Pandas 資料分析 (4)

張家瑋 副教授
國立臺中科技大學資訊工程系

資料轉換與缺失值處理

```
import pandas as pd
import numpy as np
pd.set_option('max_columns', 4, 'max_rows', 10, 'max_colwidth', 12)

fueleco = pd.read_csv('data/vehicles.csv.zip')
fueleco
```

```
c:\users\test\appdata\local\programs\python\python37\lib\site-packag
umns (70,71,72,73,74,76,79) have mixed types.Specify dtype option on
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	barrels08	barrelsA08	...	phevHwy	phevComb
0	15.695714	0.0	...	0	0
1	29.964545	0.0	...	0	0
2	12.207778	0.0	...	0	0
3	29.964545	0.0	...	0	0
4	17.347895	0.0	...	0	0
...	...	...	...	...	...
39096	14.982273	0.0	...	0	0
39097	14.330870	0.0	...	0	0
39098	15.695714	0.0	...	0	0
39099	15.695714	0.0	...	0	0
39100	18.311667	0.0	...	0	0

39101 rows × 83 columns

資料轉換與缺失值處理

```
fueleco.select_dtypes(object).columns
```

```
Index(['drive', 'eng_dscr', 'fuelType', 'fuelType1', 'make', 'model',
       'mpgData', 'trany', 'VClass', 'guzzler', 'trans_dscr', 'tCharger',
       'sCharger', 'atvType', 'fuelType2', 'rangeA', 'evMotor', 'mfrCode',
       'c240Dscr', 'c240bDscr', 'createdOn', 'modifiedOn', 'startStop'],
      dtype='object')
```

```
fueleco.drive.nunique()
```

```
7
```

```
fueleco.drive.sample(5, random_state=42)
```

```
4217     4-Wheel ...
1736     4-Wheel ...
36029    Rear-Whe...
37631    Front-Wh...
1668     Rear-Whe...
Name: drive, dtype: object
```

```
fueleco.drive.isna().sum()
```

```
1189
```

```
fueleco.drive.isna().mean() * 100
```

```
3.0408429451932175
```

資料轉換與缺失值處理

```
fueleco.drive.value_counts()
```

```
Front-Wheel Drive                13653
Rear-Wheel Drive                 13284
4-Wheel or All-Wheel Drive        6648
All-Wheel Drive                   2401
4-Wheel Drive                     1221
2-Wheel Drive                      507
Part-time 4-Wheel Drive            198
Name: drive, dtype: int64
```

```
fueleco.drive.value_counts(dropna=False)
```

```
Front-Wheel Drive                13653
Rear-Wheel Drive                 13284
4-Wheel or All-Wheel Drive        6648
All-Wheel Drive                   2401
4-Wheel Drive                     1221
NaN                               1189
2-Wheel Drive                      507
Part-time 4-Wheel Drive            198
Name: drive, dtype: int64
```

#保留前六項分類

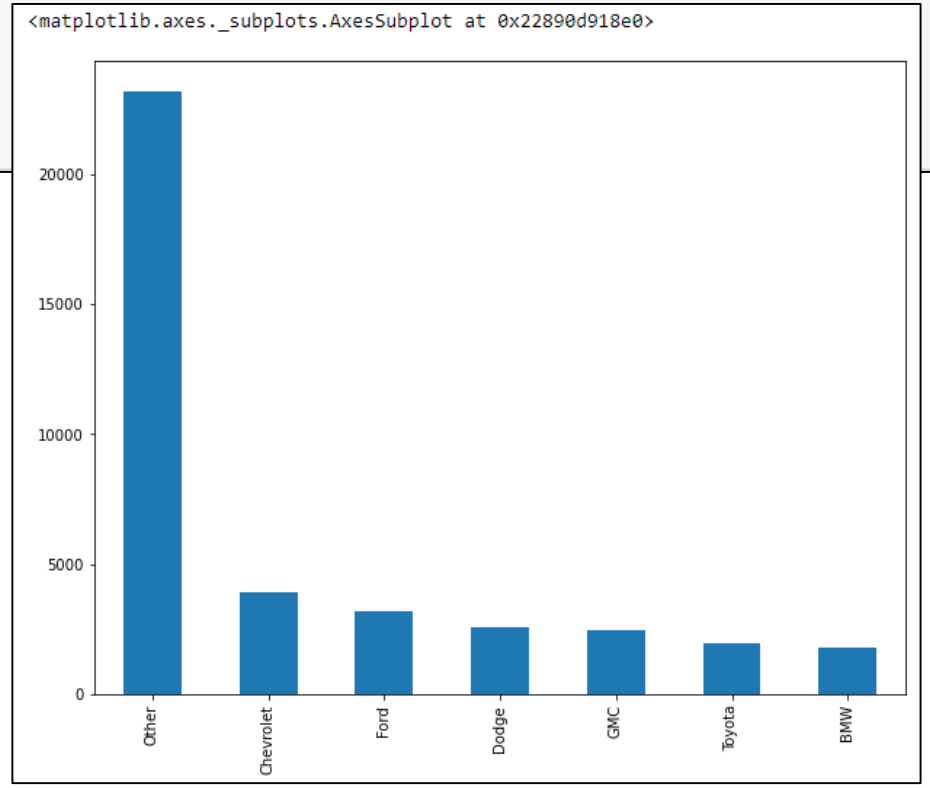
```
top_n = fueleco.make.value_counts().index[:6]
(fueleco
   .assign(make=fueleco.make.where(
              fueleco.make.isin(top_n), 'Other'))
   .make
   .value_counts()
)
```

#使用 where()更改 make 欄的內容：若不是前六項分類則改為 Other

```
Other        23211
Chevrolet     3900
Ford          3208
Dodge         2557
GMC           2442
Toyota        1976
BMW           1807
Name: make, dtype: int64
```

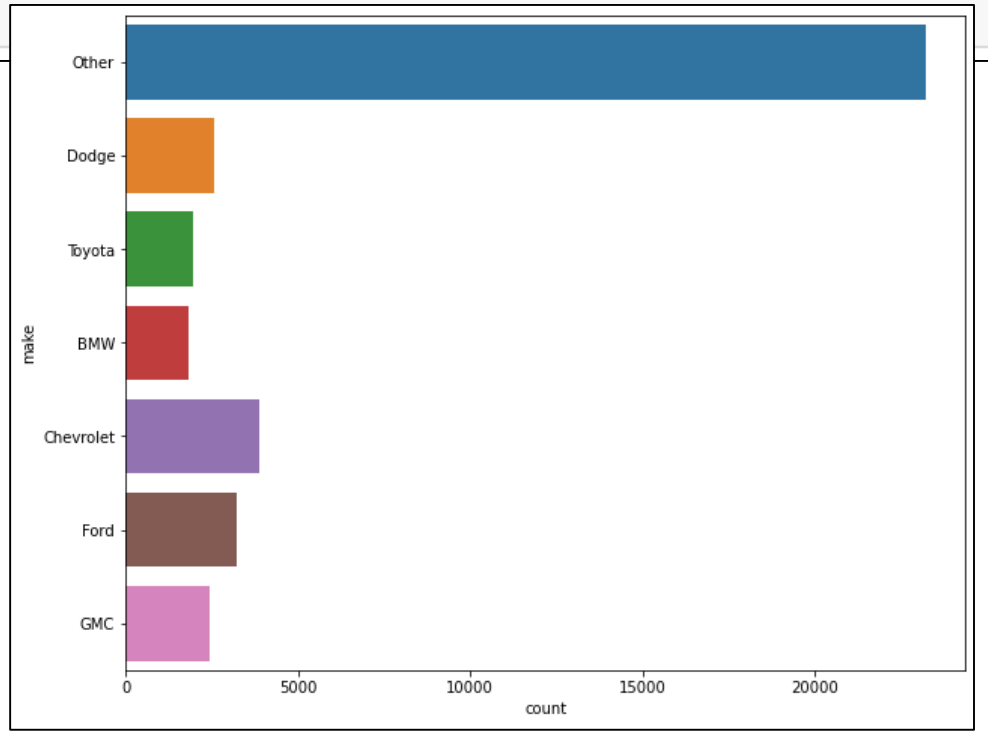
資料轉換與缺失值處理

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 8))
top_n = fueleco.make.value_counts().index[:6]
(fueleco
   .assign(make=fueleco.make.where(
               fueleco.make.isin(top_n),
               'Other'))
   .make
   .value_counts()
   .plot.bar(ax=ax)
)
```



