Exam 1 Review - Solutions

In [1]:
```
#4.1.1 What python package does dataframe manipulation?
import pandas as pd

#5.2.1. What three Python packages that have we used in this class that
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1. Formulating Research Questions and Collecting Data sample)

1.1.1 Three common types of wheat seeds are: kama, Canadian, and rosa. The **populations** kama seeds are very large, therefore we cannot possibly collect data about all kama seeds. Similarly, the populations of Canadian and rosa seeds are too large to collect. So what we do collect three **random samples**, each of size 70. One sample of kama seeds, one sample of Canadian seeds, and one sample of rosa seeds. For each **observation** we measure various **numerical variables**.

1.1.2 Can we answer questions about the population of kama seeds using the random sampl kama seeds?

Yes. Because this data is from a random sample of the population.

1.2.2 and 1.2.3 Using this dataset, can we determine if having the human-defined label of a r seed (as opposed to a kama seed) **causes** the seed to have a larger area?

No. This is because we did not **randomly assign** our sample to "seed class" values. We just simply observed the seed class labels and the area of the seeds.

2. Data Management

2.1 Read the 'seeds.csv' dataset into a dataframe called df.

4.2.1 Display the first 7 rows.

In [2]:
```
df=pd.read_csv('seeds.csv')
df.head(7)
```

Out[2]:

	area	perimeter	compactness	kernel_length	width	asymmetry_coefficient	kernel_
0	15.26	14.84	0.871	5.762	2.212	2.221	
1	14.88	14.57	0.8811	5.554	2.222	1.018	
2	14.29	14.09	0.905	5.291	2.227	2.699	
3	12.84	12.94	0.8955	5.224	2.279	2.259	
4	16.14	14.99	0.9024	5.658	2.562	1.255	
5	14.28	14.21	0.8951	5.286	2.212	2.462	
6	14.69	14.49	0.8799	5.562	2.259	2.586	

2.2 What do the rows of this dataframe represent?

-a seed in the samples

2.3 What does a column in this dataframe represent?

-a numerical properties of the seeds

3. Data Cleaning

3.1 Basic Ensuring Data is of the Correct "Data Type"

3.1.1 and 3.1.2 We define three new objects below. Determine what types they are.

In [3]:
```
a=df['seed_class'][0]
b=df['compactness'][1]
c=df['perimeter'][2]
```

In [4]:
```
print(a)
print(type(a))
print(b)
print(type(b))
print(c)
print(type(c))
```

```
kama
<class 'str'>
0.8811
<class 'str'>
14.09
<class 'numpy.float64'>
```

- Object a is a "string" object.
- Object b is currently also a "string" object. (Ideally we want this to b a float... see next pa
- Object c is a "float" object.

3.1.3 While compactness is supposed to be a numerical attribute, what do we get when we a compactness of the first two seeds in the dataset?

In [5]:
```
print('First seed compactness:', df['compactness'][0])
print('Second seed compactness:', df['compactness'][1])

