CSE 242 Assignment 1, Fall 2021

4 Questions, 100 pts, due: 23:59 pm, Oct 12th, 2021

Your name: Nayan Sanjay Bhatia Student ID:1926648

Instruction

- Submit your assignments onto **Gradescope** by the due date. Upload a `zip` file containing:

  (1) The saved/latest `.ipynb` file.

  (2) All other materials to make your `.ipynb` file runnable.

- This is an **individual** assignment. All help from others (from the web, books other than text, or people other than the TA or instructor) must be clearly acknowledged.
- Most coding parts can be finished with only 1-2 lines of codes.
- Make sure you have installed required packages: `pandas, seaborn, matplotlib`

Objective

- **Task 1:** Review of **Probability**
- **Task 2:** Getting familiar with **Pandas** and **Seaborn/Matplotlib**

Question 1 (Conditional probability, 10 pts)

Assume that the conditional probability of an email (chosen uniformly and randomly from a set of emails) containing the word “payment”, given that the email is a spam email, is 72%. Suppose that the conditional probability of an email being spam, given that it contains the word “payment”, is 8%. Find the ratio of the probability that an email is spam to the probability that an email contains the word “payment”.

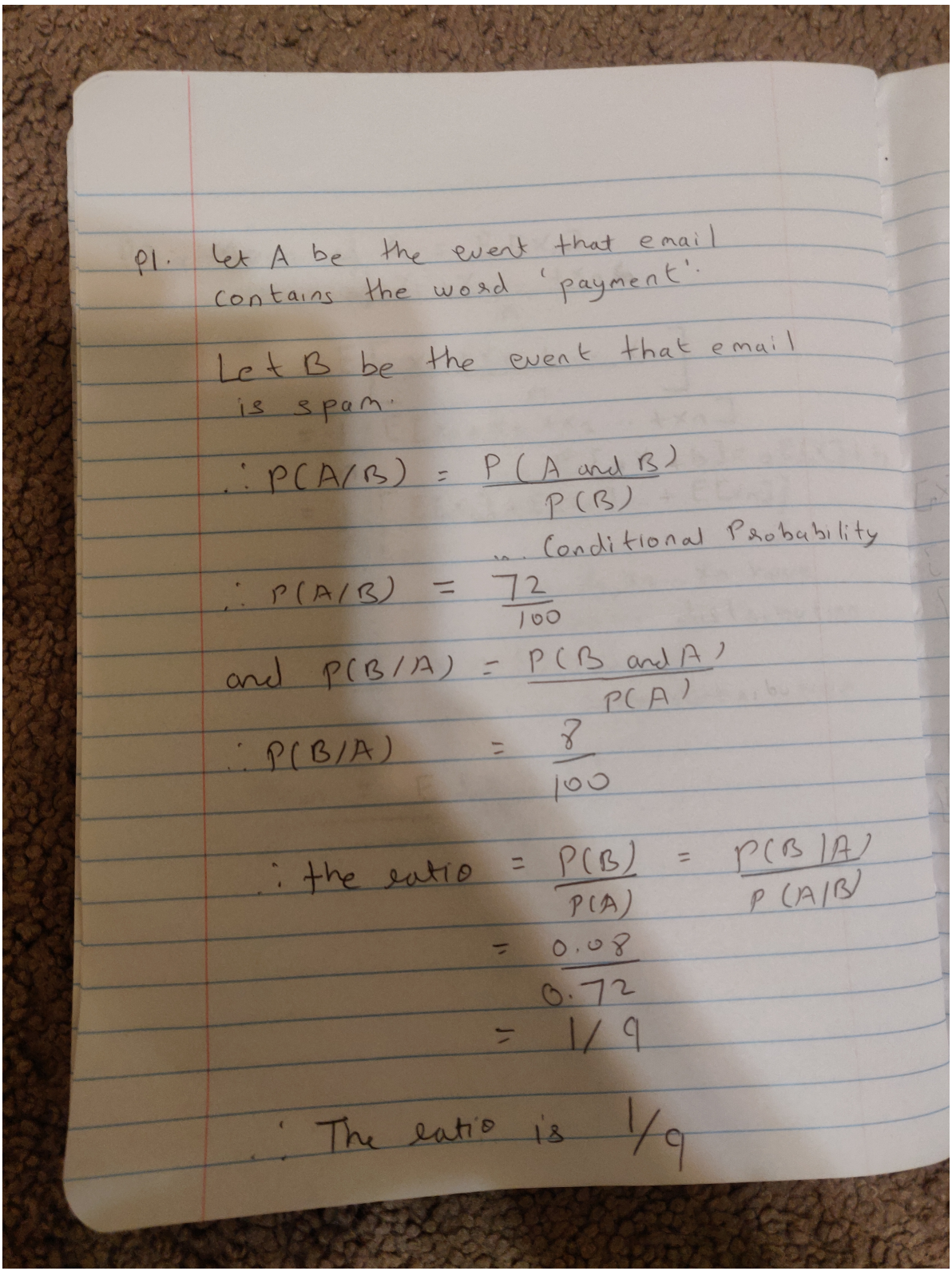
Solution:

