Imports

1. First, import all the necessary modules using the import function. For this exercise, we will be mainly using `pandas`. To learn more about these packages, you can read through the documentation:
    - https://pandas.pydata.org/
    - https://numpy.org/
    - https://seaborn.pydata.org/
    - https://matplotlib.org/

```
# Let's import the relevant Python packages here
# Feel free to import any other packages for this project

#Data Wrangling
import numpy as np
import pandas as pd
from pandas.core.algorithms import is_numeric_dtype

#Plotting
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

Data

In the following exercise, we will familiarize ourselves with Python by studying the College dataset, which can be found in the file `College.csv`. This dataset contains the following variables from 777 different universities and colleges in the US:

Column	Description
Private	Public/private indicator
Apps	Number of applications received
Accept	Number of applicants accepted
Enroll	Number of new students enrolled
Top10perc	New students from top 10% of high school class
Top25perc	New students from top 25% of high school class
F.Undergrad	Number of full-time undergraduates
P.Undergrad	Number of part-time undergraduates
Outstate	Out-of-state tuition
Room.Board	Room and board costs

✓ 1s completed at 6:17 PM

Personal	Estimated personal spending
PhD	Percent of faculty with Ph.D.'s
Terminal	Percent of faculty with terminal degree
S.F.Ratio	Student/faculty ratio
Perc.alumni	Percent of alumni who donate
Expend	Instructional expenditure per student
Grad.Rate	Graduation rate

2. Load the College dataset using `pandas`

```
###code here
df = pd.read_csv('sample_data/College.csv')
```

3. Use the `head()` function to view the data.

```
###code here

df.head()
```

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	C
0	Yes	1660	1232	721	23	52	2885	537	
1	Yes	2186	1924	512	16	29	2683	1227	
2	Yes	1428	1097	336	22	50	1036	99	
3	Yes	417	349	137	60	89	510	63	
4	Yes	193	146	55	16	44	249	869	

4. Notice that there is a column 'Names' of each university's name. As we don't want to use these names as predictors, they are natural candidates to index our data.

```
###code here
```

```
df.set_index('Names', inplace = True)
```

5. Use the `head()` function again. You should now see that the indices have been replaced with the name of each university in the data set. This means that Python has given each row a name corresponding to the appropriate university. Python will not try to perform calculations on the row names.

```
###code here

df.head()
```

Names	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Unde
Abilene Christian University	Yes	1660	1232	721	23	52	2885	
Adelphi University	Yes	2186	1924	512	16	29	2683	
Adrian College	Yes	1428	1097	336	22	50	1036	
Agnes Scott College	Yes	417	349	137	60	89	510	
Alaska Pacific University	Yes	193	146	55	16	44	249	

6. Use the `info()` function to check and produce a numerical summary of your variables.

```
###code here

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 777 entries, Abilene Christian University to York College of Pennsylvania
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Private      777 non-null    object
 1   Apps         777 non-null    int64
 2   Accept       777 non-null    int64
 3   Enroll       777 non-null    int64
```

```
 4   Top10perc    777 non-null    int64
 5   Top25perc    777 non-null    int64
 6   F.Undergrad  777 non-null    int64
 7   P.Undergrad  777 non-null    int64
 8   Outstate     777 non-null    int64
 9   Room.Board   777 non-null    int64
 10  Books        777 non-null    int64
 11  Personal     777 non-null    int64
 12  PhD          777 non-null    int64
 13  Terminal     777 non-null    int64
 14  S.F.Ratio    777 non-null    float64
 15  perc.alumni  777 non-null    int64
 16  Expend       777 non-null    int64
 17  Grad.Rate    777 non-null    int64
dtypes: float64(1), int64(16), object(1)
memory usage: 115.3+ KB
```

7. Examine if there are any duplicates and drop them if needed.

```
###code here

df.duplicated().sum()

#df.drop_duplicates(keep = 'last') #use this if there are any duplications
```

```
0
```

8. Replace any missing values in the `Apps' column with 0. This dataframe will henceforth be our original dataframe.

```
###code here

df.isnull().sum()

df.Apps.fillna(0, inplace = True)


#Generalize the fill missing value with mean procedure (Mean imputation)

def fillNaWithMean(df):
  for column in df.columns:
    if is_numeric_dtype(df[column]):
        average = df[column].mean()
        df[column].fillna(average, inplace = True)

fillNaWithMean(df)
```

9. Find the college with the least out-of-state tuition and name this variable `college_least_tuition` . The variable should return the name of a college, not its tuition. **Q:Which college in this dataset has the least amount of out-of-state tuition?**

```
###code here
df.Outstate.idxmin() #method 1
```

```
'Brigham Young University at Provo'
```

```
df[df.Outstate == df.Outstate.min()].index #method 2
```

```
Index(['Brigham Young University at Provo'], dtype='object', name='Names')
```

10. From the original dataframe, select the 'PhD' column and name this dataframe `phd_column`. Find the length of this column and name this variable `phd_column_length`.

```
# Note: Double bracket for dataframe

phd_column = df[['PhD']]
phd_column_length = len(phd_column)
```

```
phd_column_length
```

```
777
```

11. From the original dataframe, select both the 'Private' and 'Top10perc' columns, and slice them such that only the rows with index 15 and 16 remain. Name this dataframe `private_top10`. From this dataframe, find the length of the filtered 'Private' column and name this variable `private_column_length`.

```
# Double bracket to obtain multiple columns
private_top10 = df[['Private', 'Top10perc']].iloc[15:17]
#private_top10
```

```
private_column_length = len(private_top10)
private_column_length
```

```
2
```

12. From the original dataframe, select the row that only contains data about the "Amherst College". Name this dataframe `Amherst`.

```
###code here

Amherst = df[df.index == 'Amherst College']
```

```
Amherst
```

Names	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Under
Amherst College	Yes	4302	992	418	83	96	1593	

13. From the original dataframe, select the rows that contain all colleges with the name "Penn" included. The "P" should be capitalized. Name this dataframe `many_penns`. Comment on your observations. **Q: How many universities are there in this dataset that include "Penn"?**

