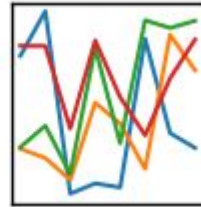


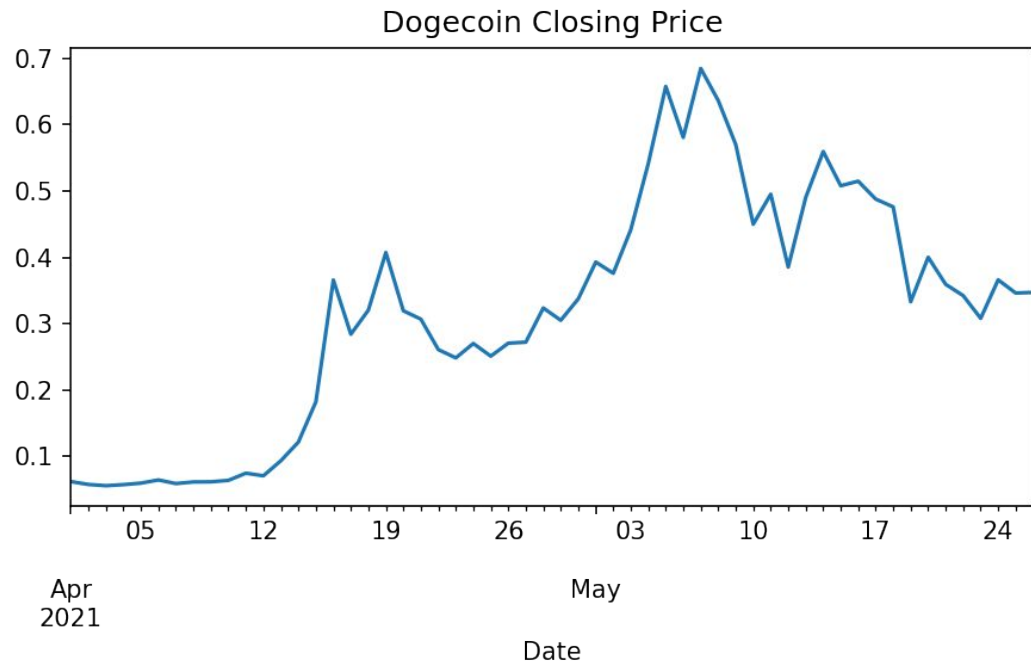
Rolling Window Calculations with *pandas*

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Rolling Windows



When tracking a statistic over time, there can be a lot of short-term fluctuations.

Sometimes it is better to “zoom out” and look for the general trend rather than the exact value at any point in time.

Rolling Windows

| | Date | Close |
|-----|------------|----------|
| 356 | 2021-05-17 | 0.487892 |
| 357 | 2021-05-18 | 0.476115 |
| 358 | 2021-05-19 | 0.333123 |
| 359 | 2021-05-20 | 0.400194 |
| 360 | 2021-05-21 | 0.359382 |
| 361 | 2021-05-22 | 0.342371 |
| 362 | 2021-05-23 | 0.308071 |
| 363 | 2021-05-24 | 0.366162 |
| 364 | 2021-05-25 | 0.346302 |
| 365 | 2021-05-26 | 0.347167 |

A common way to uncover trends is to look at a rolling average.

For example, for each day, we can average the last 7 closing prices.

This smooths out some of the “noise” in the closing price.

