Time Series Visualizations

Report a content mistake

Learning Objectives:

By the end of this lesson, students will be able to:

- Visualize time series in pandas and matplotlib.
- Use Matplotlib Tick Formatters and Locators.
- Perform advanced customization to visuals

Time Series Visualizations

We will continue to use the weather dataset from the previous lessons. Make sure you have changed the date to a datetime and set it as the index. (See previous lessons if needed)

- Let's focus on the average temperature for now ("meantemp")

```
ts = df['meantemp'].copy()
ts
```

```
datetime
2013-01-01    10.000000
2013-01-02     7.400000
2013-01-03     7.166667
2013-01-04     8.666667
2013-01-05     6.000000                ...    2016-12-28    17.217391
2016-12-29    15.238095
2016-12-30    14.095238
2016-12-31    15.052632
2017-01-01    10.000000
Name: meantemp, Length: 1462, dtype: float64
```

To start, we can make a very simple plot of the mean temperature by date.

```
ax = ts.plot();
```

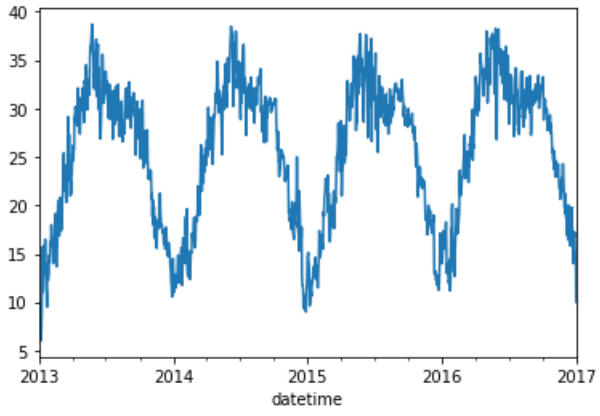


Figure Sizes for Time Series

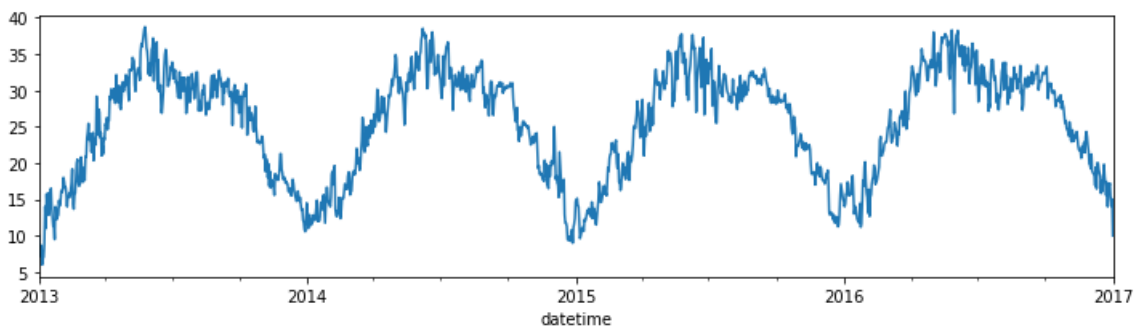
- In general, when we are visualizing time series, we usually want a wider and shorter plot than we typically use.
- To see what is the default figsize used by matplotlib, we can check the values stored in the "rcParams" dictionary in pyplot.

```
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize']
```

```
[6.0, 4.0]
```

- The default figsize is 6" wide by 4" tall. Let's try something wider and shorter by setting the "figsize"

```
## plot a 12x3 figure
ts.plot(figsize=(12,3));
```