```
###code here
many_penns = df[df.index.str.contains('Penn')]
```

Exploratory Data Analysis (EDA)

we will perform exploratory data analysis (EDA) to extract insights

First, we return to the College dataset. This dataset contains the following variables from 777 different universities and colleges in the US:

Column	Description
Private	Public/private indicator
Apps	Number of applications received
Accept	Number of applicants accepted
Enroll	Number of new students enrolled
Top10perc	New students from top 10% of high school class
Top25perc	New students from top 25% of high school class
F.Undergrad	Number of full-time undergraduates
P.Undergrad	Number of part-time undergraduates
Outstate	Out-of-state tuition
Room.Board	Room and board costs
Books	Estimated book costs
Personal	Estimated personal spending
PhD	Percent of faculty with Ph.D.'s
Terminal	Percent of faculty with terminal degree
S.F.Ratio	Student/faculty ratio

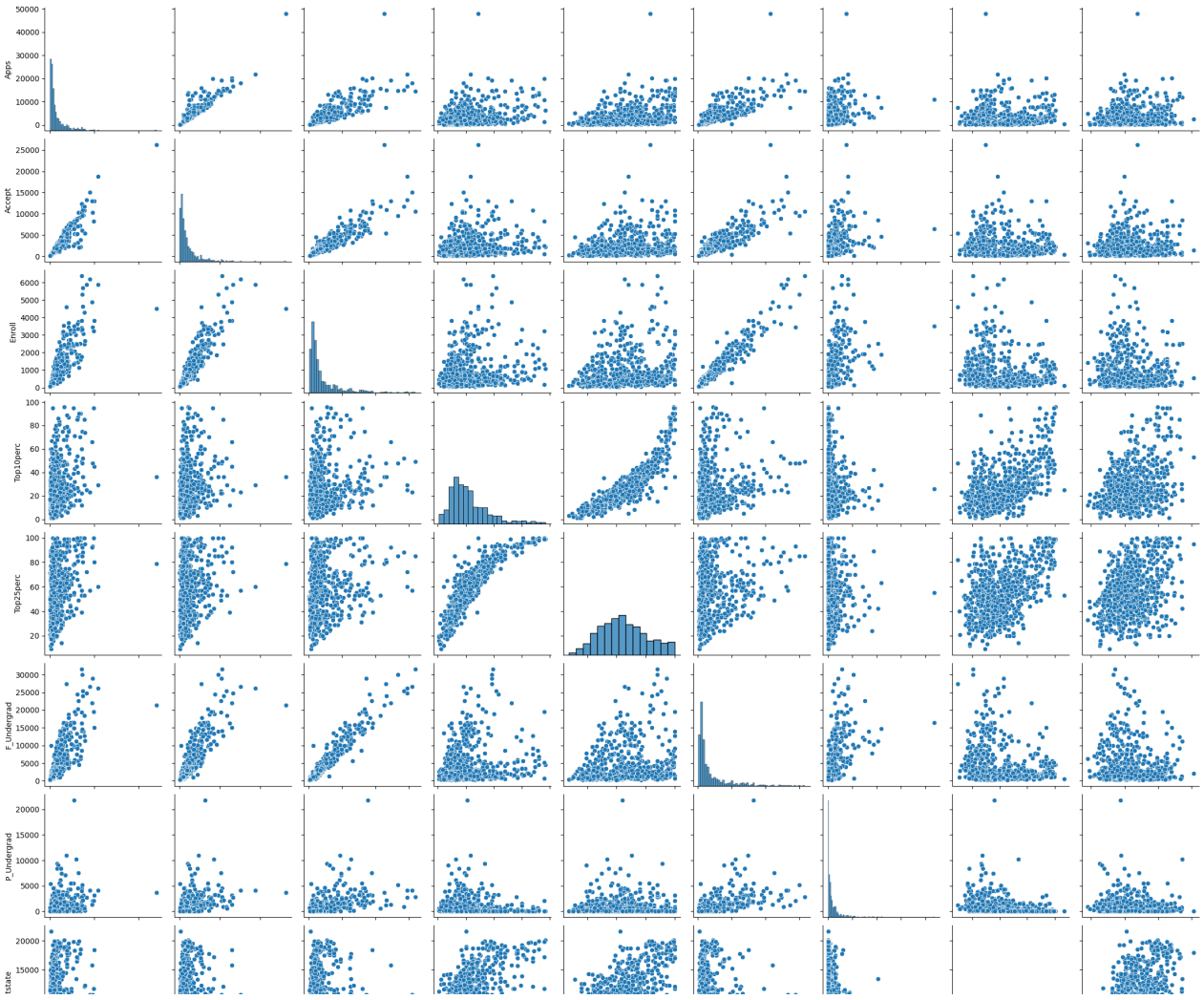
Perc.alumni	Percent of alumni who donate
Expend	Instructional expenditure per student
Grad.Rate	Graduation rate

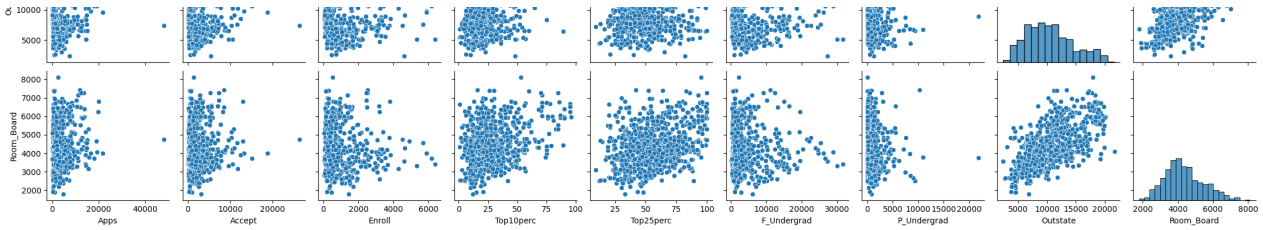
First, Let's rename some columns.

```
df = df.rename(columns = {'Grad.Rate':'Grad_Rate',
                          'S.F.Ratio': 'S_F_Ratio',
                          'perc.alumni':'perc_alumni',
                          'Room.Board':'Room_Board',
                          'F.Undergrad':'F_Undergrad',
                          'P.Undergrad':'P_Undergrad'})
```

1. Use the `seaborn pairplot()` function on the original dataframe with all universities to produce a scatterplot matrix of the first ten columns of the data. Remember to use the `%matplotlib` inline command for plotting in Jupyter. Comment on your observations.

```
###code here

sns.pairplot(df.iloc[:, :10])
plt.show()
```

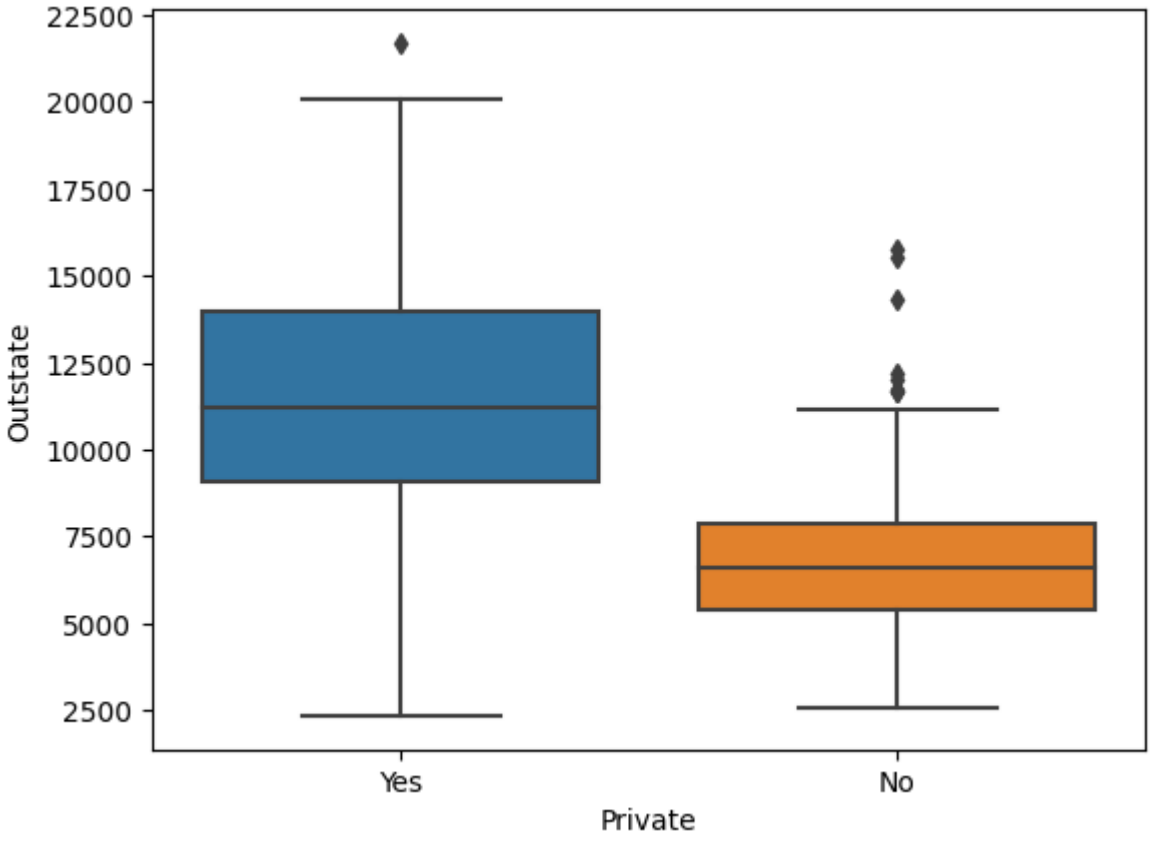




2. Use the `seaborn boxplot()` function on the original dataframe with all universities to produce side-by-side box plots of 'Outstate' versus 'Private'. **Q: On average, are private universities more expensive than public universities?**

```
###code here
sns.boxplot(x = 'Private', y = 'Outstate', data = df)

##On average, private universities are more expensive than public universities.
```

```
<Axes: xlabel='Private', ylabel='Outstate'>
```



3. Create a new qualitative variable named `large_university` by binning the 'Enroll' column. We are going to divide universities into two groups based on whether the number of new students enrolled exceeds the average (mean) of all new students enrolled.

