case_4.4

April 22, 2020

1 How do we deploy the Chicago police force to efficiently fight crime?

[2]:
```
import pandas              as pd # The gold standard of Python data analysis,␣
↪to create and manipulate tables of data
import numpy               as np # The Python module for processing arrays␣
↪which/Pandas is based on
import matplotlib.pyplot   as plt # The gold standard of Python data␣
↪visualization, but can be complex to use
import seaborn             as sns; sns.set() # A package to make Matplotlib␣
↪visualizations more aesthetic
import branca
import geopandas

import folium # package for making maps, please make sure to use a version␣
↪older than 1.0.0.
from wordcloud import WordCloud # A package that will allow us to make a␣
↪wordcloud
from scipy.stats import ttest_ind # A module for Python machine learning--we'll␣
↪stick to T-Tests here
from IPython.display import display
#from folium.plugins import TimeSliderChoropleth
from time_slider_choropleth import TimeSliderChoropleth
%matplotlib inline
plt.rcParams["figure.figsize"] = (8,5)
```

"I think that's how Chicago got started. A bunch of people in New York said, 'Gee, I'm enjoying the crime and the poverty, but it just isn't cold enough. Let's go west." - Richard Jeni

"Chicago is known for good steaks, expensive stores, and beautiful architecture. Unfortunately, the Windy City also enjoys a reputation for corrupt politics [and] violent crime." - Bob Barr

1.1 Introduction (5 mts)

Business Context. Congratulations! You were recently promoted to regional Chief of Strategy of the Chicago Police Department. You have many years of experience with field work, but this is

your first time having to think about the bigger picture. Chicago is a large city, and your resources are limited. Thus you need to devise a comprehensive plan to enhance the efficiency of police force deployment to fight crime. Making data-driven decisions is essential, even in law enforcement where prior knowledge usually dominates the decision-making process.

Business Problem. Your main task is to **explore the data and identify patterns of crime in Chicago, and come up with strategies to efficiently deploy your workforce to fight crime.**

Analytical Context. So, you found a dataset available to the Chicago PD from 2017 with information on crimes committed throughout the city. In this case, we will focus on exploratory analysis to construct some preliminary strategies for police deployment. These strategies can be further consolidated or dismissed using more rigorous statistical analysis. One of the key aspects of this case is that our data contains records of crime incidents where often we do not have a clear definition of outcome (such as "severity" of a crime). We will discuss ways of dealing with such data, and how they can be incorporated to have meaningful conclusions.

The case is structured as follows. We will (1) look at univariate summaries (2) come up with a preliminary strategy based on this (3) look at joint distributions and revise our strategy, and finally (4) think about changing strategies depending on our priorities and severity of the crimes.

1.2 Exploring patterns associated with individual variables of interest

Let's read in and view our dataset. This dataset is downloaded from this website. It contains reported crime incidents (with the exception of murders, where data exists for each victim) that occured in the City of Chicago in 2017.

[3]:
```
df = pd.read_csv('Chicago_crime_data.csv', dtype={'ID': object, 'beat_num':␣
↪object})
df
```

[3]:

	ID	Case Number	Date	\
0	11192233	JB100016	12/31/17 23:58	
1	11196379	JB105867	12/31/17 23:50	
2	11192540	JB100551	12/31/17 23:48	
3	11192239	JB100032	12/31/17 23:45	
4	11192254	JB100003	12/31/17 23:45	
5	11192231	JB100018	12/31/17 23:43	
6	11192238	JA569474	12/31/17 23:40	
7	11192213	JB100015	12/31/17 23:39	
8	11192991	JB100001	12/31/17 23:30	
9	11192682	JB100690	12/31/17 23:30	
10	11192349	JB100035	12/31/17 23:30	
11	11206183	JB119133	12/31/17 23:30	
12	11192413	JB100031	12/31/17 23:28	
13	11192385	JB100030	12/31/17 23:26	
14	11192266	JA569467	12/31/17 23:25	
15	11192368	JA569472	12/31/17 23:20	

```
16      11192302    JA569461  12/31/17 23:20
17      11192237    JA569460  12/31/17 23:20
18      11192198    JA569469  12/31/17 23:16
19      11192221    JA569473  12/31/17 23:15
20      11192209    JB100006  12/31/17 23:14
21      11192216    JA569455  12/31/17 23:12
22      11192264    JA569456  12/31/17 23:11
23      11192279    JB100058  12/31/17 23:10
24      11192204    JA569450  12/31/17 23:07
25      11193107    JB101337  12/31/17 23:00
26      11193671    JB101759  12/31/17 23:00
27      11193298    JB101665  12/31/17 23:00
28      11192687    JB100655  12/31/17 23:00
29      11192657    JB100597  12/31/17 23:00
…         …          …            …
268273  10939343    JA252267     1/1/17 0:00
268274  11103097    JA450689     1/1/17 0:00
268275  10979113    JA304447     1/1/17 0:00
268276  11156625    JA521559     1/1/17 0:00
268277  11614273    JC174951     1/1/17 0:00
268278  11217575    JB133919     1/1/17 0:00
268279  11095070    JA440244     1/1/17 0:00
268280  10983012    JA307979     1/1/17 0:00
268281  10983010    JA307926     1/1/17 0:00
268282  11088530    JA426202     1/1/17 0:00
268283  11010296    JA337038     1/1/17 0:00
268284  11095178    JA440856     1/1/17 0:00
268285  11095180    JA440688     1/1/17 0:00
268286  11160883    JA526992     1/1/17 0:00
268287  10993875    JA320714     1/1/17 0:00
268288  11329960    JB283477     1/1/17 0:00
268289  11160892    JA527074     1/1/17 0:00
268290  11079429    JA420285     1/1/17 0:00
268291  11722919    JC306738     1/1/17 0:00
268292  11160897    JA527112     1/1/17 0:00
268293  10802448    JA101652     1/1/17 0:00
268294  10801104    JA100015     1/1/17 0:00
268295  11488630    JB491364     1/1/17 0:00
268296  11587511    JC142200     1/1/17 0:00
268297  10967496    JA289346     1/1/17 0:00
268298  11035993    JA367627     1/1/17 0:00
268299  10942975    JA261045     1/1/17 0:00
268300  10942796    JA260938     1/1/17 0:00
268301  10801141    JA100083     1/1/17 0:00
268302  11255786    JB185271     1/1/17 0:00

                                      Block  IUCR                 Primary Type  \
```

0	046XX N ST LOUIS AVE	630	BURGLARY
1	024XX N LAKE SHORE DR NB	460	BATTERY
2	001XX E SUPERIOR ST	890	THEFT
3	019XX S CANAL ST	1320	CRIMINAL DAMAGE
4	115XX S STATE ST	041A	BATTERY
5	039XX N LONG AVE	1811	NARCOTICS
6	048XX N AVERS AVE	1310	CRIMINAL DAMAGE
7	107XX S WENTWORTH AVE	502P	OTHER OFFENSE
8	087XX S PEORIA ST	1320	CRIMINAL DAMAGE
9	0000X W HUBBARD ST	1150	DECEPTIVE PRACTICE
10	012XX W 97TH ST	497	BATTERY
11	039XX W 69TH ST	610	BURGLARY
12	007XX E 111TH ST	143A	WEAPONS VIOLATION
13	031XX W FULTON BLVD	497	BATTERY
14	023XX S ALBANY AVE	486	BATTERY
15	072XX S ST LAWRENCE AVE	1310	CRIMINAL DAMAGE
16	077XX S LAFLIN ST	051A	ASSAULT
17	013XX W MORSE AVE	460	BATTERY
18	051XX W STRONG ST	1365	CRIMINAL TRESPASS
19	016XX S CENTRAL PARK AVE	143A	WEAPONS VIOLATION
20	011XX W ARGYLE ST	2890	PUBLIC PEACE VIOLATION
21	037XX E 105TH ST	454	BATTERY
22	020XX W WARREN BLVD	486	BATTERY
23	075XX S KINGSTON AVE	610	BURGLARY
24	024XX E 78TH ST	497	BATTERY
25	047XX N MONTICELLO AVE	620	BURGLARY
26	028XX W WILCOX ST	1320	CRIMINAL DAMAGE
27	065XX S HARVARD AVE	820	THEFT
28	006XX N FAIRBANKS CT	870	THEFT
29	002XX W OHIO ST	281	CRIM SEXUAL ASSAULT
…	… …		…
268273	058XX S SANGAMON ST	1754	OFFENSE INVOLVING CHILDREN
268274	079XX S DR MARTIN LUTHER KING JR DR	1585	SEX OFFENSE
268275	024XX N TRIPP AVE	560	ASSAULT
268276	012XX W 83RD ST	1153	DECEPTIVE PRACTICE
268277	018XX S CALIFORNIA AVE	1153	DECEPTIVE PRACTICE
268278	021XX N MOODY AVE	1153	DECEPTIVE PRACTICE
268279	042XX W LELAND AVE	1752	OFFENSE INVOLVING CHILDREN
268280	086XX S ELIZABETH ST	1752	OFFENSE INVOLVING CHILDREN
268281	003XX W 108TH PL	1752	OFFENSE INVOLVING CHILDREN
268282	063XX S TALMAN AVE	1754	OFFENSE INVOLVING CHILDREN
268283	049XX N HAMLIN AVE	1752	OFFENSE INVOLVING CHILDREN
268284	105XX S PEORIA ST	1753	OFFENSE INVOLVING CHILDREN
268285	037XX N OKETO AVE	1751	OFFENSE INVOLVING CHILDREN
268286	001XX W OHIO ST	1752	OFFENSE INVOLVING CHILDREN
268287	075XX N OAKLEY AVE	2825	OTHER OFFENSE
268288	083XX S ASHLAND AVE	1154	DECEPTIVE PRACTICE

```
268289                    015XX W FARGO AVE  1753  OFFENSE INVOLVING CHILDREN
268290                  095XX S PRAIRIE AVE  1754  OFFENSE INVOLVING CHILDREN
268291                   010XX N PULASKI RD  1585                SEX OFFENSE
268292                  065XX N RICHMOND ST  1752  OFFENSE INVOLVING CHILDREN
268293                      006XX W 62ND ST  1310             CRIMINAL DAMAGE
268294                      004XX W 66TH ST   486                     BATTERY
268295                081XX S SOUTH SHORE DR  1153          DECEPTIVE PRACTICE
268296                  022XX N PARKSIDE AVE  1752  OFFENSE INVOLVING CHILDREN
268297                    084XX S PAULINA ST   486                     BATTERY
268298                    026XX W COYLE AVE  1752  OFFENSE INVOLVING CHILDREN
268299                    035XX S GILES AVE   281         CRIM SEXUAL ASSAULT
268300                  028XX N WESTERN AVE  1330           CRIMINAL TRESPASS
268301                  011XX W DICKENS AVE   486                     BATTERY
268302                     045XX N HAZEL ST  1155          DECEPTIVE PRACTICE

                                        Description  \
0                            ATTEMPT FORCIBLE ENTRY
1                                            SIMPLE
2                                      FROM BUILDING
3                                        TO VEHICLE
4                                  AGGRAVATED: HANDGUN
5                        POSS: CANNABIS 30GMS OR LESS
6                                      TO PROPERTY
7                              FALSE/STOLEN/ALTERED TRP
8                                        TO VEHICLE
9                                   CREDIT CARD FRAUD
10      AGGRAVATED DOMESTIC BATTERY: OTHER DANG WEAPON
11                                   FORCIBLE ENTRY
12                           UNLAWFUL POSS OF HANDGUN
13      AGGRAVATED DOMESTIC BATTERY: OTHER DANG WEAPON
14                            DOMESTIC BATTERY SIMPLE
15                                     TO PROPERTY
16                                 AGGRAVATED: HANDGUN
17                                           SIMPLE
18                                     TO RESIDENCE
19                           UNLAWFUL POSS OF HANDGUN
20                                    OTHER VIOLATION
21                           AGG PO HANDS NO/MIN INJURY
22                              DOMESTIC BATTERY SIMPLE
23                                   FORCIBLE ENTRY
24      AGGRAVATED DOMESTIC BATTERY: OTHER DANG WEAPON
25                                   UNLAWFUL ENTRY
26                                       TO VEHICLE
27                                   $500 AND UNDER
28                                   POCKET-PICKING
29                                   NON-AGGRAVATED
…                                               …
```

```
268273                 AGG SEX ASSLT OF CHILD FAM MBR
268274                                          OTHER
268275                                         SIMPLE
268276            FINANCIAL IDENTITY THEFT OVER $ 300
268277            FINANCIAL IDENTITY THEFT OVER $ 300
268278            FINANCIAL IDENTITY THEFT OVER $ 300
268279                  AGG CRIM SEX ABUSE FAM MEMBER
268280                  AGG CRIM SEX ABUSE FAM MEMBER
268281                  AGG CRIM SEX ABUSE FAM MEMBER
268282                 AGG SEX ASSLT OF CHILD FAM MBR
268283                  AGG CRIM SEX ABUSE FAM MEMBER
268284                  SEX ASSLT OF CHILD BY FAM MBR
268285                   CRIM SEX ABUSE BY FAM MEMBER
268286                  AGG CRIM SEX ABUSE FAM MEMBER
268287                        HARASSMENT BY TELEPHONE
268288       FINANCIAL IDENTITY THEFT $300 AND UNDER
268289                  SEX ASSLT OF CHILD BY FAM MBR
268290                 AGG SEX ASSLT OF CHILD FAM MBR
268291                                          OTHER
268292                  AGG CRIM SEX ABUSE FAM MEMBER
268293                                    TO PROPERTY
268294                        DOMESTIC BATTERY SIMPLE
268295            FINANCIAL IDENTITY THEFT OVER $ 300
268296                  AGG CRIM SEX ABUSE FAM MEMBER
268297                        DOMESTIC BATTERY SIMPLE
268298                  AGG CRIM SEX ABUSE FAM MEMBER
268299                                NON-AGGRAVATED
268300                                       TO LAND
268301                        DOMESTIC BATTERY SIMPLE
268302               AGGRAVATED FINANCIAL IDENTITY THEFT

                Location Description  Arrest  Domestic  …  Ward  \
0                           APARTMENT   False     False  …  33.0
1             MOVIE HOUSE/THEATER   False     False  …  43.0
2                     HOTEL/MOTEL   False     False  …  42.0
3                          STREET   False      True  …  25.0
4                        RESIDENCE   False      True  …  34.0
5                          STREET    True     False  …  38.0
6                        APARTMENT   False     False  …  39.0
7                          STREET    True     False  …  34.0
8                          STREET   False     False  …  21.0
9                   BAR OR TAVERN   False     False  …  42.0
10                       RESIDENCE   False     False  …  21.0
11                RESIDENCE-GARAGE   False     False  …  13.0
12                         STREET    True     False  …   9.0
13                       APARTMENT    True      True  …  27.0
14                       APARTMENT   False      True  …  24.0
```

