SciPy LATAM · 2019

High Performance Pandas 2

Time Based Analysis Using Pandas

David Camacho – Lead Data Architect

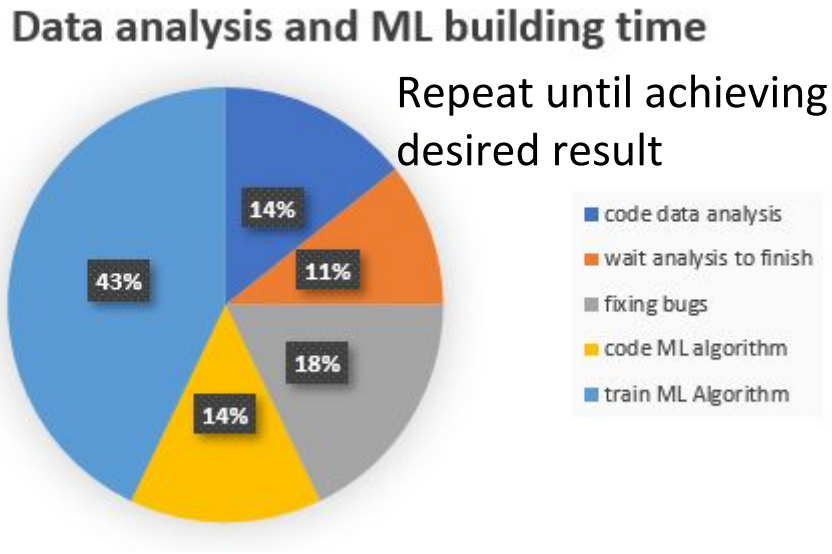
Cyxtera

Outline

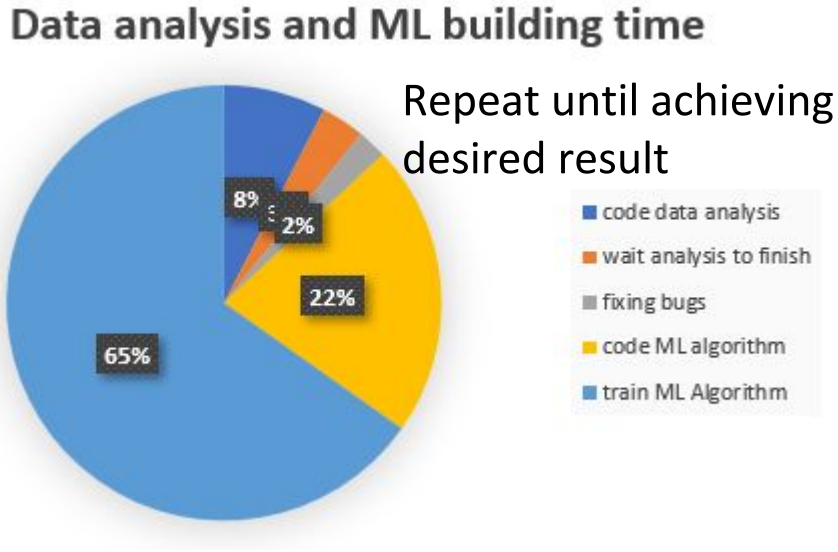
- Intro
- Pandas Datetime Objects
- Time Based Querying
- Grouping Information Using Time
- Aggregations and Custom Operations

Data Analysis is the most time consuming task of an ML project… some times is waiting the algorithm to train!

Intro



Repeat until achieving desired result



Repeat until achieving desired result

Total time per iteration:
~10+ hours
Total coffee consumption:
2lt

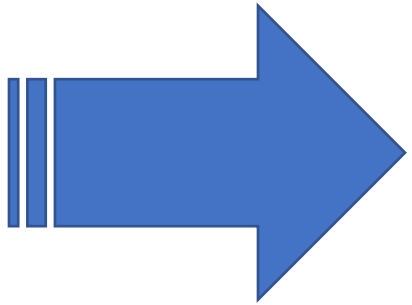
Total time per iteration:
~7 hours
Total coffee consumption: 2lt