Rolling Windows

| | Date | Close | rolling_7_day_average |
|-----|------------|----------|-----------------------|
| 356 | 2021-05-17 | 0.487892 | |
| 357 | 2021-05-18 | 0.476115 | |
| 358 | 2021-05-19 | 0.333123 | |
| 359 | 2021-05-20 | 0.400194 | |
| 360 | 2021-05-21 | 0.359382 | |
| 361 | 2021-05-22 | 0.342371 | |
| 362 | 2021-05-23 | 0.308071 | |
| 363 | 2021-05-24 | 0.366162 | |
| 364 | 2021-05-25 | 0.346302 | |
| 365 | 2021-05-26 | 0.347167 | |

Rolling Windows

| | Date | Close | rolling_7_day_average |
|-----|------------|----------|-----------------------|
| 356 | 2021-05-17 | 0.487892 | |
| 357 | 2021-05-18 | 0.476115 | |
| 358 | 2021-05-19 | 0.333123 | |
| 359 | 2021-05-20 | 0.400194 | |
| 360 | 2021-05-21 | 0.359382 | |
| 361 | 2021-05-22 | 0.342371 | |
| 362 | 2021-05-23 | 0.308071 | |
| 363 | 2021-05-24 | 0.366162 | |
| 364 | 2021-05-25 | 0.346302 | |
| 365 | 2021-05-26 | 0.347167 | |

Rolling Windows

| | Date | Close | rolling_7_day_average |
|-----|------------|----------|-----------------------|
| 356 | 2021-05-17 | 0.487892 | |
| 357 | 2021-05-18 | 0.476115 | |
| 358 | 2021-05-19 | 0.333123 | |
| 359 | 2021-05-20 | 0.400194 | |
| 360 | 2021-05-21 | 0.359382 | |
| 361 | 2021-05-22 | 0.342371 | |
| 362 | 2021-05-23 | 0.308071 | |
| 363 | 2021-05-24 | 0.366162 | |
| 364 | 2021-05-25 | 0.346302 | |
| 365 | 2021-05-26 | 0.347167 | |

Average Close in this
window = \$0.352807



Rolling Windows

| | Date | Close | rolling_7_day_average |
|-----|------------|----------|-----------------------|
| 356 | 2021-05-17 | 0.487892 | |
| 357 | 2021-05-18 | 0.476115 | |
| 358 | 2021-05-19 | 0.333123 | |
| 359 | 2021-05-20 | 0.400194 | |
| 360 | 2021-05-21 | 0.359382 | |
| 361 | 2021-05-22 | 0.342371 | |
| 362 | 2021-05-23 | 0.308071 | |
| 363 | 2021-05-24 | 0.366162 | |
| 364 | 2021-05-25 | 0.346302 | |
| 365 | 2021-05-26 | 0.347167 | |


Average Close in this
window = \$0.352807

0.352807

Rolling Windows

| | Date | Close | rolling_7_day_average |
|-----|------------|----------|-----------------------|
| 356 | 2021-05-17 | 0.487892 | |
| 357 | 2021-05-18 | 0.476115 | |
| 358 | 2021-05-19 | 0.333123 | |
| 359 | 2021-05-20 | 0.400194 | |
| 360 | 2021-05-21 | 0.359382 | |
| 361 | 2021-05-22 | 0.342371 | |
| 362 | 2021-05-23 | 0.308071 | |
| 363 | 2021-05-24 | 0.366162 | |
| 364 | 2021-05-25 | 0.346302 | |
| 365 | 2021-05-26 | 0.347167 | 0.352807 |

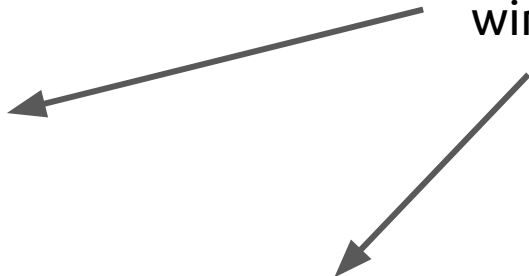
Average Close in this
window = \$0.350801



Rolling Windows

| | Date | Close | rolling_7_day_average |
|-----|------------|----------|-----------------------|
| 356 | 2021-05-17 | 0.487892 | |
| 357 | 2021-05-18 | 0.476115 | |
| 358 | 2021-05-19 | 0.333123 | |
| 359 | 2021-05-20 | 0.400194 | |
| 360 | 2021-05-21 | 0.359382 | |
| 361 | 2021-05-22 | 0.342371 | |
| 362 | 2021-05-23 | 0.308071 | |
| 363 | 2021-05-24 | 0.366162 | |
| 364 | 2021-05-25 | 0.346302 | 0.350801 |
| 365 | 2021-05-26 | 0.347167 | 0.352807 |