15	APARTMENT	False	False	…	6.0
16	APARTMENT	False	False	…	17.0
17	CTA TRAIN	True	False	…	49.0
18	APARTMENT	False	False	…	45.0
19	STREET	True	False	…	24.0
20	CTA PLATFORM	True	False	…	48.0
21	RESIDENCE	True	False	…	10.0
22	RESIDENCE PORCH/HALLWAY	False	False	…	2.0
23	APARTMENT	False	False	…	7.0
24	RESIDENCE	False	True	…	7.0
25	ANIMAL HOSPITAL	False	False	…	33.0
26	STREET	False	False	…	2.0
27	RESIDENCE	False	False	…	20.0
28	BAR OR TAVERN	False	False	…	42.0
29	BAR OR TAVERN	False	False	…	42.0
…	…	…	…	…	…	…
268273	RESIDENCE	False	False	…	16.0
268274	RESIDENCE	False	False	…	6.0
268275	RESIDENCE	False	True	…	31.0
268276	RESIDENCE	False	False	…	21.0
268277	RESIDENCE	False	False	…	12.0
268278	OTHER	False	False	…	29.0
268279	RESIDENCE	False	False	…	39.0
268280	RESIDENCE	False	True	…	21.0
268281	RESIDENCE	False	False	…	34.0
268282	RESIDENCE	False	False	…	15.0
268283	RESIDENCE	False	True	…	39.0
268284	RESIDENCE	False	False	…	34.0
268285	RESIDENCE	False	False	…	36.0
268286	RESIDENCE	True	False	…	42.0
268287	APARTMENT	False	False	…	49.0
268288	RESIDENCE	False	False	…	21.0
268289	RESIDENCE	False	False	…	49.0
268290	RESIDENCE	False	False	…	6.0
268291	RESIDENCE	False	False	…	37.0
268292	RESIDENCE	False	False	…	50.0
268293	RESIDENCE-GARAGE	False	False	…	16.0
268294	APARTMENT	True	True	…	20.0
268295	RESIDENCE	False	False	…	7.0
268296	RESIDENCE	False	False	…	36.0
268297	RESIDENCE	False	True	…	18.0
268298	RESIDENCE	False	False	…	50.0
268299	SCHOOL, PUBLIC, BUILDING	True	False	…	2.0
268300	OTHER	False	False	…	1.0
268301	APARTMENT	False	True	…	43.0
268302	COMMERCIAL / BUSINESS OFFICE	False	False	…	46.0

	Community Area	FBI Code	X Coordinate	Y Coordinate	Year	\
0	14	5	1152214.0	1930694.0	2017	
1	7	08B	1175293.0	1916610.0	2017	
2	8	6	1177508.0	1905401.0	2017	
3	31	14	1173432.0	1891037.0	2017	
4	53	04B	1178329.0	1828012.0	2017	
5	15	18	1139656.0	1925536.0	2017	
6	14	14	1149845.0	1931713.0	2017	
7	49	26	1176851.0	1833743.0	2017	
8	71	14	1171830.0	1846925.0	2017	
9	8	11	1176142.0	1903314.0	2017	
10	73	04B	1169792.0	1840606.0	2017	
11	65	5	1151377.0	1858558.0	2017	
12	50	15	1183476.0	1831507.0	2017	
13	27	04B	1155463.0	1901838.0	2017	
14	30	08B	1156063.0	1888248.0	2017	
15	69	14	1181487.0	1857163.0	2017	
16	71	04A	1167458.0	1853409.0	2017	
17	1	08B	1165794.0	1946184.0	2017	
18	11	26	1141289.0	1932431.0	2017	
19	29	15	1152641.0	1891613.0	2017	
20	3	26	1167773.0	1933570.0	2017	
21	52	08B	1202943.0	1835949.0	2017	
22	28	08B	1162769.0	1900351.0	2017	
23	43	5	1194505.0	1855397.0	2017	
24	43	04B	1193681.0	1853719.0	2017	
25	14	5	1151198.0	1931301.0	2017	
26	27	14	1157580.0	1899231.0	2017	
27	68	6	1175193.0	1861522.0	2017	
28	8	6	1178393.0	1904480.0	2017	
29	8	2	1174498.0	1904149.0	2017	
…	…	…	…	…	…	
268273	68	2	1170978.0	1865944.0	2017	
268274	44	17	1180276.0	1852475.0	2017	
268275	20	08A	1147641.0	1915933.0	2017	
268276	71	11	1169650.0	1849748.0	2017	
268277	29	11	NaN	NaN	2017	
268278	19	11	NaN	NaN	2017	
268279	14	20	1147339.0	1931163.0	2017	
268280	71	20	1169500.0	1847499.0	2017	
268281	49	20	1176019.0	1832977.0	2017	
268282	66	2	1159801.0	1862443.0	2017	
268283	14	20	1150166.0	1932498.0	2017	
268284	73	2	1172184.0	1834879.0	2017	
268285	17	20	1126087.0	1923814.0	2017	
268286	8	20	1174846.0	1904157.0	2017	
268287	2	26	1159680.0	1949930.0	2017	

268288	71	11	NaN	NaN	2017
268289	1	2	1164806.0	1949525.0	2017
268290	49	2	1179602.0	1841777.0	2017
268291	23	17	NaN	NaN	2017
268292	2	20	1155481.0	1943206.0	2017
268293	68	14	1173000.0	1863775.0	2017
268294	68	08B	1174537.0	1861156.0	2017
268295	46	11	NaN	NaN	2017
268296	19	17	NaN	NaN	2017
268297	71	08B	1166482.0	1848636.0	2017
268298	2	20	1157570.0	1946019.0	2017
268299	35	2	1178842.0	1881615.0	2017
268300	21	26	1159863.0	1918955.0	2017
268301	7	08B	1168452.0	1914119.0	2017
268302	3	11	NaN	NaN	2017