df['compactness'][0]+df['compactness'][1]
```

```
First seed compactness: 0.871
Second seed compactness: 0.8811
```

Out[5]:
```
'0.8710.8811'
```

3.2.3 How can we quickly look at all possible values of of the compactness column?

In [6]:
```
df['compactness'].unique()
```

Out[6]:
```
array(['0.871', '0.8811', '0.905', '0.8955', '0.9024', '0.8951', '0.879
       '0.8911', '0.8747', '0.888', '0.8696', '0.8796', '0.8759',
       '0.8744', '0.8992', '0.9182', '0.9058', '0.9152', '0.8686',
       '0.8584', '0.8722', '0.8988', '0.8664', 'missing value', '0.8849
       '0.8641', '0.8564', '0.882', '0.8604', '0.8662', '0.8724',
       '0.8529', '0.8728', '0.8779', '0.9', '0.9079', '0.8822', '0.8944
       '0.8871', '0.8852', '0.9009', '0.8986', '0.8794', '0.8861',
       '0.8882', '0.8819', '0.8676', '0.8751', '0.8922', '0.8528',
       '0.8821', '0.8557', '0.8658', '0.8818', '0.9006', '0.8857',
       '0.8292', '0.8682', '0.884', '0.868', '0.8716', '0.8879', '0.872
       '0.8625', '0.8458', '0.8672', '0.8622', '0.8762', '0.9081',
       '0.8786', '0.8628', '0.8599', '0.875', '0.8892', '0.8977',
       '0.8894', '0.878', '0.887', '0.8969', '0.859', '0.8989', '0.9021
       '0.8746', '0.8984', '0.8906', '0.9066', '0.8452', '0.8648',
       '0.8815', '0.8687', '0.8627', '0.881', '0.8866', '0.8985',
       '0.8717', '0.8829', '0.8917', '0.9056', '0.88', '0.8752', '0.886
       '0.8921', '0.9025', '0.8859', '0.8854', '0.9077', '0.889',
       '0.9008', '0.8897', '0.8772', '0.8588', '0.9064', '0.8999',
       '0.8698', '0.8725', '0.8991', '0.9108', '0.8942', '0.8706',
       '0.8644', '0.899', '0.8785', '0.8527', '0.858', '0.885', '0.848'
       '0.8612', '0.862', '0.8652', '0.8274', '0.8167', '0.8225',
       '0.8491', '0.8107', '0.8496', '0.8249', '0.8222', '0.8266',
       '0.8282', '0.8252', '0.8596', '0.8081', '0.8082', '0.8262',
       '0.8425', '0.8502', '0.8416', '0.8558', '0.8792', '0.8462',
       '0.8442', '0.8291', '0.8259', '0.8189', '0.8455', '0.8419',
       '0.8275', '0.8099', '0.8255', '0.8229', '0.8472', '0.856',
       '0.8579', '0.8575', '0.8541', '0.8198', '0.8272', '0.8594',
       '0.829', '0.8562', '0.8795', '0.8256', '0.8629', '0.886', '0.848
       '0.8964', '0.8609', '0.8874', '0.8567', '0.8782', '0.8511',
       '0.8521', '0.8684'], dtype=object)
```

3.2 Basic Dealing with Missing Values

3.2.1. How do we read a csv into a dataframe and indicate, which types of values we would li
Python to convert into a “NaN” object?

In [7]:
```
df=pd.read_csv('seeds.csv', na_values=['missing value'])
```

3.2.2 Are there any "NaN" values in this dataframe?

In [8]:
```
df.isna().sum()
```

Out[8]:
```
area                    0
perimeter               0
compactness             1
kernel_length           0
width                   0
asymmetry_coefficient   0
kernel_groove_length    0
seed_class              0
perimeter_size          0
dtype: int64
```

Yes, df has one "NaN" value in the compactness column.

3.2.3 Overwrite the dataframe df to be the dataframe that has all rows with na values droppe

In [9]:
```
df=df.dropna()
```

In [10]:
```
df
```

Out[10]:

	area	perimeter	compactness	kernel_length	width	asymmetry_coefficient	kerne
0	15.26	14.84	0.8710	5.762	2.212	2.221	
1	14.88	14.57	0.8811	5.554	2.222	1.018	
2	14.29	14.09	0.9050	5.291	2.227	2.699	
3	12.84	12.94	0.8955	5.224	2.279	2.259	
4	16.14	14.99	0.9024	5.658	2.562	1.255	
...	...	...	...	...	...	...	
205	12.19	12.20	0.8782	5.127	2.981	2.621	
206	11.22	12.88	0.8511	5.140	2.795	4.225	
207	12.20	12.66	0.8882	5.226	2.222	8.215	
208	11.84	12.21	0.8521	5.175	2.826	2.598	
209	12.20	12.24	0.8684	5.242	2.974	5.627	

209 rows × 9 columns

3.2.4 What would most functions in Python do if we had not dropped this row with missing da

- Drop them and then run the function.

4. Dataframe Manipulation

4.2. Basic Dataframe Description

4.2.2 Show the shape of the dataframe (ie. # of rows and # of columns).

4.2.3 Display the columns of the dataframe.

4.2.4 Display the index of the dataframe.

In [11]:
```
df.shape
```

Out[11]:
```
(209, 9)
```

- 209 rows
- 8 columns

In [12]:
```
df.columns.values
```

Out[12]:
```
array(['area', 'perimeter', 'compactness', 'kernel_length', 'width',
       'asymmetry_coefficient', 'kernel_groove_length', 'seed_class',
       'perimeter_size'], dtype=object)
```

In [13]:
```
df.index
```

Out[13]:
```
Int64Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
            ...
            200, 201, 202, 203, 204, 205, 206, 207, 208, 209],
           dtype='int64', length=209)
```

4.3 Creating a dataframe

- 4.3.1 Create a dataframe (called df2) that contains the following information.
  - a column called 'name', containing the names 'joe', 'nick', 'kevin'
  - a column called 'age', containing the ages 31, 28, 32

In [14]:
```
df2=pd.DataFrame({'name': ['joe', 'nick', 'kevin'], 'age':[31,28,32]})
df2.head()
```

Out[14]:

	name	age
0	joe	31
1	nick	28
2	kevin	32

4.4 How to isolate a column from a dataframe.

4.4.1 Display just the name column of the df2 dataframe.

In [15]:
```
df2['name']
```

Out[15]:
```
0      joe
1     nick
2    kevin
Name: name, dtype: object
```

4.4.2 Display just the name of the column of the df2 dataframe (using the .iloc function)

In [16]:
```
df2.iloc[:,0]
```

Out[16]:
```
0      joe
1     nick
2    kevin
Name: name, dtype: object
```

4.5 How to isolate ranges of values in a dataframe

- 4.5.1 Display the third row in the df2 dataframe.

In [17]:
```
df2.iloc[2,:]
```

Out[17]:
```
name    kevin
age        32
Name: 2, dtype: object
```

- 4.5.2-4.5.7 Display the first and third row and the first two columns of the df2 dataframe.

In [18]:
```
df2.iloc[[0,2],0:2]
```

Out[18]:

	name	age
0	joe	31
2	kevin	32

4.6 Filtering a Dataframe

4.6.2 Create a new dataframe called df3 that is just comprised of people in df2 that are in the 30's.

In [19]:
```
df3=df2[df2['age']>=30]
df3
```

Out[19]:

	name	age
0	joe	31
2	kevin	32

4.6.3 Create a new dataframe called df4 that is just comprised of people named 'joe'.

In [20]:
```
df4=df2[df2['name']=='joe']
df4
```

Out[20]:

	name	age
0	joe	31

4.7 What are two ways to vertically stack two dataframes?

- Create a new dataframe called df5 that is df3 vertically stacked on top of df4.

In [21]:
```
df5=pd.concat([df3,df4])
df5
```

Out[21]:

	name	age
0	joe	31
2	kevin	32
0	joe	31

- Create a new dataframe called df5 that is df3 vertically stacked on top of df4, and have tl index go from 0 to 2.

In [22]:
```
df5=pd.concat([df3,df4], ignore_index=True)
df5
```

Out[22]:

	name	age
0	joe	31
1	kevin	32
2	joe	31

4.8 How do you merge two dataframes that share simil values in a given column?

Below we create a new dataframe called df_order. Notice that they both have a column callec 'name' with some overlapping values.

Create a new dataframe called df_final that is what you get when you merge the df5 datafram the df_order dataframe on the 'name' column.

Before your run this code, think about what you would expect this dataframe to contain.

In [23]:
```
df_order=pd.DataFrame({'name': ['joe', 'nick', 'kevin'], 'order':['midd
```

In [24]:
```
df_final=pd.merge(df_order, df5, on='name')
df_final
```

Out[24]:

	name	order	age
0	joe	middle	31
1	joe	middle	31
2	kevin	oldest	32

4.9 How do you sort a dataframe by values in a given column.

- 4.9.1 Sort the df2 dataframe by age in ascending order.
- 4.9.2 Sort the df2 dataframe by age in descending order.

In [25]:
```
df2.sort_values(by='age')
```

Out[25]:

	name	age
1	nick	28
0	joe	31
2	kevin	32

In [26]:
```
df2.sort_values(by='age', ascending=False)
```

Out[26]:

	name	age
2	kevin	32
0	joe	31
1	nick	28

5.3 Describing a Single Categorical Variable

5.3.1 Count how many of each seed_class there is in the df dataframe.

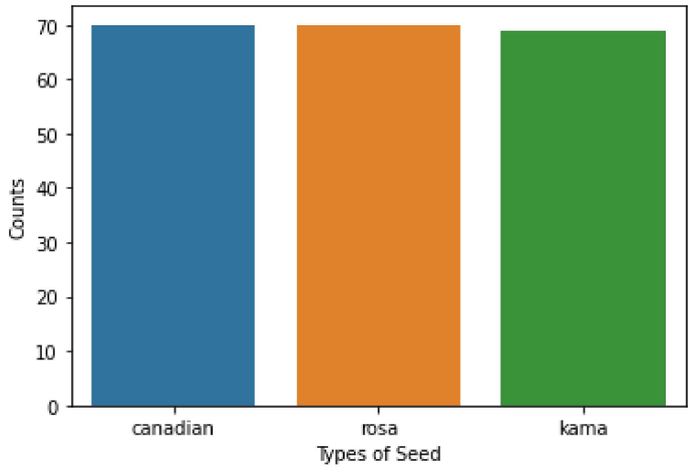
In [27]:
```
df['seed_class'].value_counts()
```

Out[27]:
```
canadian    70
rosa        70
kama        69
Name: seed_class, dtype: int64
```

5.3.2 Visualize the number of each type of seed_class.

In [28]:
```
seed_counts=df['seed_class'].value_counts()
sns.barplot(seed_counts.index, seed_counts)
plt.xlabel('Types of Seed')
plt.ylabel('Counts')
plt.show()
```

