Mean, median & mode imputations

DEALING WITH MISSING DATA IN PYTHON

Suraj Donthi
Deep Learning & Computer Vision Consultant

Basic imputation techniques

- constant (e.g. 0)
- mean
- median
- mode or most frequent

Mean Imputation

```python
from sklearn.impute import SimpleImputer

diabetes_mean = diabetes.copy(deep=True)

mean_imputer = SimpleImputer(strategy='mean')
```

Mean Imputation

```python
from sklearn.impute import SimpleImputer
diabetes_mean = diabetes.copy(deep=True)
mean_imputer = SimpleImputer(strategy='mean')
diabetes_mean.iloc[:, :] = mean_imputer.fit_transform(diabetes_mean)
```

Median imputation

```
diabetes_median = diabetes.copy(deep=True)
median_imputer = SimpleImputer(strategy='median')
diabetes_median.iloc[:, :] = median_imputer.fit_transform(diabetes_median)
```

Mode imputation

```
diabetes_mode = diabetes.copy(deep=True)
mode_imputer = SimpleImputer(strategy='most_frequent')
diabetes_mode.iloc[:, :] = mode_imputer.fit_transform(diabetes_mode)
```

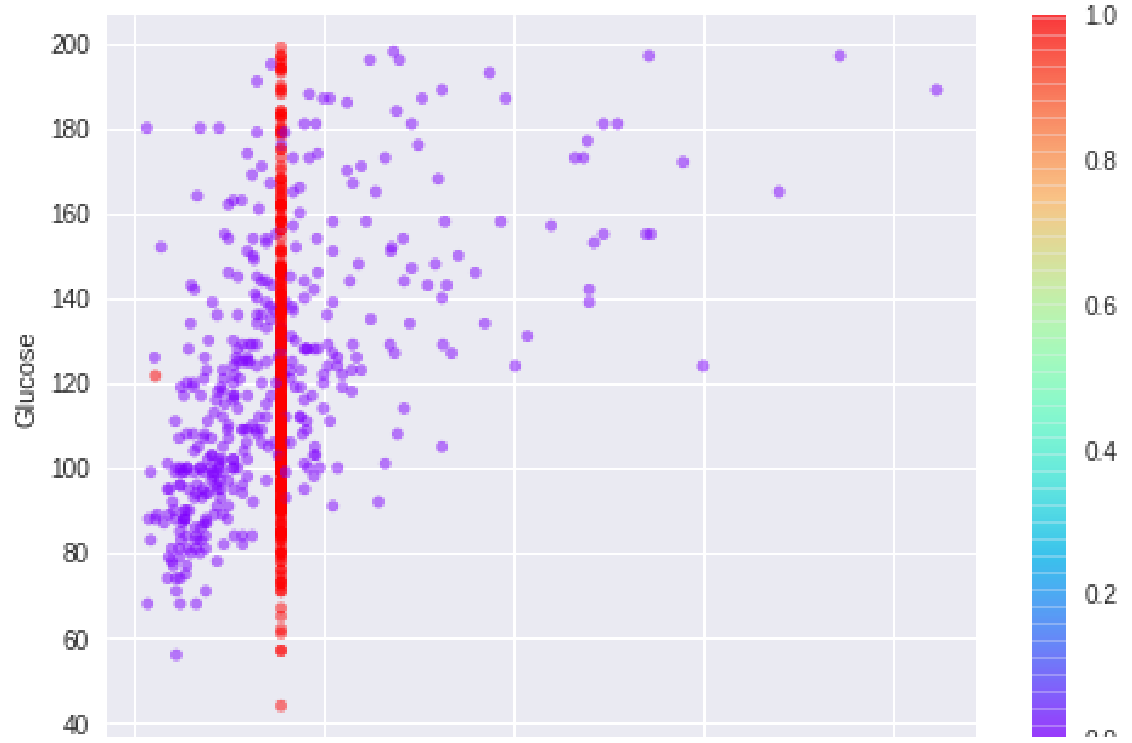
Imputing a constant

```
diabetes_constant = diabetes.copy(deep=True)
constant_imputer = SimpleImputer(strategy='constant', fill_value=0))
diabetes_constant.iloc[:, :] = constant_imputer.fit_transform(diabetes_constant)
```

Scatterplot of imputation

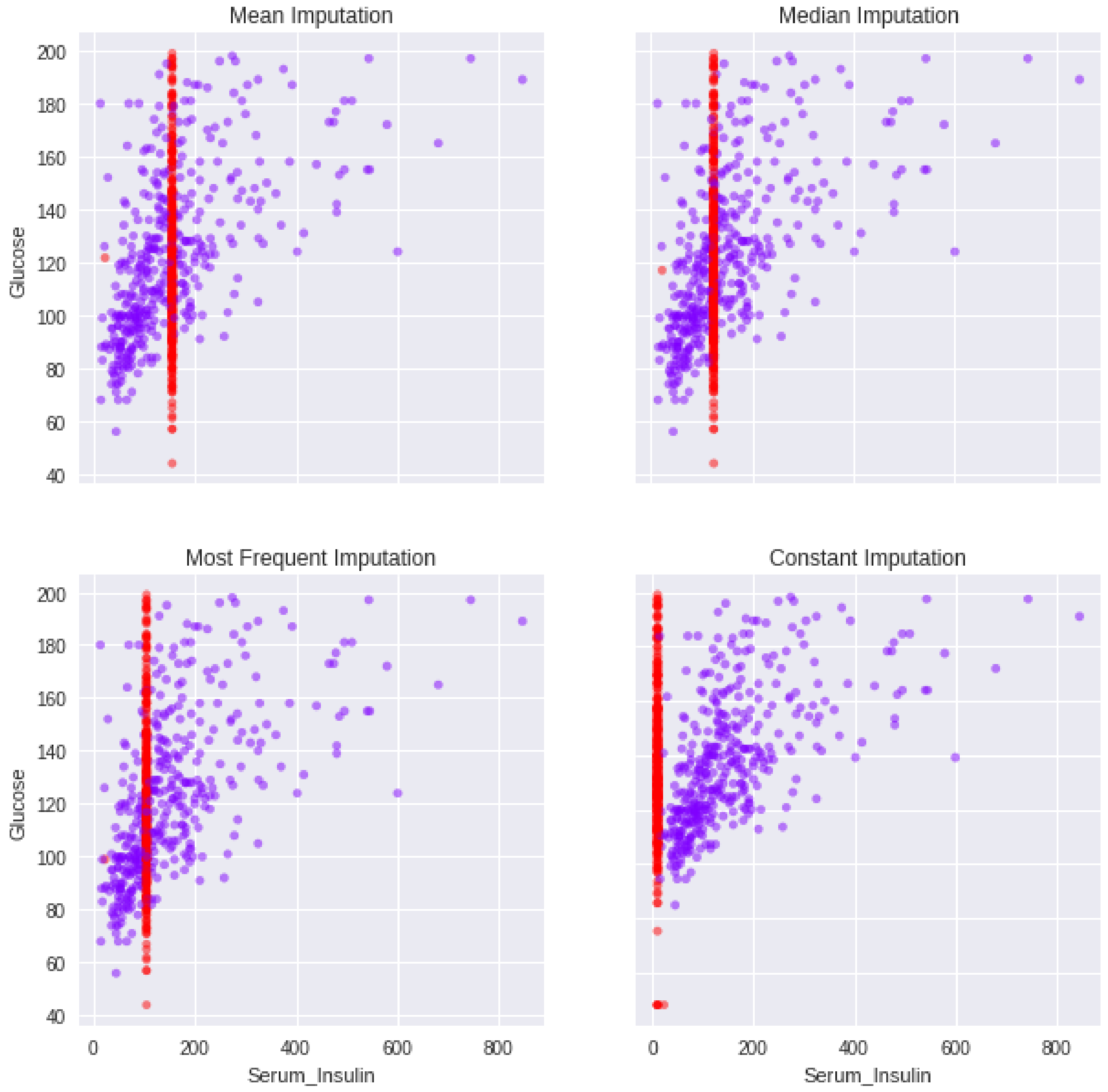
```
nullity = diabetes['Serum_Insulin'].isnull()+diabetes['Glucose'].isnull()
```

```
diabetes_mean.plot(x='Serum_Insulin', y='Glucose', kind='scatter', alpha=0.5,
                   c=nullity, cmap='rainbow', title='Mean Imputation')
```