	Updated On	Latitude	Longitude	Location
0	5/4/18 15:51	41.965694	-87.715726	(41.965693651, -87.715726125)
1	5/4/18 15:51	41.926559	-87.631294	(41.926558908, -87.631294073)
2	5/4/18 15:51	41.895751	-87.623496	(41.895750913, -87.623495923)
3	5/4/18 15:51	41.856427	-87.638893	(41.856426716, -87.638892854)
4	5/4/18 15:51	41.683369	-87.622830	(41.683369303, -87.622829524)
5	5/4/18 15:51	41.951779	-87.762026	(41.951778739, -87.76202629)
6	5/4/18 15:51	41.968536	-87.724410	(41.968536385, -87.724409882)
7	5/4/18 15:51	41.699129	-87.628068	(41.699129337, -87.628068173)
8	5/4/18 15:51	41.735414	-87.646067	(41.735414028, -87.64606746)
9	5/4/18 15:51	41.890055	-87.628576	(41.89005498, -87.628575838)
10	5/4/18 15:51	41.718118	-87.653717	(41.718118187, -87.653716662)
11	5/4/18 15:51	41.767761	-87.720696	(41.767760717, -87.720696433)
12	5/4/18 15:51	41.692842	-87.603880	(41.692842029, -87.603879988)
13	8/17/18 15:59	41.886446	-87.704558	(41.886445649, -87.704558181)
14	5/4/18 15:51	41.849141	-87.702721	(41.849141176, -87.702721436)
15	5/4/18 15:51	41.763291	-87.610373	(41.763291164, -87.610373104)
16	5/4/18 15:51	41.753302	-87.661899	(41.753301733, -87.661899406)
17	5/4/18 15:51	42.007919	-87.665351	(42.007919177, -87.665351128)
18	5/4/18 15:51	41.970669	-87.755853	(41.970669192, -87.755852495)
19	5/4/18 15:51	41.858443	-87.715192	(41.858443405, -87.715191718)
20	5/4/18 15:51	41.973263	-87.658436	(41.973263481, -87.658435722)
21	5/4/18 15:51	41.704557	-87.532458	(41.704556764, -87.532457986)
22	5/4/18 15:51	41.882215	-87.677770	(41.882215163, -87.677770467)
23	5/4/18 15:51	41.758135	-87.562718	(41.758134982, -87.562718432)
24	5/4/18 15:51	41.753551	-87.565793	(41.753550633, -87.565793115)
25	5/4/18 15:51	41.967379	-87.719446	(41.967379335, -87.71944578)
26	5/4/18 15:51	41.879249	-87.696855	(41.879248965, -87.696855018)
27	5/4/18 15:51	41.775396	-87.633312	(41.775395602, -87.633311659)
28	5/4/18 15:51	41.893204	-87.620274	(41.893203509, -87.620273669)
29	5/4/18 15:51	41.892383	-87.634588	(41.892383161, -87.634588313)

```
…                …          …          …                                  …
268273   2/10/18 15:50  41.787623 -87.648634  (41.787623197, -87.648634237)
268274   2/10/18 15:50  41.750455 -87.614955  (41.750454627, -87.614955093)
268275   2/10/18 15:50  41.925277 -87.732920  (41.925277458, -87.732920414)
268276   2/10/18 15:50  41.743208 -87.653972  (41.743208229, -87.653972475)
268277    3/6/19 16:20        NaN        NaN                            NaN
268278   1/30/18 15:52        NaN        NaN                            NaN
268279   2/10/18 15:50  41.967076 -87.733639  (41.967075649, -87.733638571)
268280   2/10/18 15:50  41.737040 -87.654587  (41.737039906, -87.654587076)
268281   2/10/18 15:50  41.697046 -87.631137  (41.697045977, -87.631137431)
268282   2/10/18 15:50  41.778253 -87.689712  (41.778252913, -87.689711865)
268283   2/10/18 15:50  41.970684 -87.723209  (41.970684211, -87.723209014)
268284   2/10/18 15:50  41.702350 -87.645123  (41.702350298, -87.645123324)
268285   2/10/18 15:50  41.947291 -87.811945  (41.947290738, -87.811945372)
268286   2/10/18 15:50  41.892397 -87.633310  (41.892397329, -87.633310027)
268287   2/10/18 15:50  42.018327 -87.687742  (42.018327003, -87.687741942)
268288   5/30/18 15:52        NaN        NaN                            NaN
268289   2/10/18 15:50  42.017108 -87.668891  (42.017108035, -87.668890777)
268290   2/10/18 15:50  41.721113 -87.617751  (41.721113442, -87.617750867)
268291   6/15/19 16:06        NaN        NaN                            NaN
268292   2/10/18 15:50  41.999962 -87.703376  (41.999961961, -87.703375638)
268293   2/14/17 15:49  41.781627 -87.641284  (41.781626794, -87.641284443)
268294   2/14/17 15:49  41.774406 -87.635727  (41.774405884, -87.635727351)
268295  10/27/18 16:07        NaN        NaN                            NaN
268296   2/16/19 15:59        NaN        NaN                            NaN
268297   2/10/18 15:50  41.740225 -87.665612  (41.740224785, -87.665611819)
268298   2/10/18 15:50  42.007638 -87.695614  (42.007638503, -87.695613598)
268299   2/10/18 15:50  41.830450 -87.619323   (41.830450306, -87.61932306)
268300   2/10/18 15:50  41.933326 -87.687927  (41.933326413, -87.687927299)
268301   2/14/17 15:49  41.919874 -87.656504  (41.919874416, -87.656503702)
268302   3/15/18 15:55        NaN        NaN                            NaN

[268303 rows x 22 columns]
```

We can see above that the table contains 22 columns and there are 268,303 records in total. Since each homicide case could have more than one row, the actual number of cases is smaller than 268,303. Blow is a brief description of each column:

Variable name	Variable description	Note
ID	Unique identifier for the record	Each victim in a single homicide case is assigned to a different ID
Case Number	The Chicago Police Department RD (Records Division) number	Unique to the incident. Multiple IDs can share the same Case Number if the incident is a homicide case

Variable name	Variable description	Note
Date	Date when the incident occurred	Might be a best estimate for some records
Block	The partially redacted address where the incident occurred	The redacted address is in the same block as the actual address
IUCR	The Illinois Uniform Crime Reporting code	Directly linked to the primary type and the description of the crime. See details here
Primary Type	The primary description of the IUCR code	-
Description	The secondary description of the IUCR code	-
Location Description	Description of the location where the incident occurred	-
Arrest	Whether an arrest as made	-
Domestic	Whether the incident was domestic-related	Domestic-related definition is based on the Illinois Domestic Violence Act
beat_num	The police beat where the incident occurred	Smallest police geographic area - each beat has a dedicated police beat car. See details here
District	The police district where the incident occurred	Three to five beats make up a police sector and three sectors make up a police district. See details here
Ward	The ward where the incident occurred	Wards are city council districts. See details here
Community Area	The community area where the incident occurred	See details here
FBI Code	The crime classification as outlined in the FBI's NIBRS	NIBRS stands for National Incident-Based Reporting System (NIBRS). See details here
Latitude	The latitude of the location where the incident occurred	This location is shifted from the actual location for partial redaction but falls on the same block

Variable name	Variable description	Note
Longitude	The longitude of the location where the incident occurred	-

There are quite a few factors, but most are either: (1) identifying information (e.g. `id`, `ICUR`); or (2) too granular to start with (e.g. `latitude`, `longitude`). Therefore, we will first focus on the following variables: `primary_type`, `description`, `location_description` (location types), `date` (time of occurrence) and `beat_num` (geographic location), which give valuable information without getting too granular too quickly. Our outcome of interest is the number of crime incidents.

1.2.1 Investigating crimes by types and descriptions (10 mts)

Similar to the last EDA case, it makes sense to explore the relationship of `primary_type` and `description` with our outcome of interest, crime incidents. However, we cannot repeat the exact process where we look at the pairwise correlations between the variables of interest and the outcome. This is because both `primary_type` and `description` are categorical variables, so it would not make send to place them on a scatterplot and calculating correlations makes little sense.

Luckily, both variables are discrete so we can still count the total number of records which belong to a specific category for each of these two variables using a **frequency table**. Note that `primary_type` and `description` are **nested variables** since no category within `description` appears in the same row as two different categories of `primary_type`:

[4]:
```
df["Primary Type"].value_counts()
```

[4]:
```
THEFT                                64356
BATTERY                              49218
CRIMINAL DAMAGE                      29043
ASSAULT                              19304
DECEPTIVE PRACTICE                   19204
OTHER OFFENSE                        17235
BURGLARY                             13000
ROBBERY                              11879
NARCOTICS                            11659
MOTOR VEHICLE THEFT                  11385
CRIMINAL TRESPASS                     6815
WEAPONS VIOLATION                     4686
OFFENSE INVOLVING CHILDREN            2282
CRIM SEXUAL ASSAULT                   1631
PUBLIC PEACE VIOLATION                1498
INTERFERENCE WITH PUBLIC OFFICER      1087
SEX OFFENSE                           1031
PROSTITUTION                           735
```

```
HOMICIDE                              675
ARSON                                 444
LIQUOR LAW VIOLATION                  191
GAMBLING                              191
STALKING                              190
KIDNAPPING                            190
INTIMIDATION                          151
OBSCENITY                              86
CONCEALED CARRY LICENSE VIOLATION      69
NON-CRIMINAL                           37
OTHER NARCOTIC VIOLATION               11
PUBLIC INDECENCY                       10
HUMAN TRAFFICKING                       8
NON-CRIMINAL (SUBJECT SPECIFIED)        2
Name: Primary Type, dtype: int64
```

We can see the most prevalent primary type of crime is theft, followed by battery and criminal damage. More severe types, such as homicide, arson and human trafficking, are very rare. A more detailed description of crime types is listed in the `Description` column. We can further break down the above frequencies by `Description` since `Primary Type` and `Description` are nested variables. The resulting frequency table is shown below:

1.2.2 Exercise 1: (10 mts)

Write code using the `groupby` function, to count the number of cases in all combinations of `Primary Type` and `Description`. Then sort the results in decreasing order of the number of cases. Based on the results, what are most prevalent descriptions of theft, battery and criminal damage cases in Chicago?

[5]:
```
df.groupby(["Primary Type","Description"])["ID"].count().
↪reset_index(name="count")\
.sort_values(by="count", ascending = False).reset_index(drop=True).head(20)
```

[5]:

	Primary Type	Description	count
0	THEFT	$500 AND UNDER	24516
1	BATTERY	DOMESTIC BATTERY SIMPLE	23819
2	BATTERY	SIMPLE	16185
3	THEFT	OVER $500	15352
4	CRIMINAL DAMAGE	TO PROPERTY	13843
5	CRIMINAL DAMAGE	TO VEHICLE	13555
6	ASSAULT	SIMPLE	12743
7	THEFT	FROM BUILDING	10662
8	THEFT	RETAIL THEFT	10460
9	MOTOR VEHICLE THEFT	AUTOMOBILE	9834
10	BURGLARY	FORCIBLE ENTRY	7506
11	DECEPTIVE PRACTICE	FINANCIAL IDENTITY THEFT OVER $ 300	5157
12	BURGLARY	UNLAWFUL ENTRY	4594

```
13              ROBBERY                      ARMED: HANDGUN   4550
14   DECEPTIVE PRACTICE                CREDIT CARD FRAUD   3939
15        OTHER OFFENSE                 TELEPHONE THREAT   3924
16    CRIMINAL TRESPASS                          TO LAND   3850
17    WEAPONS VIOLATION          UNLAWFUL POSS OF HANDGUN   3597
18              ROBBERY             STRONGARM - NO WEAPON   3564
19              ASSAULT               AGGRAVATED: HANDGUN   2891
```

Answer. Summarizing the data by this more detailed classification of crime types reveals what the prevalent crime descriptions are within each primary type. For example:

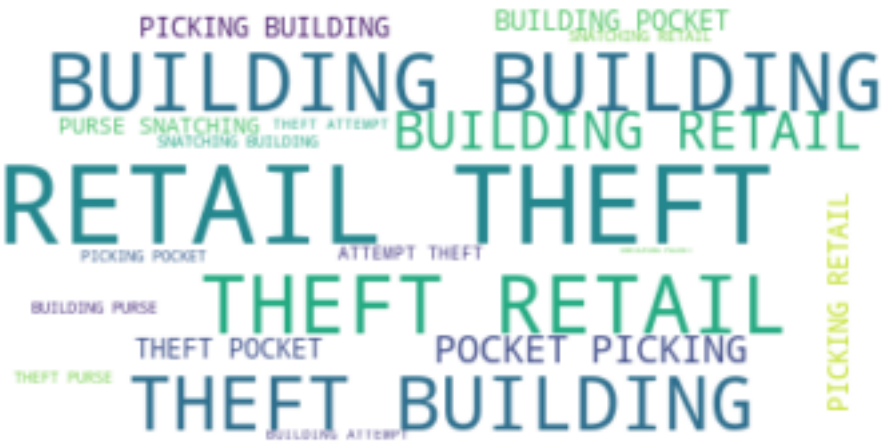
1. More than half of theft offenses involved items valued at $500 or less (petty theft)
2. Simple domestic battery is almost as prevalent as petty theft
3. Most criminal damage cases involved properties or vehicles

There are in fact 310 descriptions in total and listing them all here is not viable. However, we can use a visualization tool known as a **word cloud** to summarize the prevalent descriptions within each primary type. A word cloud visualizes the words within a collection of texts (in our case, the texts are all `Descriptions` for a specific primary type) and the size of each word is proportional to how often it appears in the datase. Below, we construct three word clouds for the top 3 most prevalent primary crime types:

[6]:
```
# wordcloud for primary type defined by rank
def wordcloud_crime( df, rank ):
    df_filter = df[df["Primary Type"]==df["Primary Type"].value_counts().
 ↪index[rank]]
    text = ' '.join(df_filter['Description'])
    wordcloud = WordCloud(max_font_size=50, max_words=100,␣
 ↪background_color="white").generate(text)
    plt.figure()
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.show()
```

[7]:
```
print("Crime type: ", df["Primary Type"].value_counts().index[0])
wordcloud_crime( df, 0 )
```

```
Crime type:  THEFT
```



From the above wordcloud, it seems that the words ”building” and ”retail” are strongly linked to theft offenses, indicating that theft likely mainly happened indoors and in malls or retail stores.