Datetime Objects

DataType	Description	Related Methods	RelatedIndexes	Base Object
Timestamp	Basic Timestamp data type. Represents a point in time	to_datetime, date_range, bdate_range	DatetimeIndex	Pd.Timestamp
TimeDelta	Absolute time duration. Represents duration between 2 given times	to_timedelta, timedelta_range	TimedeltaIndex	Pd.Timedelta
Period	Represents a span of time from a point in time.	period_range	PeriodIndex	Pd.Period
DateOffset	Represents a span of time.			Pd.DateOffset

Datetime Objects

DatetimeIndex
Series.dt



Property	Description
year	The year of the datetime
month	The month of the datetime
day	The days of the datetime
hour	The hour of the datetime
minute	The minutes of the datetime
second	The seconds of the datetime
microsecond	The microseconds of the datetime
nanosecond	The nanoseconds of the datetime
date	Returns datetime.date (does not contain timezone information)
time	Returns datetime.time (does not contain timezone information)
timetz	Returns datetime.time as local time with timezone information
dayofyear	The ordinal day of year
weekofyear	The week ordinal of the year
week	The week ordinal of the year
dayofweek	The number of the day of the week with Monday=0, Sunday=6
weekday	The number of the day of the week with Monday=0, Sunday=6
day_name	The name of the day in a week (ex: Friday)
quarter	Quarter of the date: Jan-Mar = 1, Apr-Jun = 2, etc.
days_in_month	The number of days in the month of the datetime
is_month_start	Logical indicating if first day of month (defined by frequency)
is_month_end	Logical indicating if last day of month (defined by frequency)
is_quarter_start	Logical indicating if first day of quarter (defined by frequency)
is_quarter_end	Logical indicating if last day of quarter (defined by frequency)
is_year_start	Logical indicating if first day of year (defined by frequency)
is_year_end	Logical indicating if last day of year (defined by frequency)
is_leap_year	Logical indicating if the date belongs to a leap year

Creating Time Series

Loading data

example.csv - Notepad

File Edit Format View Help

```
,Customer Number,Customer Name,2016,2017,Percent Growth,Jan Units,Month,Day,Year,Active,Start_Date
0,10002,Quest Industries,125000,162500,0.3,500,1,10,2015,TRUE,1/10/2015
1,552278,Smith Plumbing,920000,1012000,0.1,700,6,15,2014,TRUE,6/15/2014
2,23477,ACME Industrial,50000,62500,0.25,125,3,29,2016,TRUE,3/29/2016
3,24900,Brekke LTD,350000,490000,0.04,75,10,27,2015,TRUE,10/27/2015
4,651029,Harbor Co,15000,12750,-0.15,NaN,2,2,2014,FALSE,2/2/2014
```

```
1 sample_data = pd.read_csv('data/sample_data.csv', parse_dates=['Start_Date'], index_col=0)
2 sample_data
```

	Customer Number	Customer Name	2016	2017	Percent Growth	Jan Units	Month	Day	Year	Active	Start_Date
0	10002	Quest Industries	125000	162500	0.30	500.0	1	10	2015	True	2015-01-10
1	552278	Smith Plumbing	920000	1012000	0.10	700.0	6	15	2014	True	2014-06-15
2	23477	ACME Industrial	50000	62500	0.25	125.0	3	29	2016	True	2016-03-29
3	24900	Brekke LTD	350000	490000	0.04	75.0	10	27	2015	True	2015-10-27
4	651029	Harbor Co	15000	12750	-0.15	NaN	2	2	2014	False	2014-02-02

- When importing data to pandas you can load it directly as a time series (object Series fiill with timesatmp)
- The same response is returned if you pass a Series pd.to_datetime method.