```
###code here

avg_enroll = df.Enroll.mean()
large_university = df.Enroll.map(lambda x: True if x>avg_enroll else False)


large_university
```

```
Names
Abilene Christian University      False
Adelphi University                False
Adrian College                    False
Agnes Scott College               False
Alaska Pacific University         False
                                  ...
Worcester State College           False
Xavier University                 False
Xavier University of Louisiana    False
```

```
Xavier University of Louisiana    False
Yale University                    True
York College of Pennsylvania      False
Name: Enroll, Length: 777, dtype: bool
```

4. Create a dataframe called `large_universities` that only includes large universities.

```
###code here
large_universities = df[large_university == True]


small_universities = df[large_university == False]
```

5. Use the `pandas describe(include = 'all')` function on large_universities to produce a numerical summary of each column. Within this dataframe, what is the 75^{th} percentile for the column 'Enroll'?

```
###code here

df.describe(include = 'all')
```

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F_
count	777	777.000000	777.000000	777.000000	777.000000	777.000000	
unique	2	NaN	NaN	NaN	NaN	NaN	
top	Yes	NaN	NaN	NaN	NaN	NaN	
freq	565	NaN	NaN	NaN	NaN	NaN	
mean	NaN	3001.638353	2018.804376	779.972973	27.558559	55.796654	3
std	NaN	3870.201484	2451.113971	929.176190	17.640364	19.804778	4
min	NaN	81.000000	72.000000	35.000000	1.000000	9.000000	
25%	NaN	776.000000	604.000000	242.000000	15.000000	41.000000	
50%	NaN	1558.000000	1110.000000	434.000000	23.000000	54.000000	1
75%	NaN	3624.000000	2424.000000	902.000000	35.000000	69.000000	4
max	NaN	48094.000000	26330.000000	6392.000000	96.000000	100.000000	31

```
df.Enroll.describe()
```

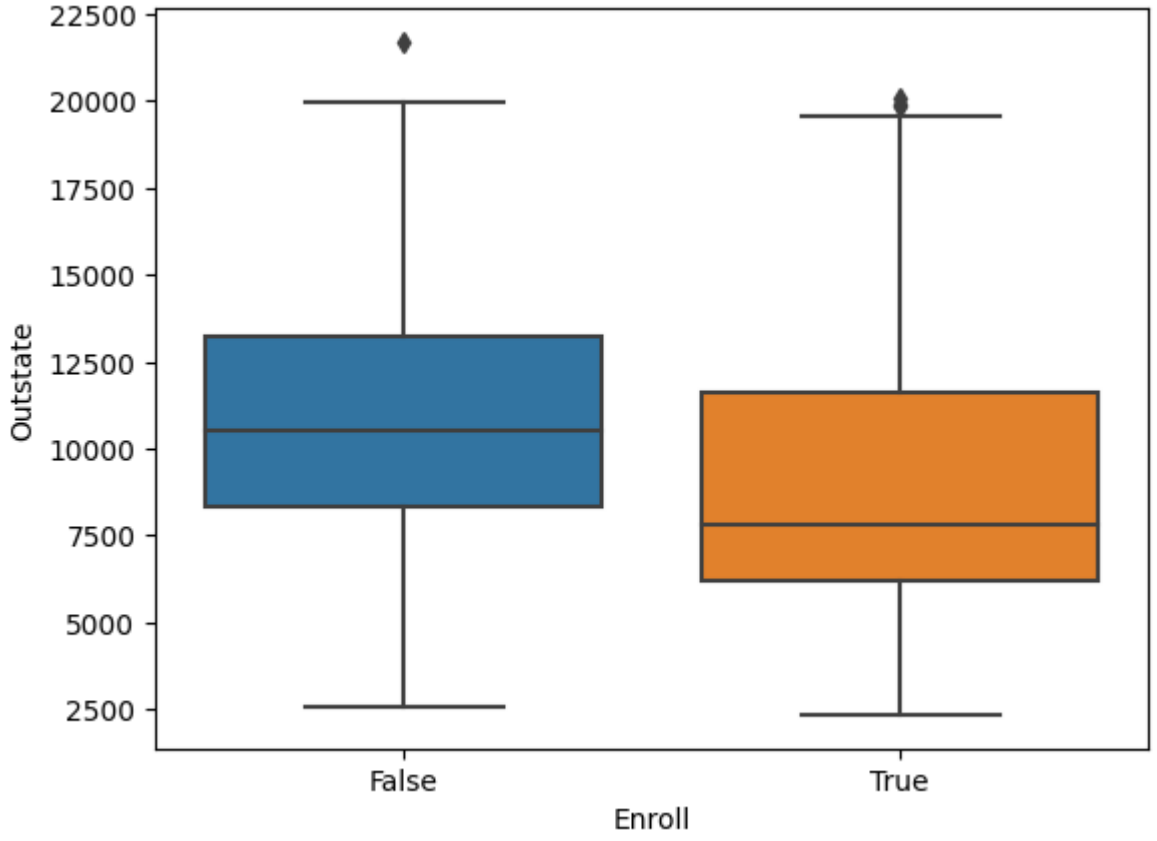
```
count     777.000000
mean      779.972973
std       929.176190
```

```
std        929.176190
min         35.000000
25%        242.000000
50%        434.000000
75%        902.000000
max       6392.000000
Name: Enroll, dtype: float64
```

6. Use the `seaborn boxplot()` function on the large_university variable to determine whether larger universities are more expensive. **Q: are large universities more expensive than small universities?**