資料轉換與缺失值處理

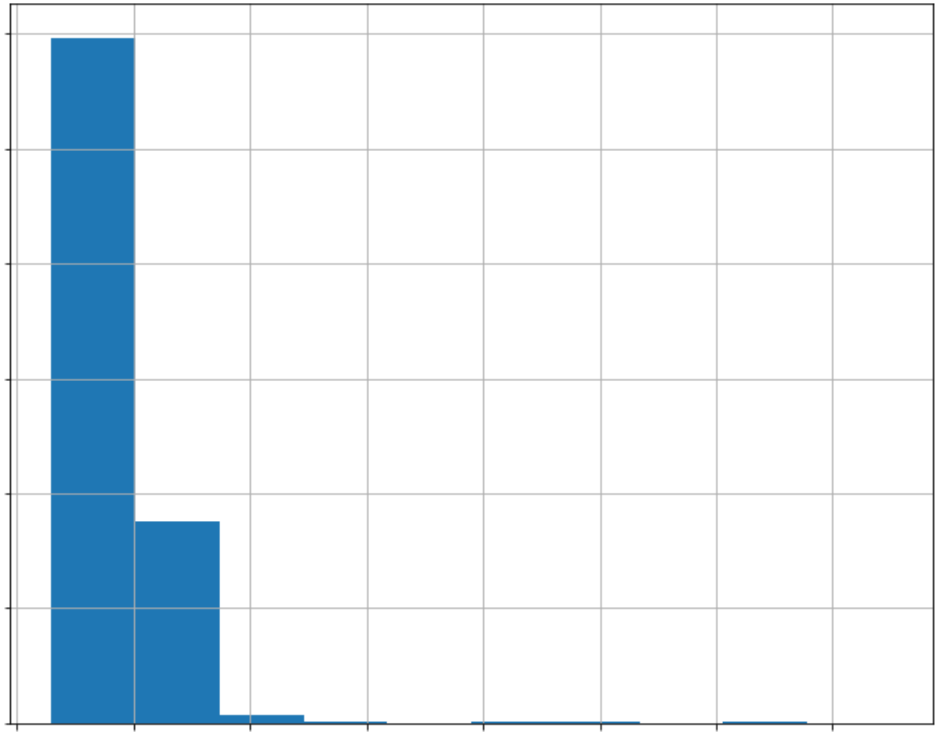
```
import seaborn as sns
fig, ax = plt.subplots(figsize=(10, 8))
top_n = fueleco.make.value_counts().index[:6]
sns.countplot(y='make',
    data= (fueleco
        .assign(make=fueleco.make.where(
                fueleco.make.isin(top_n),
                'Other'))
  )
)
```



檢視連續資料的分布狀況

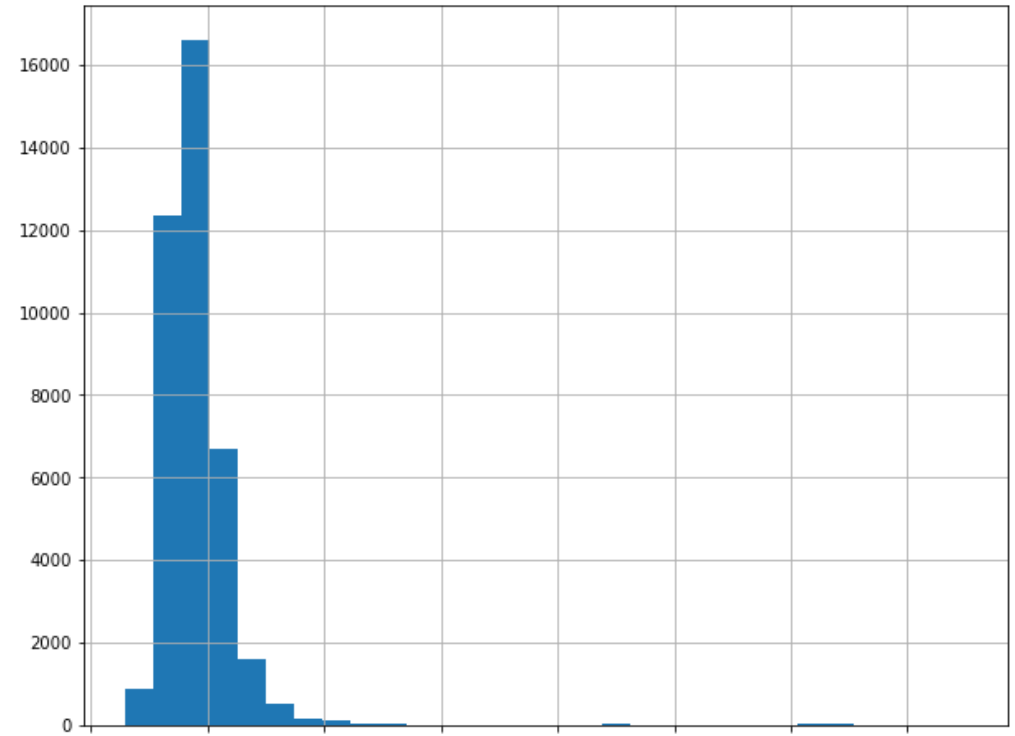
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 8))
fueleco.city08.hist(ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x228968ec3d0>
```