Creating Time Indexes

DatetimeIndex

```
1 sample_data[['Start_Date']].dtypes
```

```
Start_Date    datetime64[ns]
dtype: object
```

```
1 dates = ['2012-05-01', pd.Timestamp('2012-05-02'), dtm(2012, 5, 3), '2012/05/04']
```

```
1 pd.to_datetime(dates)
```

```
DatetimeIndex(['2012-05-01', '2012-05-02', '2012-05-03', '2012-05-04'], dtype='datetime64[ns]', freq=None)
```

```
1 pd.to_datetime(dates, infer_datetime_format=True)
```

```
DatetimeIndex(['2012-05-01', '2012-05-02', '2012-05-03', '2012-05-04'], dtype='datetime64[ns]', freq=None)
```

```
1 pd.date_range(start='2012-05-01', periods=4, freq='D')
```

```
DatetimeIndex(['2012-05-01', '2012-05-02', '2012-05-03', '2012-05-04'], dtype='datetime64[ns]', freq='D')
```

Creating Time Indexes

TimedeltaIndex

```
1 pd.Timedelta('1D')
```

```
Timedelta('1 days 00:00:00')
```

```
1 pd.Timedelta('1Day')
```

```
Timedelta('1 days 00:00:00')
```

```
1 pd.Timedelta('1H')
```

```
Timedelta('0 days 01:00:00')
```

```
1 pd.Timedelta('1H30T')
```

```
Timedelta('0 days 01:30:00')
```

```
1 pd.Timestamp('2019/10/07')+pd.Timedelta('1Day')
```

```
Timestamp('2019-10-08 00:00:00')
```

```
1 dtm(2019,10,7)+pd.Timedelta('1H20T')
```

```
datetime.datetime(2019, 10, 7, 1, 20)
```

```
1 dtm(2019,10,7)-pd.Timedelta('1H20T')
```

```
datetime.datetime(2019, 10, 6, 22, 40)
```

```
1 pd.Timestamp(2019,10,7)-\
2 pd.Timestamp(2019,10,6)+\
3 pd.Timedelta('1H10T')
```

```
Timedelta('1 days 01:10:00')
```

Creating Time Indexes

TimedeltaIndex

```
1 pd.date_range(start='2019/10/07', periods=5, freq='D')+\
2 pd.Timedelta('1H30T10S')
```

```
DatetimeIndex(['2019-10-07 01:30:10', '2019-10-08 01:30:10',
               '2019-10-09 01:30:10', '2019-10-10 01:30:10',
               '2019-10-11 01:30:10'],
              dtype='datetime64[ns]', freq='D')
```

```
1 pd.date_range(start='2019/10/07', periods=5, freq='H')+\
2 pd.timedelta_range(start='1Day', end='2Days', freq='6H')
```

```
DatetimeIndex(['2019-10-08 00:00:00', '2019-10-08 07:00:00',
               '2019-10-08 14:00:00', '2019-10-08 21:00:00',
               '2019-10-09 04:00:00'],
              dtype='datetime64[ns]', freq='7H')
```

Creating Time Indexes

PerioIndex

```
display(HTML('<pre style="font-size: 2em;">{}</pre>'.\
             format(pd.Period('2019/10'))))
```

```
2019-10
```

```
display(HTML('<pre style="font-size: 2em;">{}</pre>'.\
             format(pd.Period('2019/10', freq='30T'))))
```

```
2019-10-01 00:00
```

```
per = pd.Period('2019/10', freq='30T')
display(HTML('<pre style="font-size: 2em;">{}</pre>'.\
             format(per.start_time)))
display(HTML('<pre style="font-size: 2em;">{}</pre>'.\
             format(per.end_time)))
```

```
2019-10-01 00:00:00

2019-10-01 00:29:59.999999999
```

Creating Time Indexes

PerioIndex

```
1 display(HTML('<pre style="font-size: 1.7em;">{}</pre>'.\
2              format(pd.period_range('2019/01', periods=10, freq='30T'))))
```

```
PeriodIndex(['2019-01-01 00:00', '2019-01-01 00:30', '2019-01-01 01:00',
             '2019-01-01 01:30', '2019-01-01 02:00', '2019-01-01 02:30',
             '2019-01-01 03:00', '2019-01-01 03:30', '2019-01-01 04:00',
             '2019-01-01 04:30'],
            dtype='period[30T]', freq='30T')
```

Creating Time Indexes

PerioIndex

```
1 per = pd.Period('2019/10', freq='30T')
2 display(HTML('<span style="font-size: 2em;">{}</span>'.\
              format(per.start_time)))
4 display(HTML('<span style="font-size: 2em;">{}</span>'.\
              format(per.end_time)))
```