Average Close in this
window = \$0.350801



Rolling Windows

| | Date | Close | rolling_7_day_average |
|-----|------------|----------|-----------------------|
| 356 | 2021-05-17 | 0.487892 | |
| 357 | 2021-05-18 | 0.476115 | |
| 358 | 2021-05-19 | 0.333123 | |
| 359 | 2021-05-20 | 0.400194 | |
| 360 | 2021-05-21 | 0.359382 | |
| 361 | 2021-05-22 | 0.342371 | |
| 362 | 2021-05-23 | 0.308071 | |
| 363 | 2021-05-24 | 0.366162 | |
| 364 | 2021-05-25 | 0.346302 | 0.350801 |
| 365 | 2021-05-26 | 0.347167 | 0.352807 |

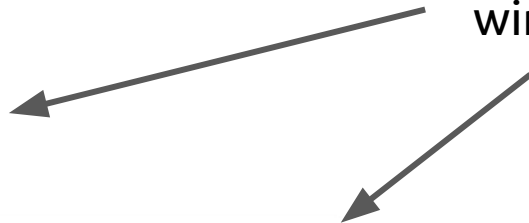
Average Close in this
window = \$0.369345



Rolling Windows

| | Date | Close | rolling_7_day_average |
|-----|------------|----------|-----------------------|
| 356 | 2021-05-17 | 0.487892 | |
| 357 | 2021-05-18 | 0.476115 | |
| 358 | 2021-05-19 | 0.333123 | |
| 359 | 2021-05-20 | 0.400194 | |
| 360 | 2021-05-21 | 0.359382 | |
| 361 | 2021-05-22 | 0.342371 | |
| 362 | 2021-05-23 | 0.308071 | |
| 363 | 2021-05-24 | 0.366162 | 0.369345 |
| 364 | 2021-05-25 | 0.346302 | 0.350801 |
| 365 | 2021-05-26 | 0.347167 | 0.352807 |

Average Close in this
window = \$0.369345




Rolling Windows

| | Date | Close | rolling_7_day_average |
|------------|------------|----------|-----------------------|
| 356 | 2021-05-17 | 0.487892 | 0.491621 |
| 357 | 2021-05-18 | 0.476115 | 0.488890 |
| 358 | 2021-05-19 | 0.333123 | 0.481425 |
| 359 | 2021-05-20 | 0.400194 | 0.468542 |
| 360 | 2021-05-21 | 0.359382 | 0.439939 |
| 361 | 2021-05-22 | 0.342371 | 0.416282 |
| 362 | 2021-05-23 | 0.308071 | 0.386735 |
| 363 | 2021-05-24 | 0.366162 | 0.369345 |
| 364 | 2021-05-25 | 0.346302 | 0.350801 |
| 365 | 2021-05-26 | 0.347167 | 0.352807 |

Continue this for all
dates to fill in the rest
of the values.

```
doge['rolling_7_day_average'] = doge.rolling('7d', on = 'Date')['Close'].mean()
```

| | Date | Close |
|-----|------------|----------|
| 356 | 2021-05-17 | 0.487892 |
| 357 | 2021-05-18 | 0.476115 |
| 358 | 2021-05-19 | 0.333123 |
| 359 | 2021-05-20 | 0.400194 |
| 360 | 2021-05-21 | 0.359382 |
| 361 | 2021-05-22 | 0.342371 |
| 362 | 2021-05-23 | 0.308071 |
| 363 | 2021-05-24 | 0.366162 |
| 364 | 2021-05-25 | 0.346302 |
| 365 | 2021-05-26 | 0.347167 |



