Forward and backward filling

Missing data

Contents

FORWARD AND BACKWARD FILLING METHODS

IMPLEMENTATION IN PANDAS

PRACTICAL CONSIDERATIONS

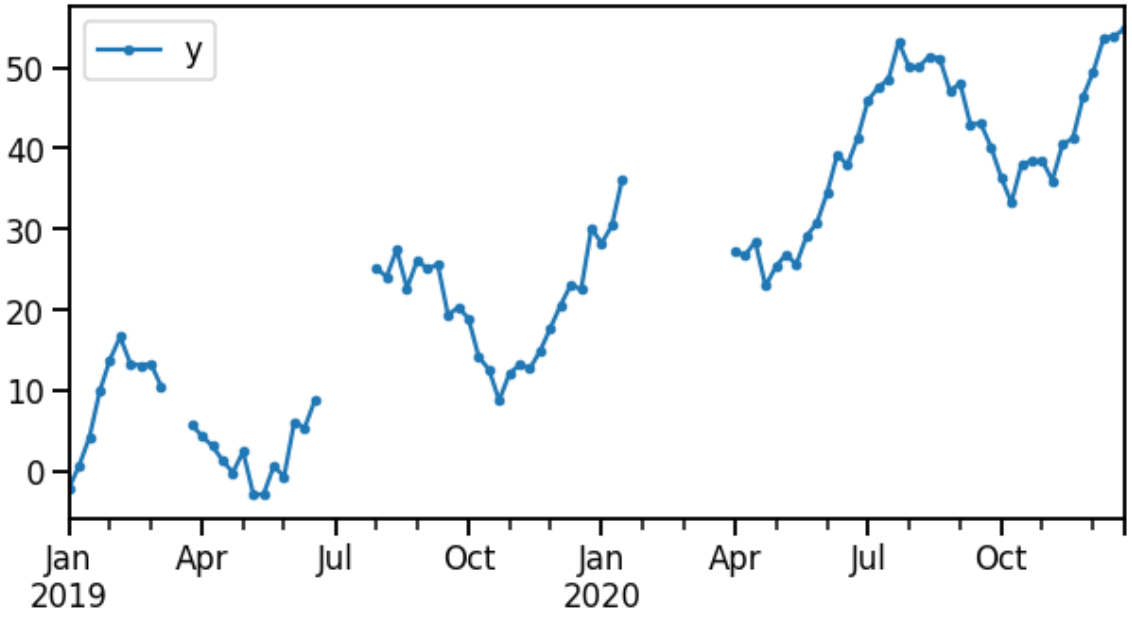
Forward filling (aka last observation carried forward)

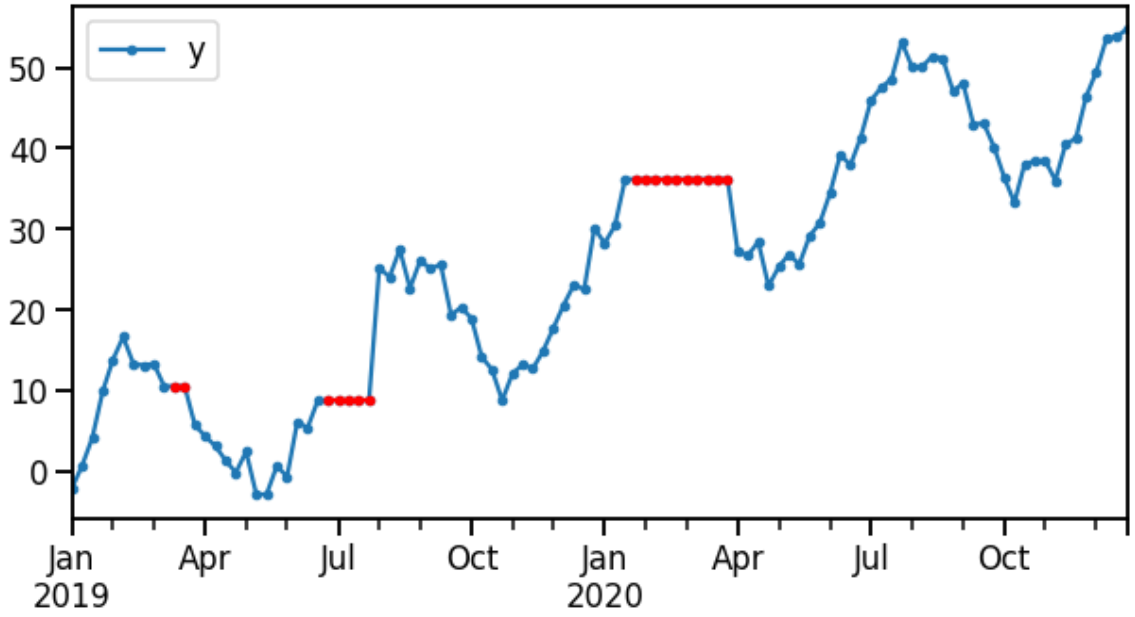
Date	Sales
2020-01-01	3
2020-01-02	10
2020-01-03	23
2020-01-04	nan
2020-01-05	nan
2020-01-06	nan
2020-01-07	58
2020-01-08	5