```
###code here

sns.boxplot(x = large_university, y = df.Outstate)
```

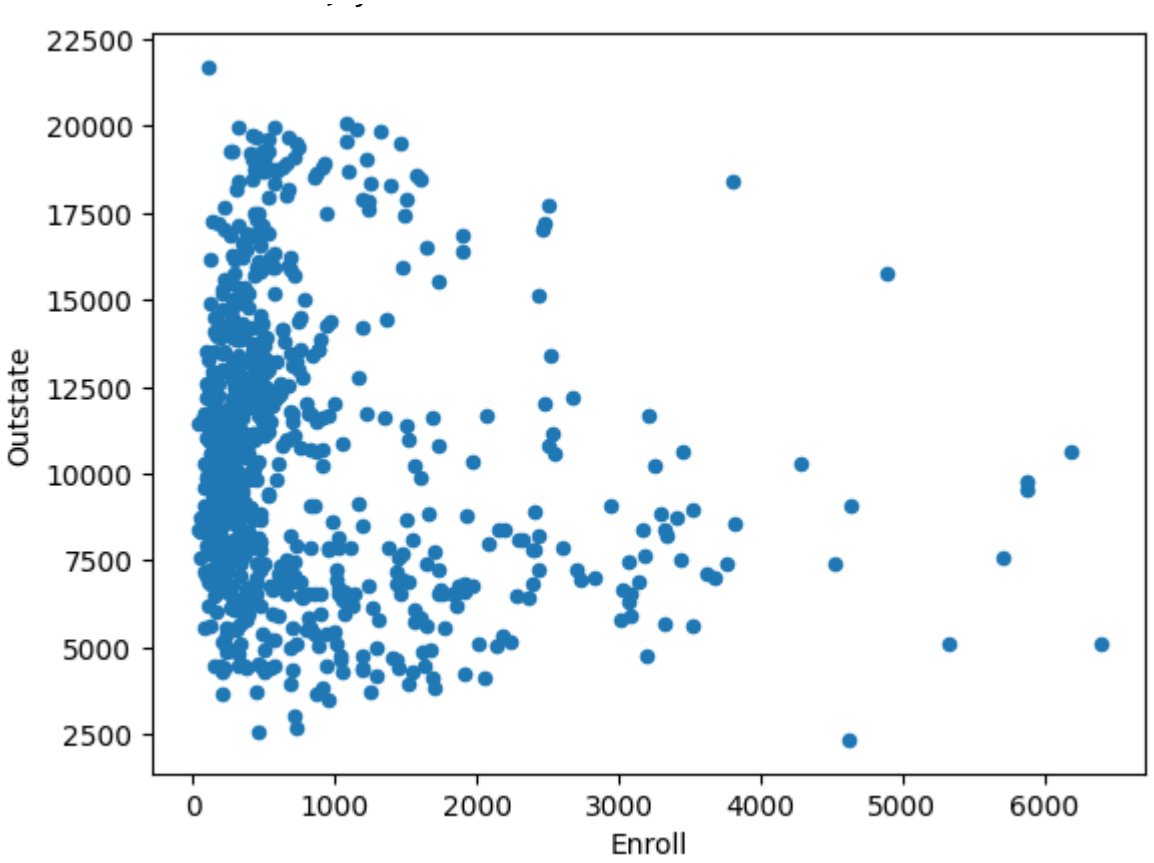
```
<Axes: xlabel='Enroll', ylabel='Outstate'>
```



7. To further examine the relationship between the size of a university against its out-of-state tuition, plot a scatter plot. **Q: What type of relationship does this show (weak, medium, strong)?**

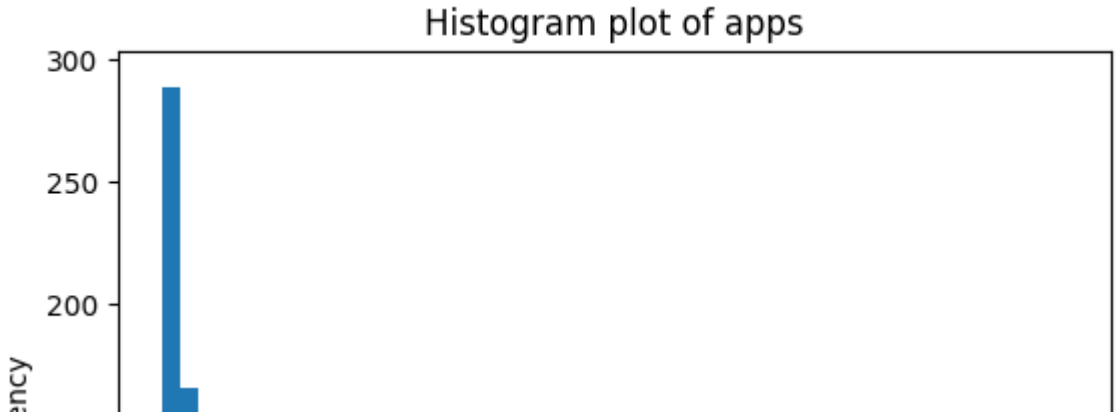
```
###code here

df.plot.scatter('Enroll', 'Outstate')
```

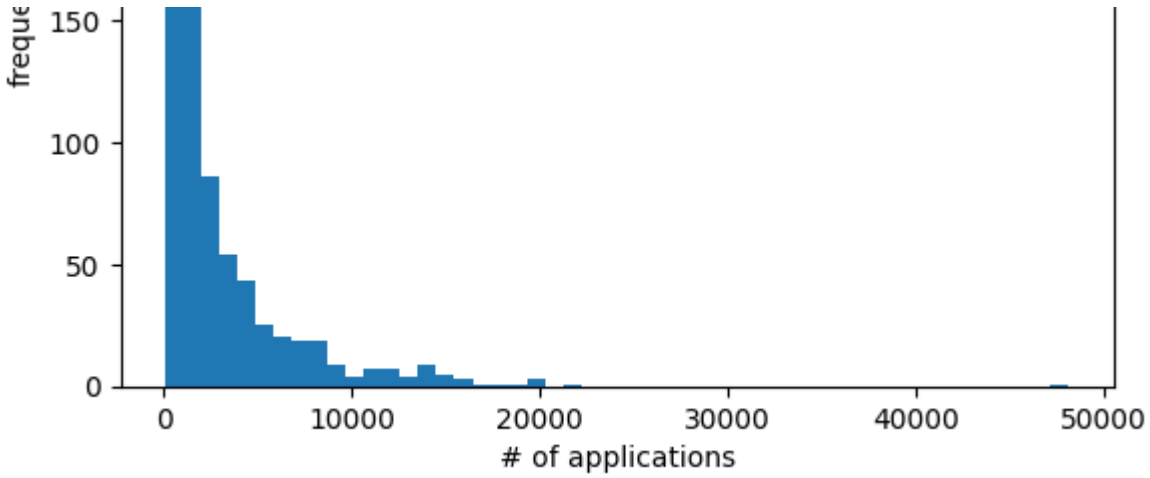
```
<Axes: xlabel='Enroll', ylabel='Outstate'>
```



8. Use `matplotlib.pyplot.hist()` or `seaborn.distplot()` to produce some histograms with different numbers of bins for a few of the quantitative variables. You may find the `matplotlib.pyplot.subplot()` function useful to plot multiple graphs under a single codeblock. In particular, create a histogram to examine the number of applications received with a bin size of 200. **Q: Are there more universities that receive 0 to 10,000 applications or 10,000 to 20,000 applications?**