Visualizing imputations

```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 10))

nullity = diabetes['Serum_Insulin'].isnull()+diabetes['Glucose'].isnull()

imputations = {'Mean Imputation': diabetes_mean,
               'Median Imputation': diabetes_median,
               'Most Frequent Imputation': diabetes_mode,
               'Constant Imputation': diabetes_constant}

for ax, df_key in zip(axes.flatten(), imputations):

    imputations[df_key].plot(x='Serum_Insulin', y='Glucose', kind='scatter',
                             alpha=0.5, c=nullity, cmap='rainbow', ax=ax,
                             colorbar=False, title=df_key)
```


Mean Imputation
Median Imputation
Most Frequent Imputation
Constant Imputation
Glucose
Serum_Insulin
200
180
160
140
120
100
80
60
40
0
200
400
600
800

Summary

You learned to

- Impute with statistical parameters like mean, median and mode
- Graphically compare the imputations
- Analyze the imputations

Let's practice!

DEALING WITH MISSING DATA IN PYTHON

Imputing time-series data

DEALING WITH MISSING DATA IN PYTHON

Suraj Donthi
Deep Learning & Computer Vision Consultant

Airquality Dataset

```python
import pandas as pd
airquality = pd.read_csv('air-quality.csv', parse_dates='Date',
                                index_col='Date')

airquality.head()
```

```
            Ozone   Solar   Wind   Temp
Date
1976-05-01   41.0   190.0    7.4   67
1976-05-02   36.0   118.0    8.0   72
1976-05-03   12.0   149.0   12.6   74
1976-05-04   18.0   313.0   11.5   62
1976-05-05    NaN     NaN   14.3   56
```

Airquality Dataset

```
airquality.isnull().sum()
```

```
Ozone     37
Solar      7
Wind       0
Temp       0
dtype: int64
```

```
airquality.isnull.mean() * 100
```

```
Ozone     24.183007
Solar      4.575163
Wind       0.000000
Temp       0.000000
dtype: float64
```

The .fillna() method

The attribute `method` in `.fillna()` can be set to

- `'ffill'` or `'pad'`
- `'bfill'` or `'backwardfill'`

Ffill method

- Replace `NaN` s with last observed value
- `pad` is the same as `'ffill'`

```
airquality.fillna(method='ffill', inplace=True)
```

```python
airquality['Ozone'][30:40]
```

```
Date           Ozone
1976-05-31      37.0
1976-06-01       NaN
1976-06-02       NaN
1976-06-03       NaN
1976-06-04       NaN
1976-06-05       NaN
1976-06-06       NaN
1976-06-07      29.0
1976-06-08       NaN
1976-06-09      71.0
```

```python
airquality.fillna(method='ffill',
                  inplace=True)
airquality['Ozone'][30:40]
```

```
Date           Ozone
1976-05-31      37.0
1976-06-01      37.0
1976-06-02      37.0
1976-06-03      37.0
1976-06-04      37.0
1976-06-05      37.0
1976-06-06      37.0
1976-06-07      29.0
1976-06-08      29.0
1976-06-09      71.0
```

Bfill method

- Replace `NaN` s with next observed value
- `backfill` is the same as `'bfill'`

```
df.fillna(method='bfill', inplace=True)
```

```python
airquality['Ozone'][30:40]
```

```
Date          Ozone
1976-05-31     37.0
1976-06-01      NaN
1976-06-02      NaN
1976-06-03      NaN
1976-06-04      NaN
1976-06-05      NaN
1976-06-06      NaN
1976-06-07     29.0
1976-06-08      NaN
1976-06-09     71.0
```

```python
airquality.fillna(method='bfill',
                         inplace=True)
airquality['Ozone'][30:40]
```

```
Date          Ozone
1976-05-31     37.0
1976-06-01     29.0
1976-06-02     29.0
1976-06-03     29.0
1976-06-04     29.0
1976-06-05     29.0
1976-06-06     29.0
1976-06-07     29.0
1976-06-08     71.0
1976-06-09     71.0
```

The .interpolate() method

- The `.interpolate()` method extends the sequence of values to the missing values

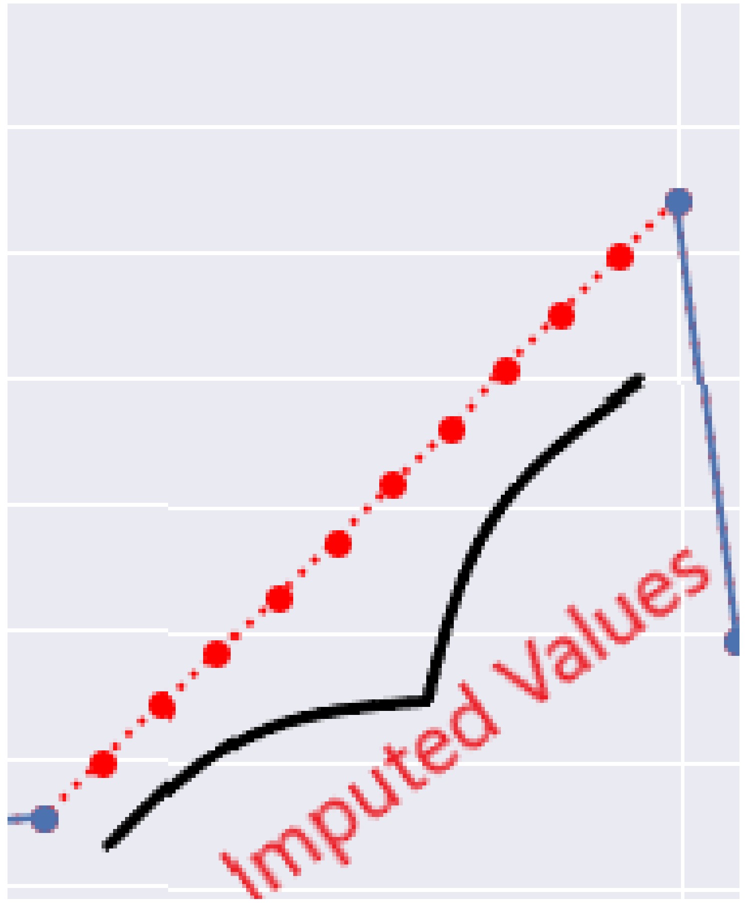
The attribute `method` in `.interpolate()` can be set to