```
C:\Users\vellison\Miniconda3\lib\site-packages\seaborn\_decorators.py:3
utureWarning: Pass the following variables as keyword args: x, y. From
ion 0.12, the only valid positional argument will be `data`, and passin
her arguments without an explicit keyword will result in an error or mi
erpretation.
  warnings.warn(
```



5.4 Determining if there is an association between two categorical variables.

5.4.1 How many of the following are there:

- (large perimeter Canadian seeds)
- (small perimeter Canadian seeds)
- (large perimeter kama seeds)
- (small perimeter kama seeds)
- (large perimeter rosa seeds)
- (small perimeter rosa seeds)

In [29]:
```
pd.crosstab(df['seed_class'], df['perimeter_size'])
```

Out[29]:

perimeter_size	large	small
seed_class		
canadian	0	70
kama	51	18
rosa	70	0

5.4.2

- What percent of Canandian seeds have a large perimeter size, a small perimeter size?

- What percent of kama seeds have a large perimeter size, a small perimeter size?
- What percent of rosa seeds have a large perimeter size, a small perimeter size?

In [30]:
```
pd.crosstab(df['seed_class'], df['perimeter_size'], normalize='index')
```

Out[30]:

perimeter_size	large	small
seed_class		
canadian	0.00000	1.00000
kama	0.73913	0.26087
rosa	1.00000	0.00000

5.4.2

- What percent of large_perimeter seeds are: canadian, kama, and rosa?
- What percent of small_perimeter seeds are: canadian, kama, and rosa?

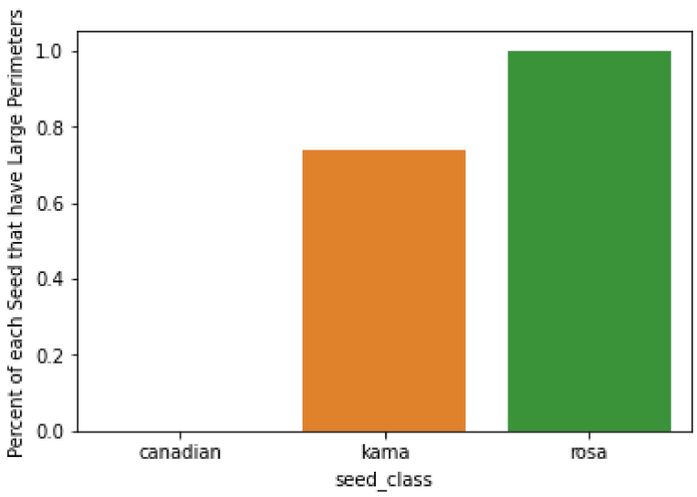
In [31]:
```
pd.crosstab(df['perimeter_size'], df['seed_class'], normalize='index')
```

Out[31]:

seed_class	canadian	kama	rosa
perimeter_size			
large	0.000000	0.421488	0.578512
small	0.795455	0.204545	0.000000

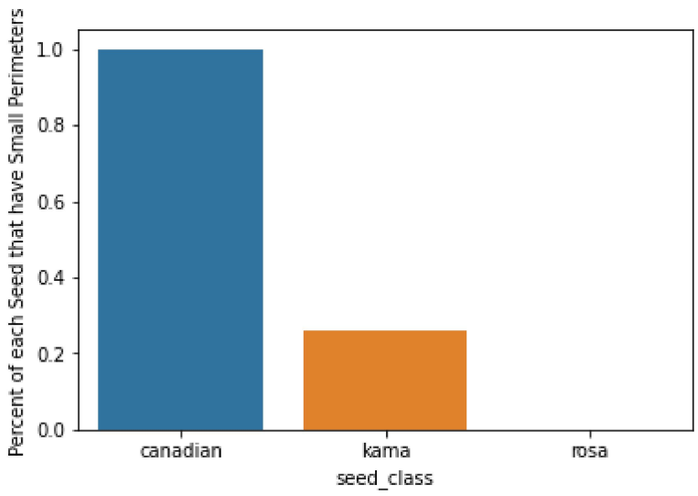
5.4.3 Visualize these distributions from 5.4.1 and 5.4.2

In [32]:
```
seed_size_df=pd.crosstab(df['seed_class'], df['perimeter_size'], normal
sns.barplot(x=seed_size_df.index, y="large", data=seed_size_df)
plt.ylabel('Percent of each Seed that have Large Perimeters')
plt.show()
```



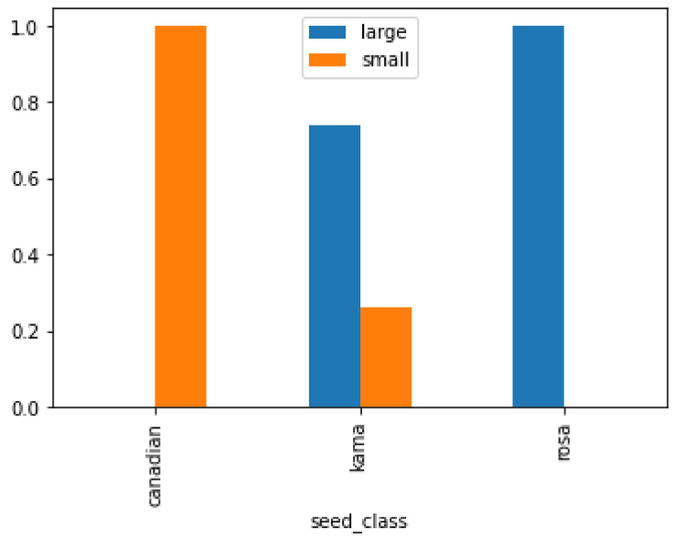
Because the orange (or equivalently) blue bars are of different heights, there is an associatio
between seed type and perimeter size in this dataset.

In [33]:
```
sns.barplot(x=seed_size_df.index, y="small", data=seed_size_df)
plt.ylabel('Percent of each Seed that have Small Perimeters')
plt.show()
```



Because the bars are of different heights, there is an association between seed type and peri size in this dataset.

In [34]:
```
seed_size_df.plot.bar()
plt.legend()
plt.show()
```



Because the orange (or equivalently) blue bars are of different heights, there is an associatio between seed type and perimeter size in this dataset.

5.5 Describing the Distribution of a Single Numerical Variable in a Dataset

5.5.1 Summary Statistics

5.5.1.1 Measures of center

- 5.5.1.1.1.Calculate the mean and the median of the seed area in df.

In [35]:
```
df['area'].mean()
```

Out[35]: 14.7433014354067

In [36]:
```
df['area'].median()
```

Out[36]: 14.28

5.5.1.1.3

- Do you think this distribution is likely to be symmetric, left-skewed, or right-skewed?

Because the mean is larger than the median, this distribution is likely to be right-skewed.

5.5.1.2 Measures of spread

- 5.5.1.2.Calculate the standard deviation, variance, IQR, Q1, Q3, and range of the area c in df.

In [37]:
```
df['area'].std()
```

Out[37]: 2.9737833824633624

In [38]:
```
df['area'].var()
```

Out[38]: 8.843387605815238

In [39]:
```
#Range
df['area'].max()-df['area'].min()
```

Out[39]: 10.59

In [40]:
```
#Q1
df['area'].quantile(.25)
```

Out[40]: 12.19

In [41]:
```
#Q3
df['area'].quantile(.75)
```

Out[41]: 17.26

In [42]:
```
#IQR=Q3-Q1
df['area'].quantile(.75)-df['area'].quantile(.25)
```

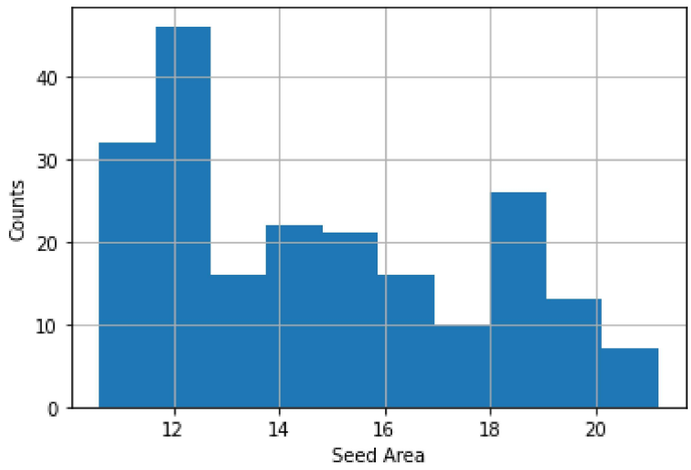
Out[42]: 5.070000000000002

5.5.2 Visualizations

5.5.2.1 Histograms

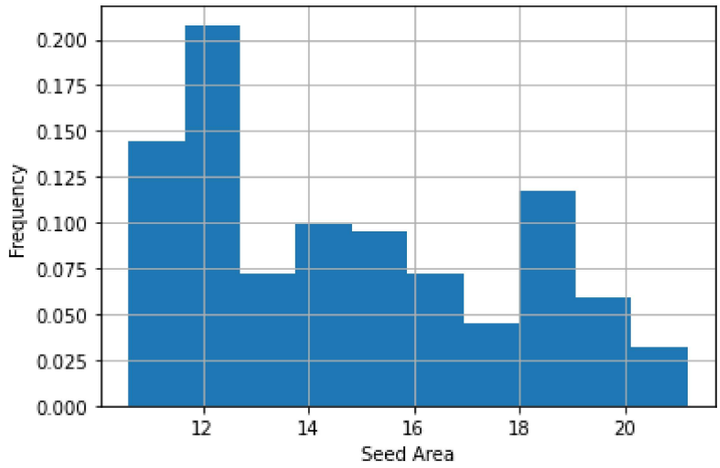
5.5.2.1.1.1 Create a histogram of seed areas (using counts on the y-axis.

In [43]:
```
df['area'].hist()
plt.ylabel('Counts')
plt.xlabel('Seed Area')
plt.show()
```



5.5.2.1.1.2 Create a histogram of seed areas (using frequency on the y-axis.)

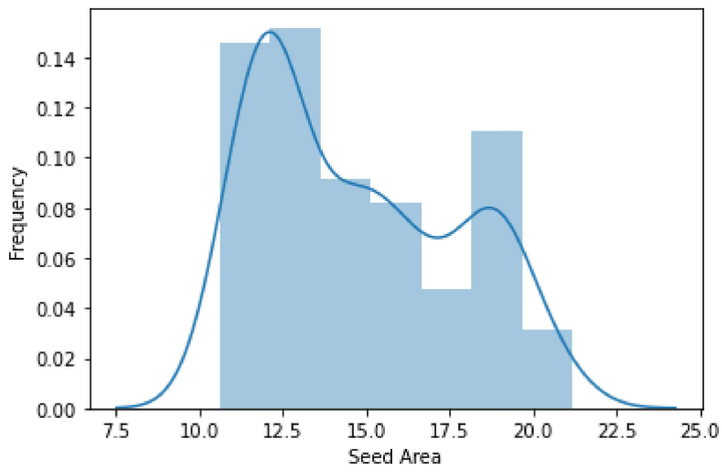
In [44]:
```
df['area'].hist(density=True)
plt.ylabel('Frequency')
plt.xlabel('Seed Area')
plt.show()
```



5.5.2.1.1.3 Create a histogram of seed areas (using frequency on the y-axis.) Overlay it with a density curve.

In [45]:
```
sns.distplot(df['area'])
plt.ylabel('Frequency')
plt.xlabel('Seed Area')
plt.show()
```

