

# Reading date and time data in Pandas

WORKING WITH DATES AND TIMES IN PYTHON



**Max Shron**  
Data Scientist and Author

# A simple Pandas example

```
# Load Pandas
import pandas as pd

# Import W20529's rides in Q4 2017
rides = pd.read_csv('capital-onebike.csv')
```

# A simple Pandas example

```
# See our data
print(rides.head(3))
```

```
      Start date      End date      Start station \
0 2017-10-01 15:23:25 2017-10-01 15:26:26      Glebe Rd & 11th St N
1 2017-10-01 15:42:57 2017-10-01 17:49:59  George Mason Dr & Wilson Blvd
2 2017-10-02 06:37:10 2017-10-02 06:42:53  George Mason Dr & Wilson Blvd

      End station Bike number Member type
0      George Mason Dr & Wilson Blvd      W20529      Member
1      George Mason Dr & Wilson Blvd      W20529      Casual
2  Ballston Metro / N Stuart & 9th St N      W20529      Member
```

# A simple Pandas example

```
rides['Start date']
```

```
0    2017-10-01 15:23:25
1    2017-10-01 15:42:57
...
Name: Start date, Length: 290, dtype: object
```

```
rides.iloc[2]
```

```
Start date    2017-10-02 06:37:10
End date      2017-10-02 06:42:53
...
Name: 1, dtype: object
```

# Loading datetimes with parse\_dates

```
# Import W20529's rides in Q4 2017
rides = pd.read_csv('capital-onebike.csv',
                    parse_dates = ['Start date', 'End date'])

# Or:
rides['Start date'] = pd.to_datetime(rides['Start date'],
                                     format = "%Y-%m-%d %H:%M:%S")
```

# Loading datetimes with parse\_dates

```
# Select Start date for row 2  
rides['Start date'].iloc[2]
```

```
Timestamp('2017-10-02 06:37:10')
```

# Timezone-aware arithmetic

```
# Create a duration column
rides['Duration'] = rides['End date'] - rides['Start date']

# Print the first 5 rows
print(rides['Duration'].head(5))
```

```
0    00:03:01
1    02:07:02
2    00:05:43
3    00:21:18
4    00:21:17
Name: Duration, dtype: timedelta64[ns]
```

# Loading datetimes with parse\_dates

```
rides['Duration']\  
    .dt.total_seconds()\  
    .head(5)
```

```
0      181.0  
1     7622.0  
2      343.0  
3     1278.0  
4     1277.0  
Name: Duration, dtype: float64
```



# Reading date and time data in Pandas

WORKING WITH DATES AND TIMES IN PYTHON

# Summarizing datetime data in Pandas

WORKING WITH DATES AND TIMES IN PYTHON



**Max Shron**  
Data Scientist and Author

# Summarizing data in Pandas

```
# Average time out of the dock  
rides['Duration'].mean()
```

```
Timedelta('0 days 00:19:38.931034')
```

```
# Total time out of the dock  
rides['Duration'].sum()
```

```
Timedelta('3 days 22:58:10')
```

# Summarizing data in Pandas

```
# Percent of time out of the dock  
rides['Duration'].sum() / timedelta(days=91)
```

```
0.04348417785917786
```

# Summarizing data in Pandas

```
# Count how many time the bike started at each station  
rides['Member type'].value_counts()
```

```
Member    236  
Casual     54  
Name: Member type, dtype: int64
```

```
# Percent of rides by member  
rides['Member type'].value_counts() / len(rides)
```

```
Member    0.813793  
Casual    0.186207  
Name: Member type, dtype: float64
```

# Summarizing datetime in Pandas

```
# Add duration (in seconds) column
rides['Duration seconds'] = rides['Duration'].dt.total_seconds()

# Average duration per member type
rides.groupby('Member type')['Duration seconds'].mean()
```

```
Member type
Casual      1994.666667
Member      992.279661
Name: Duration seconds, dtype: float64
```

# Summarizing datetime in Pandas

```
# Average duration by month  
rides.resample('M', on = 'Start date')['Duration seconds'].mean()
```

```
Start date  
2017-10-31    1886.453704  
2017-11-30     854.174757  
2017-12-31     635.101266  
Freq: M, Name: Duration seconds, dtype: float64
```

# Summarizing datetime in Pandas

```
# Size per group
rides.groupby('Member type').size()
```

```
Member type
Casual      54
Member     236
dtype: int64
```