Tell *pandas* you
want to create a
rolling window.

```
doge['rolling_7_day_average'] = doge.rolling('7d', on = 'Date')['Close'].mean()
```

| | Date | Close |
|-----|------------|----------|
| 356 | 2021-05-17 | 0.487892 |
| 357 | 2021-05-18 | 0.476115 |
| 358 | 2021-05-19 | 0.333123 |
| 359 | 2021-05-20 | 0.400194 |
| 360 | 2021-05-21 | 0.359382 |
| 361 | 2021-05-22 | 0.342371 |
| 362 | 2021-05-23 | 0.308071 |
| 363 | 2021-05-24 | 0.366162 |
| 364 | 2021-05-25 | 0.346302 |
| 365 | 2021-05-26 | 0.347167 |



How big of a
window?
In this case, 7 days.

```
doge['rolling_7_day_average'] = doge.rolling('7d', on = 'Date')['Close'].mean()
```

| | Date | Close |
|-----|------------|----------|
| 356 | 2021-05-17 | 0.487892 |
| 357 | 2021-05-18 | 0.476115 |
| 358 | 2021-05-19 | 0.333123 |
| 359 | 2021-05-20 | 0.400194 |
| 360 | 2021-05-21 | 0.359382 |
| 361 | 2021-05-22 | 0.342371 |
| 362 | 2021-05-23 | 0.308071 |
| 363 | 2021-05-24 | 0.366162 |
| 364 | 2021-05-25 | 0.346302 |
| 365 | 2021-05-26 | 0.347167 |

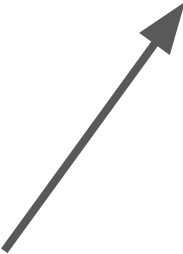
which column to
build the window
based on



```
doge['rolling_7_day_average'] = doge.rolling('7d', on = 'Date')['Close'].mean()
```

| | Date | Close |
|-----|------------|----------|
| 356 | 2021-05-17 | 0.487892 |
| 357 | 2021-05-18 | 0.476115 |
| 358 | 2021-05-19 | 0.333123 |
| 359 | 2021-05-20 | 0.400194 |
| 360 | 2021-05-21 | 0.359382 |
| 361 | 2021-05-22 | 0.342371 |
| 362 | 2021-05-23 | 0.308071 |
| 363 | 2021-05-24 | 0.366162 |
| 364 | 2021-05-25 | 0.346302 |
| 365 | 2021-05-26 | 0.347167 |