If you are not familair with Latex, you may attach a figure/screen-shoot and display the code below.

In [30]:

```
from IPython.display import Image
# Replace the figure name
Image(filename='1.jpg')
```

Out[30]:



Typesetting math: 100%

Question 2 (Expectation and variance, 30 pts)

Suppose that $X_1, ..., X_n$ are independent random variables with the same distribution.

(a --15 pts): Denote the mean of X_i as $\mathbf{E}[X_1]$, find the mean of $$\dfrac{X_1+\cdot \cdot \cdot+ X_n}{n}.$$

(b --15 pts): Denote the variance of X_i as $\text{Var}[X_1]$, find the variance of $$\dfrac{X_1+\cdot \cdot \cdot+ X_n}{n}.$$

Solution (a):

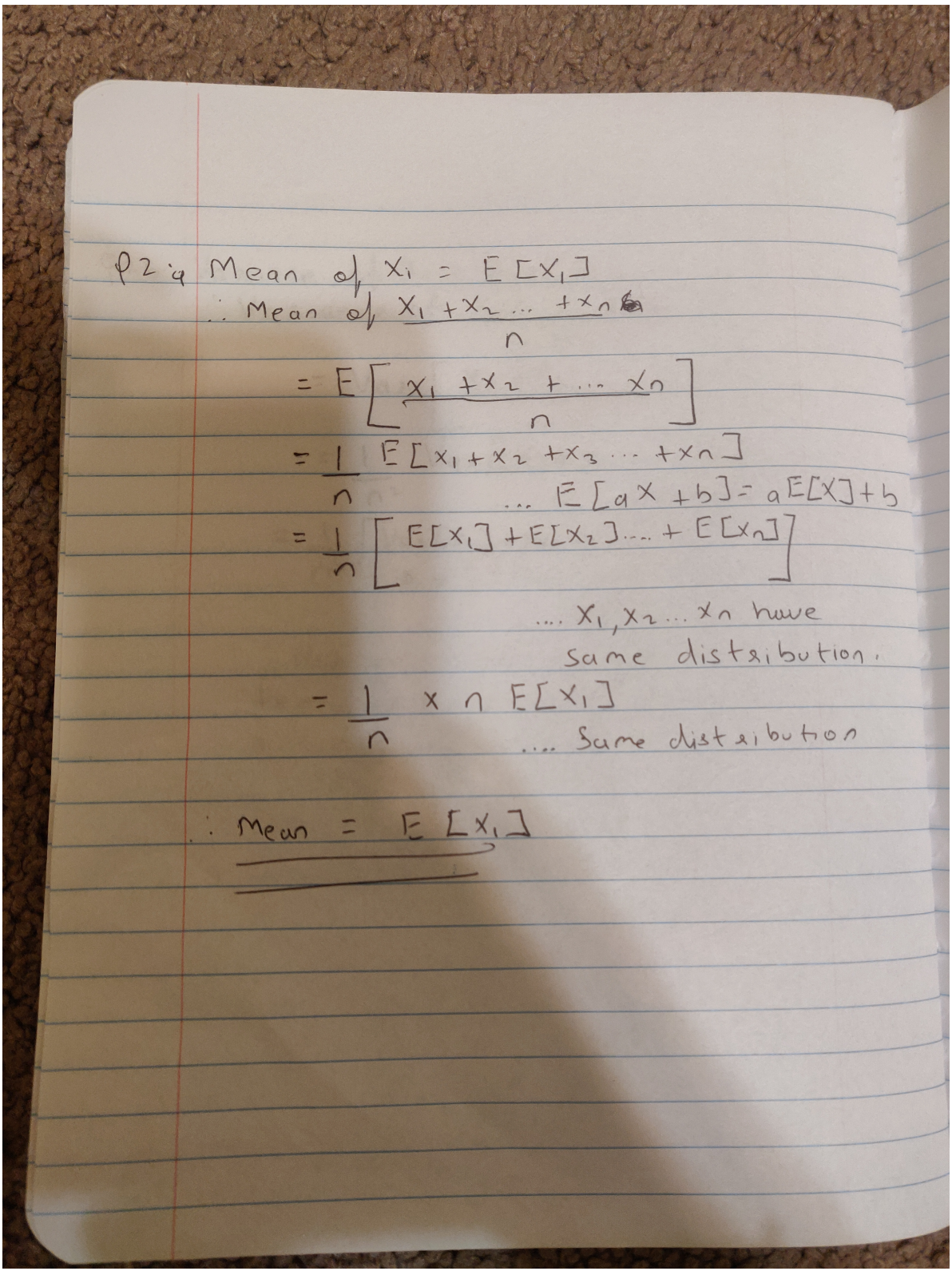
Solution (b):