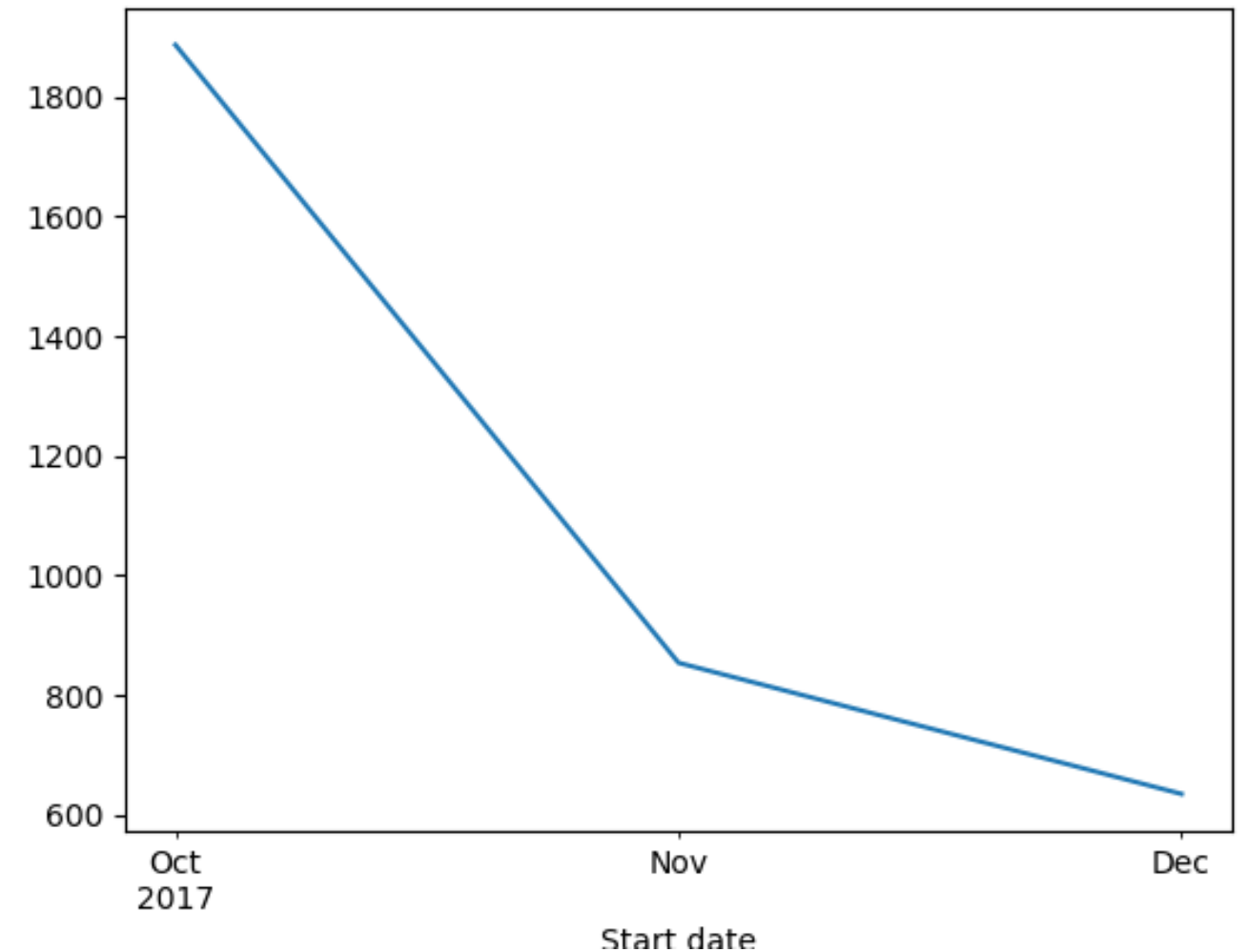
```
# First ride per group
rides.groupby('Member type').first()
```

```
          Duration  ...
Member type         ...
Casual    02:07:02  ...
Member    00:03:01  ...
```