1.2.3 Exercise 2: (5 mts)

Use the code above to generate the wordcloud for battery cases and criminal damage cases. What are the most common words to describe these two types of cases?

[8]:
```
print("Crime type: ", df["Primary Type"].value_counts().index[1])
wordcloud_crime( df, 1 )
print("Crime type: ", df["Primary Type"].value_counts().index[2])
wordcloud_crime( df, 2 )
```

```
Crime type:  BATTERY
```

```
Crime type:  CRIMINAL DAMAGE
```

Answer. Battery is strongly linked the word "domestic", implying that battery charges usually involved family members. Criminal damage was strongly associated with the words "property" and "vehicle", indicating the targets for most criminal damage cases.

1.2.4 Exercise 3: (5 mts)

As we have seen with word clouds, it seems that a given type of crime is usually linked with certain types of locations (e.g. at home, in retail stores). Write code to investigate the crime patterns associated with types of crime locations. Based on the results, which types of locations are more likely to have crime?

Answer: Since `Location Description` is a discrete variable, we can use the same code as when analyzing `Primary Type`. Based on the results, we can find that street, residence, apartments and sidewalk account for around 50% of all incidents.

[8]:
```
df["Location Description"].value_counts()
```

[8]:
```
STREET                              59975
RESIDENCE                           45880
APARTMENT                           33448
SIDEWALK                            21006
OTHER                               11327
PARKING LOT/GARAGE(NON.RESID.)       8247
RESTAURANT                           6893
SMALL RETAIL STORE                   6832
RESIDENTIAL YARD (FRONT/BACK)        5311
ALLEY                                5283
VEHICLE NON-COMMERCIAL               4723
RESIDENCE PORCH/HALLWAY              4627
```

RESIDENCE-GARAGE	4587
DEPARTMENT STORE	4533
GAS STATION	3884
GROCERY FOOD STORE	3507
SCHOOL, PUBLIC, BUILDING	3412
PARK PROPERTY	2082
BAR OR TAVERN	1967
CONVENIENCE STORE	1821
CTA TRAIN	1714
COMMERCIAL / BUSINESS OFFICE	1649
HOTEL/MOTEL	1438
DRUG STORE	1270
HOSPITAL BUILDING/GROUNDS	1156
SCHOOL, PUBLIC, GROUNDS	1052
BANK	1031
CTA STATION	925
POLICE FACILITY/VEH PARKING LOT	897
VACANT LOT/LAND	871
	…
CREDIT UNION	16
VEHICLE - OTHER RIDE SHARE SERVICE (E.G., UBER, LYFT)	15
AUTO / BOAT / RV DEALERSHIP	14
FOREST PRESERVE	11
AIRPORT TERMINAL MEZZANINE - NON-SECURE AREA	11
AIRPORT TRANSPORTATION SYSTEM (ATS)	9
PARKING LOT	8
SAVINGS AND LOAN	7
NEWSSTAND	7
CHA PARKING LOT	4
STAIRWELL	3
DRIVEWAY	3
GARAGE	3
VACANT LOT	3
CHURCH	2
SCHOOL YARD	2
HALLWAY	2
GANGWAY	2
RETAIL STORE	2
TAVERN	2
BASEMENT	1
CTA PROPERTY	1
CLUB	1
VESTIBULE	1
NURSING HOME	1
RIVER BANK	1
CTA "L" PLATFORM	1
GAS STATION DRIVE/PROP.	1

```
ROOMING HOUSE                                                    1
CHA HALLWAY                                                      1
Name: Location Description, Length: 128, dtype: int64
```

Contingency tables So far, we have seen crime patterns linked with `Primary Type` and `Location Description` separately. It makes sense to see whether a certain combination of crime type and location type is prevalent or not. We know that both `Primary Type` and `Location Description` are discrete variables. We can therefore use a **contingency table** (cross table) to summarize the total number of incidents that belong to a specific combination of values of `Primary Type` and `Location Descritpion`.

We cannot use the previous code where we analyzed `Primary Type` and `Description` together since unlike those two variables, `Location Description` and `Primary Type` are **NOT** nested variables. We can use the function `crosstab` in `pandas` to generate the contingency table of two variables.

1.2.5 Exercise 4:

4.1 `crosstab(var1, var2)` generates the contingency table for `var1` vs. `var2`. Use this function to generate a contingency table for `Primary Type` vs. `Location Description` where only the top 10 most prevalent crime locations and types are included.

[16]:
```
df_1 = df[df["Location Description"].isin(df["Location Description"].
 ↪value_counts().index[:10]) & df["Primary Type"].isin(df["Primary Type"].
 ↪value_counts().index[:10])]
pd.crosstab(df_1["Primary Type"],df_1["Location Description"])
```

[16]:
```
Location Description  ALLEY  APARTMENT  OTHER  PARKING LOT/GARAGE(NON.RESID.)  \
Primary Type
ASSAULT                 485       2839    753                             499
BATTERY                1038      11706   1021                             849
BURGLARY                 18       3956    494                              40
CRIMINAL DAMAGE         434       3767    804                            1442
DECEPTIVE PRACTICE       19       1762   2261                             201
MOTOR VEHICLE THEFT     196         70    277                             861
NARCOTICS               646        613    136                             319
OTHER OFFENSE           158       2473   1355                             158
ROBBERY                 929        235    230                             383
THEFT                   629       3716   3313                            2931

Location Description  RESIDENCE  RESIDENTIAL YARD (FRONT/BACK)  RESTAURANT  \
Primary Type
ASSAULT                    3189                            495         541
BATTERY                   10136                            815         692
BURGLARY                   4170                             29         312
CRIMINAL DAMAGE            5524                            828         433
DECEPTIVE PRACTICE         6225                             20         925
```

MOTOR VEHICLE THEFT	307	299	15
NARCOTICS	730	235	40
OTHER OFFENSE	6697	146	185
ROBBERY	234	171	195
THEFT	5048	1588	3101

Location Description	SIDEWALK	SMALL RETAIL STORE	STREET
Primary Type			
ASSAULT	2189	363	3533
BATTERY	6812	321	6732
BURGLARY	6	319	49
CRIMINAL DAMAGE	251	347	9997
DECEPTIVE PRACTICE	226	601	965
MOTOR VEHICLE THEFT	29	9	8217
NARCOTICS	3264	28	3013
OTHER OFFENSE	593	175	3451
ROBBERY	3513	366	3548
THEFT	2152	4048	15801

4.2 Based on the contingency table above, what are the hot spots for the top 10 most prevalent types of crime? Are they the same or not?

Answer. No, theft is markedly different from the others. Theft is spread widely across many locations, whereas the others are concentrated in a few spots.

4.3 How can the above table help you deploy your workforce efficiently?

Answer. This analysis is most useful when we want to identify hot spots for a specific type of crime. For instance, if the police department considers eliminating theft as a priority, then the above table can tell us where we should deploy more police forces to combat theft. The table gives us the following insights:

1. Other than theft, all major types of crime tend to concentrate in fewer than five types of locations, which indicates location-specific patrolling is more efficient for combatting those types of crime.
2. Theft offenses tend to be spread across all types of locations. It is the type of crime that would require more resources to control.

1.2.6 Investigating crime by timestamp (5 mts)

We now move on to investigate the relationship between crime incidents and time; i.e. the `Date` variable we pointed out early on. Time is one of the most important dimensions for constructing an effective deployment plan. Since we cannot patrol every location 24/7, we must target periods of time with high crime rates. `Date` gives us a timestamp for each incident, which allows us to count how many incidents happened within a given period of time. Since we have one year's worth of data, we can start with monthly total incidents to see if certain months are crime-prone.

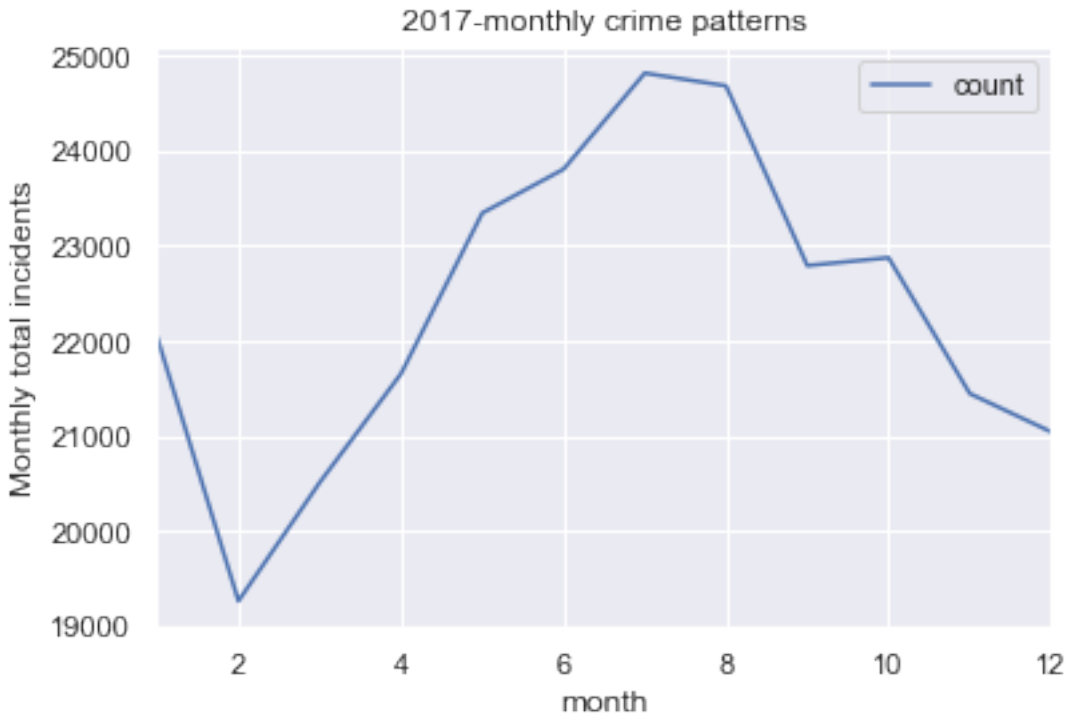
1.2.7 Exercise 5:

5.1 Convert the column `Date` into datatime type and get the month for each record. Use the calculated `month` and `groupby` function and plot the total number of cases in each month.