```
C:\Users\vellison\Miniconda3\lib\site-packages\seaborn\distributions.py
7: FutureWarning: `distplot` is a deprecated function and will be remov
n a future version. Please adapt your code to use either `displot` (a f
e-level function with similar flexibility) or `histplot` (an axes-level
ction for histograms).
  warnings.warn(msg, FutureWarning)
```



5.5.2.1.2.3 What percent of seed areas do we expect to be above 18?

- The box widths look to be about 1.25 wide.
- 18 and above looks like it starts at the second to last box.
- Total area of last two boxes are about = (1.25)(0.11) + (1.25)(.03) = 0.175

In [46]:
```
1.25*.11+1.25*.03
```

Out[46]:
```
0.17500000000000002
```

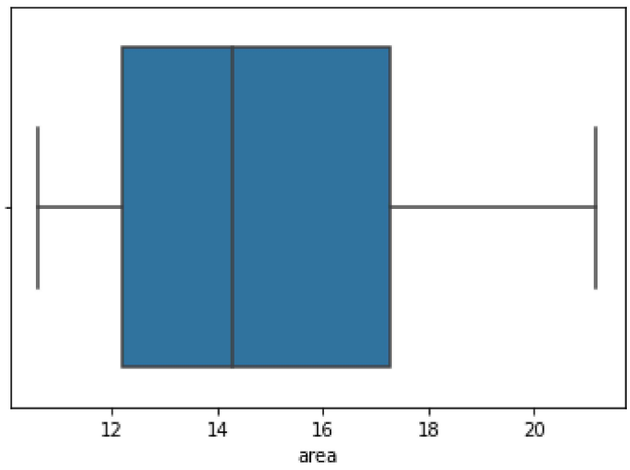
5.5.2.2 Boxplot

Create a boxplot of seed areas.

- Are there any outliers?
- Use this to estimate:
  - the median
  - the Q1
  - the Q3
  - the IQR
  - the range

In [47]:
```
sns.boxplot(df['area'])
plt.show()
```