```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 8))
fueleco.city08.hist(ax=ax, bins=30)
```

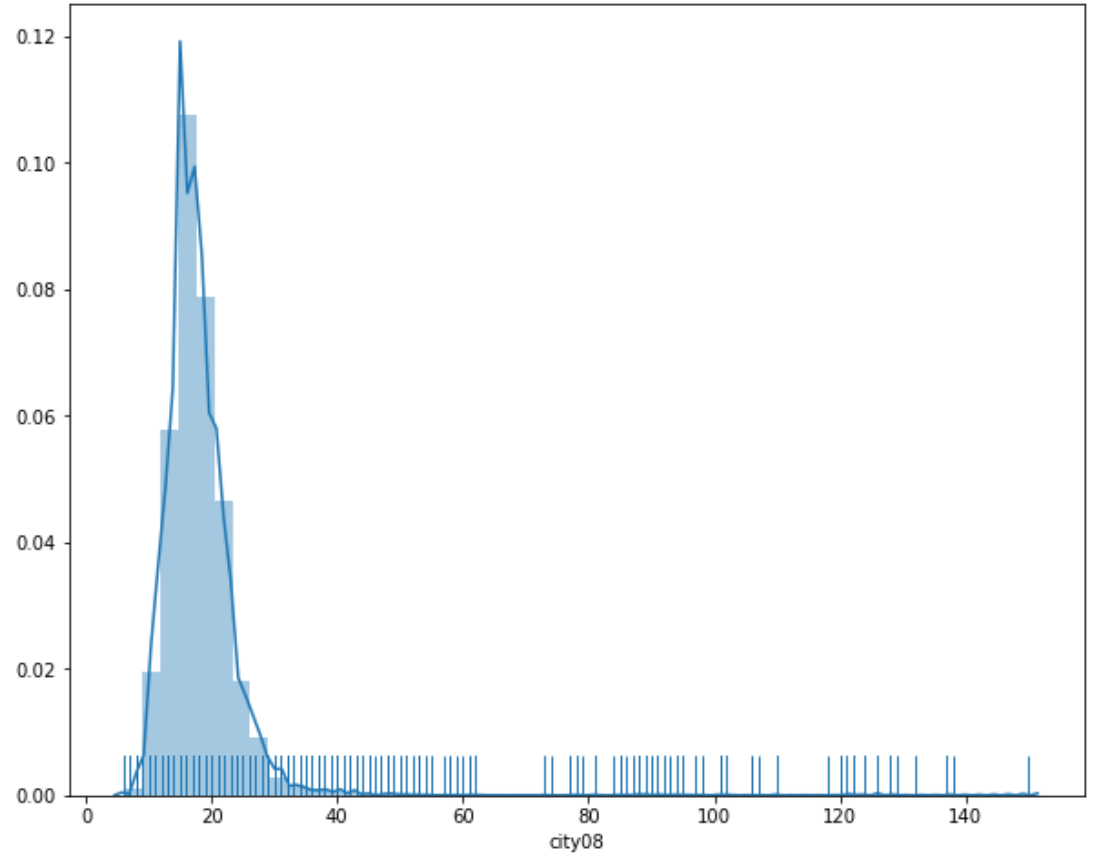
```
<matplotlib.axes._subplots.AxesSubplot at 0x2289579bcd0>
```



檢視連續資料的分布狀況

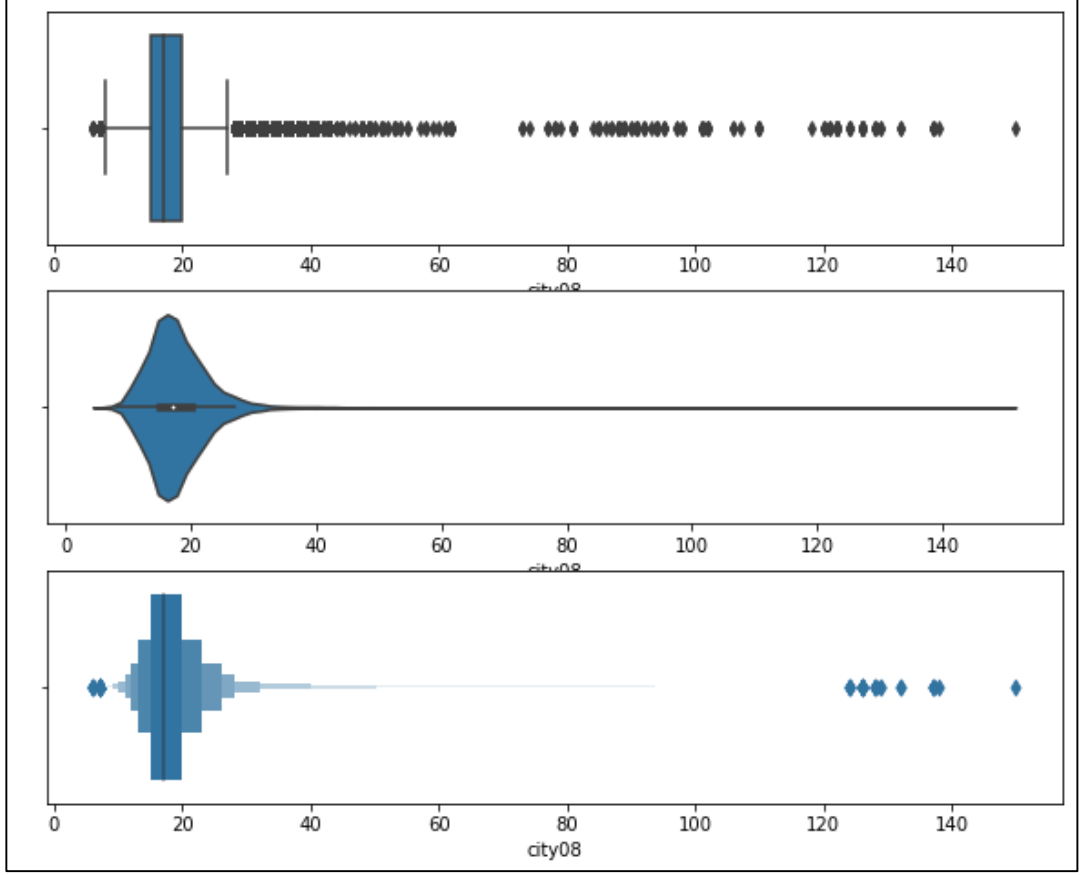
```
fig, ax = plt.subplots(figsize=(10, 8))
sns.distplot(fueleco.city08, rug=True, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x228950910d0>
```




```
fig, axs = plt.subplots(nrows=3, figsize=(10, 8))
sns.boxplot(fueleco.city08, ax=axs[0])
sns.violinplot(fueleco.city08, ax=axs[1])
sns.boxenplot(fueleco.city08, ax=axs[2])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22897d8f700>
```



檢視連續資料的分布狀況

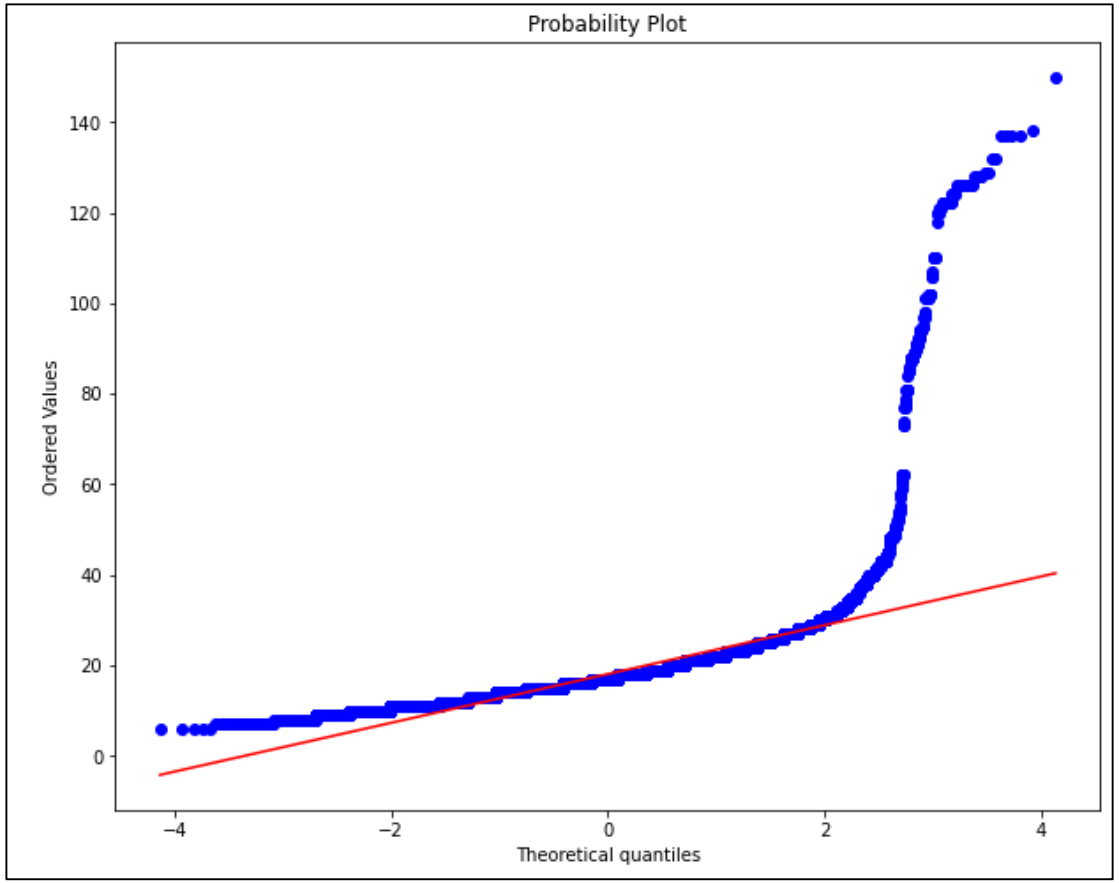
```
from scipy import stats
stats.kstest(fueleco.city08, cdf='norm')#檢查是否常態分佈
```