If you are not familair with Latex, you may attach a figure/screen-shoot and display the code below.

In [31]:

```
# Replace the figure name
from IPython.display import Image
Image(filename='2a.jpg')
```

Out[31]:

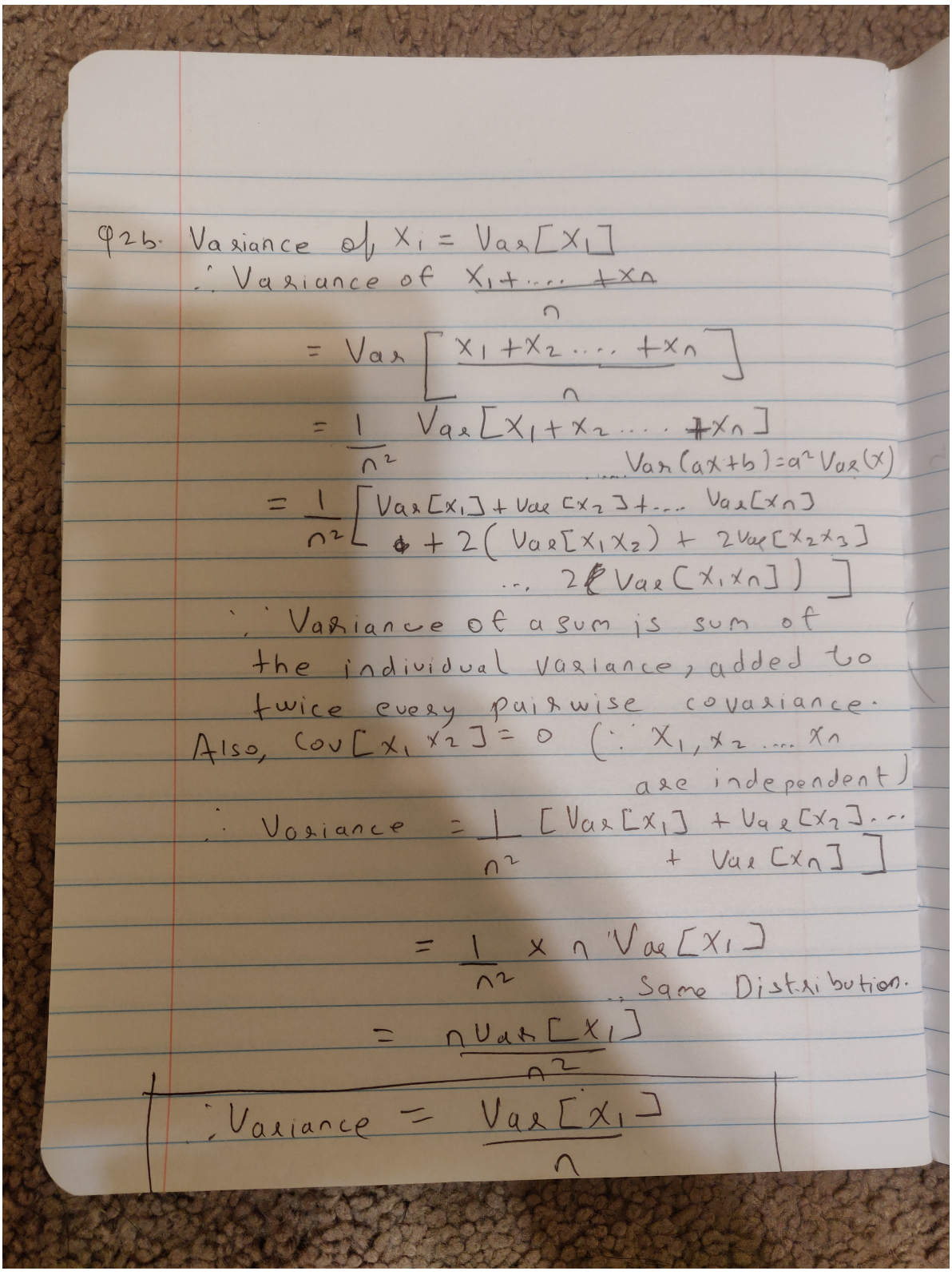


Typesetting math: 100%

In [33]:

```
# Replace the figure name
from IPython.display import Image
Image(filename='2b.jpg')
```

Out[33]:



Typesetting math: 100%

Question 3 (Pandas, 25 pts)

In this question, you will be using **Pandas** to apply exploratory data analysis of a Covid-19 dataset (from **The New York Times**).

If you have not installed the required packages, please refer to the **lab session material** for instructions.

Reading data using Pandas

In [34]:

```
# Read the dataset you will be working on
# The dataframe loaded with pandas is named as data
import pandas as pd
data = pd.read_csv('covid_19.csv')

# Take a look at the first 3 rows
data.head(3)
```

Out[34]:

	date	state	cases	deaths
0	1/21/20	Washington	1	0
1	1/22/20	Washington	1	0
2	1/23/20	Washington	1	0

Question 3.1 (Get the shape of data, 5 pts)

Print the number of rows and columns of the dataframe "data"

In [35]:

```
################## Your answer for Question 3.1   #################

################## Your code here #################
print(len(data),len(data.columns)) #rows, columns
print(data.shape) #alternate method
```

```
31089 4
(31089, 4)
```

Data information

In Pandas, there are many summary functions which contain statistics as well as other data information. The name of the columns are:

In [36]:

```
data.columns
```

Out[36]:

```
Index(['date', 'state', 'cases', 'deaths'], dtype='object')
```

A brief summary of the dataset information:

mean: Mean of the values.

std: Standard deviation of the observations.

25%: The lower percentile.

75%: The upper percentile.

You may use `.describe()` to get a brief summary of the dataframe information.