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

- Impute with the last non-missing observation

Example: Forward filling





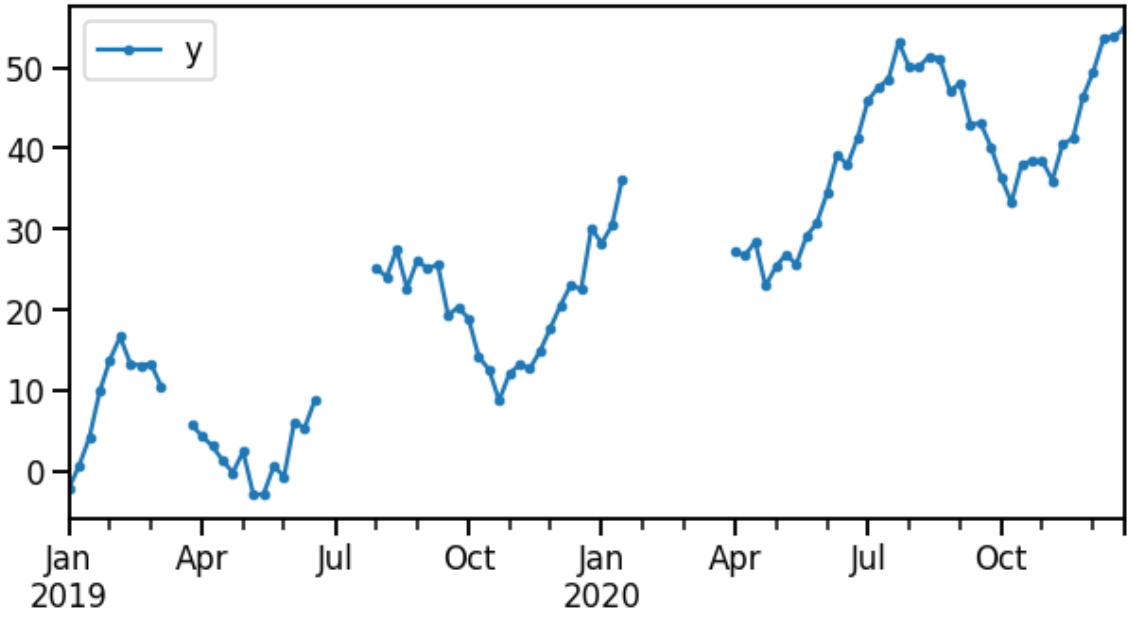
Backward filling
(aka next observation carried backwards)

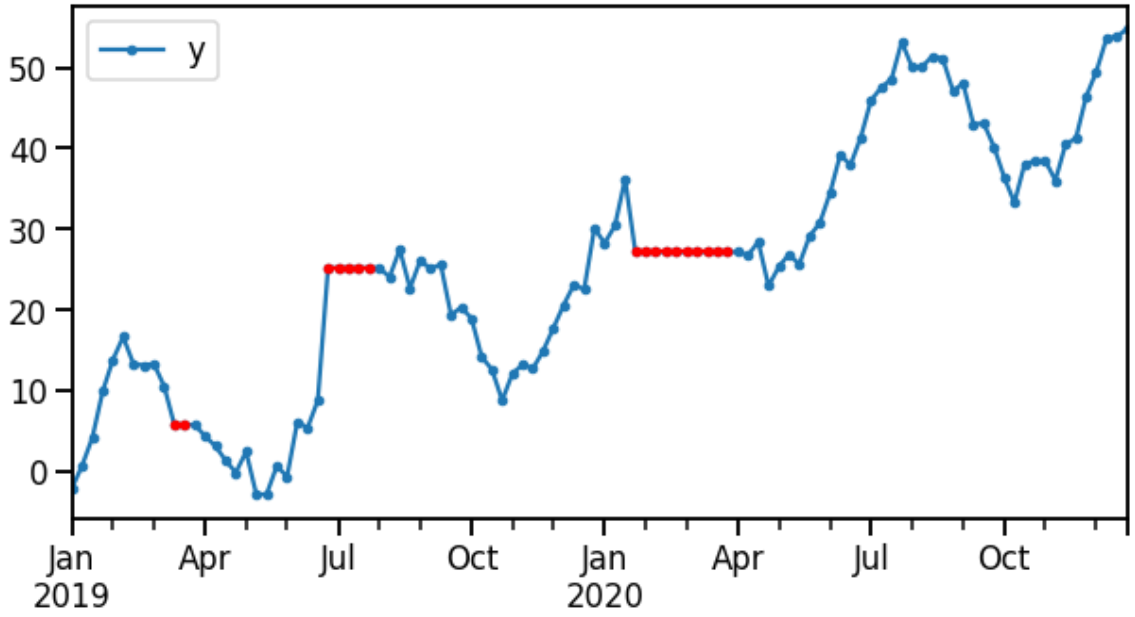
Date	Sales
2020-01-01	3
2020-01-02	10
2020-01-03	23
2020-01-04	nan
2020-01-05	nan
2020-01-06	nan
2020-01-07	58
2020-01-08	5

