata = pd.to_datetime(df['Date received'])
df['Year'] = yrdata.map(lambda t: t.year)
df = df.groupby(['State', 'Year']).size()
barplotdata = df[midwest].unstack().loc[:, 2012:2013]
print(barplotdata)
barplotdata.plot(kind="bar", stacked=True)
plt.show()
```

the numerical data

Year	2012	2013
State		
IA	99	125
IL	1178	1611
IN	308	416
KS	146	169
MI	1459	1774
MN	478	623
MO	519	629
ND	14	20
NE	108	119
OH	1045	1355
SD	33	35
WI	417	523

mortgage complaints in the midwest



Probability and Statistics

1. Probability
   - random variables in sympy
   - random variables in scipy
2. Statistics
   - bar plots
   - pie charts
   - analyzing a ratio with a histogram
   - time plots

relative size of complaints

A pie charts relates the size of some complaints
relative to other complaints.

We will query the data with the following question:

- We are interested in the state Puerto Rico (PR).
- The complaints we are interested in concern the products `Credit card`, `Credit reporting`, `Mortgage`, and `Student loan`.

We have to query the data frame based on the values of rows
in the column `Product`.

executing a query and making a pie chart

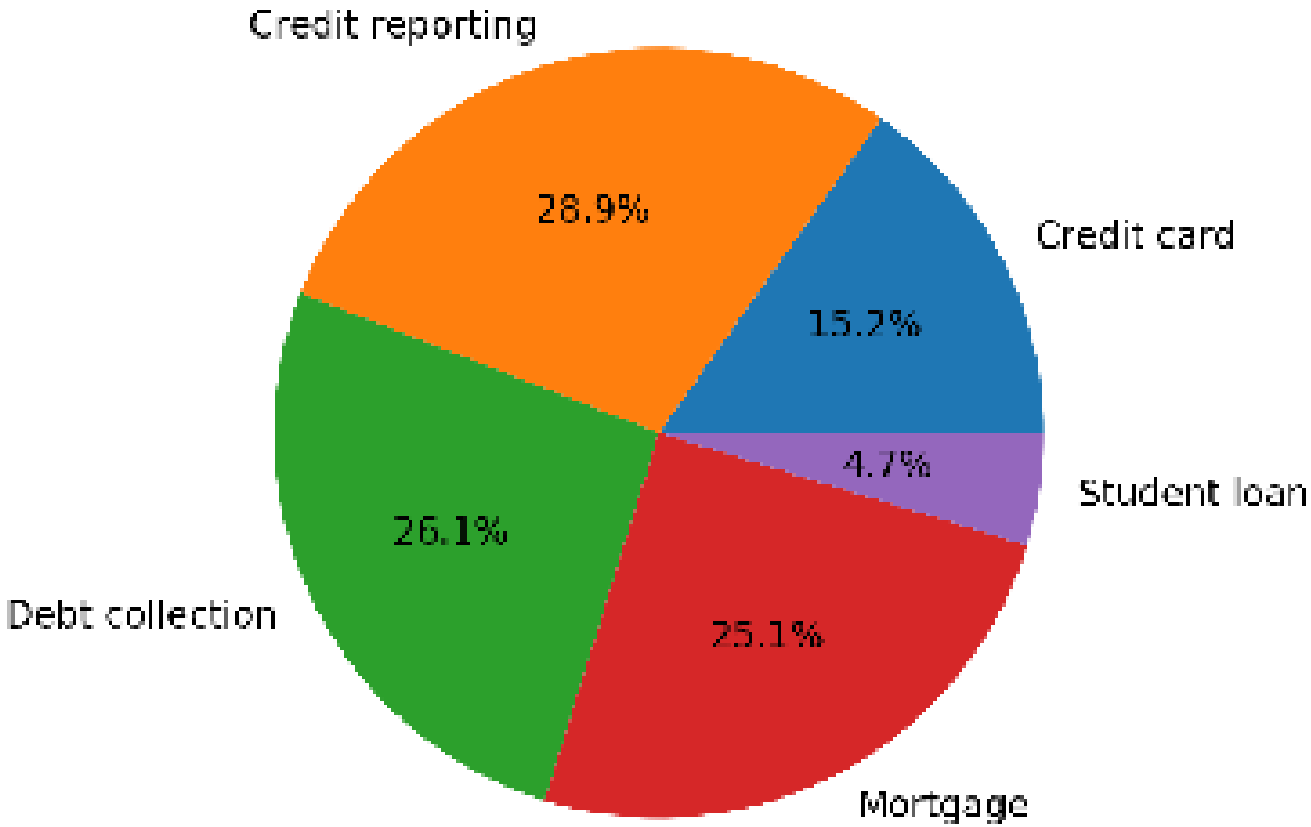
The `data` contains the data frame of the database.