```
###code here

plt.hist(df.Apps, bins = 50)
plt.xlabel('# of applications')
plt.ylabel('frequency')
plt.title('Histogram plot of apps')
plt.show()
```





9. We are interested in the acceptance rate of these universities. To calculate the acceptance rate, we take the 'Accept' column and divide by the 'Apps' column. Add this column to a copied version of the original dataframe. Remember to copy the dataframe first before performing feature transformation, as we do not wish to alter or inadvertently create new columns in our original data. **Name your new column `acceptance_rate`. Q: Is there a positive or negative relationship between the university's acceptance rate and the percentage of students in the university who graduated from the top 10% from high school?**

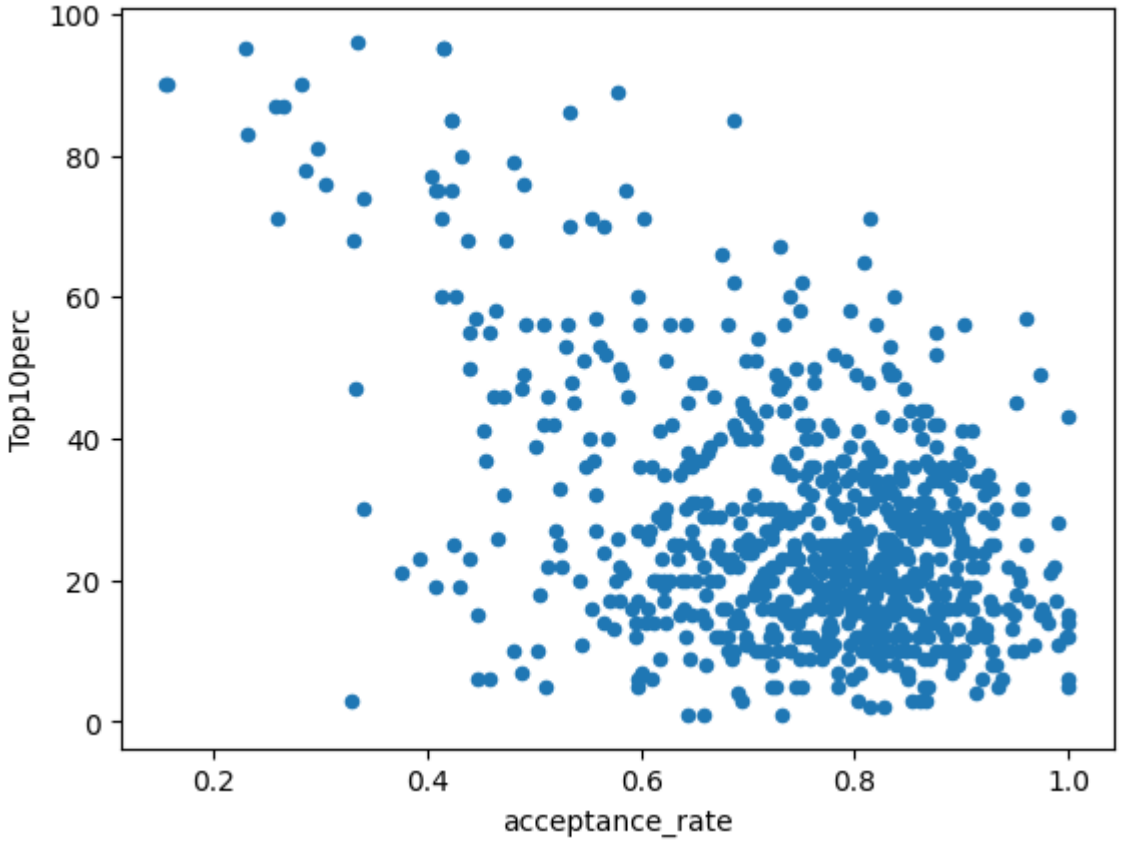
```
###code here

df['acceptance_rate'] = df.Accept/df.Apps


df.acceptance_rate.describe()
```

```
count    777.000000
mean       0.746928
std        0.147104
min        0.154486
25%        0.675647
50%        0.778750
75%        0.848522
max        1.000000
Name: acceptance_rate, dtype: float64
```

You may plot the university's acceptance rate vs the percentage of students in the university who graduated from the top 10% from high school.

```
###Code here
df.plot.scatter('acceptance_rate', 'Top10perc')

##There is a negative relationship between acceptance rate and top 10 percent high schoc
```

```
<Axes: xlabel='acceptance_rate', ylabel='Top10perc'>
```



10. Continue exploring the data and provide a brief summary of what you discover.

10.1. We will create a new qualitative variable called Elite. This categorical variable is defined by subsetting the Top10perc variable. We are going to divide universities into two groups based on whether or not the proportion of students coming from the top 10% of their high school classes exceeds 50%. **Q: Are Elite schools tuition higher than that of non-Elite schools for out-of-state students?**

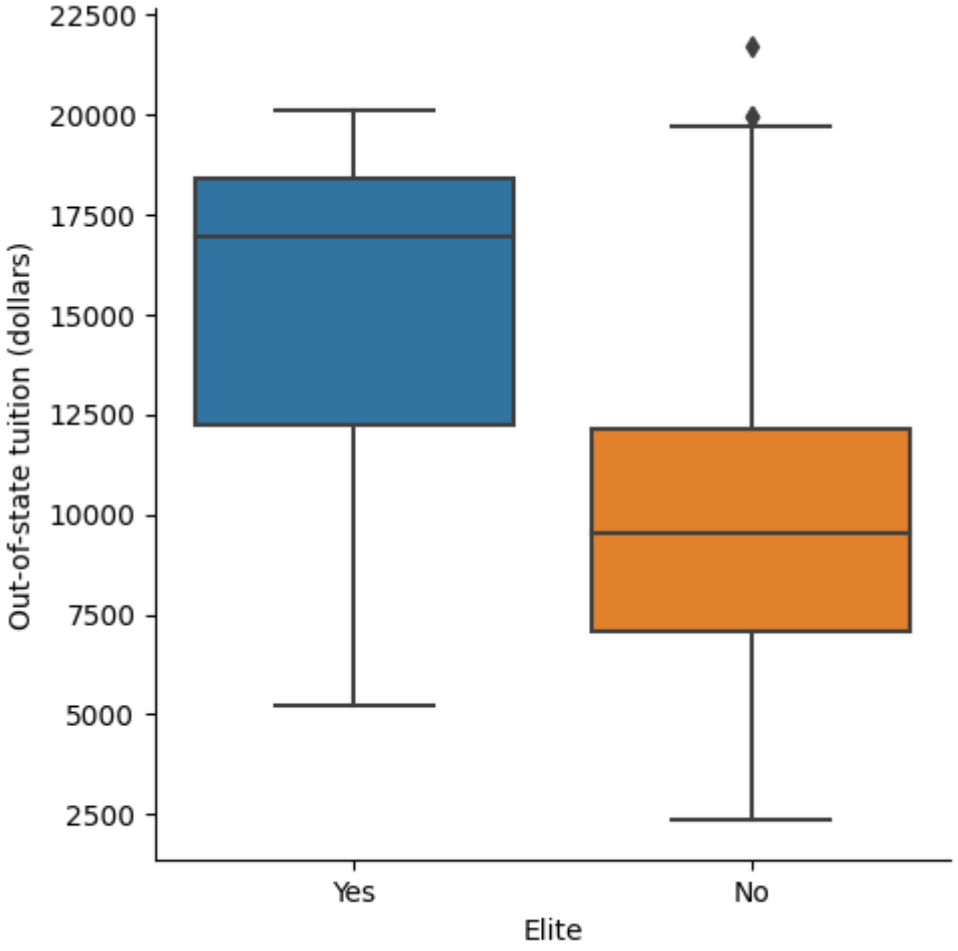
```
###Code here
# Create a new column called Elite and set the default value as "No"
df["Elite"] = "No"
# Select all rows (i.e. schools) with over 50% of their students coming from the top 10%
# Set the value of the Elite column for those schools to "Yes"
df.loc[df["Top10perc"] > 50, "Elite"] = "Yes"
#Use group.by() function to count the number of Elite schools.
df["Elite"].groupby(by = df["Elite"]).count()
```

```
Elite
No     699
Yes     78
Name: Elite, dtype: int64
```

Visualizing the tuition of Elite schools verson non-Elite for out-of-state students.