```
KstestResult(statistic=0.9999999990134123, pvalue=0.0)
```

#pvalue 小於 0.05 代表不是常態分佈

```
from scipy import stats
fig, ax = plt.subplots(figsize=(10, 8))
stats.probplot(fueleco.city08, plot=ax)
```

```
((array([-4.1352692 , -3.92687024, -3.81314873, ...,  3.81314873,
         3.92687024,  4.1352692 ]),
  array([  6,   6,   6, ..., 137, 138, 150], dtype=int64)),
 (5.385946629915974, 18.077798521776934, 0.772587941459713))
```



比較連續欄位間的關聯性

```
fueleco.city08.cov(fueleco.highway08)
```

```
46.33326023673624
```

#cov() 計算共變異數

```
fueleco.city08.cov(fueleco.comb08)
```

```
47.419946678190776
```

```
fueleco.city08.cov(fueleco.cylinders)
```

```
-5.931560263764768
```

```
fueleco.city08.corr(fueleco.highway08)
```

```
0.932494506228495
```

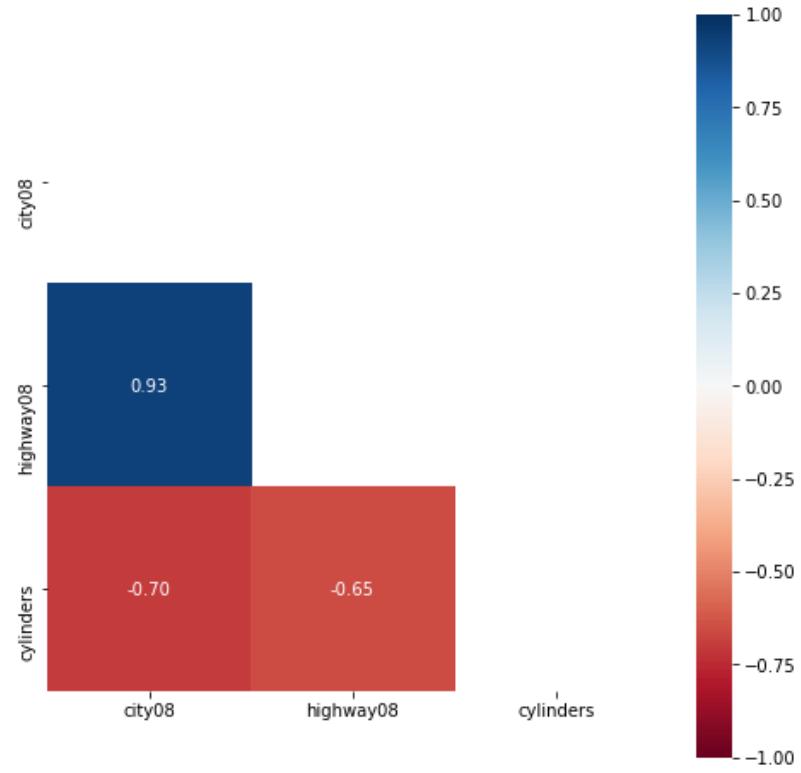
#corr() 計算皮爾森相關性

```
fueleco.city08.corr(fueleco.cylinders)
```

```
-0.7016548423827895
```

```
import seaborn as sns
fig, ax = plt.subplots(figsize=(8,8))
corr = fueleco[['city08', 'highway08', 'cylinders']].corr()
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
sns.heatmap(corr, mask=mask,
            fmt='.2f', annot=True, ax=ax, cmap='RdBu', vmin=-1, vmax=1,
            square=True)
```

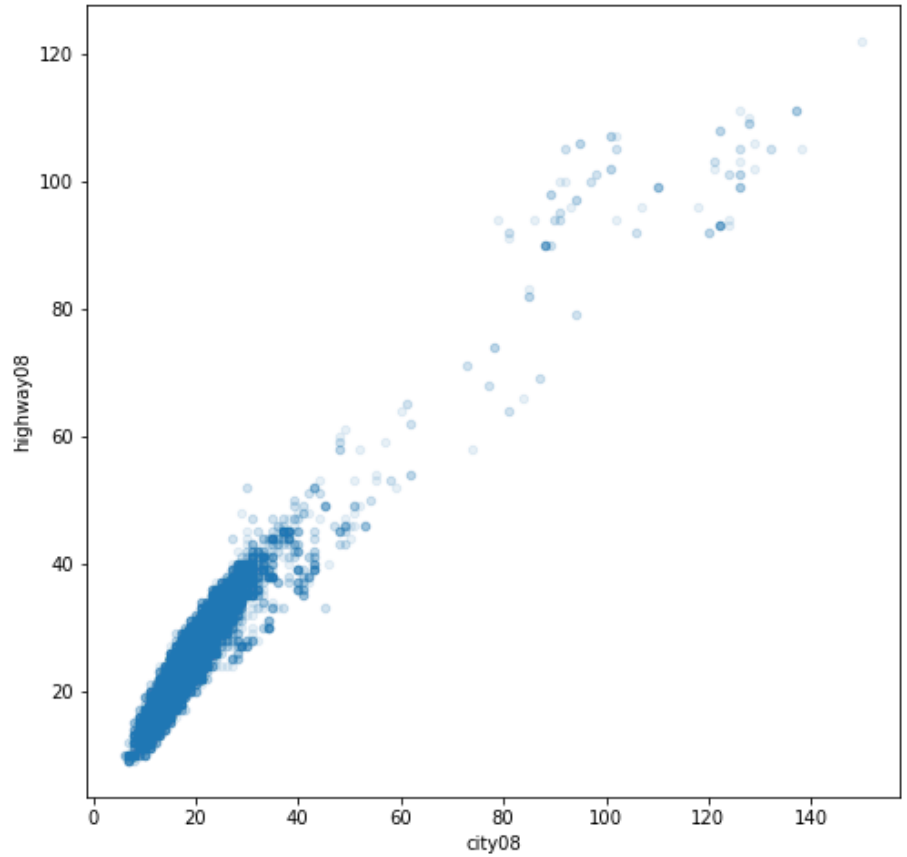
```
<matplotlib.axes._subplots.AxesSubplot at 0x228992a3eb0>
```



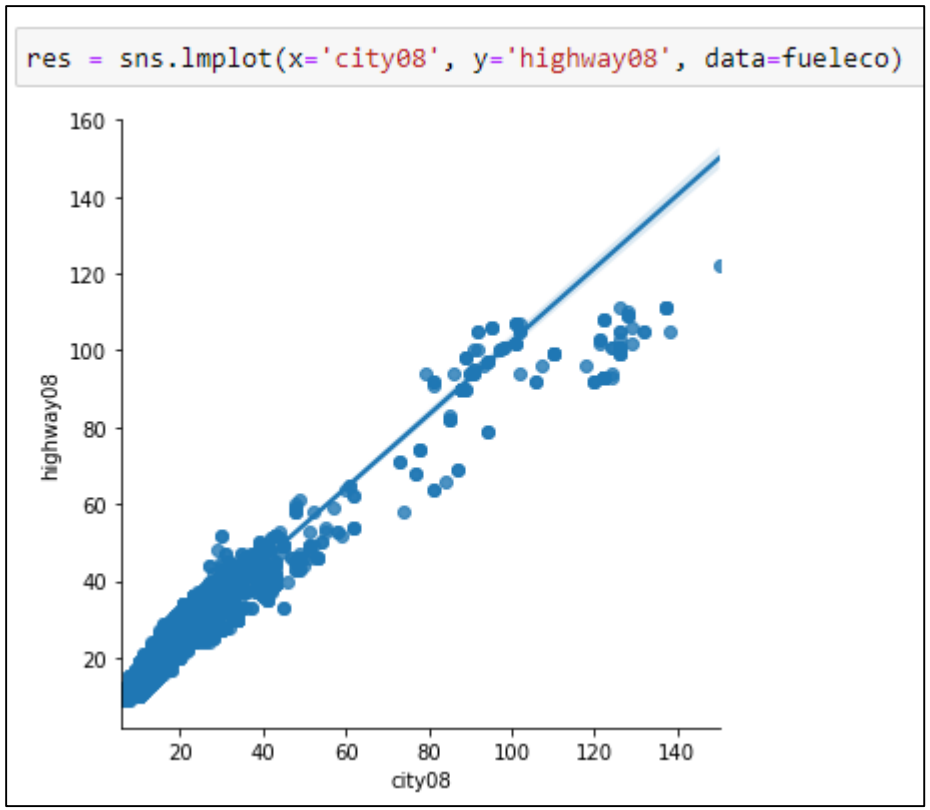
比較連續欄位間的關聯性