  - `'linear'`
  - `'quadratic'`
  - `'nearest'`

Linear interpolation

- Impute linearly or with equidistant values

```
df.interpolate(method='linear', inplace=True)
```



```python
airquality['Ozone'][30:40]
```

```
Date          Ozone
1976-05-31     37.0
1976-06-01      NaN
1976-06-02      NaN
1976-06-03      NaN
1976-06-04      NaN
1976-06-05      NaN
1976-06-06      NaN
1976-06-07     29.0
1976-06-08      NaN
1976-06-09     71.0
```

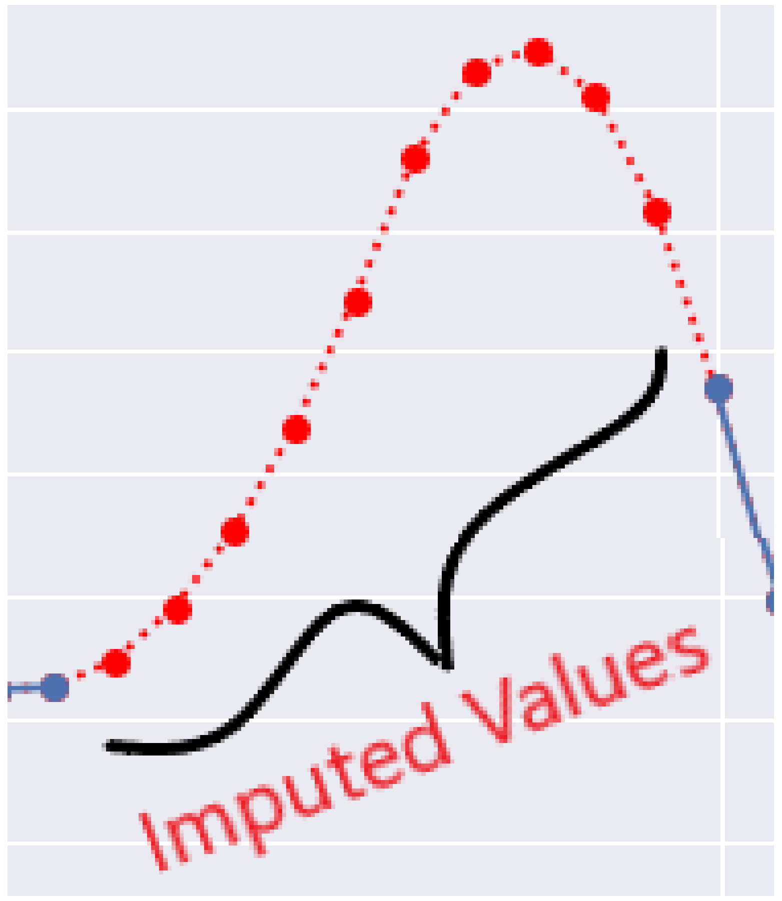
```python
airquality.interpolate(
             method='linear', inplace=True)
airquality['Ozone'][30:40]
```

```
Date          Ozone
1976-05-31     37.0
1976-06-01     35.9
1976-06-02     34.7
1976-06-03     33.6
1976-06-04     32.4
1976-06-05     31.3
1976-06-06     30.1
1976-06-07     29.0
1976-06-08     50.0
1976-06-09     71.0
```

Quadratic interpolation

- Impute the values quadratically

```
df.interpolate(method='quadratic', inplace=True)
```



```python
airquality['Ozone'][30:39]
```

```
            Ozone
Date
1976-05-31   37.0
1976-06-01    NaN
1976-06-02    NaN
1976-06-03    NaN
1976-06-04    NaN
1976-06-05    NaN
1976-06-06    NaN
1976-06-07   29.0
1976-06-08    NaN
```

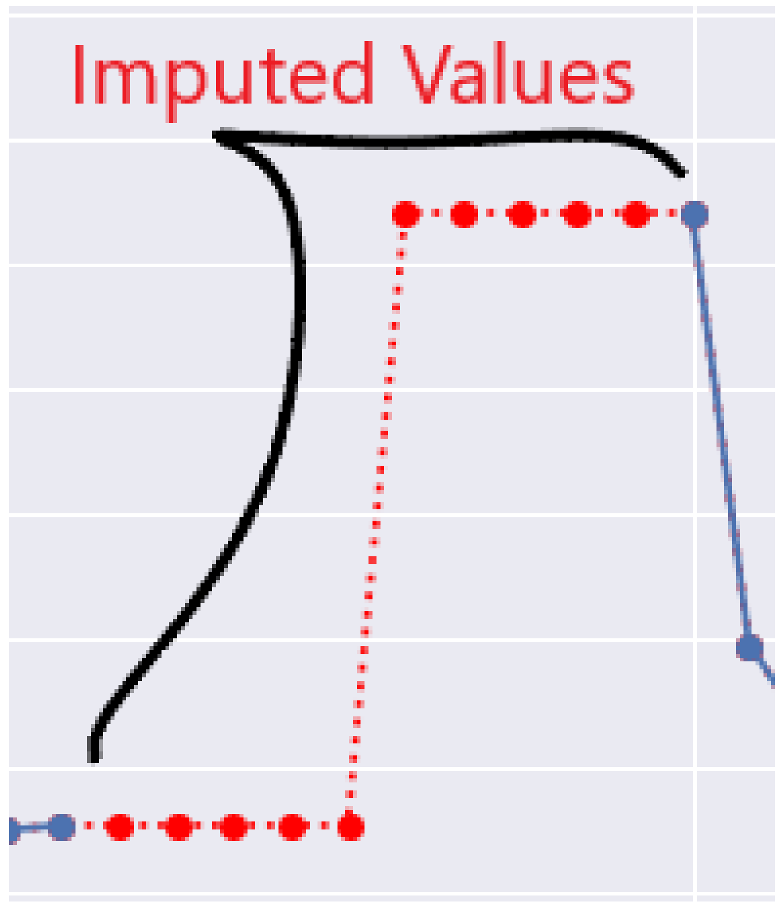
```python
airquality.interpolate(
  method='quadratic', inplace=True)
airquality['Ozone'][30:39]
```

```
            Ozone
Date
1976-05-31   37.0
1976-06-01  -38.4
1976-06-02  -79.4
1976-06-03  -85.9
1976-06-04  -62.4
1976-06-06   -2.8
1976-06-07   29.0
1976-06-08   62.2
```