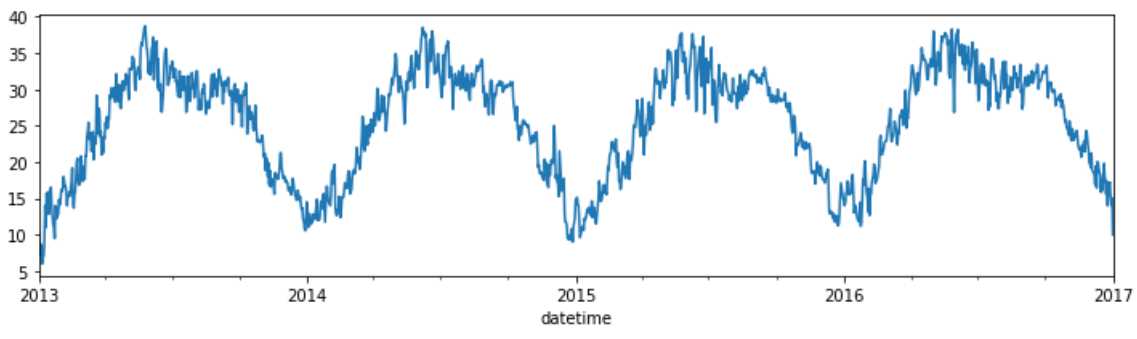


- That looks much better! It would be tedious to constantly have to set the figsize for every visualization, so we can actually change the default figsize using plt.rcParams!
- All we have to do is set the figure.figsize param equal to the new (Width,Height) we want as our default.

```
## setting the default figsize for this notebook
plt.rcParams['figure.figsize'] = (12,3)
```

Now try the basic plot again, and notice it is 12X3 without needing to specify the "figsize"

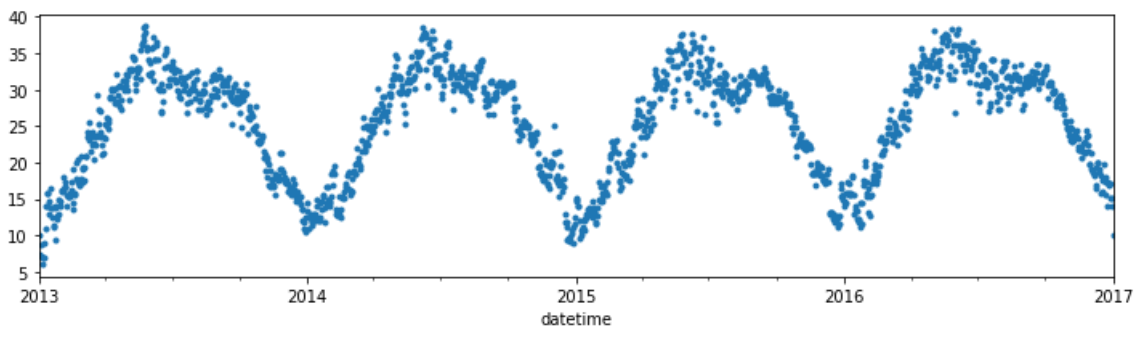
```
ax = ts.plot();
```



Plot - Style/ Level of Detail

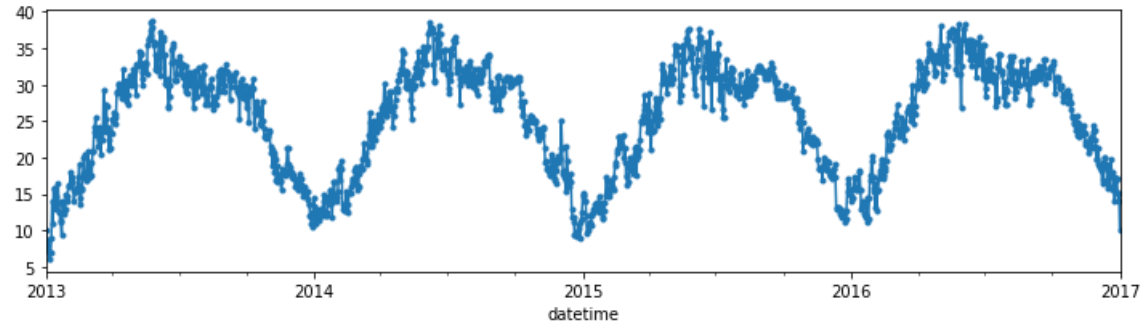
- If we just want to visualize the dates without the connecting lines, we can add `style='.'`

```
## we can also visualize just the markers without the line
ax = ts.plot(style='.');
```