[15]:
```
# convert string to datatime type
df["date_py"] = pd.to_datetime(df.Date)
```

[14]:
```
def plot_time( df, time_var, title, rot = 0):
    res = df.groupby([time_var])['ID'].count().reset_index(name="count")
    p = res.plot(x = time_var, y = "count", title = title, rot = rot)
    return p
```

[13]:
```
df["month"] = df.date_py.dt.month
p_monthly = plot_time( df, "month", "2017-monthly crime patterns")
_ = plt.ylabel("Monthly total incidents")
```



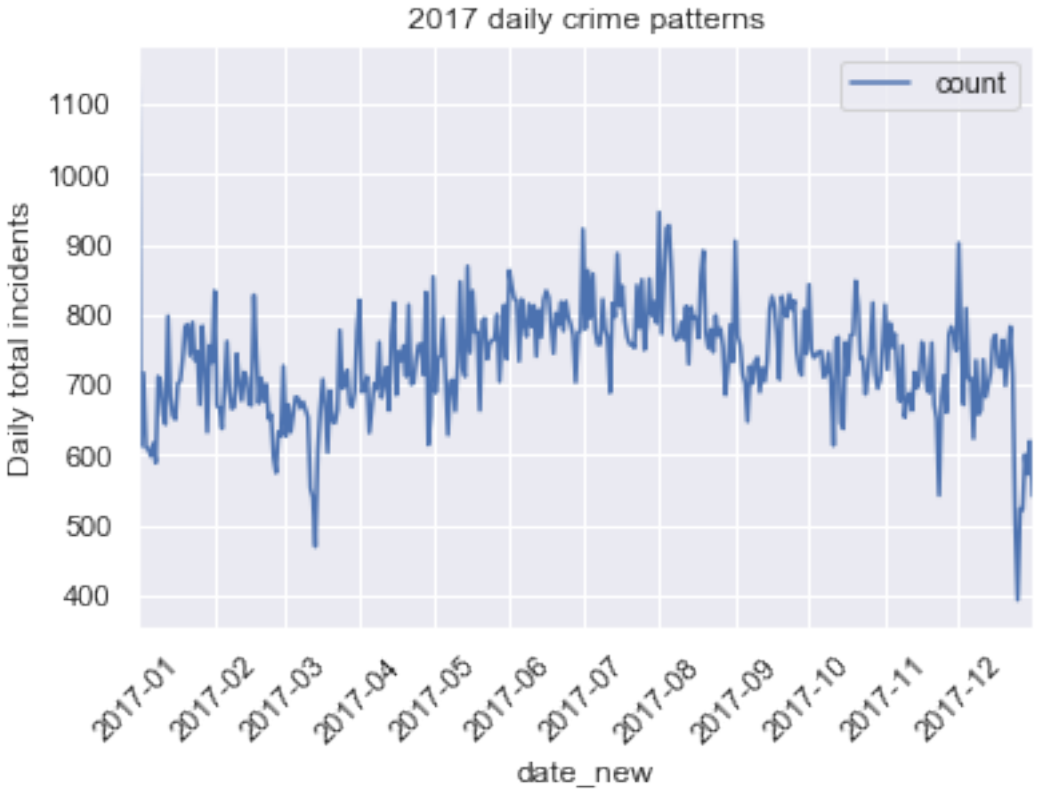
5.2 Which months have relatively higher crime rates? Why?

Answer. Feburary has the lowest number of total incidents and crime incidents peak in July. Overall, more incidents occurred during the summer. This makes sense because Chicago is cold and windy in the winter, and neither perpetrators nor victims like to be out and about much then!

5.3 Modify your code for monthly total incidents and instead plot the time series of daily total incidents throughout 2017. Do you still believe that February is the time when crime is least concerning?

Answer. One possible solution is given below:

[17]:
```
df["date_new"] = df.date_py.dt.date
# rotate the x-axis tick labels for better visualization
p_daily = plot_time( df, "date_new", "2017 daily crime patterns", rot = 45)
_ = plt.ylabel("Daily total incidents")
```

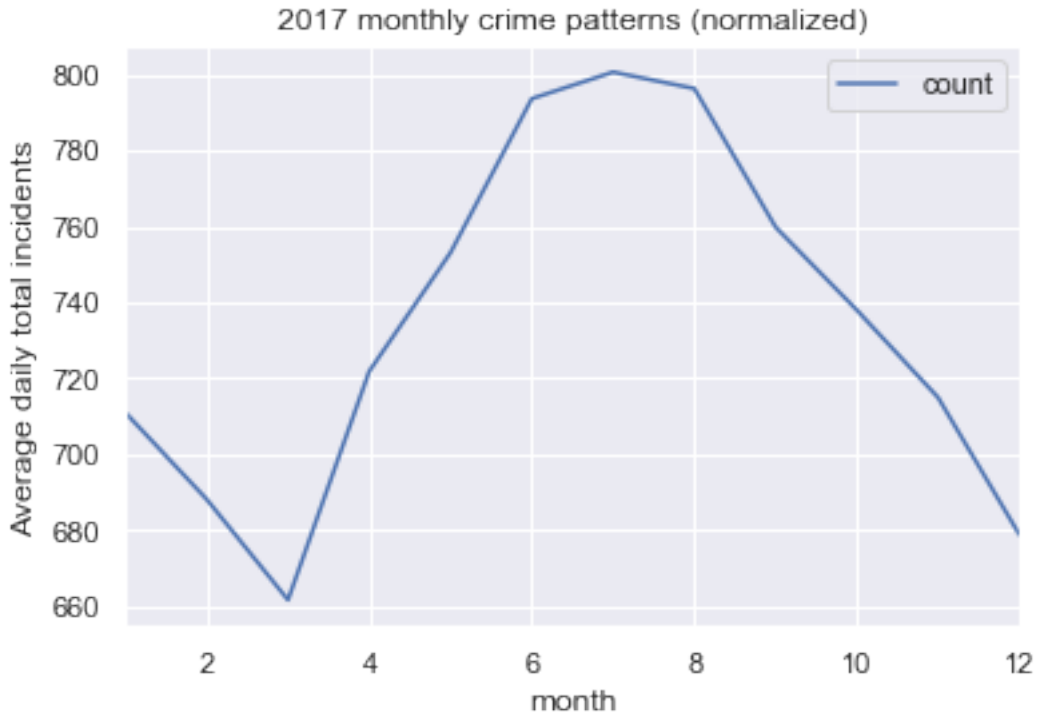


We can see that most of the days in March had fewer total incidents than most of the days in February. So why do we observe a discrepancy between the monthly chart and daily chart? The reason is simple: February has 28 days whereas March has 31 days, so the monthly total for February is likely to be the smallest.

Using normalization for fair comparison (5 mts) Therefore, if we want to compare level of crime across different months, monthly total is probably not a good metric since different months have different numbers of days. To resolve this issue, we will normalize the monthly total into some metric that does not depend on the number of days in a month. A natural choice is to divide monthly total by the number of days in a month. The normalized value is indeed the average daily incidents in a month, which can be compared across different months. Let's take a look at the

results if we use this normalization. From this, it is clear that March is in fact the least concerning month:

[18]:
```
res = df.groupby(["month"])["ID"].count().reset_index(name="count")
res["count"] = res["count"]/[31,28,31,30,31,30,31,31,30,31,30,31]
_ = res.plot(x = "month", y = "count", title = "2017 monthly crime patterns␣
 ↪(normalized)")
_ = plt.ylabel("Average daily total incidents")
```

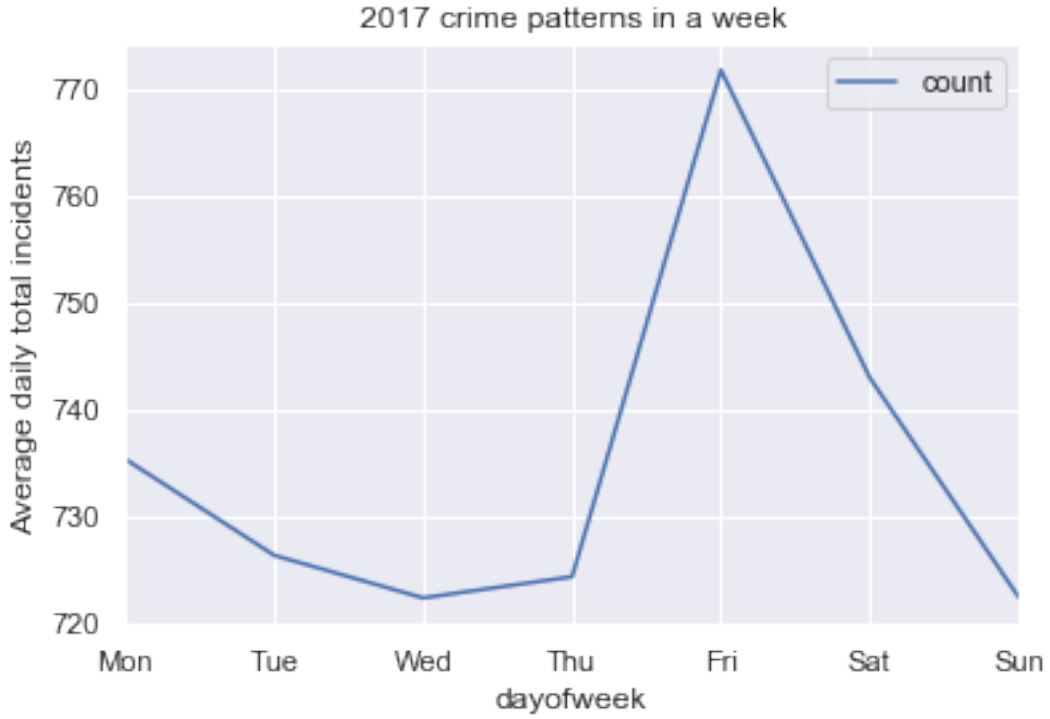


1.2.8 Exercise 6: (10 mts)

Choose the correct normalization approach and modify the code above to visualize the crime patterns in every day of a week. Which day in a week has the largest amount of cases?

Answer. We can visualize the total number of incidents that happened in a specific day of the week and plot these counts across all days of the week to examine the patterns we are looking for. But as in the case of monthly pattern, we need to normalize these raw counts since for example, there is a different number of Mondays and Sundays in 2017. For a given day of the week, we can divide the total number of incidents that happened on that day of the week by the total number of that day of the week in 2017. The result is interpreted as the average daily total incidents. Note this average is across the whole year where the average we considered in the monthly trend case is only over a given month.

[19]:
```
df["dayofweek"] = df.date_py.dt.dayofweek.astype("category")
df.dayofweek = df.dayofweek.cat.
 ↪rename_categories(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
res = df.groupby(["dayofweek"])['ID'].count().reset_index(name="count")
res['count'] = res['count']/pd.date_range("2017-1-1","2017-12-31",freq="D").
 ↪dayofweek.value_counts()[::-1]
_ = res.plot(x="dayofweek", y = "count", title = "2017 crime patterns in a␣
 ↪week")
_ = plt.ylabel("Average daily total incidents")
```



We find that Friday has significantly higher total incidents compared to all other days of week. The lowest total incidents is reached on Wednesday.