```
C:\Users\vellison\Miniconda3\lib\site-packages\seaborn\_decorators.py:3
utureWarning: Pass the following variable as a keyword arg: x. From ver
0.12, the only valid positional argument will be `data`, and passing ot
arguments without an explicit keyword will result in an error or misint
etation.
  warnings.warn(
```



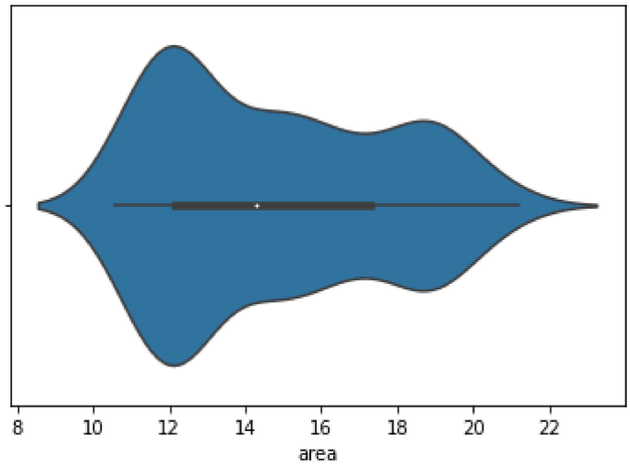
- There are no outliers (as there are no points above the whiskers).
- The median is about 14.1
- The Q1 is about 12.1
- The Q3 is about 17.5.
- Thus the IQR=Q3-Q1=17.5-12.1=5.4 (approximately).
- The range = max-min (is about 21-13=8).

5.5.2.3 Violin Plot

Create a violin plot of seed areas.

In [48]:
```
sns.violinplot(df['area'])
plt.show()
```

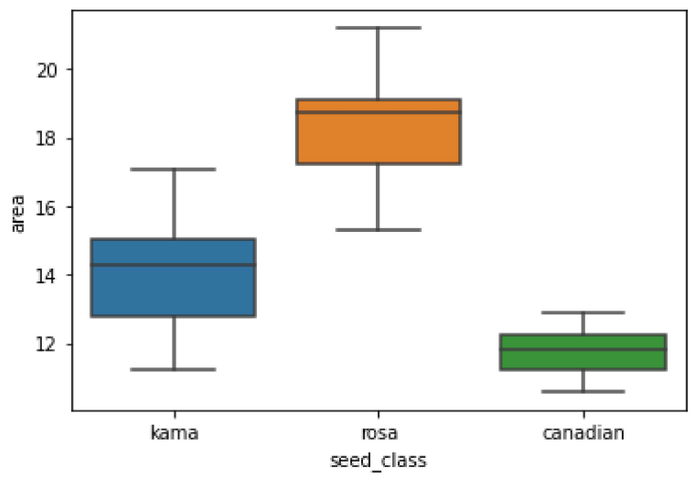
```
C:\Users\vellison\Miniconda3\lib\site-packages\seaborn\_decorators.py:3
utureWarning: Pass the following variable as a keyword arg: x. From ver
0.12, the only valid positional argument will be `data`, and passing ot
arguments without an explicit keyword will result in an error or misint
etation.
  warnings.warn(
```



5.6 Is there an association between seed class and area?

In [49]:
```
sns.boxplot(df['seed_class'],df['area'])
plt.show()
```