```
prdata = data[data['State'] == 'PR']
print('size of prdata : ', prdata.shape)

somevalues = ['Mortgage', 'Credit card', 'Student loan', \
              'Credit reporting', 'Debt collection']

complaints = prdata.loc[prdata['Product'].isin(somevalues)]
print('size of complaints : ', complaints.shape)

piedata = complaints.groupby('Product').size()
print(piedata)

piedata.plot(kind='pie', autopct="%1.1f%%")
plt.ylabel("") # remove the None
plt.show()
```

the numerical output

```
size of prdata :  (3526, 18)
size of complaints :  (2164, 18)

Product
Credit card           329
Credit reporting      625
Debt collection       565
Mortgage              543
Student loan          102
dtype: int64
```

a pie chart of complaints in Puerto Rico



Probability and Statistics

1. Probability
   - random variables in sympy
   - random variables in scipy
2. Statistics
   - bar plots
   - pie charts
   - analyzing a ratio with a histogram
   - time plots

analyzing a ratio with a histogram

We continue to explore the Comsumer Complaints database.

We are interested in the ratio of

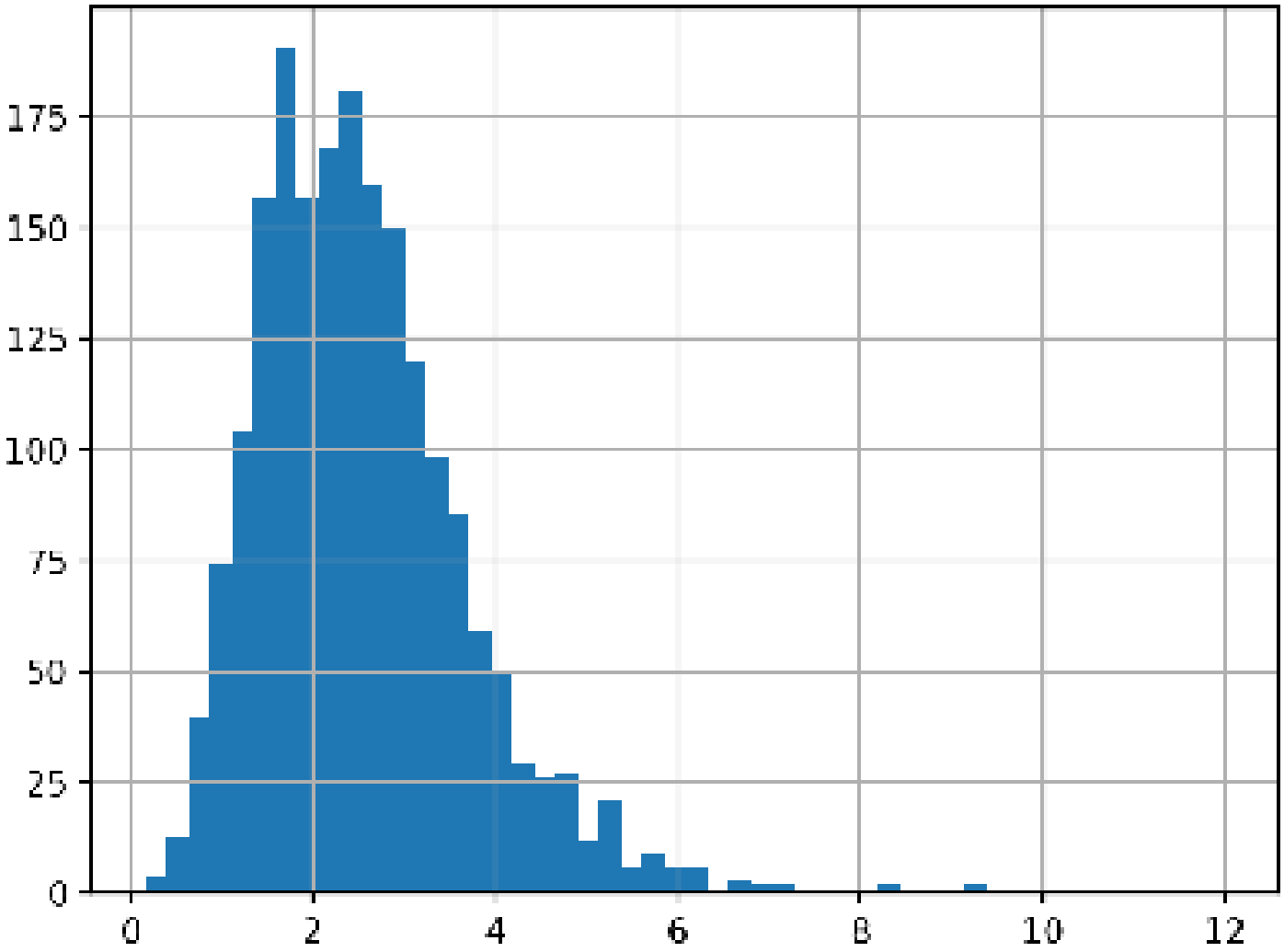
- the number of daily complaints on mortgages, against
- the number of daily complaints on credit cards.

To analyze the ratio, we will make a histogram.

plotting the ratio with a histogram

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("Consumer_Complaints.csv")
df = data.groupby(['Date received',
                   'Product']).size()
df = df.unstack()
ratios = df['Mortgage'] / df['Credit card']
ratios.hist(bins=50)
plt.show()
```

