Linear interpolation

Missing data

Contents

LINEAR INTERPOLATION

IMPLEMENTATION IN PANDAS

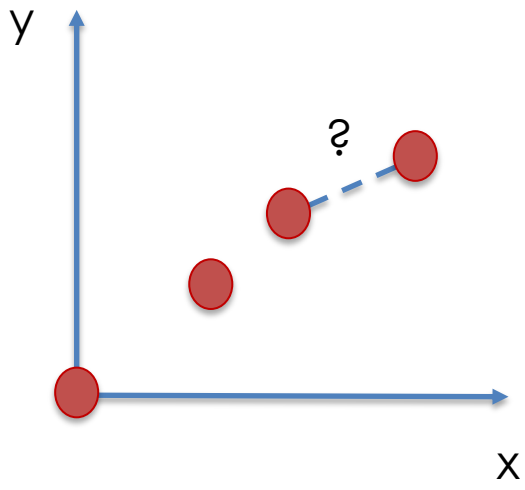
PRACTICAL CONSIDERATIONS

What is interpolation?

- Interpolation is the estimation of the value of a function, $y = f(x)$, between discrete known values
- Typically, we know the inputs and outputs but not the equation for $f(x)$ itself

x	y = f(x)
0	0
1.5	3
2	5
3	6

Known values

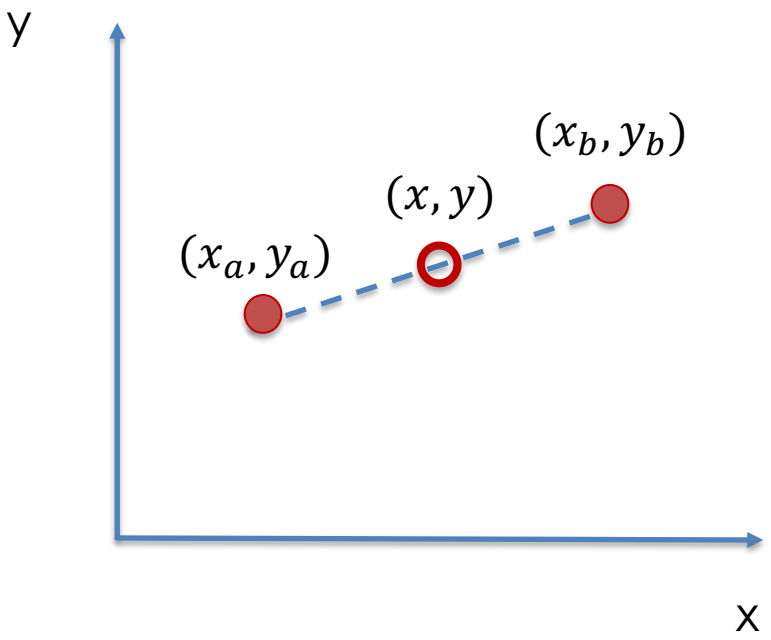


What is a good estimate for $f(x = 2.5)$?

Interpolation methods try to answer this

Linear interpolation

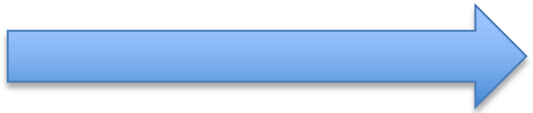
- In linear interpolation a line is drawn between known values
- The interpolated value is given by this line



$$y = y_a + (y_b - y_a)\,\frac{x - x_a}{x_b - x_a} \text{ at the point } (x, y)$$