2019-10-01 00:00:00

2019-10-01 00:29:59.999999999

Creating Time Indexes

DeteOffset

```
# adds the international labor day and colombian independence
# holidays for a few years
holidays = pd.date_range('2016-05-01', periods=4, freq='365D')\
    .append(pd.date_range('2016-07-20', periods=4, freq='365D'))
bday_col = pd.offsets.CustomBusinessDay(holidays=holidays)
```

```
test_dates = pd.Timestamp('2019-04-30')
display(HTML('<span style="font-size: 2em;">{}</span>'.\
             format(test_dates.day_name())))
display(HTML('<span style="font-size: 2em;">{}</span>'.\
             format((test_dates + pd.DateOffset(days=2)).day_name())))
display(HTML('<span style="font-size: 2em;">{}</span>'.\
             format((test_dates + 2 * bday_col).day_name())))
```

Tuesday

Thursday

Friday

Creating Time Indexes

DeteOffset

```
1 offset = pd.offsets.BusinessHour(start='09:00')
2 display(HTML('<pre style="font-size: 2em;">{}</pre>'.\
3              format(pd.Timestamp('2019-04-30'))))
4 display(HTML('<pre style="font-size: 2em;">{}</pre>'.\
               format(offset.rollback(pd.Timestamp('2019-04-30')))))
6 display(HTML('<pre style="font-size: 2em;">{}</pre>'.\
               format(offset.rollforward(pd.Timestamp('2019-04-30')))))
```

2019-04-30 00:00:00

2019-04-29 17:00:00

2019-04-30 09:00:00

Time Based Querying

```
1 sample_data.loc['2018-01':'2018-03']
```

date	name	sku	quantity	unit price	ext price	date
2018-01-01 07:21:51	Barton LLC	B1-20000	39	86.69	3380.91	2018-01-01 07:21:51
2018-01-01 10:00:47	Trantow-Barrows	S2-77896	-1	63.16	-63.16	2018-01-01 10:00:47
2018-01-01 13:24:58	Kulas Inc	B1-69924	23	90.70	2086.10	2018-01-01 13:24:58
2018-01-01 15:05:22	Kassulke, Ondricka and Metz	S1-65481	41	21.05	863.05	2018-01-01 15:05:22
2018-01-01 23:26:55	Jerde-Hilpert	S2-34077	6	83.21	499.26	2018-01-01 23:26:55
2018-01-02 10:07:15	Trantow-Barrows	S2-77896	17	87.63	1489.71	2018-01-02 10:07:15
2018-01-02 10:57:23	Kulas Inc	B1-65551	2	31.10	62.20	2018-01-02 10:57:23
2018-01-03 06:32:11	Koepp Ltd	S1-30248	8	33.25	266.00	2018-01-03 06:32:11
2018-01-03 11:29:02	Trantow-Barrows	S1-50961	22	84.09	1849.98	2018-01-03 11:29:02
2018-01-03 19:07:37	Fritsch, Russel and Anderson	S2-82423	14	81.92	1146.88	2018-01-03 19:07:37
2018-01-03 19:39:53	Kiehn-Spinka	S2-82423	15	67.74	1016.10	2018-01-03 19:39:53
2018-01-04 00:02:36	Keeling LLC	S2-00301	7	20.26	141.82	2018-01-04 00:02:36
2018-01-04 06:51:53	Frami, Hills and Schmidt	S2-23246	6	61.31	367.86	2018-01-04 06:51:53
2018-01-04 07:53:01	Kassulke, Ondricka and Metz	S2-10342	17	12.44	211.48	2018-01-04 07:53:01
2018-01-04 08:57:48	Fritsch, Russel and Anderson	B1-53102	23	71.56	1645.88	2018-01-04 08:57:48
2018-01-04 11:34:58	Stokes LLC	S1-06532	34	71.51	2431.34	2018-01-04 11:34:58