```
C:\Users\vellison\Miniconda3\lib\site-packages\seaborn\_decorators.py:3
utureWarning: Pass the following variables as keyword args: x, y. From
ion 0.12, the only valid positional argument will be `data`, and passin
her arguments without an explicit keyword will result in an error or mi
erpretation.
  warnings.warn(
```



Yes. The median areas for the three seed classes are different and furthermore their IQRs ha
overlap.

6 Probability Theory

6.1 Collect a random sample of 7 seeds from df with replacement.

In [50]:
```
df_sample1=df.sample(7, replace=True)
df_sample1
```

Out[50]:

	area	perimeter	compactness	kernel_length	width	asymmetry_coefficient	kerne
197	12.27	12.78	0.8849	5.220	2.128	4.670	
115	19.06	16.45	0.8854	6.416	2.719	2.248	
131	18.94	16.22	0.8942	6.144	2.825	2.908	
47	14.99	14.56	0.8882	5.570	2.277	2.958	
35	16.12	15.00	0.9000	5.709	2.485	2.270	
70	17.62	15.98	0.8672	6.191	2.561	4.076	
106	18.85	16.17	0.9056	6.152	2.806	2.842	

6.1 Collect a random sample of 8 seeds from df without replacement.

In [51]:
```
df_sample2=df.sample(8)
df_sample2
```

Out[51]:

	area	perimeter	compactness	kernel_length	width	asymmetry_coefficient	kerne
51	15.78	14.91	0.8922	5.674	2.424	5.592	
133	16.16	15.22	0.8644	5.845	2.295	4.266	
97	18.98	16.57	0.8687	6.449	2.552	2.144	
153	11.18	12.04	0.8266	5.220	2.692	2.222	
28	14.11	14.18	0.8820	5.541	2.221	2.754	
6	14.69	14.49	0.8799	5.562	2.259	2.586	
108	19.94	16.92	0.8752	6.675	2.762	2.252	
168	11.25	12.12	0.8291	5.176	2.668	4.227	

6.1.2- 6.4 (see separate pdf)

6.5 Normal Random Variables

Suppose the annual income of teachers in a local county is normally distributed with a mean $\$64,000$ and a standard deviation of $\$5,000$.

Define X to be the income of a randomly selected teacher from the county.

6.5.2-6.5.5 Use Python to calculate the mean, variance, and standard deviation of X.

In [52]:
```
from scipy.stats import norm
print('Mean:',norm.mean(loc=64000, scale=5000))
print('Variance:',norm.var(loc=64000, scale=5000))
print('Standard Deviation:',norm.std(loc=64000, scale=5000))
```

```
Mean: 64000.0
Variance: 25000000.0
Standard Deviation: 5000.0
```

6.2.6 What percent of teachers in the county make less than $\$70,000$?

$P(X < 70000) = .8849.$

In [53]:
```
norm.cdf(70000, loc=64000, scale=5000)
```

Out[53]: 0.8849303297782918

What is the probability that a randomly selected teacher makes more than $\$60,000$?

$P(X > 60000) = 1 - P(X \geq 60000) = 1 - .212 = 0.788.$

In [54]:
```
1-norm.cdf(60000, loc=64000, scale=5000)
```

Out[54]: 0.7881446014166034

What is the probability that a randomly selected teacher makes between $\$60,000$ and $\$70,000$?

$P(60000 < X < 70000) = P(X < 70000) - P(X < 60000) = 0.8849 - .212 = .673$

In [55]:
```
norm.cdf(70000, loc=64000, scale=5000)-norm.cdf(60000, loc=64000, scale
```

Out[55]: 0.6730749311948951

What is the income of the top 15% of teachers in the county that make the most?

What value of x will give us $P(X \geq x) = 0.15$?

This is equivalent to asking what value of x will give us $P(X < x) = 0.85$.

Answer: The teachers that are in the top 15% of earners make at least $69182.

In [56]:
```
norm.ppf(.85, loc=64000,scale=5000)
```

Out[56]: 69182.16694746896

Generate 10 random variable values for X.

In [93]:
```
norm.rvs(size=10, loc=64000,scale=5000)
```

Out[93]:
```
array([62626.10393815, 70712.64585693, 73408.12707494, 57219.87429245,
       64317.95817713, 67710.03618146, 63585.44154128, 64915.50516717,
       67820.89313178, 64358.66721816])
```

6.6 Geometric Random Variables

Suppose someone that uses an online dating app gets matched with 5% of the people they "s
right" on (aka people they would like to match with). Once this person gets a match, they stop
looking at new people and go on a date.

a.) What would we expect the average number of times for this person to "swipe right" until they stop?

If we let X=number of swipes until stopping, we can say $X \sim Geom(p = 0.05)$. X fits the definition of being a geometric random variable with $p = 0.05$, because:

1. we can assume that the outcome of any person matching with the user to be independer

2. we can assume that the probabilty of the "success outcome" aka "getting a match" is alw going to be p=0.05
3. geometric random variables = the number of "failure" trials until stopping at a "success" t (where all trials are independent and the probability of success is always the same (p).

Therefore we can calculate $E[X]$ using geometric random variable functions from python as follows.

$$E[X] = 20$$

So on average this online dating app user needs to swipe through 20 people before getting a match and stopping.

In [94]:
```
from scipy.stats import geom
geom.mean(p=0.05)
```

Out[94]: 20.0

b.) What is the probability that this user stops after exactly 7 swipes?

$$P(X = 7) = 0.037.$$

In [95]:
```
geom.pmf(7, p=0.05)
```

Out[95]: 0.03675459453124999

c.) What is the probability that this user stops after *at most* 7 swipes?