```
fig, ax = plt.subplots(figsize=(8,8))
fueleco.plot.scatter(x='city08', y='highway08', alpha=.1, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x228993189a0>
```




```
res = sns.lmplot(x='city08', y='highway08', data=fueleco)
```



時間序列分析

Python 和 Pandas 日期工具的區別

```
import pandas as pd
import numpy as np
pd.set_option('max_columns', 4, 'max_rows', 10,'max_colwidth', 12)
```

```
import datetime
date = datetime.date(year=2022, month=6, day=7)
time = datetime.time(hour=12, minute=30, second=19, microsecond=463198)
dt = datetime.datetime(year=2022, month=6, day=7, hour=12, minute=30, second=19,
                       microsecond=463198)
print(f'date is {date}')
```

```
date is 2022-06-07
```

```
print(f'time is {time}')
```

```
time is 12:30:19.463198
```

```
print(f'datetime is {dt}')
```

```
datetime is 2022-06-07 12:30:19.463198
```

```
td = datetime.timedelta(weeks=2, days=5, hours=10,
                        minutes=20, seconds=6.73,
                        milliseconds=99, microseconds=8)
td
```

```
datetime.timedelta(days=19, seconds=37206, microseconds=829008)
```

```
print(f'new date is {date+td}')
```

```
new date is 2022-06-26
```

```
print(f'new datetime is {dt+td}')
```

```
new datetime is 2022-06-26 22:50:26.292206
```

```
# time + td #error is existed because no date.
```

Python 和 Pandas 日期工具的區別

```
pd.Timestamp(year=2021, month=12, day=21, hour=5,
             minute=10, second=8, microsecond=99)
```

```
Timestamp('2021-12-21 05:10:08.000099')
```

```
pd.Timestamp('2016/1/10')
```

```
Timestamp('2016-01-10 00:00:00')
```

```
pd.Timestamp('2014-5/10')
```

```
Timestamp('2014-05-10 00:00:00')
```

```
pd.Timestamp('Jan 3, 2019 20:45.56')
```

```
Timestamp('2019-01-03 20:45:33')
```

```
pd.Timestamp('2016-01-05T05:34:43.123456789')
```

```
Timestamp('2016-01-05 05:34:43.123456789')
```

```
pd.to_datetime('2015-5-13')
```

```
Timestamp('2015-05-13 00:00:00')
```

```
pd.to_datetime('2015-13-5', dayfirst=True)
```

```
Timestamp('2015-05-13 00:00:00')
```

```
pd.to_datetime('Start Date: Sep 30, 2017 Start Time: 1:30 pm',
               format='Start Date: %b %d, %Y Start Time: %I:%M %p')
```

```
Timestamp('2017-09-30 13:30:00')
```

```
pd.to_datetime(100, unit='D', origin='2013-1-1')
```

```
Timestamp('2013-04-11 00:00:00')
```

Python 和 Pandas 日期工具的區別

```
pd.Timestamp(500)
```

```
Timestamp('1970-01-01 00:00:00.000000500')
```

```
pd.Timestamp(5000, unit='D')
```

```
Timestamp('1983-09-10 00:00:00')
```

```
s = pd.Series([10, 100, 1000, 10000])
pd.to_datetime(s, unit='D')
```

```
0   1970-01-11
1   1970-04-11
2   1972-09-27
3   1997-05-19
dtype: datetime64[ns]
```

```
s = pd.Series(['12-5-2015', '14-1-2013', '20/12/2017', '40/23/2017'])
pd.to_datetime(s, dayfirst=True, errors='coerce')
```

```
0   2015-05-12
1   2013-01-14
2   2017-12-20
3          NaT
dtype: datetime64[ns]
```

Python 和 Pandas 日期工具的區別

```
s = pd.Series([10, 100])
pd.to_timedelta(s, unit='s')
```

```
0   00:00:10
1   00:01:40
dtype: timedelta64[ns]
```

```
pd.Timedelta('12 days 5 hours 3 minutes') * 2
```

```
Timedelta('24 days 10:06:00')
```

```
(pd.Timestamp('1/1/2022') + pd.Timedelta('12 days 5 hours 3 minutes') * 2)
```

```
Timestamp('2022-01-25 10:06:00')
```

```
td1 = pd.to_timedelta([10, 100], unit='s')
td2 = pd.to_timedelta(['3 hours', '4 hours'])
td1 + td2
```

```
TimedeltaIndex(['03:00:10', '04:01:40'], dtype='timedelta64[ns]', freq=None)
```

```
ts = pd.Timestamp('2021-10-1 4:23:23.9')
ts.ceil('h')
```

```
Timestamp('2021-10-01 05:00:00')
```

```
ts.year, ts.month, ts.day, ts.hour, ts.minute, ts.second
```

```
(2021, 10, 1, 4, 23, 23)
```

```
ts.dayofweek, ts.dayofyear, ts.daysinmonth
```

```
(4, 274, 31)
```

```
ts.to_pydatetime()
```

```
datetime.datetime(2021, 10, 1, 4, 23, 23, 900000)
```

```
td = pd.Timedelta(125.8723, unit='h')
td
```

```
Timedelta('5 days 05:52:20.280000')
```

```
td.round('min')
```

```
Timedelta('5 days 05:52:00')
```

```
td.components
```

```
Components(days=5, hours=5, minutes=52, seconds=20, milliseconds=280, microseconds=0, nanoseconds=0)
```

```
td.total_seconds()
```

```
453140.28
```

對時間序列切片

```
!pip install tables
```

```
Collecting tables
  Downloading tables-3.7.0-cp37-cp37m-win_amd64.whl (6.8 MB)
Requirement already satisfied: numpy>=1.19.0 in c:\users\jwchang\appdata\local\programs\python\python37\lib\site-packages (from
tables) (1.21.1)
Collecting numexpr>=2.6.2
  Downloading numexpr-2.8.3-cp37-cp37m-win_amd64.whl (92 kB)
Requirement already satisfied: packaging in c:\users\jwchang\appdata\local\programs\python\python37\lib\site-packages (from tab
les) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\jwchang\appdata\local\programs\python\python37\lib\site-pac
kages (from packaging->tables) (2.4.7)
Installing collected packages: numexpr, tables
Successfully installed numexpr-2.8.3 tables-3.7.0
```

```
crime = pd.read_hdf('data/crime.h5', 'crime')
crime.dtypes
```

```
OFFENSE_TYPE_ID          category
OFFENSE_CATEGORY_ID      category
REPORTED_DATE          datetime...
GEO_LON                    float64
GEO_LAT                    float64
NEIGHBORHOOD_ID          category
IS_CRIME                    int64
IS_TRAFFIC                  int64
dtype: object
```

對時間序列切片