Time Based Querying

```
1 sample_data.query('date >= "2018-01-01 09:00:00" and date <= "2018-03-31 17:00:00"')
```

date	name	sku	quantity	unit price	ext price	date
2018-01-01 10:00:47	Trantow-Barrows	S2-77896	-1	63.16	-63.16	2018-01-01 10:00:47
2018-01-01 13:24:58	Kulas Inc	B1-69924	23	90.70	2086.10	2018-01-01 13:24:58
2018-01-01 15:05:22	Kassulke, Ondricka and Metz	S1-65481	41	21.05	863.05	2018-01-01 15:05:22
2018-01-01 23:26:55	Jerde-Hilpert	S2-34077	6	83.21	499.26	2018-01-01 23:26:55
2018-01-02 10:07:15	Trantow-Barrows	S2-77896	17	87.63	1489.71	2018-01-02 10:07:15
2018-01-02 10:57:23	Kulas Inc	B1-65551	2	31.10	62.20	2018-01-02 10:57:23
2018-01-03 06:32:11	Koepp Ltd	S1-30248	8	33.25	266.00	2018-01-03 06:32:11
2018-01-03 11:29:02	Trantow-Barrows	S1-50961	22	84.09	1849.98	2018-01-03 11:29:02
2018-01-03 19:07:37	Fritsch, Russel and Anderson	S2-82423	14	81.92	1146.88	2018-01-03 19:07:37
2018-01-03 19:39:53	Kiehn-Spinka	S2-82423	15	67.74	1016.10	2018-01-03 19:39:53
2018-01-04 00:02:36	Keeling LLC	S2-00301	7	20.26	141.82	2018-01-04 00:02:36
2018-01-04 06:51:53	Frami, Hills and Schmidt	S2-23246	6	61.31	367.86	2018-01-04 06:51:53
2018-01-04 07:53:01	Kassulke, Ondricka and Metz	S2-10342	17	12.44	211.48	2018-01-04 07:53:01
2018-01-04 08:57:48	Fritsch, Russel and Anderson	B1-53102	23	71.56	1645.88	2018-01-04 08:57:48
2018-01-04 11:34:58	Stokes LLC	S1-06532	34	71.51	2431.34	2018-01-04 11:34:58
2018-01-04 19:59:02	Kuhn-Gusikowski	S1-30248	14	72.75	1018.50	2018-01-04 19:59:02
2018-01-04 22:14:32	Stokes LLC	B1-50809	14	16.23	227.22	2018-01-04 22:14:32
2018-01-05 15:12:16	Herman LLC	S1-82801	10	94.30	943.00	2018-01-05 15:12:16
2018-01-05 22:39:21	Jerde-Hilpert	S2-00301	33	31.18	1028.94	2018-01-05 22:39:21

Time Based Querying

```
1 sample_data.loc[dtm(2018, 1, 1):'2018/03/31 17']
```

date	name	sku	quantity	unit price	ext price	date
2018-01-01 07:21:51	Barton LLC	B1-20000	39	86.69	3380.91	2018-01-01 07:21:51
2018-01-01 10:00:47	Trantow-Barrows	S2-77896	-1	63.16	-63.16	2018-01-01 10:00:47
2018-01-01 13:24:58	Kulas Inc	B1-69924	23	90.70	2086.10	2018-01-01 13:24:58
2018-01-01 15:05:22	Kassulke, Ondricka and Metz	S1-65481	41	21.05	863.05	2018-01-01 15:05:22
2018-01-01 23:26:55	Jerde-Hilpert	S2-34077	6	83.21	499.26	2018-01-01 23:26:55
2018-01-02 10:07:15	Trantow-Barrows	S2-77896	17	87.63	1489.71	2018-01-02 10:07:15
2018-01-02 10:57:23	Kulas Inc	B1-65551	2	31.10	62.20	2018-01-02 10:57:23
2018-01-03 06:32:11	Koepp Ltd	S1-30248	8	33.25	266.00	2018-01-03 06:32:11
2018-01-03 11:29:02	Trantow-Barrows	S1-50961	22	84.09	1849.98	2018-01-03 11:29:02
2018-01-03 19:07:37	Fritsch, Russel and Anderson	S2-82423	14	81.92	1146.88	2018-01-03 19:07:37
2018-01-03 19:39:53	Kiehn-Spinka	S2-82423	15	67.74	1016.10	2018-01-03 19:39:53
2018-01-04 00:02:36	Keeling LLC	S2-00301	7	20.26	141.82	2018-01-04 00:02:36
2018-01-04 06:51:53	Frami, Hills and Schmidt	S2-23246	6	61.31	367.86	2018-01-04 06:51:53
2018-01-04 07:53:01	Kassulke, Ondricka and Metz	S2-10342	17	12.44	211.48	2018-01-04 07:53:01