Nearest value imputation

- Impute with the nearest observable value

```
df.interpolate(method='nearest', inplace=True)
```



```python
airquality['Ozone'][30:39]
```

```
Date          Ozone
1976-05-31     37.0
1976-06-01      NaN
1976-06-02      NaN
1976-06-03      NaN
1976-06-04      NaN
1976-06-05      NaN
1976-06-06      NaN
1976-06-07     29.0
1976-06-08      NaN
```

```python
airquality.interpolate(
  method='nearest', inplace=True)
airquality['Ozone'][30:39]
```

```
Date          Ozone
1976-05-31     37.0
1976-06-01     37.0
1976-06-02     37.0
1976-06-03     37.0
1976-06-04     29.0
1976-06-05     29.0
1976-06-06     29.0
1976-06-07     29.0
1976-06-08     29.0
```

Let's practice!

DEALING WITH MISSING DATA IN PYTHON

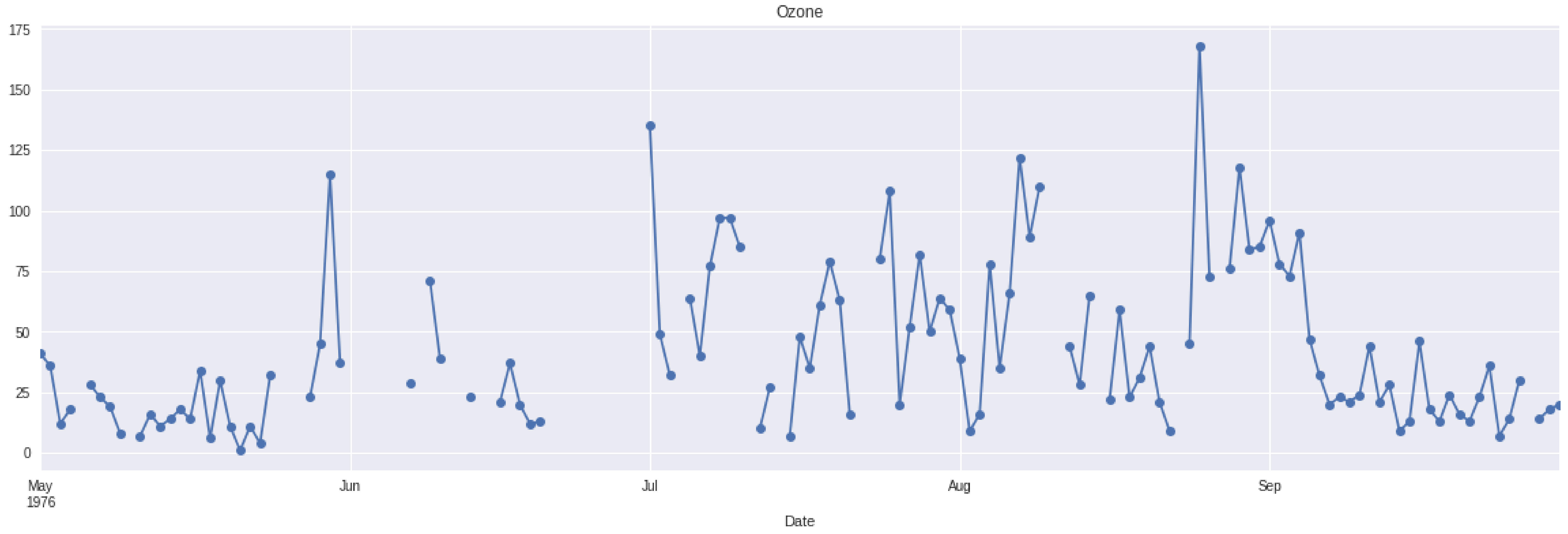
Visualizing time-series imputations

DEALING WITH MISSING DATA IN PYTHON

Suraj Donthi
Deep Learning & Computer Vision Learning

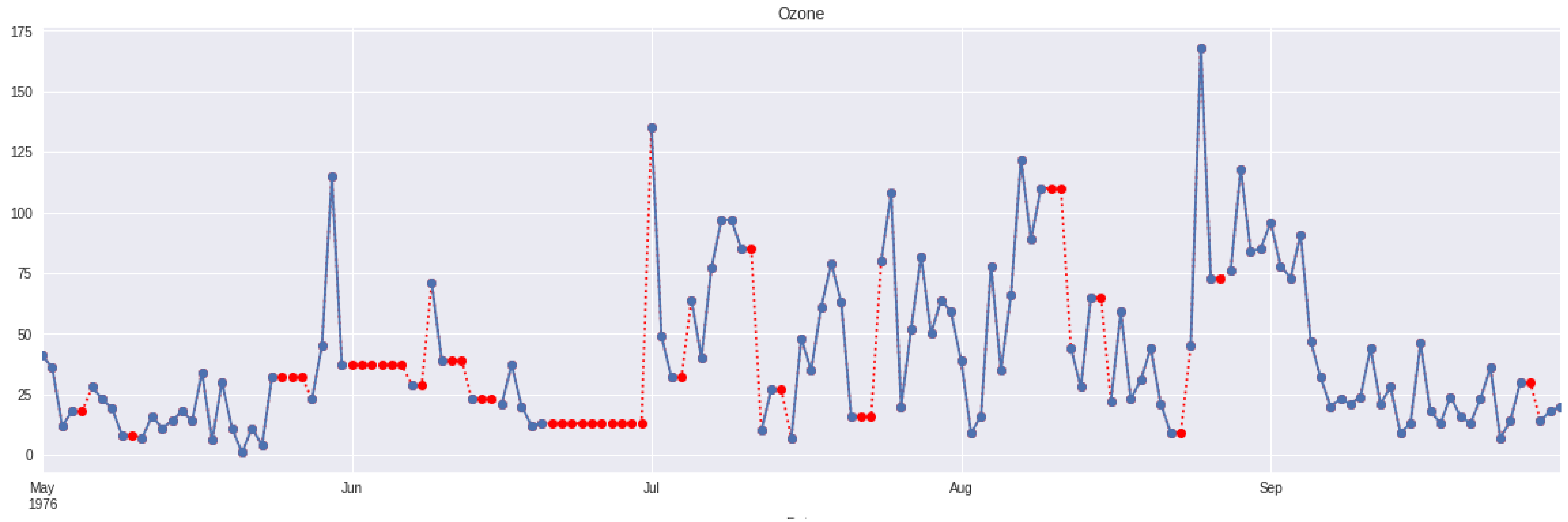
Air quality time-series plot

```
airquality['Ozone'].plot(title='Ozone', marker='o', figsize=(30, 5))
```