1.2.9 Investigating crime by geographic location

Another important dimension we need to consider is the relationship between crime incidents and geographic location. We have the rough geographic coordinate of each incident and based on these, we can explore the geographic patterns of crime in Chicago. To identify geographic hot spots of crimes, we can partition the City of Chicago into non-overlapping regions and count the total number of cases in 2017 in each region. In this case, we divide Chicago by police beats. We then visualize the results on the map:

[20]:
```
# format the beat variable to have leading zeros, count by beat
df["beat_num"] = df["beat_num"].str.zfill(4)
beat_cn = df.groupby("beat_num")["ID"].count().reset_index(name="crime_count")

# color scheme
min_cn, max_cn = beat_cn['crime_count'].quantile([0.01,0.99]).apply(round, 2)

colormap = branca.colormap.LinearColormap(
    colors=['white','yellow','orange','red','darkred'],
    #index=beat_cn['count'].quantile([0.2,0.4,0.6,0.8]),b
    vmin=min_cn,
    vmax=max_cn
)

colormap.caption="Total crimes in Chicago by police beats"
```

[21]:
```
# load the shape file for Chicago police beats
beat_orig = geopandas.read_file("Boundaries_beat.geojson", driver = "GeoJSON")
beat_data = beat_orig.join(beat_cn.set_index("beat_num"), how = "left", on =␣
 ↪"beat_num")
beat_data.fillna(0, inplace = True)
```

```
      ␣
↪---------------------------------------------------------------------------

       NotImplementedError                       Traceback (most recent call␣
↪last)

       <ipython-input-21-c688d8d56527> in <module>
         2 beat_orig = geopandas.read_file("Boundaries_beat.geojson", driver =␣
↪"GeoJSON")
         3 beat_data = beat_orig.join(beat_cn.set_index("beat_num"), how =␣
↪"left", on = "beat_num")
    ----> 4 beat_data.fillna(0, inplace = True)


       ~/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in␣
↪fillna(self, value, method, axis, inplace, limit, downcast, **kwargs)
      4032                       self).fillna(value=value, method=method,␣
↪axis=axis,
      4033                                    inplace=inplace, limit=limit,
    -> 4034                                    downcast=downcast, **kwargs)
      4035
      4036     @Appender(_shared_docs['replace'] % _shared_doc_kwargs)
```

```
      ~/anaconda3/lib/python3.7/site-packages/pandas/core/generic.py in␣
↪fillna(self, value, method, axis, inplace, limit, downcast)
     6121                 new_data = self._data.fillna(value=value,␣
↪limit=limit,
     6122                                              inplace=inplace,
   -> 6123                                              downcast=downcast)
     6124             elif isinstance(value, DataFrame) and self.ndim == 2:
     6125                 new_data = self.where(self.notna(), value)


      ~/anaconda3/lib/python3.7/site-packages/pandas/core/internals/managers.
↪py in fillna(self, **kwargs)
      523
      524     def fillna(self, **kwargs):
  --> 525         return self.apply('fillna', **kwargs)
      526
      527     def downcast(self, **kwargs):


      ~/anaconda3/lib/python3.7/site-packages/pandas/core/internals/managers.
↪py in apply(self, f, axes, filter, do_integrity_check, consolidate, **kwargs)
      393                                               copy=align_copy)
      394
  --> 395             applied = getattr(b, f)(**kwargs)
      396             result_blocks = _extend_blocks(applied, result_blocks)
      397


      ~/anaconda3/lib/python3.7/site-packages/pandas/core/internals/blocks.py␣
↪in fillna(self, value, limit, inplace, downcast)
     1832     def fillna(self, value, limit=None, inplace=False,␣
↪downcast=None):
     1833         values = self.values if inplace else self.values.copy()
   -> 1834         values = values.fillna(value=value, limit=limit)
     1835         return [self.make_block_same_class(values=values,
     1836                                            placement=self.mgr_locs,


      ~/anaconda3/lib/python3.7/site-packages/geopandas/array.py in␣
↪fillna(self, value, method, limit)
      831         elif not isinstance(value, BaseGeometry):
      832             raise NotImplementedError(
  --> 833                 "fillna currently only supports filling with a␣
↪scalar geometry"
      834             )
      835
```

```
        NotImplementedError: fillna currently only supports filling with a␣
↪scalar geometry
```

[22]:
```
# interactive visualization for beat-specific crime rate in 2017

m_crime = folium.Map(location=[41.88, -87.63],
                     zoom_start=12,
                     tiles="OpenStreetMap")
style_function = lambda x: {
    'fillColor': colormap(x['properties']['crime_count']),
    'color': 'black',
    'weight':2,
    'fillOpacity':0.5
}

stategeo = folium.GeoJson(
    beat_data.to_json(),
    name='Chicago beats',
    style_function=style_function,
    tooltip=folium.GeoJsonTooltip(
        fields=['beat_num', 'crime_count'],
        aliases=['Beat', 'Total crime'],
        localize=True
    )
).add_to(m_crime)

colormap.add_to(m_crime)
m_crime
```

[22]:
```
<folium.folium.Map at 0x1a23ae10f0>
```

Overall, we find there are three hot spots in Chicago: Downtown Chicago, West Chicago, and South Chicago. You can hover over each region to see the beat number and the total number of crimes in the beat.

1.2.10 Exercise 7 (15 mts):

7.1 (5 mts) Based on the analysis so far, what are your preliminary strategies for police deployment based on times, locations and types of crimes? What is the potential business problem you are solving here?

Answer: Based on the results, we have several suggestions: put more police force to work on Fridays, between May and September, and place them in Downtown Chicago as well as the West and South Sides of Chicago. They probably need to focus on streets and residence areas and should pay attention to theft, battery, criminal damage, and assault.

7.2 (10 mts) What is the main shortcoming of our analysis and recommendations in 4.1?

Answer: We see two main pitfalls:

1. The above investigations are for individual patterns associated with each individual variable, and the patterns associated with all these independent variables combined might not be independent. In other words, similar to the previous EDA case, there may exist interaction effects among these variables.
2. We noticed above that most crimes tend to be theft-related. However, most theft offenses are petty, which might not be the type of crime we should focus our limited police force on. More severe crimes, such as homicide cases, are overlooked in the above analyses since they are very rare. But perhaps these rare and severe crimes are the most important offenses to prevent.

The rest of this case study aims to tackle these two shortcomings.

1.3 Investigating joint distributions and interactions

In Exercise 7.2, we cited two potential problems with naively tacking together the patterns we noticed for each individual variable into a recommendation. We tackle the first issue to start: there may exist interaction effects among the variables of interest. Similar to the previous EDA case, we now investigate each potential interaction effect in more detail.

1.3.1 Crime type vs. day of the week (5 mts)

Again, we can use a contingency table to answer this question just like we did for `Primary Type` and `Location Description`. One catch here is that you need to normalize the data so that the comparisons are fair across different days of the week:

[36]:
```
res_raw = pd.crosstab(df["Primary Type"], df.dayofweek)
res_raw/pd.date_range("2017-1-1","2017-12-31",freq="D").dayofweek.
 ↪value_counts().tolist()[::-1]
```

[36]:

dayofweek	Mon	Tue	Wed	\
Primary Type				
ARSON	1.346154	1.173077	1.403846	
ASSAULT	54.230769	53.942308	55.153846	
BATTERY	129.884615	125.769231	123.596154	
BURGLARY	37.076923	36.442308	38.788462	
CONCEALED CARRY LICENSE VIOLATION	0.192308	0.153846	0.115385	
CRIM SEXUAL ASSAULT	4.057692	3.634615	4.057692	
CRIMINAL DAMAGE	80.057692	76.634615	73.423077	
CRIMINAL TRESPASS	19.423077	18.519231	18.596154	
DECEPTIVE PRACTICE	54.153846	52.403846	52.057692	
GAMBLING	0.519231	0.480769	0.500000	
HOMICIDE	1.942308	1.519231	1.846154	
HUMAN TRAFFICKING	0.000000	0.038462	0.038462	

INTERFERENCE WITH PUBLIC OFFICER	3.307692	2.750000	2.961538
INTIMIDATION	0.423077	0.442308	0.576923
KIDNAPPING	0.403846	0.519231	0.442308
LIQUOR LAW VIOLATION	0.288462	0.326923	0.403846
MOTOR VEHICLE THEFT	31.403846	29.500000	30.000000
NARCOTICS	31.000000	30.769231	32.000000
NON-CRIMINAL	0.096154	0.115385	0.076923
NON-CRIMINAL (SUBJECT SPECIFIED)	0.019231	0.000000	0.000000
OBSCENITY	0.173077	0.211538	0.384615
OFFENSE INVOLVING CHILDREN	5.557692	5.884615	5.673077
OTHER NARCOTIC VIOLATION	0.038462	0.057692	0.000000
OTHER OFFENSE	47.788462	49.730769	49.115385
PROSTITUTION	0.384615	2.750000	2.615385
PUBLIC INDECENCY	0.038462	0.057692	0.019231
PUBLIC PEACE VIOLATION	3.826923	4.538462	4.000000
ROBBERY	33.211538	30.942308	30.807692
SEX OFFENSE	2.942308	2.673077	2.711538
STALKING	0.403846	0.711538	0.442308
THEFT	176.250000	177.038462	175.096154
WEAPONS VIOLATION	11.673077	12.807692	11.692308

dayofweek	Thu	Fri	Sat	\
Primary Type				
ARSON	1.230769	1.115385	1.038462	
ASSAULT	53.730769	53.403846	49.076923	
BATTERY	123.519231	129.942308	148.115385	
BURGLARY	36.019231	41.903846	30.961538	
CONCEALED CARRY LICENSE VIOLATION	0.230769	0.192308	0.250000	
CRIM SEXUAL ASSAULT	3.403846	3.923077	5.538462	
CRIMINAL DAMAGE	74.634615	78.269231	86.480769	
CRIMINAL TRESPASS	20.000000	19.269231	18.096154	
DECEPTIVE PRACTICE	53.057692	56.423077	44.576923	
GAMBLING	0.461538	0.692308	0.519231	
HOMICIDE	1.634615	1.846154	1.615385	
HUMAN TRAFFICKING	0.019231	0.019231	0.038462	
INTERFERENCE WITH PUBLIC OFFICER	2.730769	3.019231	2.980769	
INTIMIDATION	0.576923	0.250000	0.307692	
KIDNAPPING	0.538462	0.615385	0.442308	
LIQUOR LAW VIOLATION	0.634615	0.538462	0.961538	
MOTOR VEHICLE THEFT	29.480769	33.923077	33.423077	
NARCOTICS	33.461538	38.769231	30.557692	
NON-CRIMINAL	0.057692	0.211538	0.057692	
NON-CRIMINAL (SUBJECT SPECIFIED)	0.019231	0.000000	0.000000	
OBSCENITY	0.250000	0.211538	0.173077	
OFFENSE INVOLVING CHILDREN	5.480769	6.923077	5.865385	
OTHER NARCOTIC VIOLATION	0.038462	0.000000	0.019231	
OTHER OFFENSE	47.269231	49.384615	43.711538	

PROSTITUTION	2.923077	2.038462	1.769231
PUBLIC INDECENCY	0.019231	0.000000	0.038462
PUBLIC PEACE VIOLATION	3.846154	4.653846	4.153846
ROBBERY	31.230769	32.634615	33.980769
SEX OFFENSE	2.596154	2.923077	2.769231
STALKING	0.557692	0.576923	0.461538
THEFT	179.634615	190.038462	178.557692
WEAPONS VIOLATION	11.884615	14.865385	14.346154

dayofweek	Sun
Primary Type	
ARSON	1.207547
ASSAULT	50.716981
BATTERY	162.547170
BURGLARY	28.264151
CONCEALED CARRY LICENSE VIOLATION	0.188679
CRIM SEXUAL ASSAULT	6.622642
CRIMINAL DAMAGE	87.339623
CRIMINAL TRESPASS	16.830189
DECEPTIVE PRACTICE	34.377358
GAMBLING	0.490566
HOMICIDE	2.528302
HUMAN TRAFFICKING	0.000000
INTERFERENCE WITH PUBLIC OFFICER	3.094340
INTIMIDATION	0.320755
KIDNAPPING	0.679245
LIQUOR LAW VIOLATION	0.509434
MOTOR VEHICLE THEFT	30.622642
NARCOTICS	27.132075
NON-CRIMINAL	0.094340
NON-CRIMINAL (SUBJECT SPECIFIED)	0.000000
OBSCENITY	0.245283
OFFENSE INVOLVING CHILDREN	8.339623
OTHER NARCOTIC VIOLATION	0.056604
OTHER OFFENSE	43.603774
PROSTITUTION	1.622642
PUBLIC INDECENCY	0.018868
PUBLIC PEACE VIOLATION	3.716981
ROBBERY	34.962264
SEX OFFENSE	3.150943
STALKING	0.490566
THEFT	157.924528
WEAPONS VIOLATION	12.603774

1.3.2 Exercise 8: (5 mts)

Your colleague claims that most thefts happen on Mondays, Tuesdays, or Wednesdays. How do you validate or disprove their claim based on the data and the table above? What can you say about the days which have the most battery and assault incidents?