Linear interpolation for missing data

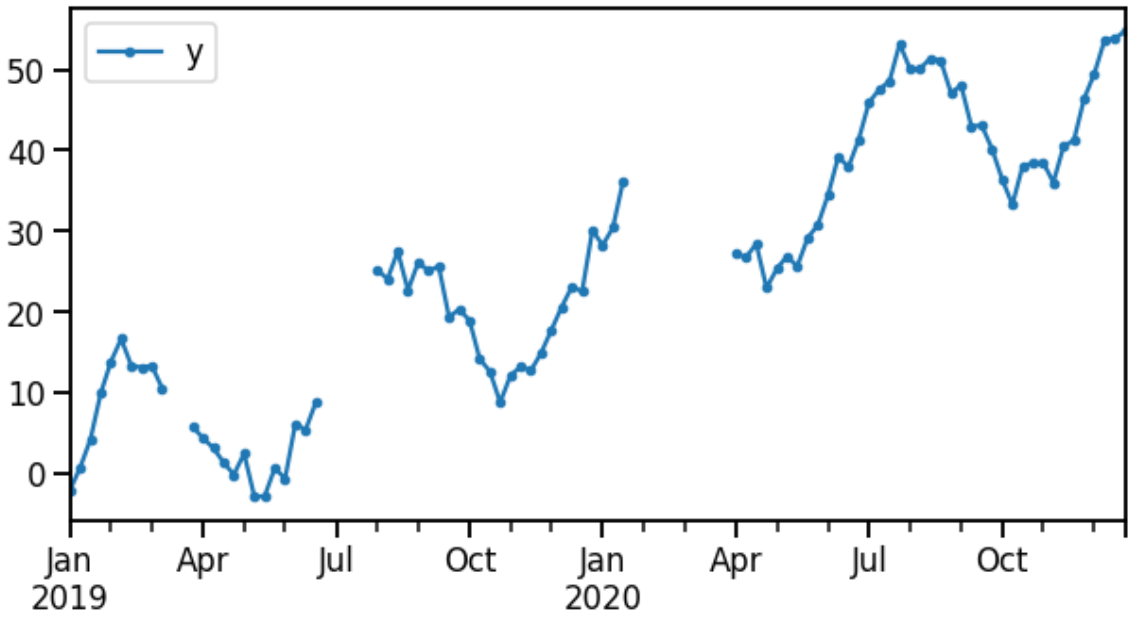
Date	Sales
2020-01-01	3
2020-01-02	10
2020-01-03	23
2020-01-04	nan
2020-01-05	nan
2020-01-06	nan
2020-01-07	58
2020-01-08	5

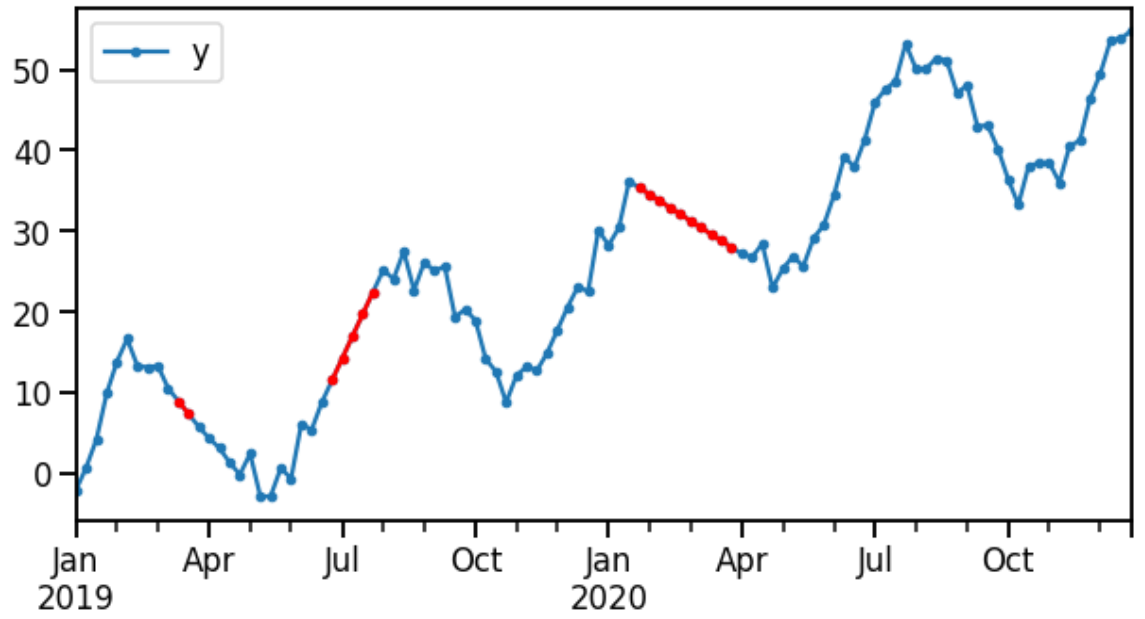


Date	Sales
2020-01-01	3
2020-01-02	10
2020-01-03	23
2020-01-04	31.75
2020-01-05	40.50
2020-01-06	49.25
2020-01-07	58
2020-01-08	5

- Impute missing values with linear interpolation between two known points

Example: Linear interpolation





Implementation

pandas.DataFrame.interpolate

`DataFrame.interpolate`(*method='linear', axis=0, limit=None, inplace=False, limit_direction=None, limit_area=None, downcast=None, **kwargs*) [source]

Fill NaN values using an interpolation method.