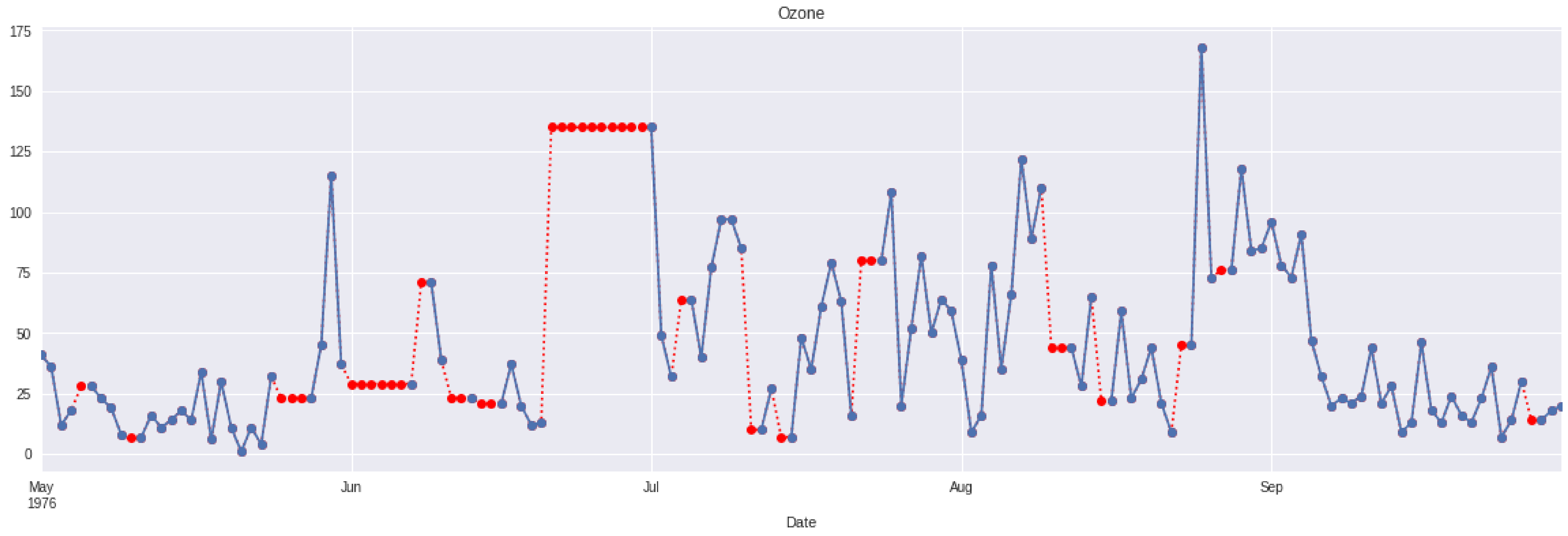
Ffill Imputation

```
ffill_imp['Ozone'].plot(color='red', marker='o', linestyle='dotted', figsize=(30, 5))
airquality['Ozone'].plot(title='Ozone', marker='o')
```



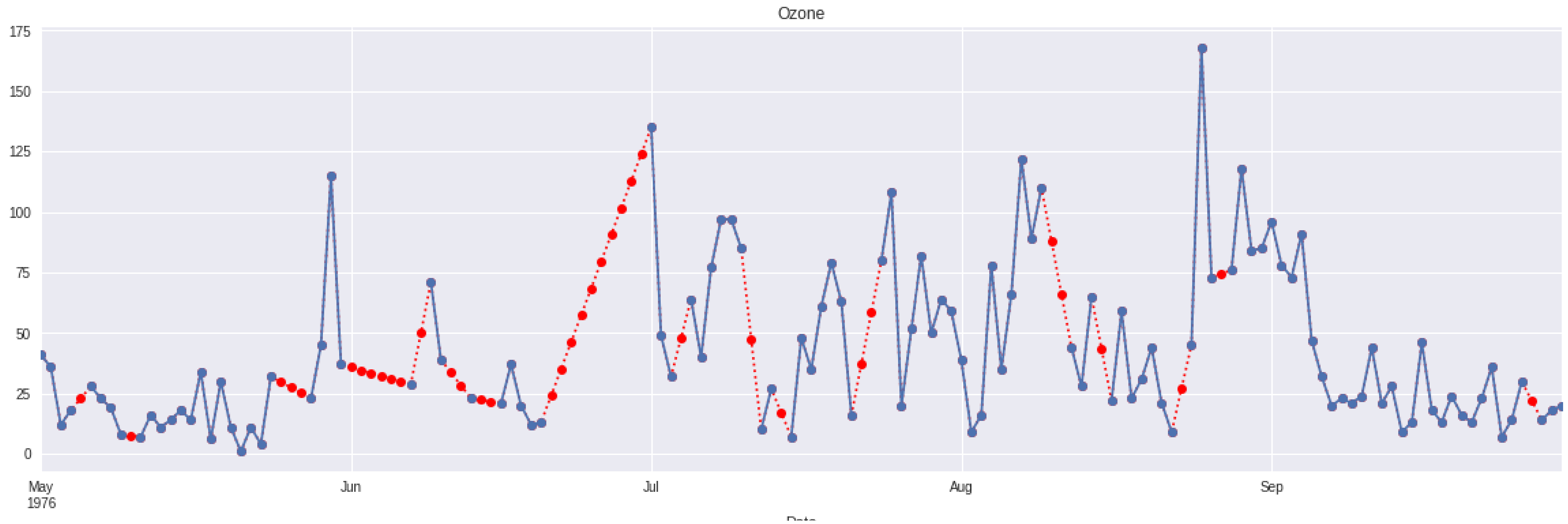
Bfill Imputation

```python
bfill_imp['Ozone'].plot(color='red', marker='o', linestyle='dotted', figsize=(30, 5))
airquality['Ozone'].plot(title='Ozone', marker='o')
```