Answer. From the results we see that Friday has the most theft cases (~10% more than other days) which disproves the first claim. Battery and assault have different patterns. Assault cases are more prevalent on weekdays than weekends (~10% more) while battery cases are more prevalent on weekends (~15% more). So deploying more police forces on Friday might not work for eliminating battery offenses.

1.3.3 Crime time vs. location of the crime (10 mts)

The next potential interaction is between crime time and crime location. The geographic hot spots might shift from time to time and targeting different regions at different times is a natural strategy to increase efficiency. The following map shows how the crime rate varies geographically over time. Here, the outcome of interest is average daily total incidents:

[23]:
```
def folium_slider( beat_cn, beat_orig, tmp_drange, index_var, index_lab,
                   value_var = "crime_count", caption = "Crimes in Chicago" ):
    # get colorbar
    min_cn, max_cn = beat_cn[value_var].quantile([0.01,0.99]).apply(round, 2)
    colormap = branca.colormap.LinearColormap(
        colors=['white','yellow','orange','red','darkred'],
        #index=beat_cn['count'].quantile([0.2,0.4,0.6,0.8]),
        vmin=min_cn,
        vmax=max_cn
    )
    colormap.caption=caption

    # get styledata for folium
    styledata = {}

    for beat in range(beat_orig.shape[0]):
        res_beat = beat_cn[beat_cn.beat_num==beat_orig.iloc[beat,:].beat_num]
        #fill missing value by zero: no recorded crime that month
        c_count = res_beat.set_index(index_var)[value_var].reindex(tmp_drange).
↪fillna(0)
        df_tmp = pd.DataFrame(
            {'color': [colormap(count) for count in c_count], 'opacity':0.5},
            index = index_lab
        )
        styledata[str(beat)] = df_tmp

    styledict = {
        str(beat): data.to_dict(orient='index') for
```

```
        beat, data in styledata.items()
    }

    # plot map and time slider
    m = folium.Map(location=[41.88, -87.63],
                       zoom_start=12,
                       tiles="OpenStreetMap")

    g = TimeSliderChoropleth(
        beat_orig.to_json(),
        styledict=styledict
    ).add_to(m)

    folium.GeoJson(beat_orig.to_json(), style_function = lambda x: {
        'color': 'black',
        'weight':2,
        'fillOpacity':0
    }, tooltip=folium.GeoJsonTooltip(
            fields=['beat_num'],
            aliases=['Beat'],
            localize=True
        )).add_to(m)

    colormap.add_to(m)

    return m
```

[24]:
```
# cycle in a year
beat_cn_month = df.groupby(["beat_num","month"])["ID"].count().reset_index(name␣
↪= "crime_count")
nd = pd.DataFrame({"month":range(1,13), "days":
↪[31,28,31,30,31,30,31,31,30,31,30,31]})
beat_cn_month = beat_cn_month.merge(nd, how = "left", on = "month")
beat_cn_month["crime_count"] = beat_cn_month["crime_count"]/
↪beat_cn_month["days"]
folium_slider( beat_cn_month, beat_orig, list(range(1,13)), "month",
               list(pd.date_range( "2017-1", "2017-12", freq = "MS").
↪strftime("%Y-%m")),
               caption = "Average daily total incidents in a month")
```

[24]:
```
<folium.folium.Map at 0x1a22539438>
```

1.3.4 Exercise 9: (10 mts)

From the plot above, what patterns do you observe over time in Downtown Chicago, as well as the West and South Sides of Chicago? Based on this, do we need to refine the strategies we outlined

in Exercise 7.1?

Answer. It is clear that Downtown Chicago remains a hot spot no matter which month we are looking at. Beat 0114 is a special region. It has an elevated crime rate only in July and August, probably due to tourists.

West Side Chicago requires particular attention from April to October. Beat 1011 in this area is crime-prone year-round.

South Side Chicago has a hot period from April to August. Beat 0511 in this area has a constantly high crime rate year-round while Beats 0833 and 0834 only have high crime rates in January.

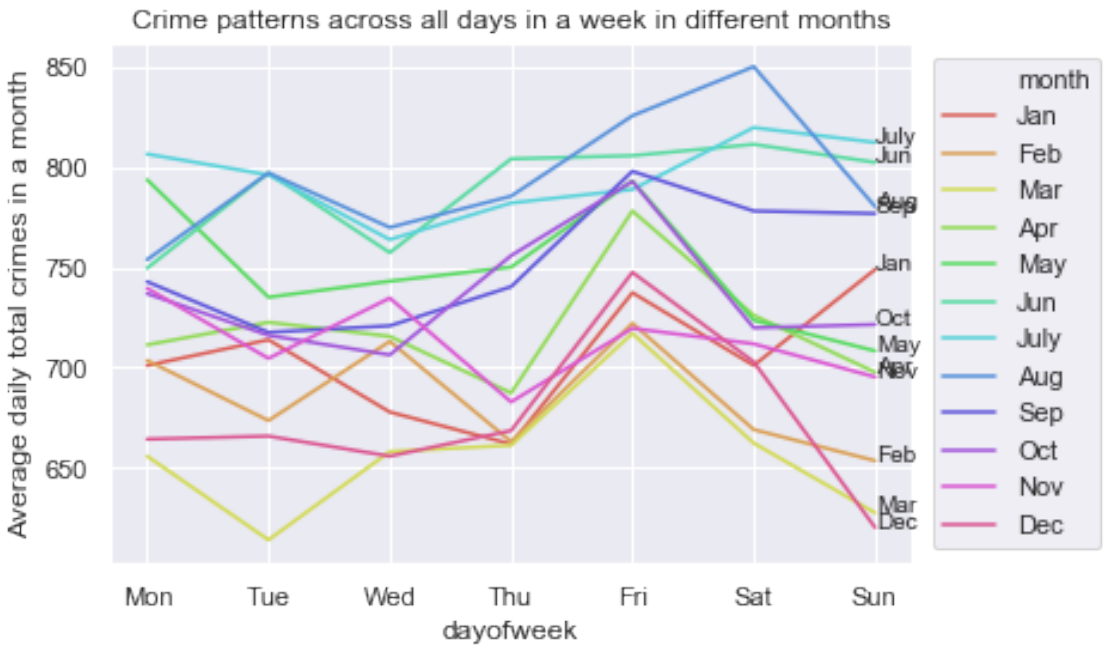
As we can see, the strategies we developed in Exercise 7.1 need refinement. The above observations tell us that we need to deploy in different areas at different times of the year. May to August is a hot period for most of the regions we are concerned about but a few beats are swamped with criminal activity year-round.

Exercise 10: Our strategies developed in the previous section suggest that we should pay attention to Fridays as well as the months from May to August. However, is Friday always the most crime-prevalent day of the week regardless of the month we are looking at? Using the code we have developed above, we plot the crime patterns in a week for every month in 2017. Note that normalization is still required here. Based on the results, is our strategy in Exercise 7.1 to focus on Friday valid?

[40]:
```
res_md = df.groupby(['dayofweek','month'])['ID'].count().
 ↪reset_index(name="count")
# normalization
date_2017 = pd.DataFrame(
    {"dayofweek": pd.date_range("2017-1-1","2017-12-31",freq="D").dayofweek.
 ↪astype("category"),
     "month": pd.date_range("2017-1-1","2017-12-31",freq="D").month } )
date_2017["dayofweek"] = date_2017["dayofweek"].cat.
 ↪rename_categories(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
nd_2017 = date_2017.groupby(['month'])['dayofweek'].value_counts().sort_index().
 ↪reset_index(name="day_count")
res_md_norm = nd_2017.merge( res_md, how = "left", on = ["month","dayofweek"]).
 ↪fillna(0)
res_md_norm['count_norm'] = res_md_norm['count']/res_md_norm['day_count']

res_md_norm['dayofweek'] = res_md_norm['dayofweek'].astype("category").cat.
 ↪reorder_categories(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
res_md_norm['month'] = res_md_norm['month'].astype('category').cat.
 ↪rename_categories(["Jan","Feb","Mar","Apr","May","Jun","July","Aug","Sep","Oct","Nov","Dec"]
mp = sns.lineplot(data=res_md_norm, x='dayofweek', hue = 'month',␣
 ↪y='count_norm',
                  palette = sns.color_palette("hls",12))
mp = mp.legend(loc='center left', bbox_to_anchor=(1.01, 0.5), ncol=1)
_ = plt.ylabel("Average daily total crimes in a month")
```

```
_ = plt.title("Crime patterns across all days in a week in different months")
for i in range(12):
    tmp = res_md_norm[res_md_norm.dayofweek=="Sun"]
    _ = plt.text( 6, tmp['count_norm'].iloc[i], tmp['month'].iloc[i])
```



Answer. We can immediately spot that **not all months have the same day-of-the-week patterns**. Friday stands out as the crime rate peak in a week only in the non-summer months. In August, the peak shifts to Saturday. In June, there seems to be no apparent peak across a week. The results here tell us that our "Friday strategy" is probably not going to work during the summer, which unfortunately also happens to be the time of year with the highest crime rate overall.

This section has only explored three pairs of potential interactions across all pairs of variables we have. There are of course many other combinations that are important and will lead to further refinement of our deployment plan. We should be able to identify patterns associated with any pair of variables using the tools we developed here. After taking any interactions we find into account, we can then propose an actionable plan.

1.4 Prioritizing by the severity of the crime: IUCR score

So far, we have based our analysis on just the total crime rate. But not all crimes are equally harmful. A homicide case would severely affect a neighborhood even after several years and hinder business development in the area. On the other hand, a case of petty theft is usually not as destructive and would be dismissed after several weeks.

We can define a different type of outcome which emphasizes crime types that need to be controlled

to the minimal level. These crimes are generally determined by municipal development plans of Chicago. For example, if the government aims to promote tourism, crimes targeted at tourists, such as theft and deceptive activities should be the main focus of the police department.

In our dataset, the column `IUCR` is a reporting code that partially measures the damage of an incident to the general well-being of the public. Let's use this code to define a new type of outcome which roughly measures the accumulated damage of all crimes in an area over a period of time. The key idea is that we should focus on places and times that are harmed by crimes the most, not necessarily the ones with the highest total crime incidents.

1.4.1 Crime severity: IUCR score (15 mts)

`IUCR` column stands for Illinois Uniform Crime Reporting code, and we can find a full key here. The most important take-away is that **as code number increases, the severity of associated crimes generally decreases**.