complaints on mortgages versus on credit cards



Probability and Statistics

1. Probability
   - random variables in sympy
   - random variables in scipy

2. Statistics
   - bar plots
   - pie charts
   - analyzing a ratio with a histogram
   - time plots

looking at the evolution over time

Time plots show variables measured at intervals over time.

Some questions:

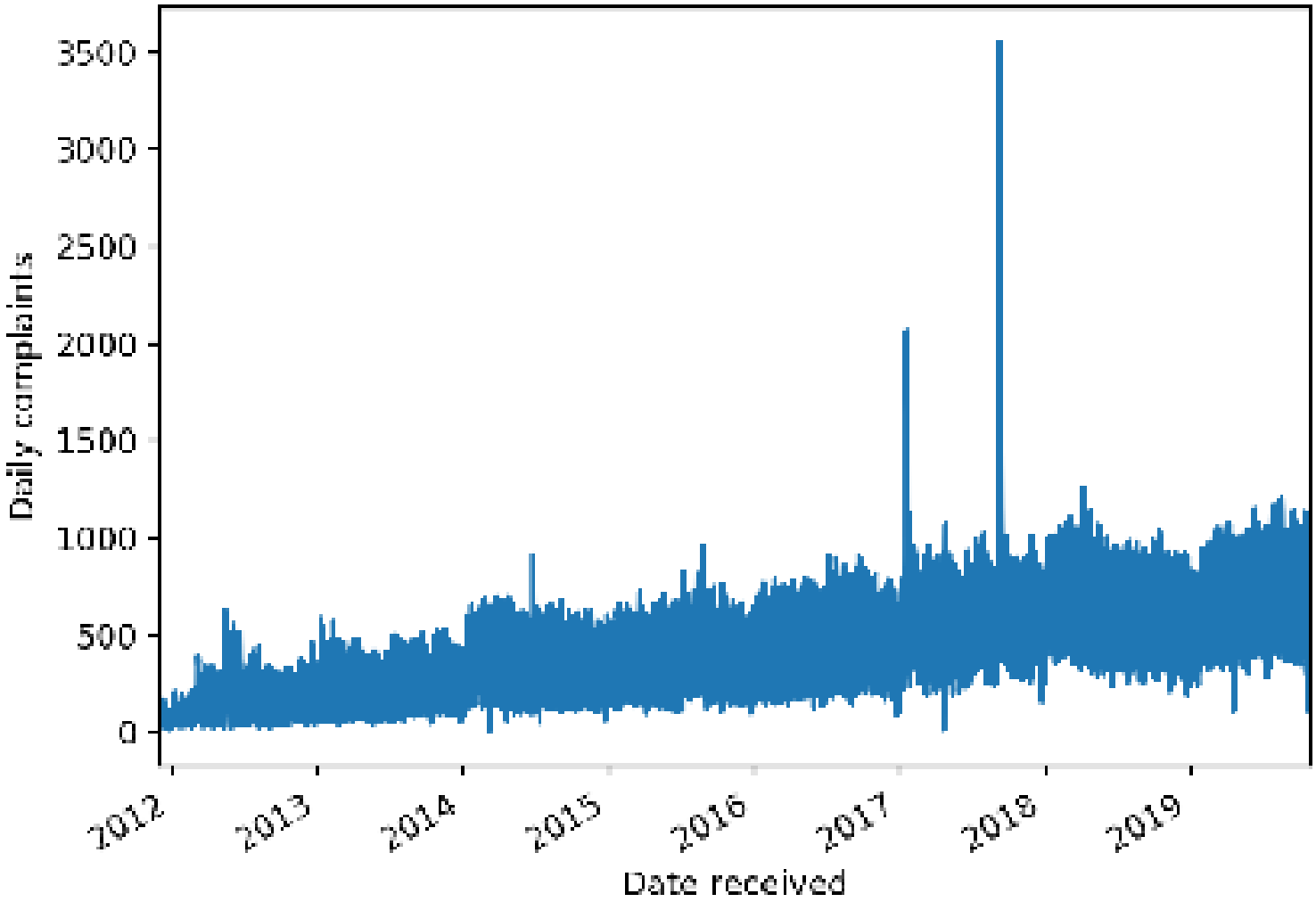
- Is there a growing trend?
  Do consumers complain more over time?
- Is there a seasonal pattern over the year?
  Are there months with more complaints than others?

Let us see how `pandas` explores time plots.

plotting the daily complaints over time

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("Consumer_Complaints.csv")
data['Date received']
    = pd.to_datetime(data['Date received'])