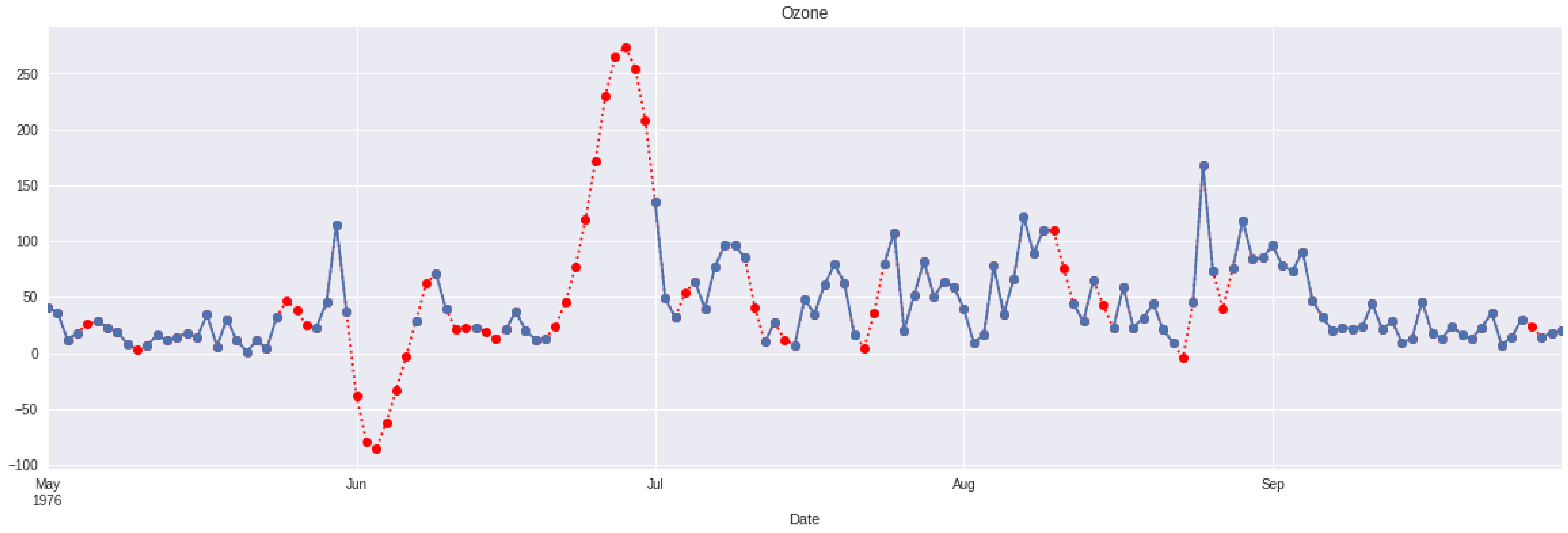
Linear Interpolation

```
linear_interp['Ozone'].plot(color='red', marker='o', linestyle='dotted', figsize=(30, 5)
airquality['Ozone'].plot(title='Ozone', marker='o')
```



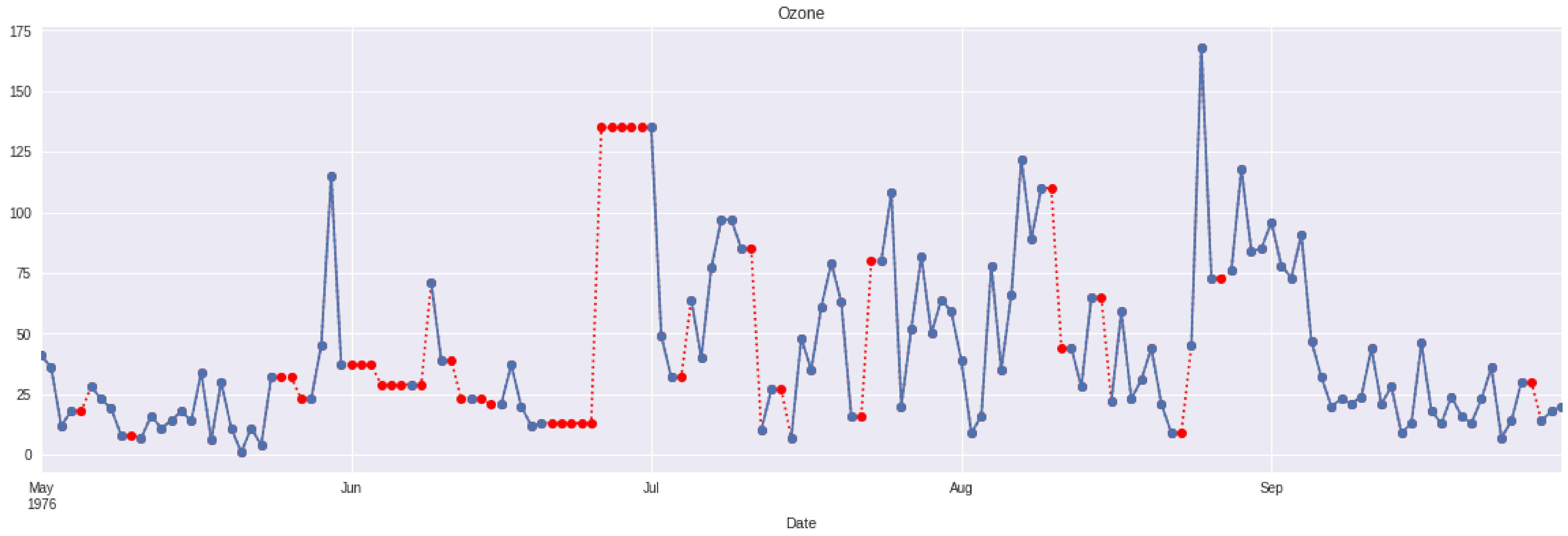
Quadratic Interpolation

```
quadratic_interp['Ozone'].plot(color='red', marker='o', linestyle='dotted', figsize=(30, 5))
airquality['Ozone'].plot(title='Ozone', marker='o')
```