```
tuition = sns.catplot(x = "Elite", y = "Outstate", kind = "box", order = ["Yes", "No"])
```

```
tuition = sns.catplot(x = "Elite", y = "Outstate", kind = "box", order = ["Yes", "No"],
tuition.set(ylabel = "Out-of-state tuition (dollars)")
plt.show()
```



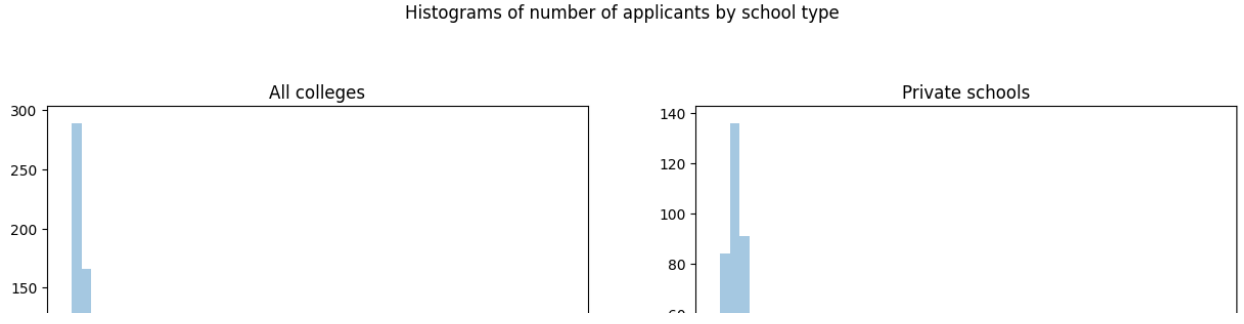
Elite schools have higher tuition for out-of-state students than non-Elite schools.

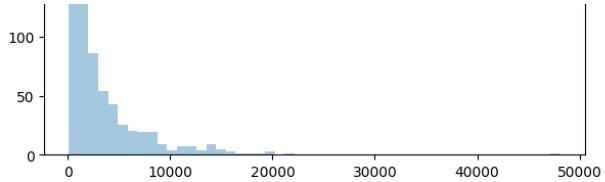
10.2. **Q: Does the school type, public, private, or elite affect the number of applications?**

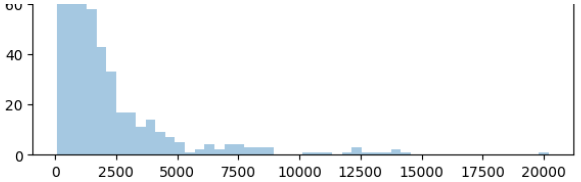
```
# Set the figure size so the plots aren't all squished together
fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize = (15, 10))

# Create histogram for number of applicants across all colleges
sns.distplot(df["Apps"], kde = False, ax = axes[0, 0])
axes[0, 0].set(xlabel = "", title = "All colleges")

# Create histogram for number of applicants at private colleges
sns.distplot(df.loc[df["Private"] == "Yes", "Apps"], kde = False, ax = axes[0, 1])
axes[0, 1].set(xlabel = "", title = "Private schools")

# Create histogram for number of applicants at elite colleges
sns.distplot(df.loc[df["Elite"] == "Yes", "Apps"], kde = False, ax = axes[1, 0])
axes[1, 0].set(xlabel = "", title = "Elite schools")

# Create histogram for number of applicants at public colleges
sns.distplot(df.loc[df["Private"] == "No", "Apps"], kde = False, ax = axes[1, 1])
```

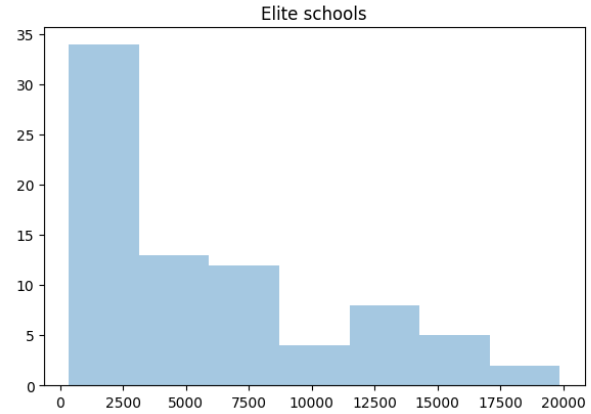
```
axes[1, 1].set(xlabel = "", title = "Public schools")

fig.suptitle("Histograms of number of applicants by school type")
```

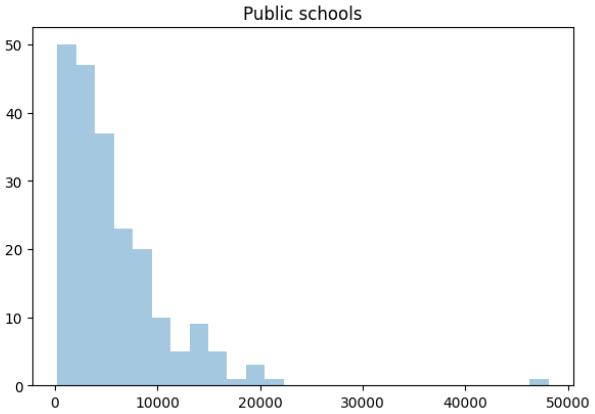
```
<ipython-input-135-ddf464b8a6eb>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Apps"], kde = False, ax = axes[0, 0])
<ipython-input-135-ddf464b8a6eb>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.loc[df["Private"] == "Yes", "Apps"], kde = False, ax = axes[0, 1
<ipython-input-135-ddf464b8a6eb>:13: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.loc[df["Elite"] == "Yes", "Apps"], kde = False, ax = axes[1, 0])
<ipython-input-135-ddf464b8a6eb>:17: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.loc[df["Private"] == "No", "Apps"], kde = False, ax = axes[1, 1]
Text(0.5, 0.98, 'Histograms of number of applicants by school type')
```












```
# Generate numerical summary of applicants by public vs private school
df["Apps"].groupby(by = df["Private"]).describe()
```

Private	count	mean	std	min	25%	50%	75%	max
No	212.0	5729.919811	5370.675335	233.0	2190.75	4307.0	7722.5	48094.0
Yes	565.0	1977.929204	2443.341319	81.0	619.00	1133.0	2186.0	20192.0

```
# Generate numerical summary of applicants by elite vs non-elite school
```

```
df["Apps"].groupby(by = df["Elite"]).describe()
```

Elite	count	mean	std	min	25%	50%	75%	max
No	699.0	2669.226037	3572.632737	81.0	695.50	1420.0	3187.50	48094.0
Yes	78.0	5980.564103	5025.659837	346.0	2175.75	3849.0	8595.25	19873.0

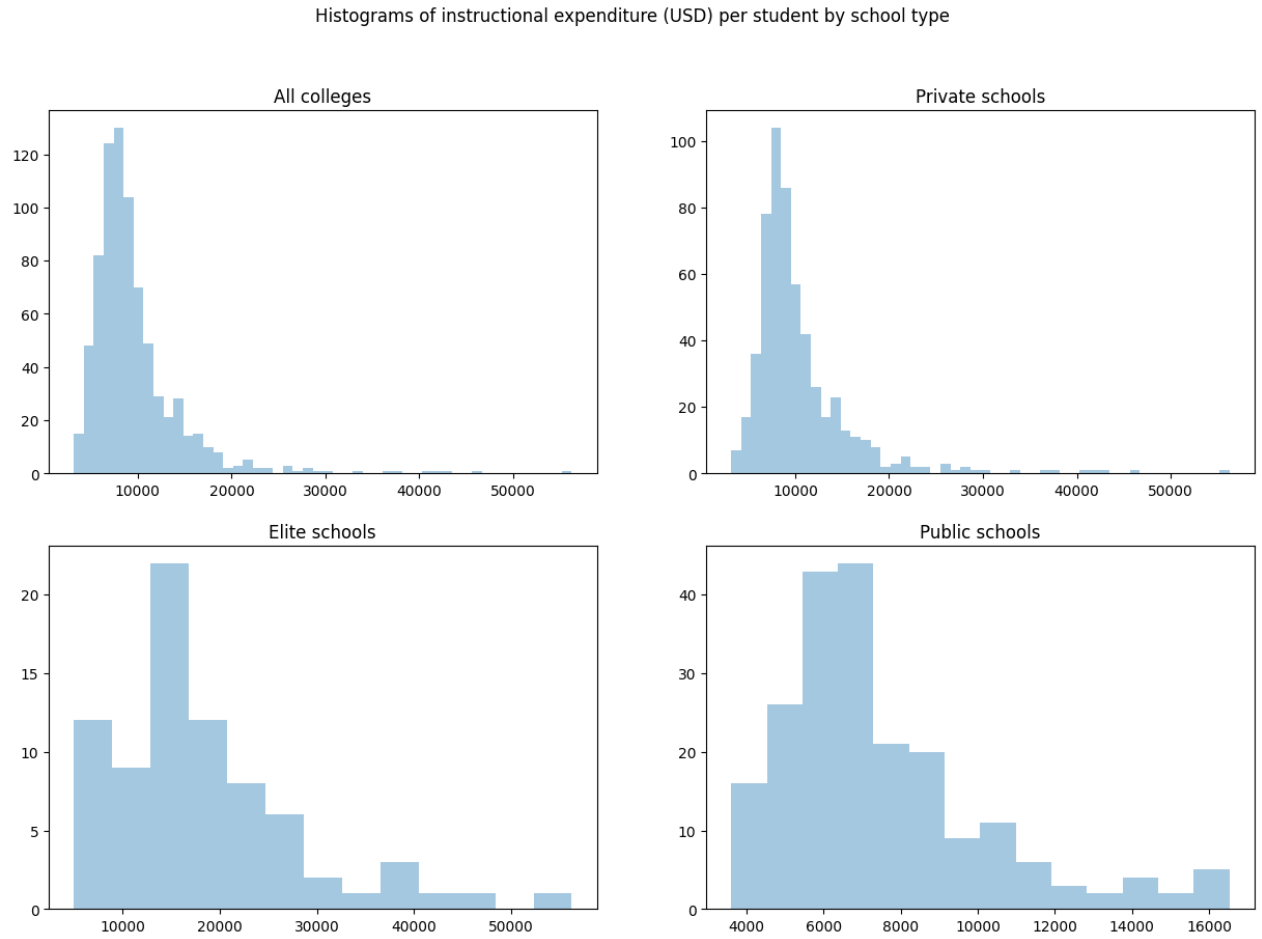
Private schools receive more applications than Public schools. Elite schools receive far less applications than non-Elite schools.

10.3 **Q: Which school type has higher instructional expenditure (in USD) per student?**