```
mem_cat = crime.memory_usage().sum()
mem_obj = (crime
    .astype({'OFFENSE_TYPE_ID':'object',
             'OFFENSE_CATEGORY_ID':'object',
            'NEIGHBORHOOD_ID':'object'})
    .memory_usage(deep=True)
    .sum()
)
mb = 2 ** 20
round(mem_cat / mb, 1), round(mem_obj / mb, 1)
```

```
(22.9, 116.2) #從 category 變成 object，記憶體用量暴增
```

對時間序列切片

```
crime
```

	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
0	traffic-...	traffic-...	...	0	1
1	vehicula...	all-othe...	...	1	0
2	disturbi...	public-d...	...	1	0
3	curfew	public-d...	...	1	0
4	aggravat...	aggravat...	...	1	0
...	...	...	...	...	...
460906	burglary...	burglary	...	1	0
460907	weapon-u...	all-othe...	...	1	0
460908	traf-hab...	all-othe...	...	1	0
460909	criminal...	public-d...	...	1	0
460910	theft-other	larceny	...	1	0

460911 rows × 8 columns

```
crime = crime.set_index('REPORTED_DATE')
crime
```

REPORTED_DATE	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
2014-06-29 02:01:00	traffic-...	traffic-...	...	0	1
2014-06-29 01:54:00	vehicula...	all-othe...	...	1	0
2014-06-29 02:00:00	disturbi...	public-d...	...	1	0
2014-06-29 02:18:00	curfew	public-d...	...	1	0
2014-06-29 04:17:00	aggravat...	aggravat...	...	1	0
...	...	...	...	...	...
2017-09-13 05:48:00	burglary...	burglary	...	1	0
2017-09-12 20:37:00	weapon-u...	all-othe...	...	1	0
2017-09-12 16:32:00	traf-hab...	all-othe...	...	1	0
2017-09-12 13:04:00	criminal...	public-d...	...	1	0
2017-09-12 09:30:00	theft-other	larceny	...	1	0

460911 rows × 7 columns

對時間序列切片

```
crime.index[:2]
```

```
DatetimeIndex(['2014-06-29 02:01:00', '2014-06-29 01:54:00'], dtype='datetime64[ns]', name='REPORTED_DATE', freq=None)
```

```
crime.loc['2016-05-12 16:45:00']
```

REPORTED_DATE	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
2016-05-12 16:45:00	traffic-...	traffic-...	...	0	1
2016-05-12 16:45:00	traffic-...	traffic-...	...	0	1
2016-05-12 16:45:00	fraud-id...	white-co...	...	1	0

3 rows × 7 columns

```
crime.loc['2016-05-12']
```

REPORTED_DATE	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
2016-05-12 23:51:00	criminal...	public-d...	...	1	0
2016-05-12 18:40:00	liquor-p...	drug-alc...	...	1	0
2016-05-12 22:26:00	traffic-...	traffic-...	...	0	1
2016-05-12 20:35:00	theft-bi...	larceny	...	1	0
2016-05-12 09:39:00	theft-of...	auto-theft	...	1	0
...	...	...	...	...	...
2016-05-12 17:55:00	public-p...	public-d...	...	1	0
2016-05-12 19:24:00	threats-...	public-d...	...	1	0
2016-05-12 22:28:00	sex-aslt...	sexual-a...	...	1	0
2016-05-12 15:59:00	menacing...	aggravat...	...	1	0
2016-05-12 16:39:00	assault-dv	other-cr...	...	1	0

243 rows × 7 columns

對時間序列切片

```
crime.loc['2016-05'].shape
```

```
(8012, 7)
```

```
crime.loc['2016'].shape
```

```
(91076, 7)
```

```
crime.loc['2016-05-12 03'].shape
```

```
(4, 7)
```

```
crime.loc['Dec 2015'].sort_index()
```

REPORTED_DATE	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
2015-12-01 00:48:00	drug-coc...	drug-alc...	...	1	0
2015-12-01 00:48:00	theft-of...	auto-theft	...	1	0
2015-12-01 01:00:00	criminal...	public-d...	...	1	0
2015-12-01 01:10:00	traf-other	all-othe...	...	1	0
2015-12-01 01:10:00	traf-hab...	all-othe...	...	1	0
...	...	...	...	...	...
2015-12-31 23:35:00	drug-coc...	drug-alc...	...	1	0
2015-12-31 23:40:00	traffic-...	traffic-...	...	0	1
2015-12-31 23:44:00	drug-coc...	drug-alc...	...	1	0
2015-12-31 23:45:00	violatio...	all-othe...	...	1	0
2015-12-31 23:50:00	weapon-p...	all-othe...	...	1	0

6907 rows × 7 columns

```
crime.loc['2016 Sep, 15'].shape
```

```
(252, 7)
```

```
crime.loc['21st October 2014 05'].shape
```

```
(4, 7)
```

```
crime.loc['2015-3-4':'2016-1-1'].sort_index()
```

REPORTED_DATE	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
2015-03-04 00:11:00	assault-dv	other-cr...	...	1	0
2015-03-04 00:19:00	assault-dv	other-cr...	...	1	0
2015-03-04 00:27:00	theft-of...	larceny	...	1	0
2015-03-04 00:49:00	traffic-...	traffic-...	...	0	1
2015-03-04 01:07:00	burglary...	burglary	...	1	0
...	...	...	...	...	...
2016-01-01 23:15:00	traffic-...	traffic-...	...	0	1
2016-01-01 23:16:00	traffic-...	traffic-...	...	0	1
2016-01-01 23:40:00	robbery-...	robbery	...	1	0
2016-01-01 23:45:00	drug-coc...	drug-alc...	...	1	0
2016-01-01 23:48:00	drug-pos...	drug-alc...	...	1	0

75403 rows × 7 columns

對時間序列切片

```
crime.loc['2015-3-4 22':'2016-1-1 11:22:00'].sort_index()
```

REPORTED_DATE	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
2015-03-04 22:25:00	traffic-...	traffic-...	...	0	1
2015-03-04 22:30:00	traffic-...	traffic-...	...	0	1
2015-03-04 22:32:00	traffic-...	traffic-...	...	0	1
2015-03-04 22:33:00	traffic-...	traffic-...	...	0	1
2015-03-04 22:36:00	theft-un...	white-co...	...	1	0
...	...	...	...	...	...
2016-01-01 11:10:00	theft-of...	auto-theft	...	1	0
2016-01-01 11:11:00	traffic-...	traffic-...	...	0	1
2016-01-01 11:11:00	traffic-...	traffic-...	...	0	1
2016-01-01 11:16:00	traf-other	all-othe...	...	1	0
2016-01-01 11:22:00	traffic-...	traffic-...	...	0	1

75071 rows × 7 columns

```
%timeit crime.loc['2015-3-4':'2016-1-1']
```

```
12.7 ms ± 723 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

```
crime_sort = crime.sort_index()
%timeit crime_sort.loc['2015-3-4':'2016-1-1']
```

```
1.69 ms ± 427 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

過濾包含時間資料的欄位

```
crime = pd.read_hdf('data/crime.h5', 'crime')
crime.dtypes
```

```
OFFENSE_TYPE_ID           category
OFFENSE_CATEGORY_ID       category
REPORTED_DATE           datetime...
GEO_LON                    float64
GEO_LAT                    float64
NEIGHBORHOOD_ID           category
IS_CRIME                     int64
IS_TRAFFIC                   int64
dtype: object
```

```
(crime
    [crime.REPORTED_DATE == '2016-05-12 16:45:00']
)
```

	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
300905	traffic-...	traffic-...	...	0	1
302354	traffic-...	traffic-...	...	0	1
302373	fraud-id...	white-co...	...	1	0

3 rows × 8 columns

```
(crime
    [crime.REPORTED_DATE == '2016-05-12']
)
```

OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC

0 rows × 8 columns