which column(s)
to aggregate



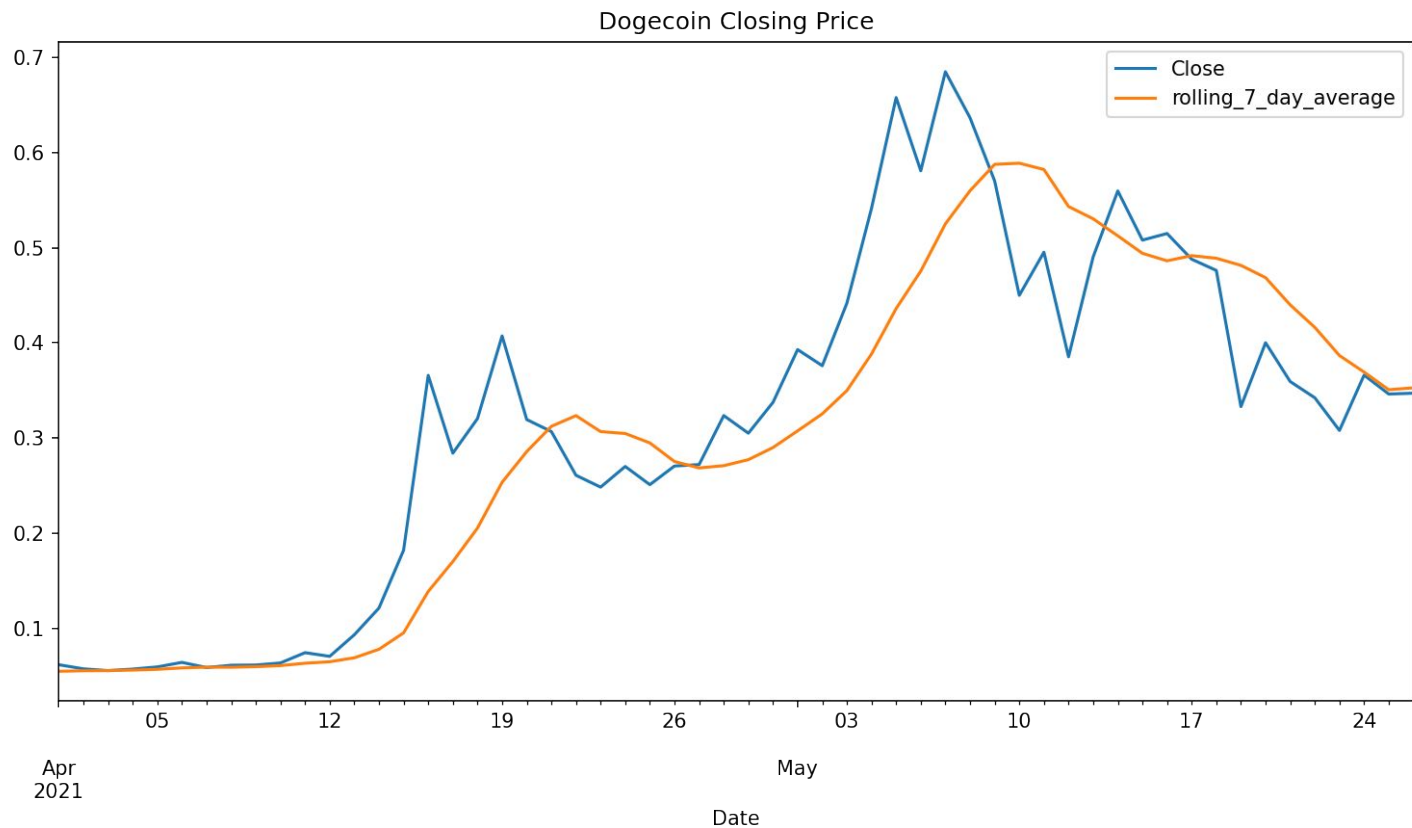

```
doge['rolling_7_day_average'] = doge.rolling('7d', on = 'Date')['Close'].mean()
```

| | Date | Close |
|-----|------------|----------|
| 356 | 2021-05-17 | 0.487892 |
| 357 | 2021-05-18 | 0.476115 |
| 358 | 2021-05-19 | 0.333123 |
| 359 | 2021-05-20 | 0.400194 |
| 360 | 2021-05-21 | 0.359382 |
| 361 | 2021-05-22 | 0.342371 |
| 362 | 2021-05-23 | 0.308071 |
| 363 | 2021-05-24 | 0.366162 |
| 364 | 2021-05-25 | 0.346302 |
| 365 | 2021-05-26 | 0.347167 |