Please note that only `method='linear'` is supported for DataFrame/Series with a MultiIndex.

Parameters: **method** : *str, default 'linear'*

Interpolation technique to use. One of:

- 'linear': Ignore the index and treat the values as equally spaced. This is the only method supported on MultiIndexes.
- 'time': Works on daily and higher resolution data to interpolate given length of interval.
- 'index', 'values': use the actual numerical values of the index.
- 'pad': Fill in NaNs using existing values.
- 'nearest', 'zero', 'slinear', 'quadratic', 'cubic', 'spline', 'barycentric', 'polynomial': Passed to *scipy.interpolate.interp1d*. These methods use the numerical values of the index. Both 'polynomial' and 'spline' require that you also specify an *order* (int), e.g. `df.interpolate(method='polynomial', order=5)`.
- 'krogh', 'piecewise_polynomial', 'spline', 'pchip', 'akima', 'cubicspline': Wrappers around the SciPy interpolation methods of similar names. See *Notes*.
- 'from_derivatives': Refers to *scipy.interpolate.BPoly.from_derivatives* which replaces 'piecewise_polynomial' interpolation method in scipy 0.18.

axis : *{{0 or 'index', 1 or 'columns', None}}, default None*

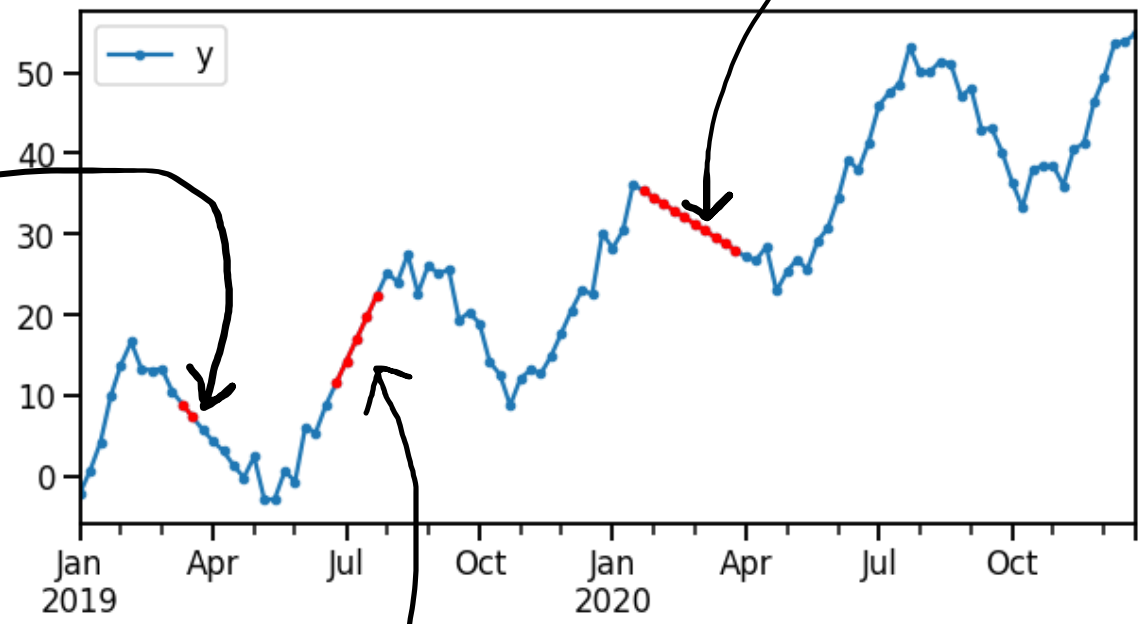
Axis to interpolate along.

```python
# Apply the linear interpolation method
df_imputed = df.interpolate(method='linear')

# Note: If the time interval between rows are not uniform then
# the method should be set as 'time' to achieve a linear fit
df_imputed = df.interpolate(method='time')
```

Practical considerations

For short gaps there is minimal distortion to the time series



For larger gaps there is greater distortion to the time series. Consider the potential impact of this on modelling and analysis (e.g., extracting seasonal information)

Linear interpolation is computationally more costly than forward or backward filling

Depending on the size and location of the missing gaps a linear interpolation will be more or less appropriate

Imputation methods for time series

1. Forward filling (aka last observation carried forward)
2. Backward filling (aka next observation carried backwards)
3. Linear interpolation
4. Spline interpolation
5. Seasonal decomposition and interpolation