```
$P(X\leq 7) = 0.302$.
```

In [96]:
```
geom.cdf(7, p=0.05)
```

Out[96]: 0.30166270390625

d.) What is the probability that this user stops after at least 10 swipes?

$$P(X \geq 10) = 1 - P(X < 10) = 1 - P(X \leq 9) = 0.63.$$

In [97]:
```
1-geom.cdf(9, p=0.05)
```

Out[97]: 0.6302494097246093

7 Inference Basics

7.1 Sampling Distribution of Sample Means

Assume that the seeds dataframe is the pouplation. Create a sampling distribution of sample seed areas. Specifically, let your samples be of size 100. Finally, your sampling distribution sh contain 300 sample means.

1. Find the mean of this sampling distribution.
2. Find the standard deviation of this sampling distribution.
3. What is the shape of this sampling distribution.

In [57]:
```
def MCmeans(df, x='', replace=True, n=1, M=1):
    #INPUT:
    # df is a data frame
    # x is a text-valued name for a variable in the data frame
    # replace = True or False depending on whether
    #    draws are with or without replacement
    # n = number of draws per sample
    # M = number of samples to draw
    MCstats = []
    for i in range(M):
        #1. Collect a random sample of size n=10 with replacement
        #2. Take the mean of this random sample
        #3. Append this random sample mean to the SampleMeans list (whi
        MCstats.append(df[x].sample(n, replace=replace).mean())
    #4. returns the sampling distribution in a dataframe format
    return pd.DataFrame({x: MCstats})
```

In [70]:
```
sampling_dist_means=MCmeans(df, x='area', n=100, M=300)
sampling_dist_means
```

Out[70]:

	area
0	14.8334
1	14.6222
2	14.5513
3	14.6054
4	14.6057
...	...
295	15.1935
296	15.1779
297	14.2875
298	15.0448
299	14.6002

300 rows × 1 columns

In [71]:
```
print('Mean of Sampling Distribution Means')
sampling_dist_means['area'].mean()
```

```
Mean of Sampling Distribution Means
```

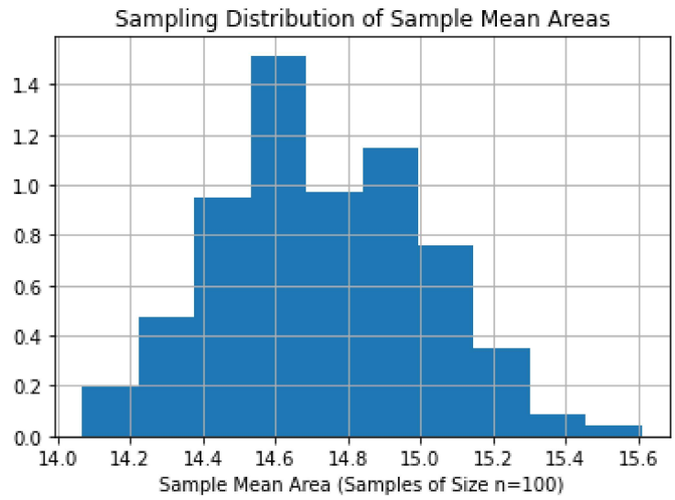
Out[71]:
```
14.728842000000002
```

In [72]:
```
print('Standard Deviation of Sampling Distribution Means')
sampling_dist_means['area'].std()
```

```
Standard Deviation of Sampling Distribution Means
```

Out[72]:
```
0.28377505193904906
```

In [73]:
```
sampling_dist_means.hist(density=True)
plt.title('Sampling Distribution of Sample Mean Areas')
plt.xlabel('Sample Mean Area (Samples of Size n=100)')
plt.show()
```



This sampling distribution is mostly symmetric and unimodal.

7.2 Sampling Distribution of Sample Proportions

Assume that the seeds dataframe is the pouplation. Create a sampling distribution of sample proportion of perimeters that are 'large'. Specifically, let your samples be of size 20. Finally, y sampling distribution should contain 400 sample proportions.

1. Find the mean of this sampling distribution.
2. Find the standard deviation of this sampling distribution.
3. What is the shape of this sampling distribution.

In [88]:
```
sampling_dist_props_list=[]
for i in range(0,400):

    #1. Collect random sample of size n=20 with replacement.
    samp=df.sample(20, replace=True)

    #2. Calculate proportion of seeds in the sample with 'large' perime
    prop_large=samp[samp['perimeter_size']=='large'].shape[0]/df.shape[

    #3. Add this sample proportion to the list.
    sampling_dist_props_list.append(prop_large)

#4. Put this list into a dataframe.
sampling_dist_props=pd.DataFrame({'sample_prop_large_perimeter':samplin
sampling_dist_props
```

Out[88]:

	sample_prop_large_perimeter
0	0.062201
1	0.047847
2	0.043062
3	0.052632
4	0.043062
...	...
395	0.057416
396	0.033493
397	0.066986
398	0.057416
399	0.052632

400 rows × 1 columns

In [89]:
```
print('Mean of Sampling Distribution Proportions')
sampling_dist_props['sample_prop_large_perimeter'].mean()
```