Time Based Querying

```
1 sample_data.loc['2018-01-01 09:00:00':'2018-03-31 17:00:00']
```

date	name	sku	quantity	unit price	ext price	date
2018-01-01 10:00:47	Trantow-Barrows	S2-77896	-1	63.16	-63.16	2018-01-01 10:00:47
2018-01-01 13:24:58	Kulas Inc	B1-69924	23	90.70	2086.10	2018-01-01 13:24:58
2018-01-01 15:05:22	Kassulke, Ondricka and Metz	S1-65481	41	21.05	863.05	2018-01-01 15:05:22
2018-01-01 23:26:55	Jerde-Hilpert	S2-34077	6	83.21	499.26	2018-01-01 23:26:55
2018-01-02 10:07:15	Trantow-Barrows	S2-77896	17	87.63	1489.71	2018-01-02 10:07:15
2018-01-02 10:57:23	Kulas Inc	B1-65551	2	31.10	62.20	2018-01-02 10:57:23
2018-01-03 06:32:11	Koepp Ltd	S1-30248	8	33.25	266.00	2018-01-03 06:32:11
2018-01-03 11:29:02	Trantow-Barrows	S1-50961	22	84.09	1849.98	2018-01-03 11:29:02
2018-01-03 19:07:37	Fritsch, Russel and Anderson	S2-82423	14	81.92	1146.88	2018-01-03 19:07:37
2018-01-03 19:39:53	Kiehn-Spinka	S2-82423	15	67.74	1016.10	2018-01-03 19:39:53
2018-01-04 00:02:36	Keeling LLC	S2-00301	7	20.26	141.82	2018-01-04 00:02:36
2018-01-04 06:51:53	Frami, Hills and Schmidt	S2-23246	6	61.31	367.86	2018-01-04 06:51:53

Creating Rolling Windows Based on Time

```
1 sample_data[['quantity']].rolling(7, center=True).mean()
```

date	quantity
2018-01-01 07:21:51	NaN
2018-01-01 10:00:47	NaN
2018-01-01 13:24:58	NaN
2018-01-01 15:05:22	18.142857
2018-01-01 23:26:55	13.714286
2018-01-02 10:07:15	17.000000
2018-01-02 10:57:23	15.714286
2018-01-03 06:32:11	12.000000
2018-01-03 11:29:02	12.142857
2018-01-03 19:07:37	10.571429
2018-01-03 19:39:53	12.714286

Aggregations and Custom Operations

```
1 start = dtm.now()
2 daily_data = sample_data.resample('1D').sum()
3 display(HTML('<pre style="font-size:2em">{}</pre>'.format(dtm.now()-start)))
4 daily_data
```

0:00:00.010306

date	quantity	unit price	ext price
2018-01-01	108	344.81	6766.16
2018-01-02	19	118.73	1551.91
2018-01-03	59	267.00	4278.96
2018-01-04	115	326.06	6044.10
2018-01-05	43	125.48	1971.94
2018-01-06	104	277.03	7711.50
2018-01-07	5	14.75	73.75
2018-01-08	94	247.76	5179.96
2018-01-09	59	158.64	1632.52

Aggregations and Custom Operations

```
1 start = dtm.now()
2 monthly_data = sample_data.groupby('name').resample('1M').mean()
3 display(HTML('<pre style="font-size:2em">{}</pre>'.format(dtm.now()-start)))
4 monthly_data
```