In [37]:

```
data.describe()
```

Out[37]:

	cases	deaths
count	3.108900e+04	31089.000000
mean	3.235684e+05	6171.822413
std	5.600332e+05	10224.348148
min	1.000000e+00	0.000000
25%	1.670600e+04	362.000000
50%	1.108810e+05	2075.000000
75%	4.098610e+05	7360.000000
max	4.647180e+06	68034.000000

To show the summarized information of a variable (i.e., the variable "deaths"):

In [38]:

```
# We can access a certain variable ("deaths") of the dataframe ('data') simply through data
data.deaths.describe()
```

Out[38]:

```
count    31089.000000
mean      6171.822413
std      10224.348148
min          0.000000
25%        362.000000
50%       2075.000000
75%       7360.000000
max      68034.000000
Name: deaths, dtype: float64
```

Missing values and data types

Entries with missing values are usually assigned with the value **NaN** ("Not a Number"), and the datatype is float64 dtype.

Question 3.2 (Check missing values, 5 pts)

Check whether there are missing values in the dataframe: print how many missing values exist in each column.

In [39]:

```
################## Hint: this dataset does not have empty values #################
################## Your answer for Question 3.2   #################

################## Your code here #################
print(data.isna().sum())
```

```
date      0
state     0
cases     0
deaths    0
dtype: int64
```

Indexing and slicing

In [40]:

```
# Get the 10-th row for variable "State"
data['state'][10]
```

Out[40]:

```
'Illinois'
```

Index based selection with `iloc` : `iloc` is **row-first**, **column-second**.

In [41]:

```
# The first row of the dataframe
print(data.iloc[0])
```

```
date         1/21/20
state     Washington
cases              1
deaths             0
Name: 0, dtype: object
```

In [42]:

```
# The first column of the dataframe
print(data.iloc[:, 0])
```

```
0        1/21/20
1        1/22/20
2        1/23/20
3        1/24/20
4        1/24/20
          ...
31084    9/18/21
31085    9/18/21
31086    9/18/21
31087    9/18/21
31088    9/18/21
Name: date, Length: 31089, dtype: object
```

In [43]:

```
# The first column (and 2nd-5th rows) of the dataframe
print(data.iloc[2:6, 0])
# or pass a list
print(data.iloc[[i+2 for i in range(4)], 0])
```

```
2    1/23/20
3    1/24/20
4    1/24/20
5    1/25/20
Name: date, dtype: object
2    1/23/20
3    1/24/20
4    1/24/20
5    1/25/20
Name: date, dtype: object
```

Question 3.3 (Conditional selection, 10 pts)

What are the number of "cases" and "deaths" for 'California' on '8/21/21'? (print the corresponding row in this dataframe with `loc`)

In [44]:

```
################## Your answer for Question 3.3   #################


################## Your code here (complete the code above) #################
print(data.loc[(data['state']=="California") &( data['date']=="8/21/21")])
```

```
          date       state    cases  deaths
29498  8/21/21  California  4316350   65082
```

Question 3.4 (Data aggregation, 5 pts)

Add a new column named "ratio" (for the dataframe "data") which defined as the ratio "deaths"/"cases" in each row.

In [45]:

```
################## Your answer for Question 3.4   #################

################## Your code here #################

ratio=data['deaths']/data['cases']
data['ratio']=ratio
print(data.tail())
```

```
          date          state   cases  deaths     ratio
31084  9/18/21       Virginia  827197   12242  0.014799
31085  9/18/21     Washington  623254    7256  0.011642
31086  9/18/21  West Virginia  221513    3370  0.015214
31087  9/18/21      Wisconsin  772089    8703  0.011272
31088  9/18/21        Wyoming   83958     918  0.010934
```

Question 4 (Seaborn and Matplotlib, 35 pts)

Visualizing pairplots using seaborn

Seaborn: Python library for statistical data visualization built on top of Matplotlib

Tutorial: detailed example codes are here (https://seaborn.pydata.org/tutorial.html) if needed.

Now we shortly switch our focus to data that only about California, Arizona and Washington.