```
Mean of Sampling Distribution Proportions
```

Out[89]:
```
0.05503588516746412
```

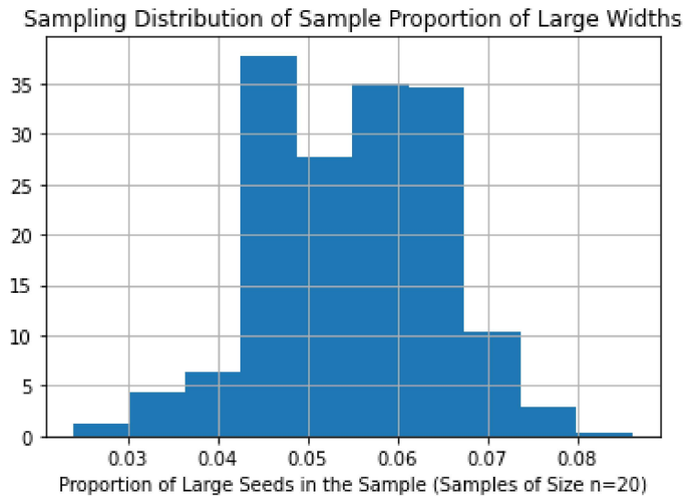
In [90]:
```
print('Standard Deviation of Sampling Distribution Proportions')
sampling_dist_props['sample_prop_large_perimeter'].std()
```

```
Standard Deviation of Sampling Distribution Proportions
```

Out[90]:
```
0.009945705193323603
```

In [91]:

```
sampling_dist_props.hist(density=True)
plt.title('Sampling Distribution of Sample Proportion of Large Widths')
plt.xlabel('Proportion of Large Seeds in the Sample (Samples of Size n=
plt.show()
```



This sampling distribution of sample proportions is relatively symmetric and unimodal.

8 Additional Coding

8.1 For Loops

What is the sum of the following 0^2 + 1^2 + 2^2 + ... + 20^2?

In [59]:

```
total_sum=0
for i in range(0,20+1):
    print('Current value of i',i)
    total_sum+=i**2
    print('Current value of total_sum',total_sum)
    print('---------------------')
print(total_sum)
```

```
Current value of i 0
Current value of total_sum 0
---------------------
Current value of i 1
Current value of total_sum 1
---------------------
Current value of i 2
Current value of total_sum 5
---------------------
Current value of i 3
Current value of total_sum 14
---------------------
Current value of i 4
Current value of total_sum 30
---------------------
Current value of i 5
Current value of total_sum 55
---------------------
Current value of i 6
Current value of total_sum 91
---------------------
Current value of i 7
Current value of total_sum 140
---------------------
Current value of i 8
Current value of total_sum 204
---------------------
Current value of i 9
Current value of total_sum 285
---------------------
Current value of i 10
Current value of total_sum 385
---------------------
Current value of i 11
Current value of total_sum 506
---------------------
Current value of i 12
Current value of total_sum 650
---------------------
Current value of i 13
Current value of total_sum 819
---------------------
Current value of i 14
Current value of total_sum 1015
---------------------
Current value of i 15
Current value of total_sum 1240
---------------------
Current value of i 16
Current value of total_sum 1496
---------------------
Current value of i 17
Current value of total_sum 1785
---------------------
Current value of i 18
Current value of total_sum 2109
---------------------
Current value of i 19
Current value of total_sum 2470
---------------------
Current value of i 20
Current value of total_sum 2870
---------------------
2870
```

Make a list of the following values [0^2, 1^2, 2^2, ... , 20^2]

In [60]:

```
total_list=[]
for i in range(0,20+1):
    print('Current value of i',i)
    total_list.append(i**2)
    print('Current List',total_list)
    print('--------------------')
print(total_list)
```

```
Current value of i 0
Current List [0]
--------------------
Current value of i 1
Current List [0, 1]
--------------------
Current value of i 2
Current List [0, 1, 4]
--------------------
Current value of i 3
Current List [0, 1, 4, 9]
--------------------
Current value of i 4
Current List [0, 1, 4, 9, 16]
--------------------
Current value of i 5
Current List [0, 1, 4, 9, 16, 25]
--------------------
Current value of i 6
Current List [0, 1, 4, 9, 16, 25, 36]
--------------------
Current value of i 7
Current List [0, 1, 4, 9, 16, 25, 36, 49]
--------------------
Current value of i 8
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64]
--------------------
Current value of i 9
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
--------------------
Current value of i 10
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
--------------------
Current value of i 11
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121]
--------------------
Current value of i 12
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144]
--------------------
Current value of i 13
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169]
--------------------
Current value of i 14
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 1
--------------------
Current value of i 15
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 1
225]
--------------------
Current value of i 16
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 1
225, 256]
--------------------
Current value of i 17
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 1
225, 256, 289]
--------------------
Current value of i 18
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 1
225, 256, 289, 324]
--------------------
Current value of i 19
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 1
225, 256, 289, 324, 361]
```

```
--------------------
Current value of i 20
Current List [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 1
225, 256, 289, 324, 361, 400]
--------------------
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256,
9, 324, 361, 400]
```

8.2 Functions

Write a function that does the following:

- Returns a value that triples the input, but
- returns a string that says "no negative numbers" if the input is a number less 0.
- Then use the function with the following inputs:
  - 4
  - -4

In [61]:
```
def my_triple_func(banana):
    if banana<0:
        return "no negative numbers"
    else:
        return banana*3
```

In [62]:
```
my_triple_func(4)
```

Out[62]:
```
12
```

In [63]:
```
my_triple_func(-4)
```

Out[63]:
```
'no negative numbers'
```

Type *Markdown* and LaTeX: α^2