ts = data.groupby('Date received').size()
ts.plot()
plt.ylabel('Daily complaints')
plt.show()
```

amount of daily complaints over time



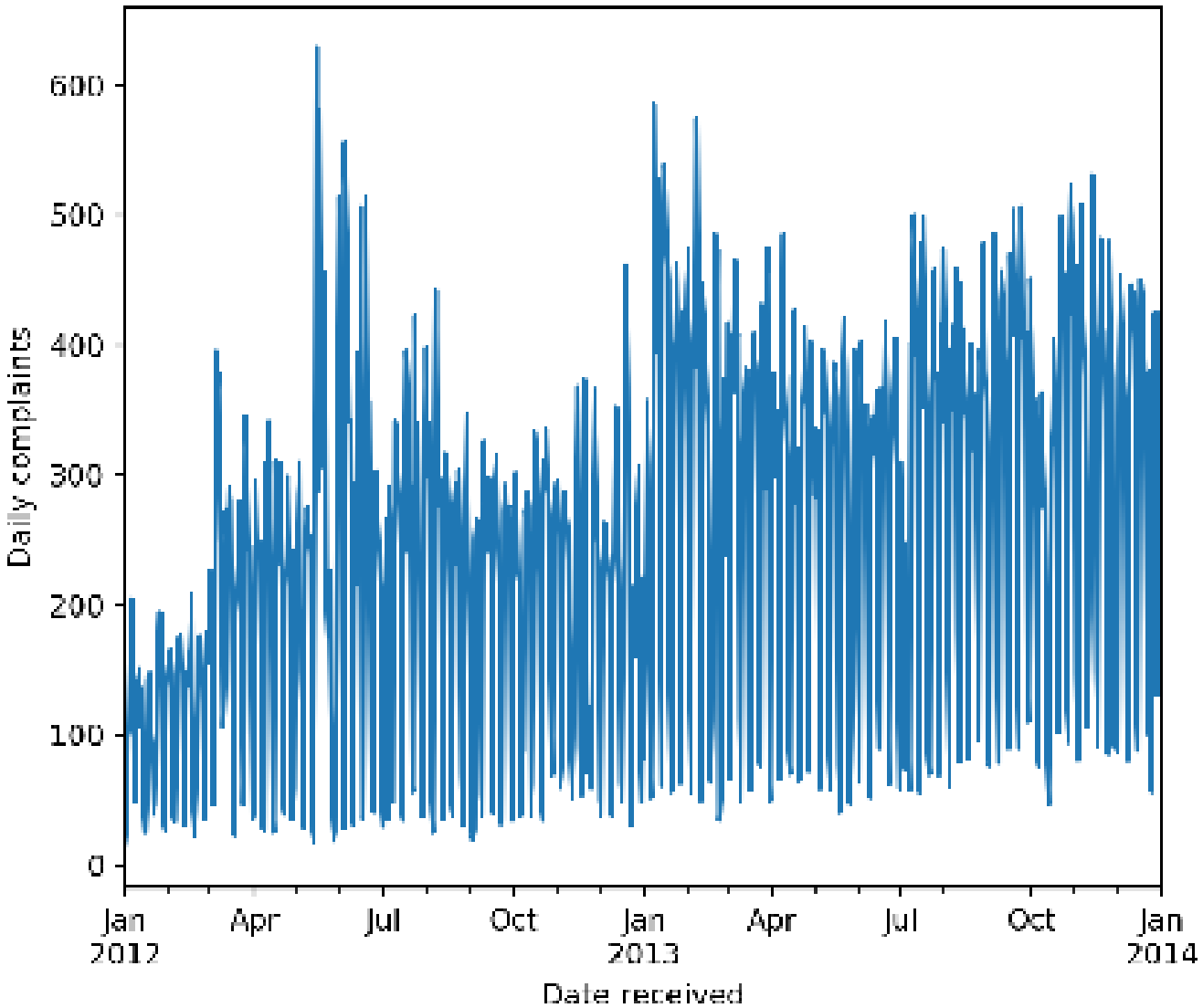
selecting a range for the time plot

Observe the conversion `to_datetime` in the previous code.

If we want to see the time plot for the size of the daily complaints in the years of 2012 and 2013, we need to declare two time stamps.

```
ts = data.groupby('Date received').size()

begintime = pd.Timestamp('2012-01-01 00:00:00')
endtime = pd.Timestamp('2014-01-01 00:00:00')

ts = ts[begintime: endtime]
ts.plot()
plt.ylabel('Daily complaints')
plt.show()
```

amount of daily complaints in 2012 and 2013



Summary and Exercises

With the `stats` modules of `sympy` and `scipy`
we can experiment with random variables.
To explore data, we use `pandas` and `matplotlib`.

Exercises:

1. Consider the histogram of the complaints plot on slide 23. Examine the `legend` settings of `matplotlib` to improve the plot so that long title no longer shows and the horizontal histogram plot looks nicer.
2. Make a bar plot to see which companies receive the most complaints.
3. Make a pie plot for the size of complaints for the five companies that receive the most complaints.