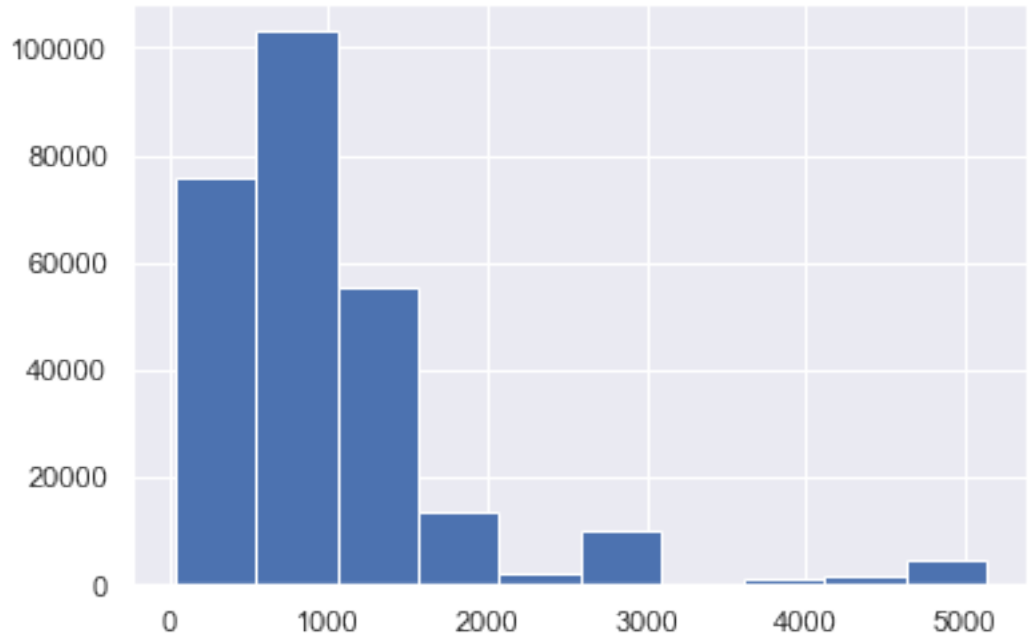
[17]:
```
df['IUCR'].head()
```

[17]:
```
0     630
1     460
2     890
3    1320
4    041A
Name: IUCR, dtype: object
```

1.4.2 Exercise 11: (5 mts)

Some of the IUCR codes have a letter after them (A, P, B, R, C, T, N). An examination of the above link shows that these letters are about convictions, which isn't too relevant to us. Write code to remove the letters and convert this column to a numeric type. Then use function `hist` to visualize the distribution of `IUCR`. Also use function `describe` to summarize `IUCR` (mean, variance, etc). Based on the histogram, do most cases have large IUCR (not severe)?

[18]:
```
# This code processes the IUCR column to be truly numeric
iucr = (df['IUCR'].str.replace("A", "")
                  .str.replace("P","")
                  .str.replace("B","")
                  .str.replace("R","")
                  .str.replace("C","")
                  .str.replace("T","")
                  .str.replace("N","")
                  .str.replace("E","")
                  .str.replace("H","")
                  .astype('int'))
_ = iucr.hist()
```



[42]:
```
iucr.describe()
```

[42]:
```
count    267178.000000
mean       1017.423085
std         839.350262
min          31.000000
25%         486.000000
50%         820.000000
75%        1310.000000
max        5132.000000
Name: IUCR, dtype: float64
```

Answer. We can see that most cases have a raw IUCR score smaller than 2000 and around 30% of cases have a raw IUCR score smaller than 500. This means that most cases in our dataset are considered to be moderately severe.

1.4.3 Exercise 12: (5 mts)

Given these IUCR scores, how would you integrate them into our crime incidence visualizations so that the final results also represent the severity of crimes in a region?

Answer. In all our analysis above, each case is counted as one towards the total number of incidents. We can amplify the effect of certain types of crimes (e.g. homicide) with a predefined scaling factor, which represents that a single homicide case should be considered just as severe as many misdemeanors such as petty theft. By incorporating these scaling factors, regions with fewer but more severe cases would be identified as more dangerous compared to the results from the

unweighted visualization. A rule of thumb is to choose the scaling factors to be positive and higher for more severe crimes.

Using the scoring scheme we discussed above, let's consider the monthly crime severity across police beats:

[43]:
```
# weighted incidence rate
iucr = iucr.fillna(0)
df["IUCR_num"] = iucr.max() - iucr
beat_cs_month = df.groupby(["beat_num","month"]).aggregate({"IUCR_num":lambda x:
 ↪ sum(x)}).reset_index()
nd = pd.DataFrame({"month":range(1,13), "days":
 ↪[31,28,31,30,31,30,31,31,30,31,30,31]})
beat_cs_month = beat_cs_month.merge(nd, how = "left", on = "month")
beat_cs_month["severity_tot"] = beat_cs_month["IUCR_num"]/beat_cs_month["days"]
folium_slider( beat_cs_month, beat_orig, list(range(1,13)), "month",
              list(pd.date_range( "2017-1", "2017-12", freq = "MS").
 ↪strftime("%Y-%m")),
              value_var = "severity_tot", caption = "Average daily crime␣
 ↪severity in Chicago")
```

[43]:
```
<folium.folium.Map at 0x23710ad6fd0>
```

It seems that the results here are not significantly different from the results we obtained when we did not incorporate `IUCR`. This might be because most cases have a relatively small raw IUCR and therefore tend to be relatively equally weighted even when we do incorporate `IUCR`.

1.4.4 Defining a severity score based on the bigger-picture business problem

Since `IUCR` was not particularly useful, let's define a different severity metric. Generally, it is good practice to define a metric that aligns with the bigger-picture goal that the city is trying to achieve. For example, if we want to attract more large companies to open up branches in Chicago, we may care a lot about homicide, sexual assault, and arson, but less about gambling, obscenity, and theft. This type of analysis is rather subjective, but with experience in the field we should gain good intuition about how every type of crime should be bucketed. If we want to be more objective, we could include another dataset of crimes which have been classified by the dollar amount in losses caused by those crimes. We start with a subjective scoring system as an example:

[44]:
```
# The zip function creates a list of tuples out of two lists
# The first element of each tuple is the crime type from the first list, and␣
 ↪the second element is the severity number

severity_10 = zip(['CRIM_SEXUAL_ASSAULT', 'ARSON', 'HOMICIDE'], [10] * 4)
severity_9 = zip(['BATTERY', 'ASSAULT','ROBBERY', 'BURGLARY'], [9] * 4)
severity_8 = zip(['MOTOR VEHICLE THEFT', 'PUBLIC PEACE VIOLATION',  'CRIMINAL␣
 ↪DAMAGE'], [8] * 3)
```

```
severity_7 = zip(['CRIMINAL TRESPASS', 'OFFENSE INVOLVING CHILDREN',␣
 ↪'KIDNAPPING'], [7] * 3)
severity_6 = zip(['STALKING', 'PUBLIC INDECENCY'], [6] * 2)
severity_5 = zip(['OTHER OFFENSE', 'HUMAN TRAFFICKING'], [5] * 2)
severity_4 = zip(['DECEPTIVE PRACTICE', 'INTIMIDATION'], [4] * 2)
severity_3 = zip(['INTERFERENCE WITH PUBLIC OFFICER', 'SEX OFFENSE'], [3] * 2)
severity_2 = zip(['NARCOTICS', 'WEAPONS VIOLATION', 'CONCEALED CARRY LICENSE␣
 ↪VIOLATION','OBSCENITY'], [2] * 4)
severity_1 = zip(['THEFT','GAMBLING', 'PROSTITUTION', 'LIQUOR LAW VIOLATION',␣
 ↪'NON-CRIMINAL (SUBJECT SPECIFIED)', 'NON-CRIMINAL'], [1] * 5)

# By turning these zipped tuples into a dictionary, we can map each row of the␣
 ↪dataset to its severity label
severities = dict()
for s in [severity_1, severity_2, severity_3, severity_4, severity_5,␣
 ↪severity_6, severity_7, severity_8, severity_9, severity_10]:
    severities.update(dict(s))

# Our last step is mapping primary_type using this dictionary
df['severity_bus'] = df['Primary Type'].apply(severities.get)
```

This weighting scheme is suitable for visualizing average per case severity but not for the total severity. The reason is that the range of the weights is rather small (1-10) and crimes with high weights are rare. As a result, the weighted sum given by this weighting scheme is very similar to the unweighted sum and visualizing the weighted sum does not provide additional information. The average per-case severity on the other hand is bounded between 1 and 10. So a small change (e.g. 0.1) in the average can still lead to apparent change in the filling color. Let's take a look at the updated average per case severity score:

[45]:
```
# note that we still need to do normalization by days
beat_cs_month = df.groupby(["beat_num","month"]).aggregate({"severity_bus":
 ↪lambda x: sum(x)}).reset_index()
nd = pd.DataFrame({"month":range(1,13), "days":
 ↪[31,28,31,30,31,30,31,31,30,31,30,31]})
beat_cs_month = beat_cs_month.merge(nd, how = "left", on = "month")
beat_cs_month["severity_bus_tot"] = beat_cs_month["severity_bus"]/
 ↪beat_cs_month["days"]
beat_cs_month["severity_bus_avg"] = beat_cs_month["severity_bus_tot"]/
 ↪beat_cn_month["crime_count"]
folium_slider( beat_cs_month, beat_orig, list(range(1,13)), "month",
              list(pd.date_range( "2017-1", "2017-12", freq = "MS").
 ↪strftime("%Y-%m")),
              value_var = "severity_bus_avg", caption = "Average crime severity␣
 ↪in Chicago")
```

[45]:
```
<folium.folium.Map at 0x237125e4160>
```

This map essentially shows the violent parts of Chicago, ignoring theft and putting emphasis on the most violent crimes. Under this prioritization scheme, we see that Downtown Chicago is no longer the worst region; rather, the worst region has now become South Side Chicago. The ”safe” North Side is now dotted with some rather violent beats.

1.5 Conclusions (10 mts)

We explored Chicago crime records in 2017 to understand crime patterns and proposed preliminary policy deployment strategies based on these patterns. We initially examined the patterns associated with each variable of interest independently. Based on the single variable analyses, we proposed that the police department should put extra forces to work on Fridays between May and August, pay more attention to theft, battery, assault and criminal damage, and deploy more forces in Downtown Chicago and the West and South Sides of Chicago.

We then conducted analyses for three pairs of variables and found that the above strategies are too rigid and don't take into account interaction effects. We found that Friday is a weekly hot spot outside of summertime, but during summertime, either Saturday becomes the hot spot or no hot spot was present at all. We also found that the time windows of high criminal activities for the three geographic hot spots are not the same: Downtown has a high rate throughout the year, whereas the crimes in the other two regions mostly accumulated between April and August. We found that theft is common in all kinds of locations whereas other types of crimes, such as battery and assault, tend to cluster in a very few number of location types.

In the last part of the analysis, we looked at how to customize weights of different crimes based on the business outcome we were optimizing for. We considered a custom severity score, which highlights the extremely violent cases, and found out that many regions in Chicago have low overall crime incident counts but high violent crime rates. Downtown Chicago, on the other hand, did not harbor violent crimes despite its high overall crime rate. This indicates that if eliminating highly violent crimes is the priority, the previously devised strategies should be changed and there should be more emphasis placed on locations like North Side Chicago.

Moving forward, there are many things we can do. Using this dataset, we can explore other pairwise interaction effects. We can also examine if the strategies we proposed here have already been implemented and whether the deployment plan in use now is successful or not. We can also consider more advanced statistical modeling for our dataset so that all variables can be included, and not just the ones we examined. For those that are interested in this topic, a good starting point is here.

1.6 Takeaways (5 mts)

In this case, we learned how to perform exploratory data analysis for records of crime. This type of data does not have as clear of an outcome of interest, so we started by assuming that the outcome of interest is the total number of crime incidents. From there, we followed a similar process as in the last EDA case, except:

1. When investigating variables of interest, we did not use correlation matrices because these make little sense for categorical variables.

2. Instead, we used tools such as frequency tables, word clouds, and contingency tables. We also used maps with a time slider to understand the spatial and temporal patterns.
3. We also learned the importance of normalization to ensure that you are making apples-to-apples comparisons.

Finally, we questioned our original assumption that the number of crime incidents was the most important outcome to optimize for. We also looked at how we ought to weight different crimes differently in our analysis based on the particular business problem at hand.