how to aggregate



Rolling Windows

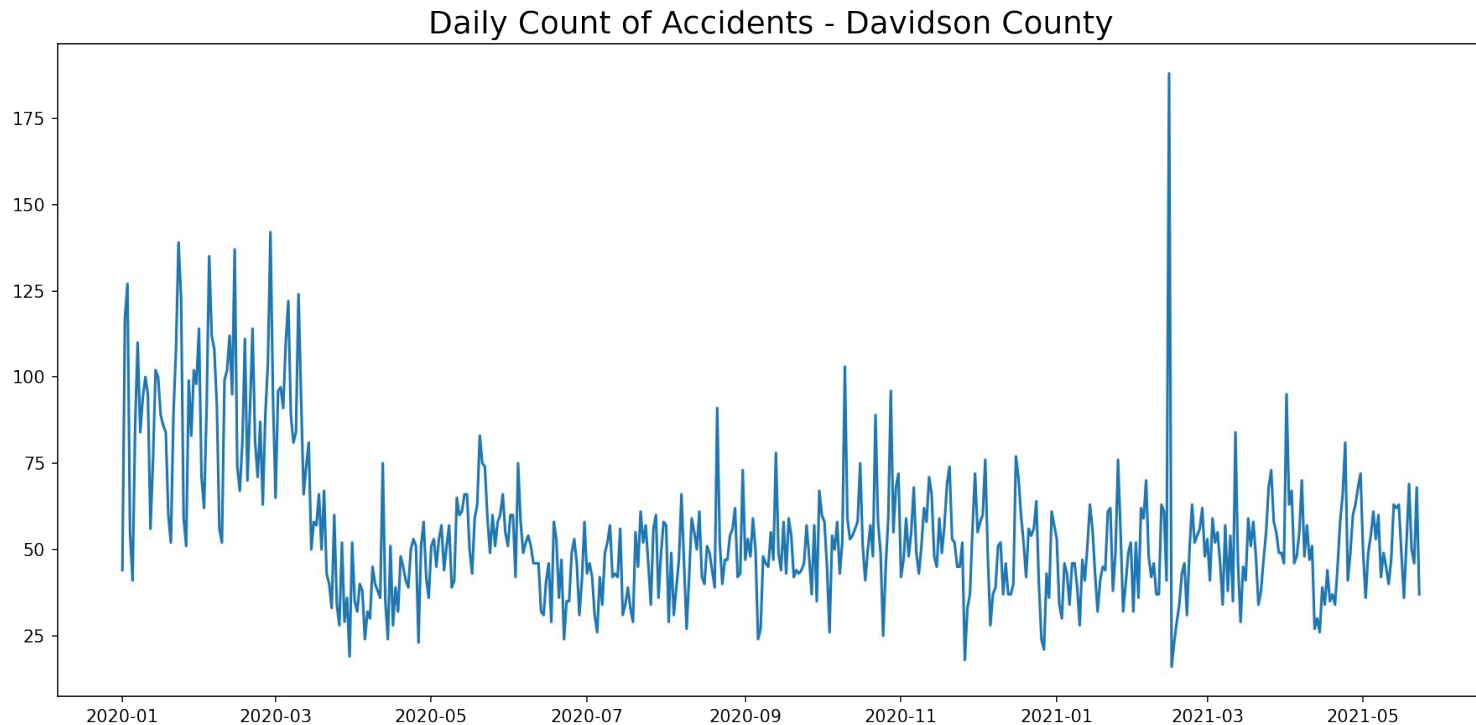


Rolling Windows - Example #2

Let's say we want to look for trends in the number of traffic accidents occurring in Davidson County over the last year and a half.

Rolling Windows - Example #2

Using raw daily counts results in a noisy picture.



Rolling Windows - Example #2

In this case, each record is a single accident, so we'll have to aggregate differently.

```
1 traffic.head()
```

| | Accident Number | Date and Time | Number of Motor Vehicles | Number of Injuries | Number of Fatalities | Property Damage |
|---------------|--------------------|------------------------|--------------------------------|--------------------------|----------------------------|--------------------|
| 199094 | 20200000247 | 2020-01-01 00:25:00 | 2.0 | 0.0 | 0 | Na |
| 216851 | 20200000511 | 2020-01-01 00:30:00 | 2.0 | 0.0 | 0 | Na |
| 206077 | 20200000105 | 2020-01-01 00:35:00 | 2.0 | 2.0 | 0 | Na |
| 215150 | 20200000203 | 2020-01-01 01:03:00 | 2.0 | 0.0 | 0 | Na |