Nearest Interpolation

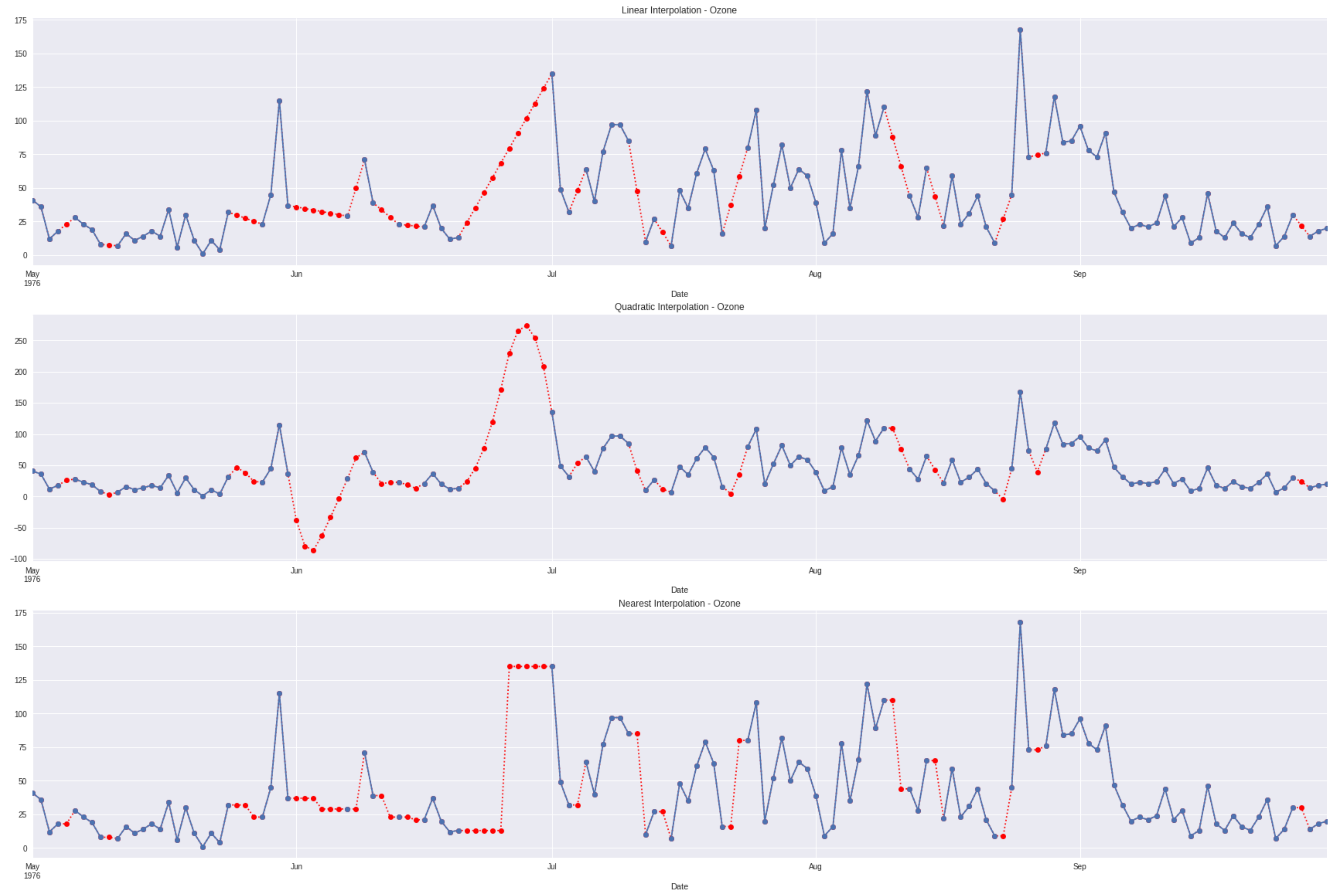
```
nearest_interp['Ozone'].plot(color='red', marker='o', linestyle='dotted', figsize=(30, 5
airquality['Ozone'].plot(title='Ozone', marker='o')
```



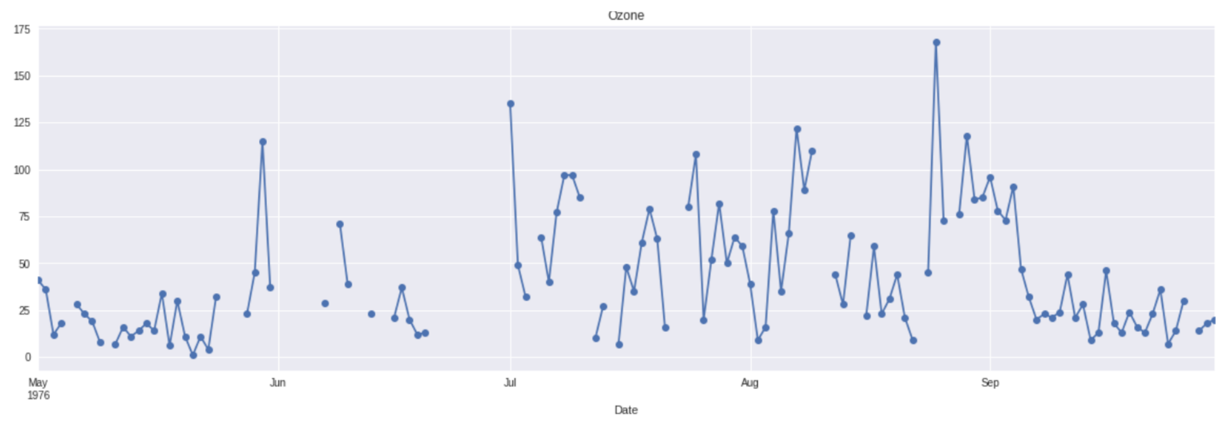
A comparison of the interpolations

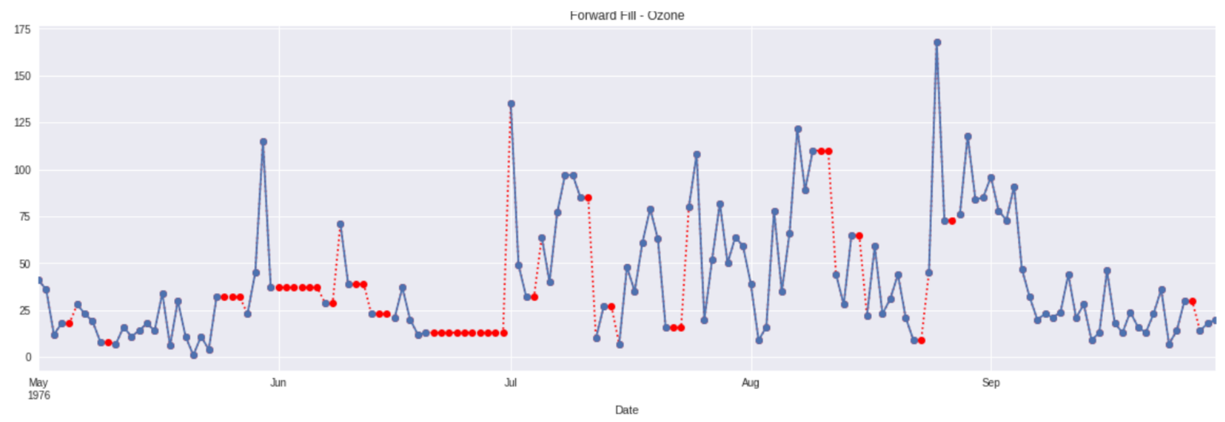
```
# Create subplots
fig, axes = plt.subplots(3, 1, figsize=(30, 20))

# Create interpolations dictionary
interpolations = {'Linear Interpolation': linear_interp,
                  'Quadratic Interpolation': quadratic_interp,
                  'Nearest Interpolation': nearest_interp}

# Visualize each interpolation
for ax, df_key in zip(axes, interpolations):
            interpolations[df_key].Ozone.plot(color='red', marker='o',
                                              linestyle='dotted', ax=ax)
            airquality.Ozone.plot(title=df_key + ' - Ozone', marker='o', ax=ax)
```