In [46]:

```
# the sub-dataframe contains only 'California', 'Arizona', 'Washington' is named as subset
subset = data.loc[data['state'].isin(['California', 'Arizona', 'Washington'])]
subset = subset.reset_index(drop=True)
subset.tail()
```

Out[46]:

	date	state	cases	deaths	ratio
1807	9/17/21	California	4637212	67945	0.014652
1808	9/17/21	Washington	623143	7256	0.011644
1809	9/18/21	Arizona	1064346	19487	0.018309
1810	9/18/21	California	4647180	68034	0.014640
1811	9/18/21	Washington	623254	7256	0.011642

In [47]:

```
# import required packages
import seaborn as sns
import matplotlib.pyplot as plt
# Allow figures to be shown in the jupyter notebook interface
%matplotlib inline
```

Questipn 4.1 (Visualizing statistical relationships, 10 pts)

In Seaborn, `relplot()` provides access to several different axes-level functions that show the relationship between two variables with semantic mappings of subsets.

Adopt `relplot()` and visualize how the variable "cases" changes w.r.t "date" for three selected states.

A basic tutorial is here (https://seaborn.pydata.org/generated/seaborn.relplot.html).

In [48]:

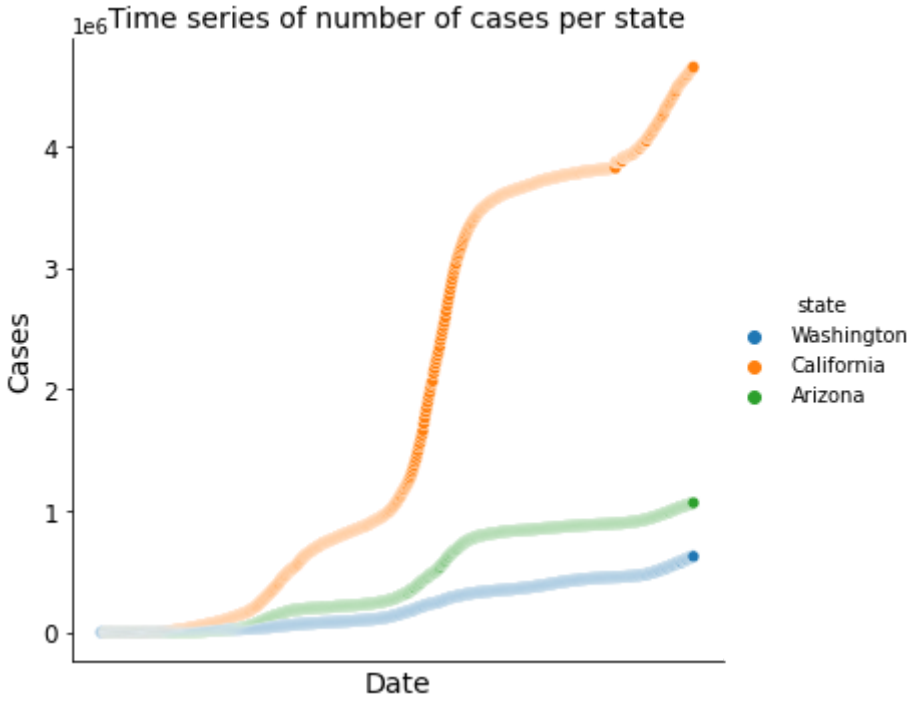
```
################## Your answer for Question 4.1   #################

################## Your code here #################



# Ignore the xticks (since too many dates)
# Use matplotlib to modify figure parameters
sns.relplot(data=subset,x="date",y="cases",hue="state")
plt.tick_params(axis='both',          # changes apply to the x-axis
    which='both',      # both major and minor ticks are affected
    bottom=False,      # ticks along the bottom edge are off
    top=False,         # ticks along the top edge are off
    labelbottom=False)


plt.ylabel('Cases', fontsize=14)
plt.yticks(fontsize=12)
plt.xlabel('Date', fontsize=14)
plt.title('Time series of number of cases per state', fontsize=14)
plt.show()
```



Question 4.2 (Regression plot with Seaborn, 5 * 3 + 10 pts)

In Seaborn, there are several statistical models to estimate a simple relationship between two sets of observations. A basic tutorial is here (https://seaborn.pydata.org/tutorial/regression.html).