```
(crime
    [crime.REPORTED_DATE.dt.date == '2016-05-12']
)
```

OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC

0 rows × 8 columns

#因為 datetime 不支援跟 string 比較

過濾包含時間資料的欄位

```
(crime[crime.REPORTED_DATE.between(left='2016-05-12', right='2016-05-13')])
```

	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
295715	criminal...	public-d...	...	1	0
296474	liquor-p...	drug-alc...	...	1	0
297204	traffic-...	traffic-...	...	0	1
299383	theft-bi...	larceny	...	1	0
299389	theft-of...	auto-theft	...	1	0
...	...	...	...	...	...
358208	public-p...	public-d...	...	1	0
358448	threats-...	public-d...	...	1	0
363134	sex-aslt...	sexual-a...	...	1	0
365959	menacing...	aggravat...	...	1	0
378711	assault-dv	other-cr...	...	1	0

243 rows × 8 columns

```
(crime[crime.REPORTED_DATE.between('2016-05', '2016-06')].shape)
```

```
(8012, 8)
```

```
(crime[crime.REPORTED_DATE.between('2016', '2017')].shape)
```

```
(91076, 8)
```

```
(crime[crime.REPORTED_DATE.between('2016-05-12 03', '2016-05-12 04')].shape)
```

```
(4, 8)
```

```
(crime[crime.REPORTED_DATE.between('2016 Sep, 15', '2016 Sep, 16')].shape)
```

```
(252, 8)
```

```
(crime[crime.REPORTED_DATE.between('21st October 2014 05',
                                   '21st October 2014 06')].shape)
```

```
(4, 8)
```

```
(crime[crime.REPORTED_DATE.between('2015-3-4 ','2016-1-1 23:59:59')].shape)
```

```
(75403, 8)
```

```
(crime
    [crime.REPORTED_DATE.between(
         '2015-3-4 22','2016-1-1 11:22:00')]
    .shape
)
```

```
(75071, 8)
```

過濾包含時間資料的欄位

```
lmask = crime.REPORTED_DATE >= '2015-3-4 22'
rmask = crime.REPORTED_DATE <= '2016-1-1 11:22:00'
crime[lmask & rmask].shape
```

```
(75071, 8)
```

```
ctseries = crime.set_index('REPORTED_DATE')
%timeit ctseries.loc['2015-3-4':'2016-1-1']
```

```
11.9 ms ± 93.1 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

```
%timeit crime[crime.REPORTED_DATE.between('2015-3-4','2016-1-1')]
```

```
15.5 ms ± 269 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

僅適用於DatetimeIndex的方法

```
crime = (pd.read_hdf('data/crime.h5', 'crime').set_index('REPORTED_DATE'))
type(crime.index)
```

```
pandas.core.indexes.datetimes.DatetimeIndex
```

```
crime.between_time('2:00', '5:00', include_end=False)
```

REPORTED_DATE	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
2014-06-29 02:01:00	traffic-...	traffic-...	...	0	1
2014-06-29 02:00:00	disturbi...	public-d...	...	1	0
2014-06-29 02:18:00	curfew	public-d...	...	1	0
2014-06-29 04:17:00	aggravat...	aggravat...	...	1	0
2014-06-29 04:22:00	violatio...	all-othe...	...	1	0
...	...	...	...	...	...
2017-08-25 04:41:00	theft-it...	theft-fr...	...	1	0
2017-09-13 04:17:00	theft-of...	auto-theft	...	1	0
2017-09-13 02:21:00	assault-...	other-cr...	...	1	0
2017-09-13 03:21:00	traffic-...	traffic-...	...	0	1
2017-09-13 02:15:00	traffic-...	traffic-...	...	0	1

29078 rows × 7 columns

```
import datetime
crime.between_time(datetime.time(2,0), datetime.time(5,0), include_end=False)
```

REPORTED_DATE	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
2014-06-29 02:01:00	traffic-...	traffic-...	...	0	1
2014-06-29 02:00:00	disturbi...	public-d...	...	1	0
2014-06-29 02:18:00	curfew	public-d...	...	1	0
2014-06-29 04:17:00	aggravat...	aggravat...	...	1	0
2014-06-29 04:22:00	violatio...	all-othe...	...	1	0
...	...	...	...	...	...
2017-08-25 04:41:00	theft-it...	theft-fr...	...	1	0
2017-09-13 04:17:00	theft-of...	auto-theft	...	1	0
2017-09-13 02:21:00	assault-...	other-cr...	...	1	0
2017-09-13 03:21:00	traffic-...	traffic-...	...	0	1
2017-09-13 02:15:00	traffic-...	traffic-...	...	0	1

29078 rows × 7 columns

僅適用於DatetimeIndex的方法

```
crime.at_time('5:47')
```

REPORTED_DATE	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	...	IS_CRIME	IS_TRAFFIC
2013-11-26 05:47:00	criminal...	public-d...	...	1	0
2017-04-09 05:47:00	criminal...	public-d...	...	1	0
2017-02-19 05:47:00	criminal...	public-d...	...	1	0
2017-02-16 05:47:00	aggravat...	aggravat...	...	1	0
2017-02-12 05:47:00	police-i...	all-othe...	...	1	0
...	...	...	...	...	...
2013-09-10 05:47:00	traffic-...	traffic-...	...	0	1
2013-03-14 05:47:00	theft-other	larceny	...	1	0
2012-10-08 05:47:00	theft-it...	theft-fr...	...	1	0
2013-08-21 05:47:00	theft-it...	theft-fr...	...	1	0
2017-08-23 05:47:00	traffic-...	traffic-...	...	0	1

118 rows × 7 columns

依據時間區段重新分組

```
crime_sort = (pd.read_hdf('data/crime.h5', 'crime')
                .set_index('REPORTED_DATE')
                .sort_index())
```

```
crime_sort.resample('W')
```

```
<pandas.core.resample.DatetimeIndexResampler object at 0x000002962D2C4040>
```

```
(crime_sort
    .resample('W')
    .size()
)
```

```
REPORTED_DATE
2012-01-08     877
2012-01-15    1071
2012-01-22     991
2012-01-29     988
2012-02-05     888
              ...
2017-09-03    1956
2017-09-10    1733
2017-09-17    1976
2017-09-24    1839
2017-10-01    1059
Freq: W-SUN, Length: 300, dtype: int64
```

```
len(crime_sort.loc[:'2012-1-8'])
```

```
877
```

```
len(crime_sort.loc['2012-1-9':'2012-1-15'])
```

```
1071
```

```
(crime_sort
    .resample('W-THU')
    .size()
)                    #以星期四為一周結束日
```

```
REPORTED_DATE
2012-01-05     462
2012-01-12    1116
2012-01-19     924
2012-01-26    1061
2012-02-02     926
              ...
2017-09-07    1803
2017-09-14    1866
2017-09-21    1926
2017-09-28    1720
2017-10-05      28
Freq: W-THU, Length: 301, dtype: int64
```

依據時間區段重新分組

```
crime_sort = (pd.read_hdf('data/crime.h5', 'crime')
                .set_index('REPORTED_DATE')
                .sort_index())
```

```
weekly_crimes = (crime_sort.groupby(pd.Grouper(freq='W'))
                           .size())
weekly_crimes
```

 # resample () 的功能可以透過 groupby()重現

```
REPORTED_DATE
2012-01-08     877
2012-01-15    1071
2012-01-22     991
2012-01-29     988
2012-02-05     888
              ...
2017-09-03    1956
2017-09-10    1733
2017-09-17    1976
2017-09-24    1839
2017-10-01    1059
Freq: W-SUN, Length: 300, dtype: int64
```

```
crime = pd.read_hdf('data/crime.h5', 'crime')
weekly_crimes2 = crime.resample('W', on='REPORTED_DATE').size()
weekly_crimes2.equals(weekly_crimes)
```