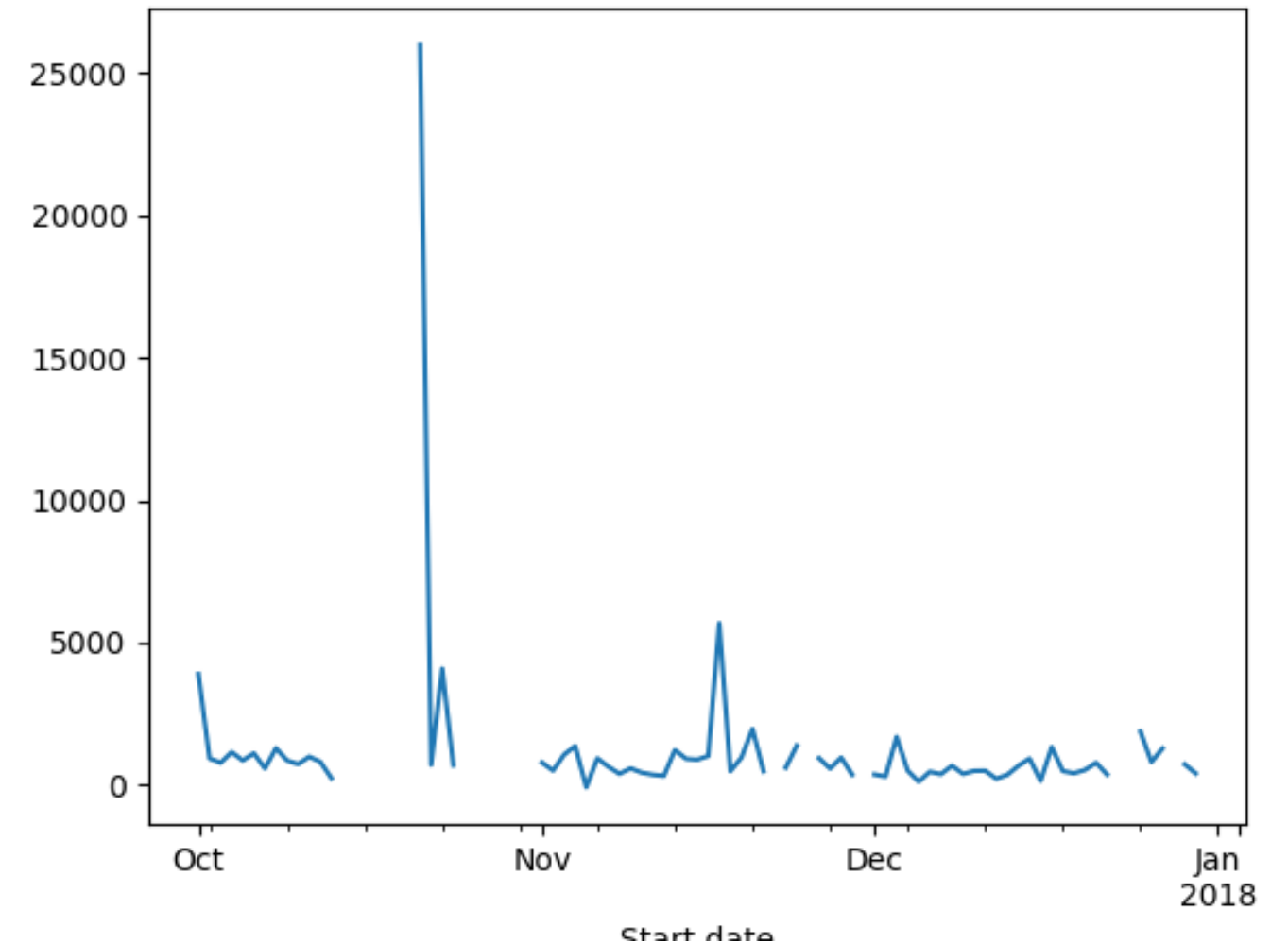
# Summarizing datetime in Pandas

```
rides\  
  .resample('M', on = 'Start date')\  
  ['Duration seconds']\  
  .mean()\  
  .plot()
```



# Summarizing datetime in Pandas

```
rides\  
  .resample('D', on = 'Start date')\  
  ['Duration seconds']\  
  .mean()\  
  .plot()
```



# Summarizing datetime data in Pandas

WORKING WITH DATES AND TIMES IN PYTHON

# Additional datetime methods in Pandas

WORKING WITH DATES AND TIMES IN PYTHON



**Max Shron**  
Data Scientist & Author

# Timezones in Pandas

```
rides['Duration'].dt.total_seconds().min()
```

```
-3346.0
```

# Timezones in Pandas

```
rides['Start date'].head(3)
```

```
0    2017-10-01 15:23:25
1    2017-10-01 15:42:57
2    2017-10-02 06:37:10
Name: Start date, dtype: datetime64[ns]
```

```
rides['Start date'].head(3)\
    .dt.tz_localize('America/New_York')
```

```
0    2017-10-01 15:23:25-04:00
1    2017-10-01 15:42:57-04:00
2    2017-10-02 06:37:10-04:00
Name: Start date, dtype: datetime64[ns, America/New_York]
```