Suppose we are only interested in covid-19 information for "California"

In [49]:

```
# Only adopt samples of California information

data_ca = subset[subset['state']=='California']
print(data_ca.shape)
data_ca = data_ca.reset_index(drop=True)
data_ca.head()
```

```
(603, 5)
```

Out[49]:

	date	state	cases	deaths	ratio
0	1/25/20	California	1	0	0.0
1	1/26/20	California	2	0	0.0
2	1/27/20	California	2	0	0.0
3	1/28/20	California	2	0	0.0
4	1/29/20	California	2	0	0.0

Question 4.2.1 Visualize 1 (5 pts)

Use seaborn `regplot()` to visualize the relationship between "date_order" "deaths". (select only first 50 rows of the dataframe "data_ca")

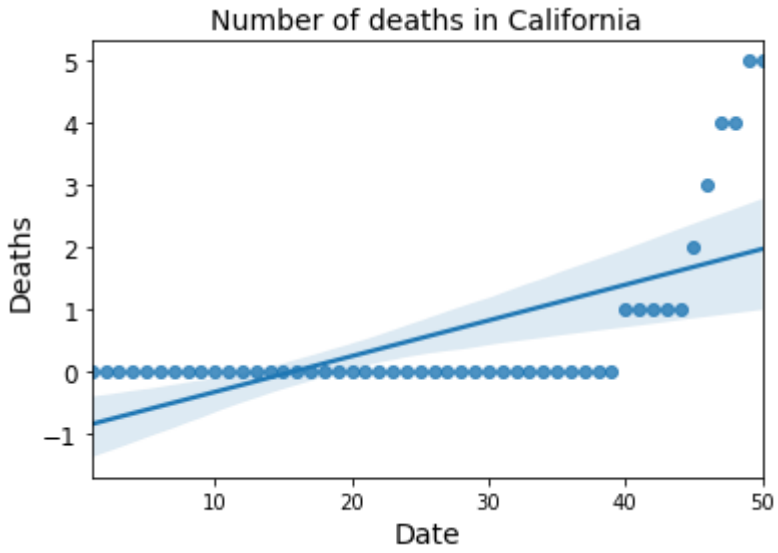
A reference link is here (https://seaborn.pydata.org/generated/seaborn.regplot.html).

In [50]:

```
data_ca['date_order'] = [i+1 for i in range(data_ca.shape[0])]
# print(data_ca[:50].shape)
data_ca_head50=data_ca[:50]
################## Your answer for Question 4.2.1   ##################

################## Your code here (reminder: select "date_order" rather than "date") #######

sns.regplot(x="date_order",y="deaths",data=data_ca_head50)
plt.ylabel('Deaths', fontsize=14)
plt.yticks(fontsize=12)
plt.xlabel('Date', fontsize=14)
plt.title('Number of deaths in California', fontsize=14)
plt.show()
```



Queation 4.2.2 Visualize 2 (5 pts)

Use seaborn `regplot()` to visualize the relationship between "date_order" "deaths". (select first 100 rows of the dataframe "data_ca")