```
True
```

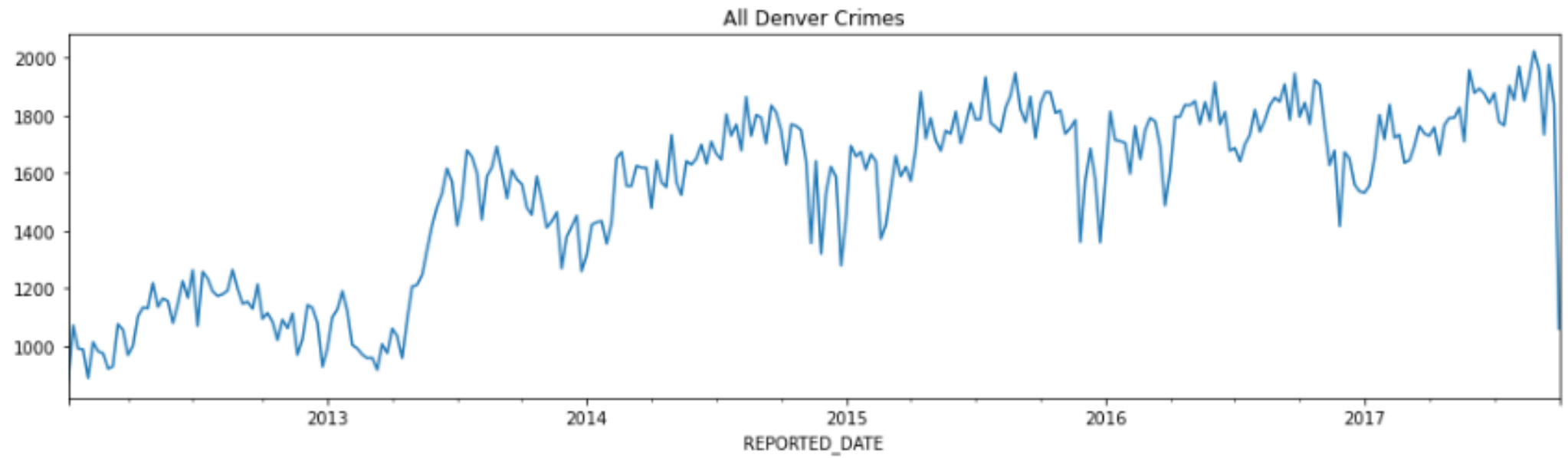
```
weekly_crimes_gby2 = (crime.groupby(pd.Grouper(key='REPORTED_DATE', freq='W'))
                           .size())
weekly_crimes_gby2.equals(weekly_crimes)
```

```
True
```

依據時間區段重新分組

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(16, 4))
weekly_crimes.plot(title='All Denver Crimes', ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28404126488>
```



分組彙總同一時間單位的多個欄位

```
crime = (pd.read_hdf('data/crime.h5', 'crime')
           .set_index('REPORTED_DATE')
           .sort_index())
```

```
(crime
    .resample('Q')
    ['IS_CRIME', 'IS_TRAFFIC']
    .sum()
)
```

Q 是以季的結束日

REPORTED_DATE	IS_CRIME	IS_TRAFFIC
2012-03-31	7882	4726
2012-06-30	9641	5255
2012-09-30	10566	5003
2012-12-31	9197	4802
2013-03-31	8730	4442
...	...	...
2016-09-30	17427	6199
2016-12-31	15984	6094
2017-03-31	16426	5587
2017-06-30	17486	6148
2017-09-30	17990	6101

23 rows × 2 columns

```
(crime
    .resample('QS')
    ['IS_CRIME', 'IS_TRAFFIC']
    .sum()
)
```

QS 是以季的開始日

REPORTED_DATE	IS_CRIME	IS_TRAFFIC
2012-01-01	7882	4726
2012-04-01	9641	5255
2012-07-01	10566	5003
2012-10-01	9197	4802
2013-01-01	8730	4442
...	...	...
2016-07-01	17427	6199
2016-10-01	15984	6094
2017-01-01	16426	5587
2017-04-01	17486	6148
2017-07-01	17990	6101

23 rows × 2 columns

```
(crime
   .loc['2012-4-1':'2012-6-30', ['IS_CRIME', 'IS_TRAFFIC']]
   .sum()
)
```

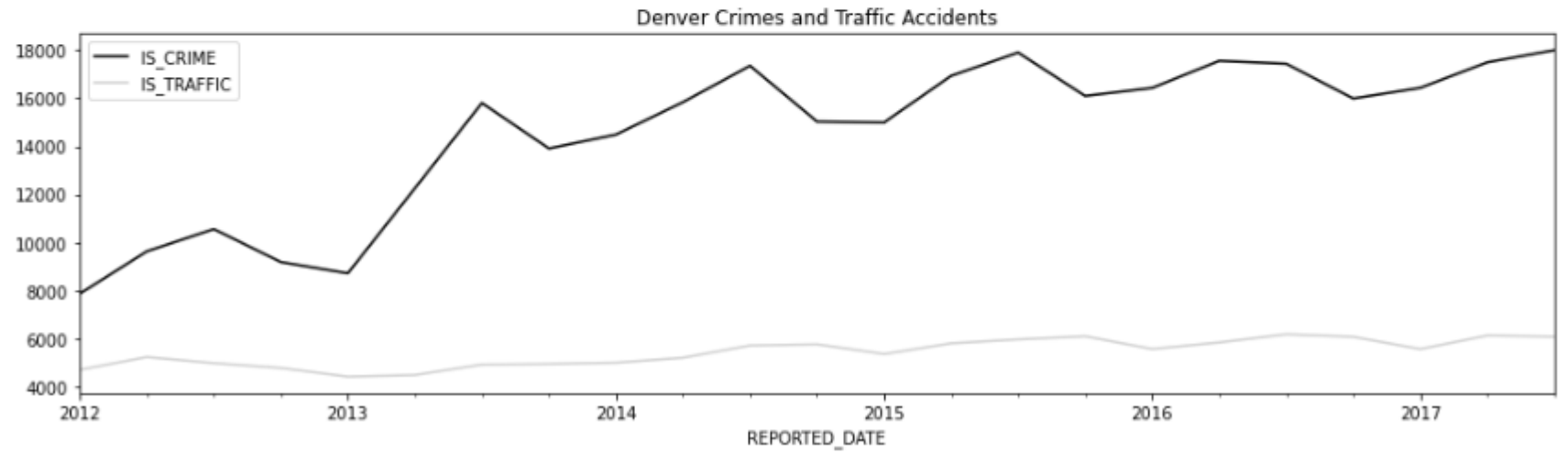
```
IS_CRIME      9641
IS_TRAFFIC    5255
dtype: int64
```

分組彙總同一時間單位的多個欄位

```
fig, ax = plt.subplots(figsize=(16, 4))
(crime
    .groupby(pd.Grouper(freq='Q'))
    ['IS_CRIME', 'IS_TRAFFIC']
    .sum()
    .plot(color=['black', 'lightgrey'], ax=ax,
          title='Denver Crimes and Traffic Accidents')
)
```

```
c:\users\jwchang\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: Indexing with
multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  after removing the cwd from sys.path.
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28404bd8a88>
```



分組彙總同一時間單位的多個欄位

```
crime_begin = (crime.resample('Q')
                    ['IS_CRIME', 'IS_TRAFFIC']
                    .sum()
                    .iloc[0])
```

```
crime_begin

IS_CRIME      7882
IS_TRAFFIC    4726
Name: 2012-03-31 00:00:00, dtype: int64
```

```
fig, ax = plt.subplots(figsize=(16, 4))
(crime
    .resample('Q')
    ['IS_CRIME', 'IS_TRAFFIC']
    .sum()
    .div(crime_begin)
    .sub(1)
    .round(2)
    .mul(100)
    .plot.bar(color=['black', 'lightgrey'], ax=ax,
              title='Denver Crimes and Traffic Accidents % Increase')
)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28404de35c8>
```