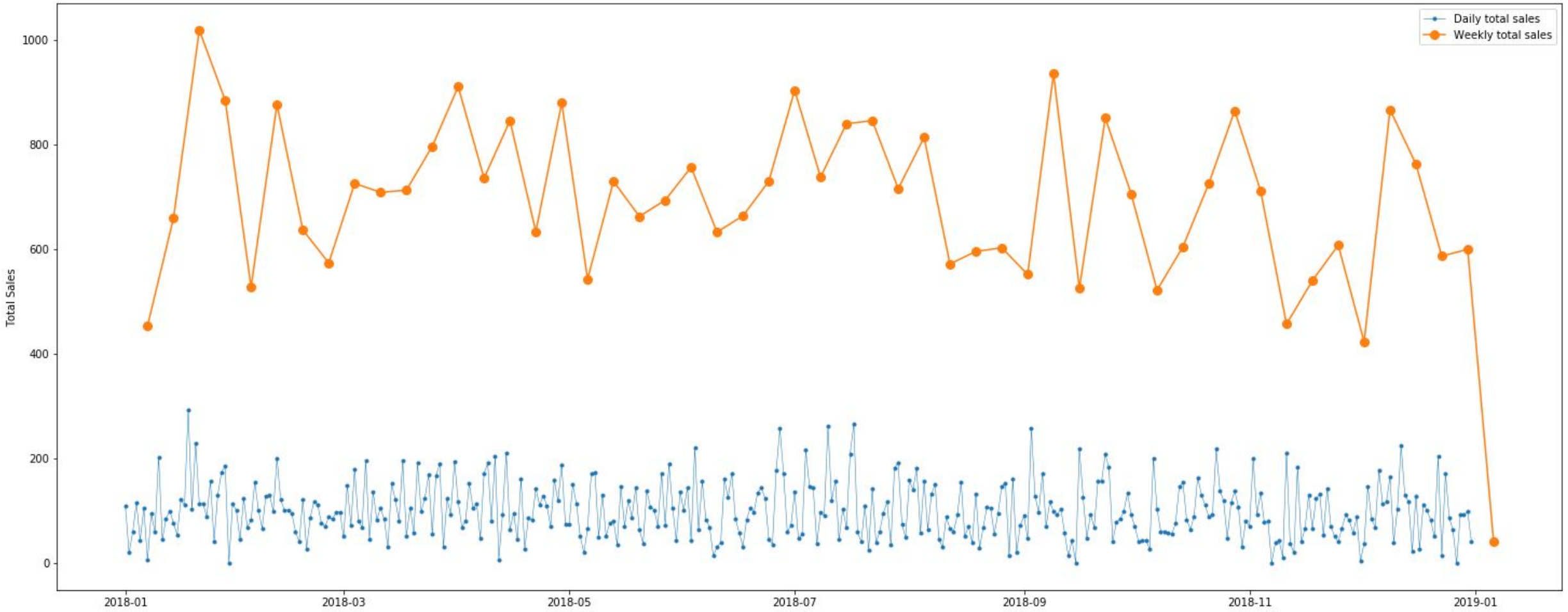
```
0:00:00.141310
```

		quantity	unit price	ext price
name	date			
Barton LLC	2018-01-31	14.125000	52.787500	772.196250
	2018-02-28	24.571429	65.202857	1745.432857
	2018-03-31	20.333333	52.453333	1171.176667
	2018-04-30	24.200000	49.400000	1147.420000
	2018-05-31	26.250000	51.661250	1277.521250
	2018-06-30	26.125000	51.691250	1307.966250
	2018-07-31	35.500000	47.730000	1687.620000
	2018-08-31	28.333333	56.796667	1461.788333
	2018-09-30	31.125000	56.666250	1756.701250
	2018-10-31	23.142857	51.558571	1335.954286

Plotting Results

```
# Start and end of the date range to extract
start, end = '2018-01', '2018-12'
# Plot daily, weekly resampled, and 7-day rolling mean time series together
fig, ax = plt.subplots(figsize=(25, 10))
ax.plot(sample_data.loc[start:end, 'quantity'].resample('1D').sum(),
marker='.', linestyle='-', linewidth=0.5, label='Daily total sales')
ax.plot(sample_data.loc[start:end, 'quantity'].resample('1W').sum(),
marker='o', markersize=8, linestyle='-', label='Weekly total sales')
ax.set_ylabel('Total Sales')
ax.legend();
```

Plotting Results



Thanks!


@luisdcamachog


luis.camacho@cyxtera.com


https://www.linkedin.com/in/luisdcamachog/


https://github.com/LuisDavidCamacho