Rolling Windows - Example #2

```
traffic['Date and Time'] = pd.to_datetime(traffic['Date and Time'])
```

```
traffic = traffic.sort_values('Date and Time')
```

```
1 traffic.head()
```

| | Accident Number | Date and Time | Number of Motor Vehicles | Number of Injuries | Number of Fatalities | Property Damage |
|--------|--------------------|------------------------|--------------------------------|--------------------------|----------------------------|--------------------|
| 199094 | 20200000247 | 2020-01-01 00:25:00 | 2.0 | 0.0 | 0 | Na |
| 216851 | 20200000511 | 2020-01-01 00:30:00 | 2.0 | 0.0 | 0 | Na |
| 206077 | 20200000105 | 2020-01-01 00:35:00 | 2.0 | 2.0 | 0 | Na |
| 215150 | 20200000203 | 2020-01-01 01:03:00 | 2.0 | 0.0 | 0 | Na |

Important: If working with datetime data, make sure that *pandas* is treating it as datetime.

Also, make sure that your dataframe is sorted by the column you are creating the rolling window on.

Rolling Windows - Example #2

```
traffic['Date and Time'] = pd.to_datetime(traffic['Date and Time'])

traffic = traffic.sort_values('Date and Time')

traffic['total_crashes_30_days'] = (traffic
                                   .rolling('30d', on = 'Date and Time')['Accident Number']
                                   .count())
```

1 traffic.head()

| | Accident Number | Date and Time | Number of Motor Vehicles | Number of Injuries | Number of Fatalities | Property Damage |
|--------|-----------------|---------------------|--------------------------|--------------------|----------------------|-----------------|
| 199094 | 20200000247 | 2020-01-01 00:25:00 | 2.0 | 0.0 | 0 | Na |
| 216851 | 20200000511 | 2020-01-01 00:30:00 | 2.0 | 0.0 | 0 | Na |
| 206077 | 20200000105 | 2020-01-01 00:35:00 | 2.0 | 2.0 | 0 | Na |
| 215150 | 20200000203 | 2020-01-01 01:03:00 | 2.0 | 0.0 | 0 | Na |

Total up the number of crashes that have occurred over the last 30 days for each entry.

Note: We are aggregating using *count* this time. This returns the number of non-NaN rows included in each window.

Rolling Windows - Example #2

```
traffic['Date and Time'] = pd.to_datetime(traffic['Date and Time'])

traffic = traffic.sort_values('Date and Time')

traffic['total_crashes_30_days'] = (traffic
                                   .rolling('30d', on = 'Date and Time')['Accident Number']
                                   .count())

traffic['rolling_daily_average'] = traffic['total_crashes_30_days'] / 30
```