# Timezones in Pandas

```
# Try to set a timezone...
rides['Start date'] = rides['Start date']\
    .dt.tz_localize('America/New_York')
```

```
AmbiguousTimeError: Cannot infer dst time from '2017-11-05 01:56:50',
try using the 'ambiguous' argument
```

```
# Handle ambiguous datetimes
rides['Start date'] = rides['Start date']\
    .dt.tz_localize('America/New_York', ambiguous='NaT')

rides['End date'] = rides['End date']\
    .dt.tz_localize('America/New_York', ambiguous='NaT')
```

# Timezones in Pandas

```
# Re-calculate duration, ignoring bad row
rides['Duration'] = rides['Start date'] - rides['End date']

# Find the minimum again
rides['Duration'].dt.total_seconds().min()
```

```
116.0
```



# Timezones in Pandas

```
# Look at problematic row  
rides.iloc[129]
```

```
Duration                NaT  
Start date              NaT  
End date               NaT  
Start station          6th & H St NE  
End station            3rd & M St NE  
Bike number            W20529  
Member type            Member  
Name: 129, dtype: object
```

# Other datetime operations in Pandas

```
# Year of first three rows
rides['Start date']\
    .head(3)\
    .dt.year
```

```
0    2017
1    2017
2    2017
Name: Start date, dtype: int64
```

```
# See weekdays for first three rides
rides['Start date']\
    .head(3)\
    .dt.weekday_name
```

```
0    Sunday
1    Sunday
2    Monday
Name: Start date, dtype: object
```

# Other parts of Pandas

```
# Shift the indexes forward one, padding with NaT
rides['End date'].shift(1).head(3)
```

```
0          NaT
1  2017-10-01 15:26:26-04:00
2  2017-10-01 17:49:59-04:00
Name: End date, dtype: datetime64[ns, America/New_York]
```

# Additional datetime methods in Pandas

WORKING WITH DATES AND TIMES IN PYTHON

# Wrap-up

WORKING WITH DATES AND TIMES IN PYTHON



**Max Shron**

Data Scientist and Author

# Recap: Dates and Calendars

- The `date()` class takes a year, month, and day as arguments
- A `date` object has accessors like `.year`, and also methods like `.weekday()`
- `date` objects can be compared like numbers, using `min()`, `max()`, and `sort()`
- You can subtract one `date` from another to get a `timedelta`
- To turn `date` objects into strings, use the `.isoformat()` or `.strftime()` methods

# Recap: Combining Dates and Times

- The `datetime()` class takes all the arguments of `date()`, plus an hour, minute, second, and microsecond
- All of the additional arguments are optional; otherwise, they're set to zero by default
- You can replace any value in a `datetime` with the `.replace()` method
- Convert a `timedelta` into an integer with its `.total_seconds()` method
- Turn strings into dates with `.strptime()` and dates into strings with `.strftime()`

# Recap: Timezones and Daylight Saving

- A `datetime` is "timezone aware" when it has its `tzinfo` set. Otherwise it is "timezone naive"
- Setting a timezone tells a `datetime` how to align itself to UTC, the universal time standard
- Use the `.replace()` method to change the timezone of a `datetime`, leaving the date and time the same
- Use the `.astimezone()` method to shift the date and time to match the new timezone
- `dateutil.tz` provides a comprehensive, updated timezone database



# Recap: Easy and Powerful Timestamps in Pandas

- When reading a csv, set the `parse_dates` argument to be the list of columns which should be parsed as datetimes
- If setting `parse_dates` doesn't work, use the `pd.to_datetime()` function
- Grouping rows with `.groupby()` lets you calculate aggregates per group. For example, `.first()`, `.min()` or `.mean()`
- `.resample()` groups rows on the basis of a `datetime` column, by year, month, day, and so on
- Use `.tz_localize()` to set a timezone, keeping the date and time the same
- Use `.tz_convert()` to change the date and time to match a new timezone

# Congratulations!

WORKING WITH DATES AND TIMES IN PYTHON