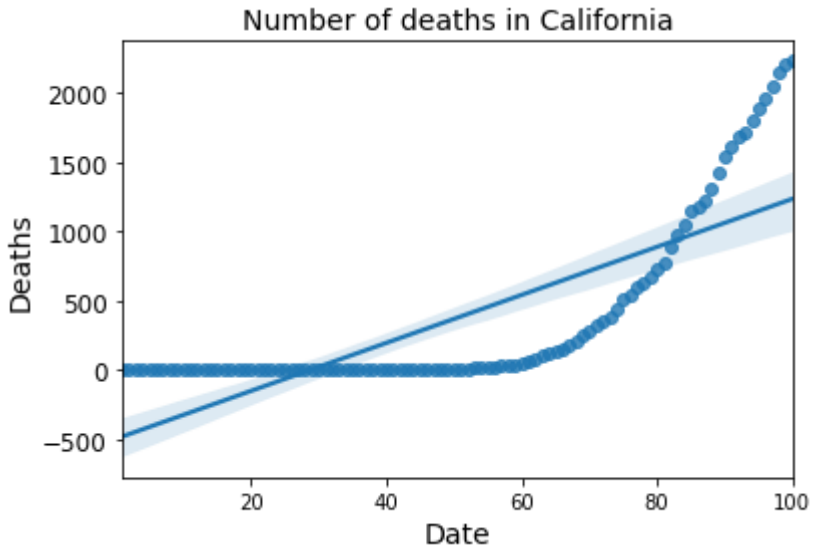
Same as Visualize 1, but with more rows of the dataframe "data_ca" included. A reference link is here (https://seaborn.pydata.org/generated/seaborn.regplot.html).

In [51]:

```
################## Your answer for Question 4.2.2   ##################

################## Your code here (only visualize w.r.t. first 100 rows of dataframe "data_c

data_ca_head100=data_ca[:100]
sns.regplot(x="date_order",y="deaths",data=data_ca_head100)
plt.ylabel('Deaths', fontsize=14)
plt.yticks(fontsize=12)
plt.xlabel('Date', fontsize=14)
plt.title('Number of deaths in California', fontsize=14)
plt.show()
```



Queation 4.2.3 Visualize 3 (5 pts)

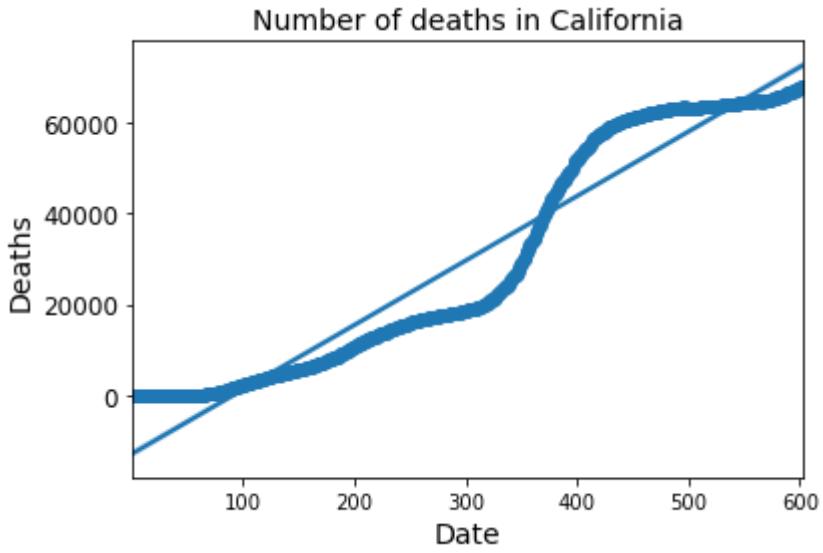
Use seaborn `regplot()` to visualize the relationship between "date_order" "deaths". (use the whole dataframe "data_ca")

Same as Visualize 1, but use the whole dataframe "data_ca".

In [52]:

```
################## Your answer for Question 4.2.3   #################

################## Your code here #################
sns.regplot(x="date_order",y="deaths",data=data_ca)
plt.ylabel('Deaths', fontsize=14)
plt.yticks(fontsize=12)
plt.xlabel('Date', fontsize=14)
plt.title('Number of deaths in California', fontsize=14)
plt.show()
```



Question 4.2.4 What is your observations from the above three figures? (10 pts, open question)

Your observations:

For the graphs, we see that the slope of regression increases as the data points increases. We see that graph with 50 data points represents less data hence likely to make wrong interpretation. As the dataset increases, the model is less likely bias and the variance starts increasing. We get more generalised model for different used cases. The mean error might be high as the dataset increases but it can be more accurate for unseen data. Hence we can get accurate prediction for the number of deaths for a given date for the third graph.