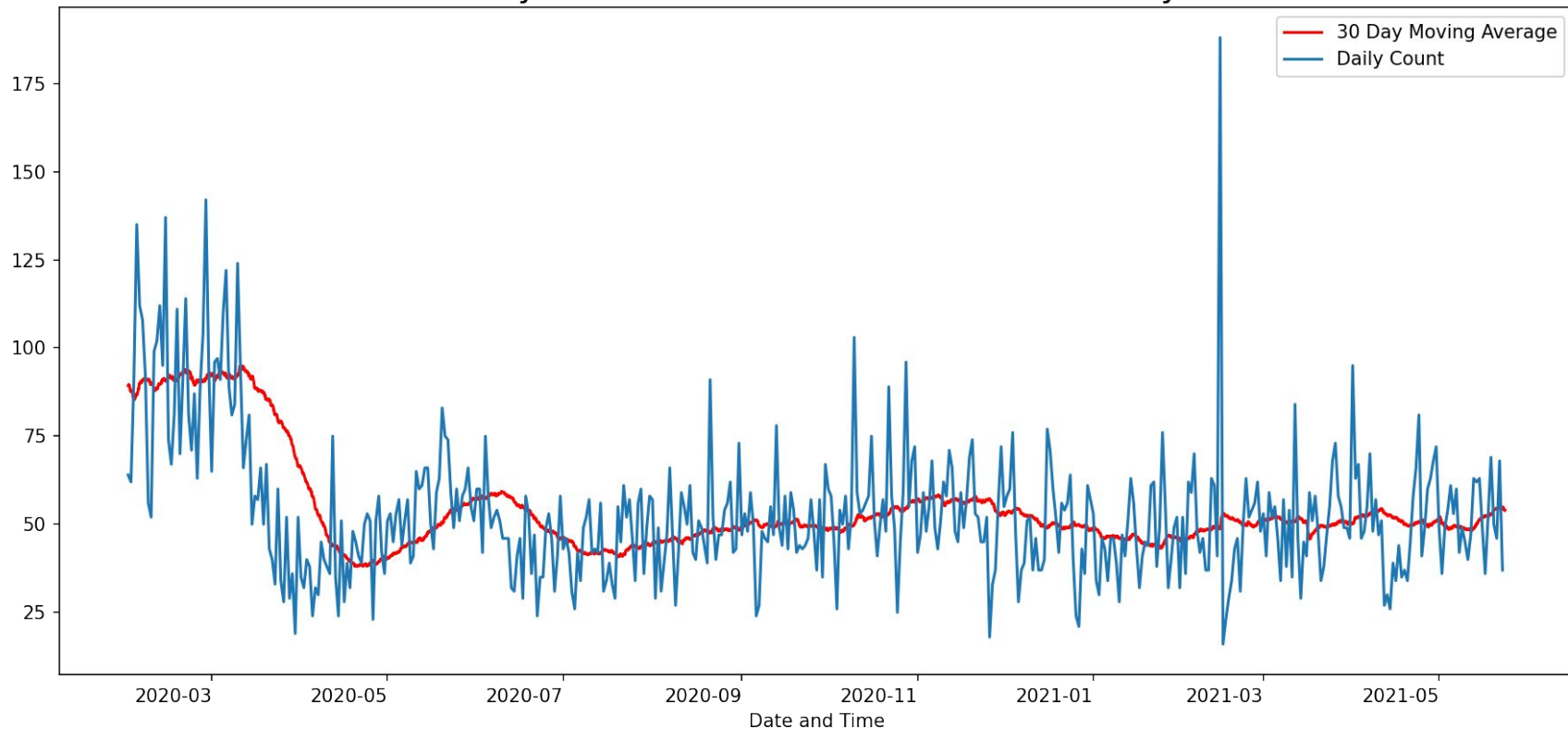
```
1 traffic.head()
```

| | Accident Number | Date and Time | Number of Motor Vehicles | Number of Injuries | Number of Fatalities | Property Damage |
|--------|-----------------|---------------------|--------------------------|--------------------|----------------------|-----------------|
| 199094 | 20200000247 | 2020-01-01 00:25:00 | 2.0 | 0.0 | 0 | Na |
| 216851 | 20200000511 | 2020-01-01 00:30:00 | 2.0 | 0.0 | 0 | Na |
| 206077 | 20200000105 | 2020-01-01 00:35:00 | 2.0 | 2.0 | 0 | Na |
| 215150 | 20200000203 | 2020-01-01 01:03:00 | 2.0 | 0.0 | 0 | Na |

Finally, divide by the number of days to get an average daily crashes for the last 30 days.

Rolling Windows - Example #2

Daily Count of Accidents - Davidson County



Rolling Windows - Example #2

```
traffic_by_precinct = (traffic
    .groupby('Precinct')
    .rolling('30d', on = 'Date and Time')['Accident Number']
    .count()
    .reset_index()
    .set_index('Date and Time')
)
```

By adding a *groupby* prior to the *rolling*, we can compare trends across precincts.

Rolling Windows - Example #2

```
traffic_by_precinct = (traffic
                        .groupby('Precinct')
                        .rolling('30d', on = 'Date and Time')['Accident Number']
                        .count()
                        .reset_index()
                        .set_index('Date and Time')
                        )
traffic_by_precinct['rolling_daily_average'] = traffic_by_precinct['Accident Number'] / 30
```

Rolling Windows - Example #2