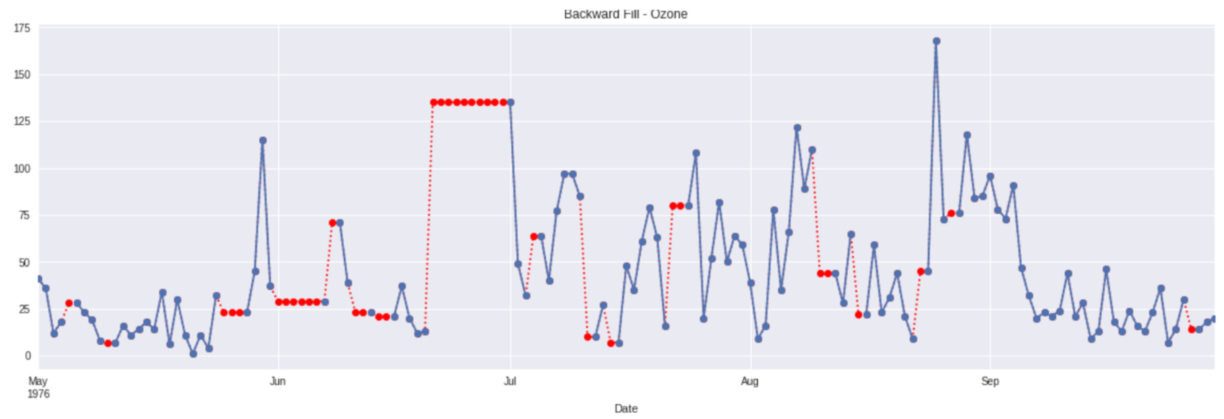
A comparison of the interpolations

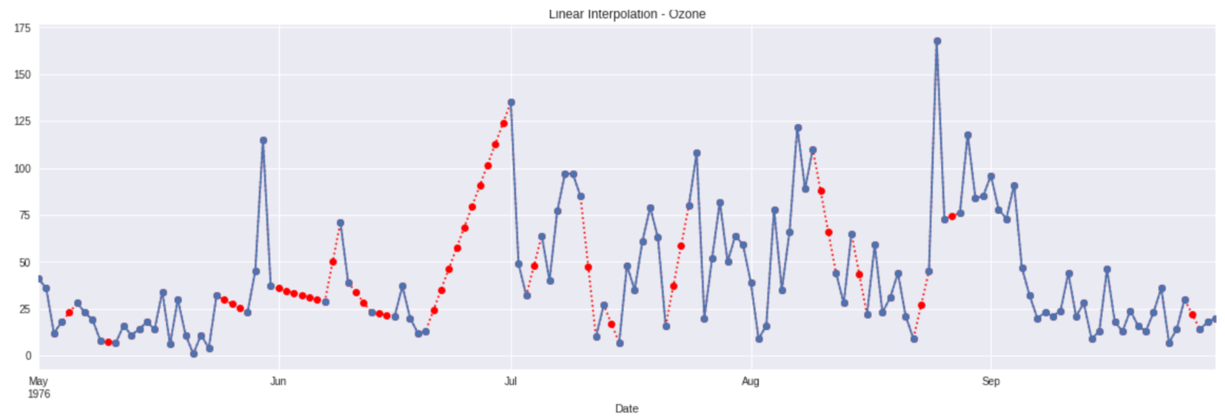


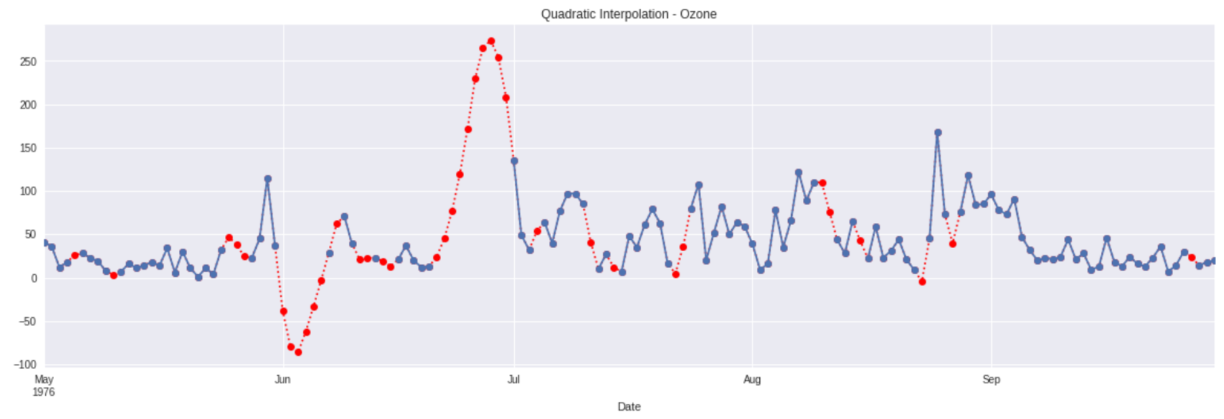
A comparison of imputation techniques

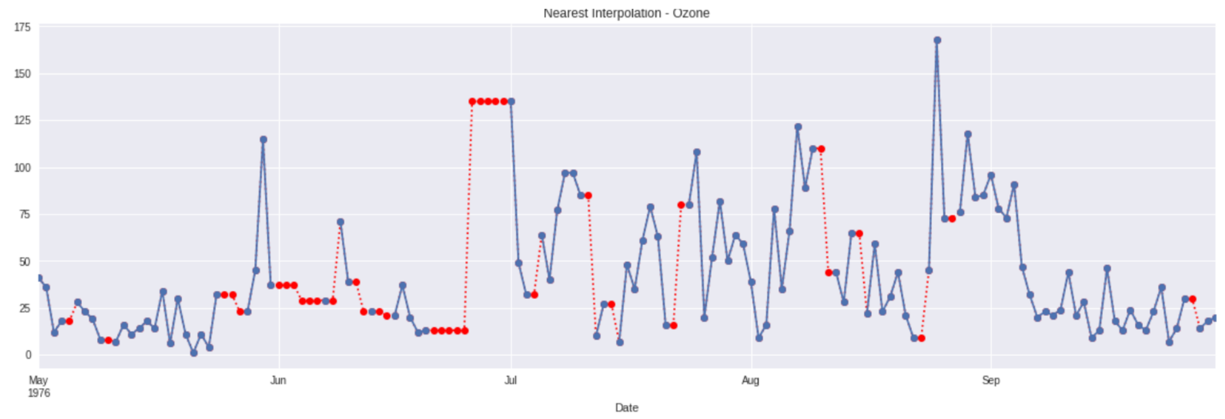












Summary

- Time-series plot of imputed DataFrame
- Comparison of imputations

Let's practice!

DEALING WITH MISSING DATA IN PYTHON