```
# Create grid of plots (fig)
# ax will be an array of four Axes objects
# Set the figure size so the plots aren't all squished together
fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize = (15, 10))

# Create histogram for instructional expenditure per student across all colleges
sns.distplot(df["Expend"], kde = False, ax = axes[0, 0])
axes[0, 0].set(xlabel = "", title = "All colleges")

# Create histogram for instructional expenditure per student at private colleges
sns.distplot(df.loc[df["Private"] == "Yes", "Expend"], kde = False, ax = axes[0, 1])
axes[0, 1].set(xlabel = "", title = "Private schools")

# Create histogram for instructional expenditure per student at elite colleges
sns.distplot(df.loc[df["Elite"] == "Yes", "Expend"], kde = False, ax = axes[1, 0])
axes[1, 0].set(xlabel = "", title = "Elite schools")

# Create histogram for instructional expenditure per student at public colleges
sns.distplot(df.loc[df["Private"] == "No", "Expend"], kde = False, ax = axes[1, 1])
axes[1, 1].set(xlabel = "", title = "Public schools")

fig.suptitle("Histograms of instructional expenditure (USD) per student by school type")
```

```
<ipython-input-138-4ab105d70d62>:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["Expend"], kde = False, ax = axes[0, 0])
<ipython-input-138-4ab105d70d62>:11: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.loc[df["Private"] == "Yes", "Expend"], kde = False, ax = axes[0,
<ipython-input-138-4ab105d70d62>:15: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.loc[df["Elite"] == "Yes", "Expend"], kde = False, ax = axes[1, 0
<ipython-input-138-4ab105d70d62>:19: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.loc[df["Private"] == "No", "Expend"], kde = False, ax = axes[1,
Text(0.5, 0.98, 'Histograms of instructional expenditure (USD) per student by
school type')
```



```
# Generate numerical summary of instructional expenditure per student by public vs priva
df["Expend"].groupby(by = df["Private"]).describe()
```

Private	count	mean	std	min	25%	50%	75%	max
No	212.0	7458.316038	2695.541611	3605.0	5715.0	6716.5	8570.25	16527.0
Yes	565.0	10486.353982	5682.576587	3186.0	7477.0	8954.0	11625.00	56233.0

```
# Generate numerical summary of instructional expenditure per student by elite vs non-el
df["Expend"].groupby(by = df["Elite"]).describe()
```

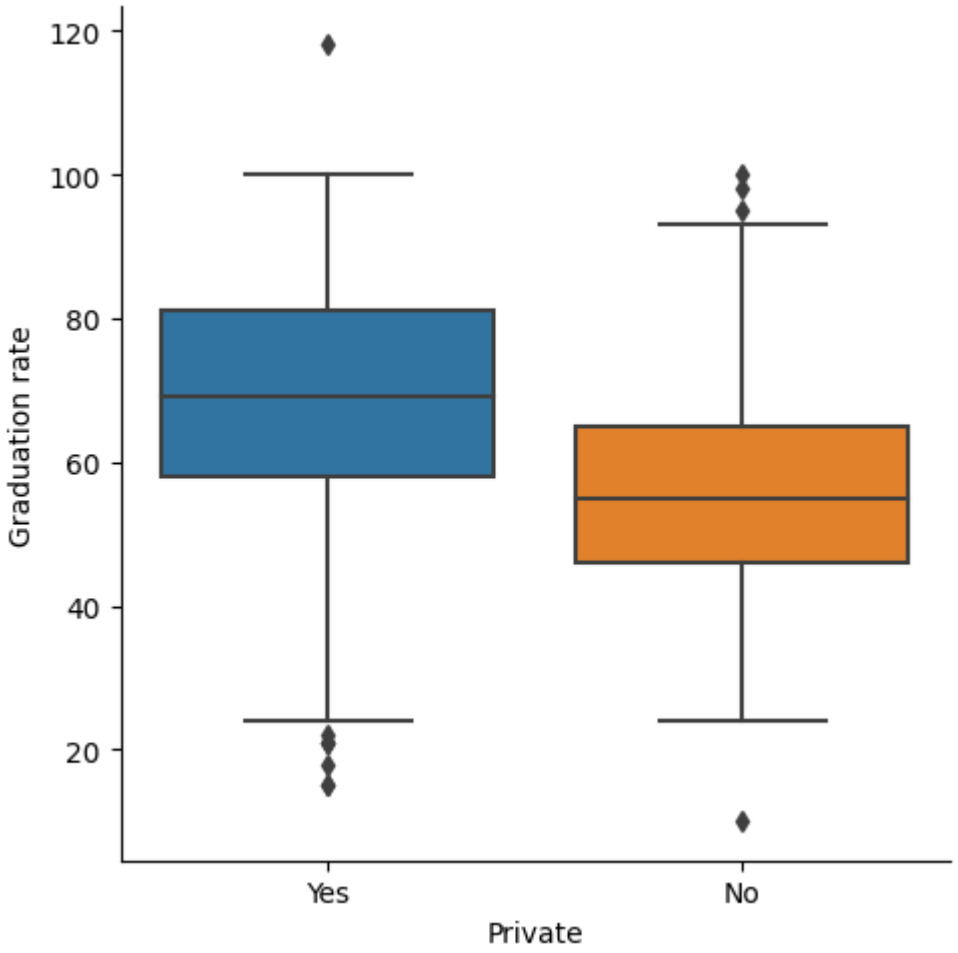
Elite	count	mean	std	min	25%	50%	75%	max
No	699.0	8684.367668	3246.275298	3186.0	6620.00	8086.0	10060.50	42926.0
Yes	78.0	18404.871795	9651.812255	4957.0	12644.25	15944.0	21420.25	56233.0

Private schools have higher intructional expenditure (USD) per student than Public schools.
Elite schools have the highest average instructional expenditure (USD) per student.

10.4. **Q: Which school type has higher graduation rate, private or public?**