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

- Impute with the next non-missing observation

Example: Backward filling





Implementation

pandas.DataFrame.fillna

`DataFrame.fillna`(*value=None, method=None, axis=None, inplace=False, limit=None, downcast=None*) [source]

Fill NA/NaN values using the specified method.

Parameters: **value** : *scalar, dict, Series, or DataFrame*

Value to use to fill holes (e.g. 0), alternately a dict/Series/DataFrame of values specifying which value to use for each index (for a Series) or column (for a DataFrame). Values not in the dict/Series/DataFrame will not be filled. This value cannot be a list.

method : *{'backfill', 'bfill', 'pad', 'ffill', None}, default None*

Method to use for filling holes in reindexed Series pad / ffill: propagate last valid observation forward to next valid backfill / bfill: use next valid observation to fill gap.

axis : *{0 or 'index', 1 or 'columns'}*

Axis along which to fill missing values.

inplace : *bool, default False*

If True, fill in-place. Note: this will modify any other views on this object (e.g., a no-copy slice for a column in a DataFrame).

limit : *int, default None*

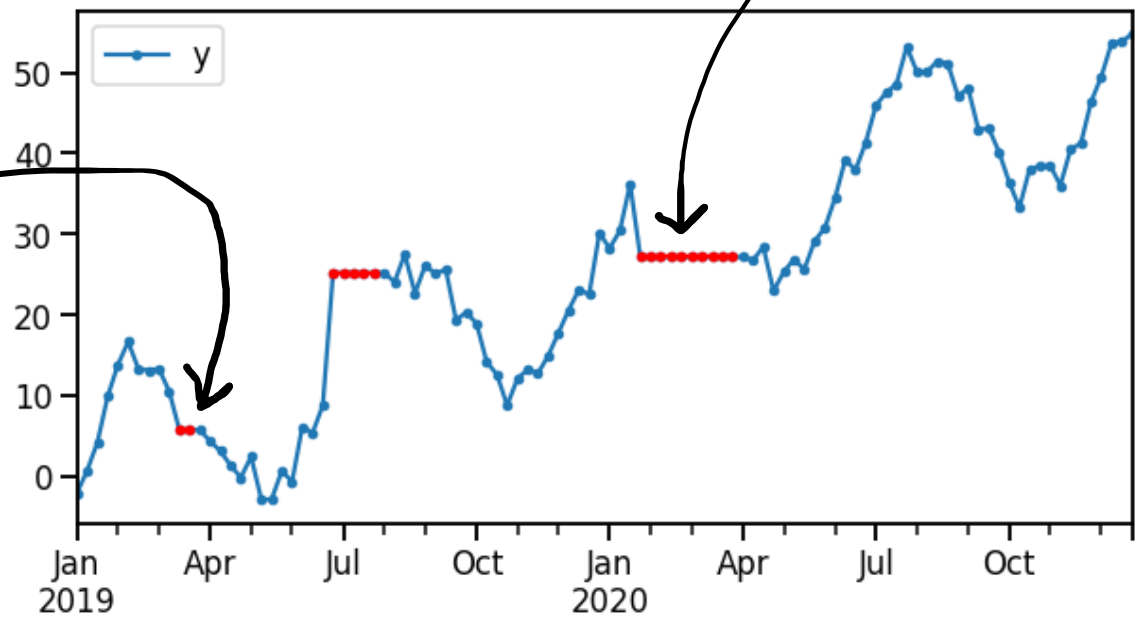
If method is specified, this is the maximum number of consecutive NaN values to forward/backward fill. In other words, if there is a gap with more than this number of consecutive NaNs, it will only be partially filled. If method is not specified, this is the maximum number of entries along the entire axis where NaNs will be filled. Must be greater than 0 if not None.

```
# Apply the backward fill method
df_imputed = df.fillna(method='bfill')

# Apply the forward fill method
df_imputed = df.fillna(method='ffill')
```

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)

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