```
# Side-by-side boxplots of public vs private schools
```

```
# Side-by-side boxplots for public vs private schools
ax = sns.catplot(x = "Private", y = "Grad_Rate", kind = "box", order = ["Yes", "No"], da
ax.set(ylabel = "Graduation rate")
plt.show()
```



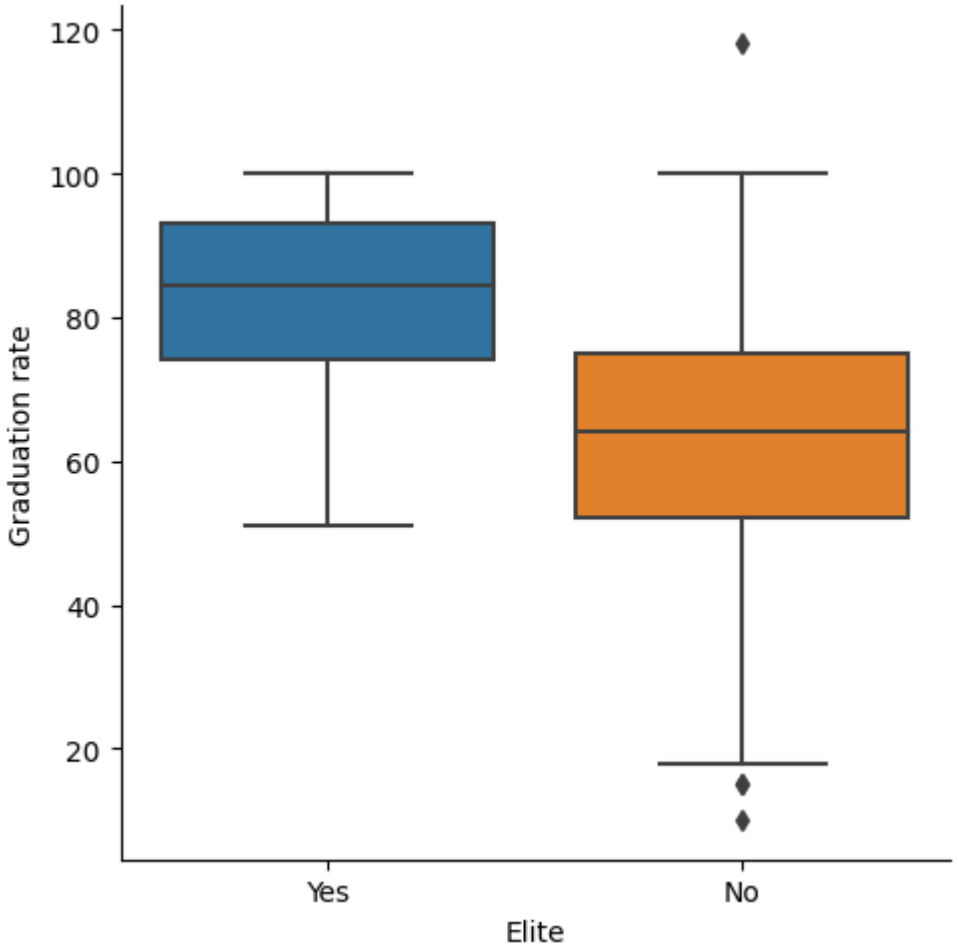

```
# Generate numerical summary of graduation rate by public vs private schools
df["Grad_Rate"].groupby(by = df["Private"]).describe()
```

Private	count	mean	std	min	25%	50%	75%	max
No	212.0	56.042453	14.583412	10.0	46.0	55.0	65.0	100.0
Yes	565.0	68.998230	16.749457	15.0	58.0	69.0	81.0	118.0

Private schools have higher graduation rate on average.

10.5. Q: Which school type has higher graduation rate, Elite or Non-Elite?

```
# Side-by-side boxplots for elite vs non-elite schools
ax = sns.catplot(x = "Elite", y = "Grad_Rate", kind = "box", order = ["Yes", "No"], data
ax.set(ylabel = "Graduation rate")
plt.show()
```




```
# Generate numerical summary of graduation rate by elite vs non-elite schools
df["Grad_Rate"].groupby(by = df["Elite"]).describe()
```

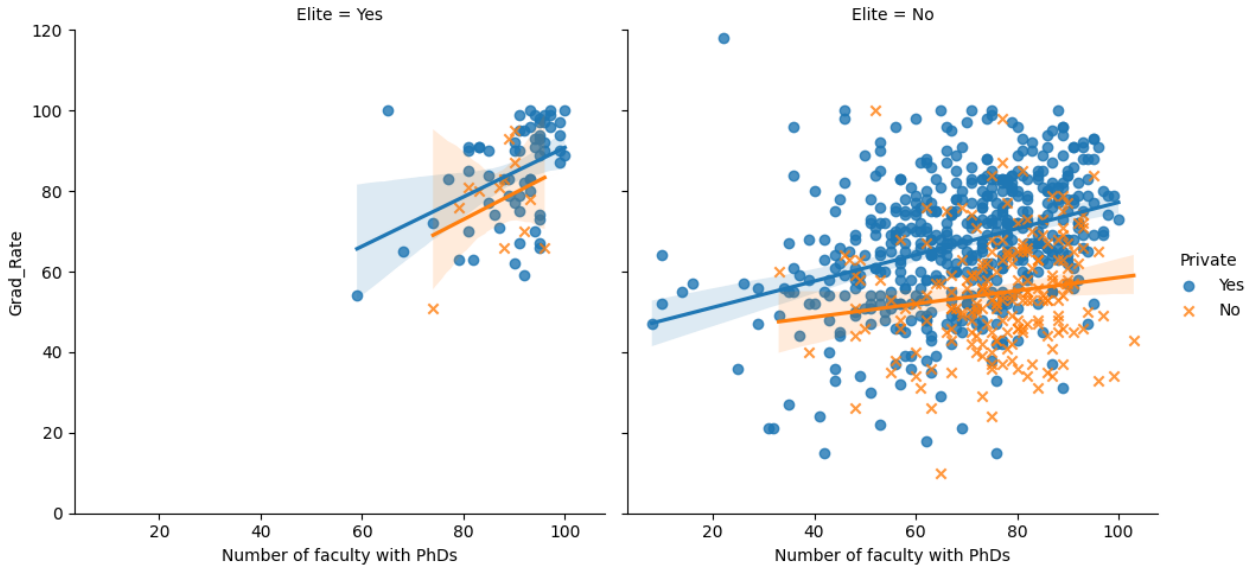
Elite	count	mean	std	min	25%	50%	75%	max
No	699.0	63.463519	16.469466	10.0	52.00	64.0	75.0	118.0
Yes	78.0	83.384615	12.380578	51.0	74.25	84.5	93.0	100.0

There is an even more drastic difference when comparing elite and non-elite schools. The minimum graduation rate among elite schools is almost exactly equal to the first quartile for non-elite schools.

10.6. **Q: Does the number of faculty with PhDs have a correlation with Graduation Rate in Elite schools versus Non-Elite schools?**

```
# Create pair of scatter plots analyzing the relationship between number of faculty with
# Include least squares regression lines to help distinguish between different facets of
# Use columns to distinguish between elite and non-elite schools
# Use hue to distinguish between public and private schools
g = sns.lmplot(x = "PhD", y = "Grad_Rate", hue = "Private", col = "Elite", col_order = [
```

```
                markers = ["o", "x"], data = df)
g.set(xlabel = "Number of faculty with PhDs", ylim = (0, 120))
plt.show()
```



There is clear positive relationship between the Graduation Rate and Number of Faculty with PhDs for elite schools, and while the relationship is weaker for non-elite ones. Also, there seems to be a sharper increase in graduation rates per additional faculty member with a PhD for Elite schools when compared to Non-Elite schools.

Colab paid products - Cancel contracts here