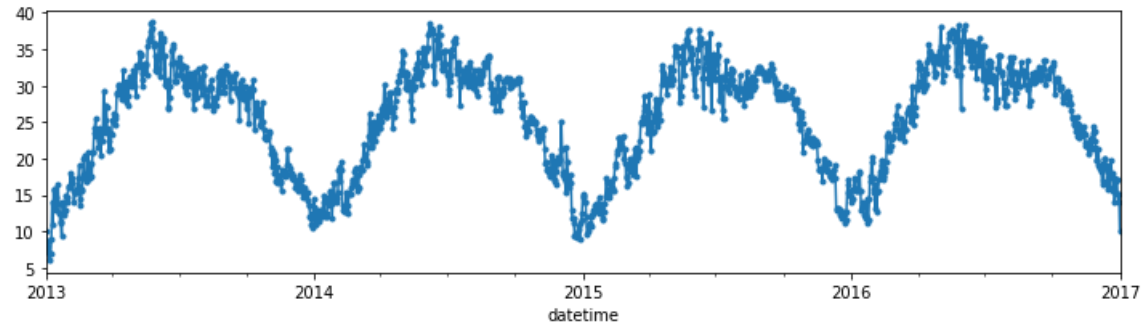


- We can also keep our line, but ADD markers as well either by:
  - Using a "style" arg for .plot that includes a valid marker (".","o", etc.) AND a line style.
    - e.g. ".-" or "o--" or "<-" etc.
    - See the marker documentation for other marker styls
    - Valid line styles are "-","--",":"
  - Use the `marker` argument and specify a valid marker symbol (see link above).

```
## Using style to add markers
ax = ts.plot(style='.-')
```




```
## Using marker to add markers
ax = ts.plot(marker='.')
```



Formatting Date Xticks

- For more complex formatting, there are tools in matplotlib designed to allow us to customize the dates on our axis.

Matplotlib's Artists

There are special classes in base matplotlib (not in matplotlib.pyplot) that are designed to update the spacing and text formatting of our x and y ticks.

- Tick-Formatting Artists:
  - For general use cases, these artists are located in matplotlib's `ticker` submodule.
  - Dates, however, have their own submodule called `dates`

```
#import tick customization tools
import matplotlib.ticker as mticks
import matplotlib.dates as mdates
```

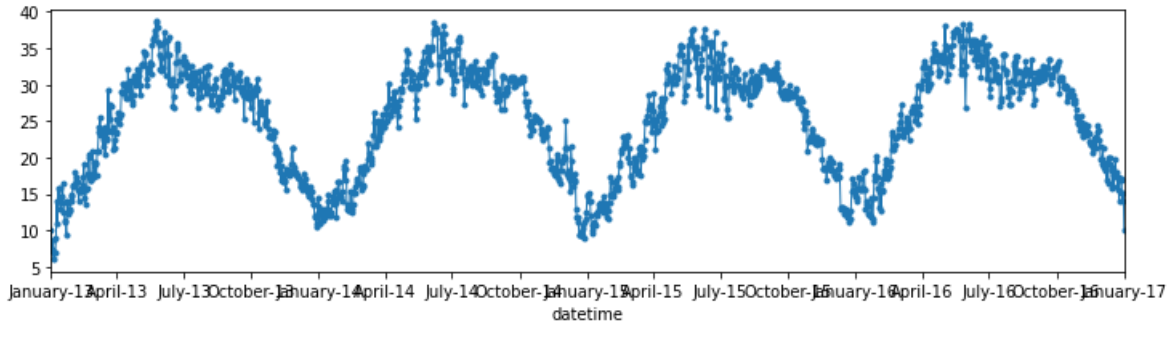
- Each matplotlib axis can have both major and minor ticks.
- Matplotlib use 2 types of artists to control ticks:
  - a Locator which determines WHERE the ticks appear
  - and a Formatter which determines the format of the tick label's text.
- The way we use these Artist objects is to:
  - Instantiate an appropriate Locator and/or Formatter
    - e.g.: `locate_months = mdates.MonthLocator()`
  - Apply your locator and/or formatter to the correct axis and types of ticks:
    - Start the line of code by slicing out the axis you want to customize from your Axis. (e.g. `ax.xaxis` or `ax.yaxis`)
    - Then chain on the "set_major_locator"/"set_major_formatter" to customize the major ticks. e.g. `ax.xaxis.set_major_locator(locate_months)` .
- These formatters won't apply rotation or changes to the text alignment, so we will do that after updating our ticks using the `ax.set_xticklabels()` method.

We will demonstrate how these tools work.

```
## creating our tick locators and formatters
# create the locator to place ticks every 3 months.
loc_3months = mdates.MonthLocator(interval=3)
# create the formatter to display 3-letter month names + 2-digit year
fmt_months = mdates.DateFormatter("%B-%y")
```

Now that we have defined our locator and formatter, lets use these to produce our visual.

```
## Plot the time series with pandas
## Adding . markers and a thinner line
ax = ts.plot(marker='.',lw=1);
ax.xaxis.set_major_locator(loc_3months)
ax.xaxis.set_major_formatter(fmt_months)
```



Ok, this isn't quite what we wanted, but you can see we are moving in the correct direction!

Stylizing visuals is an iterative process!

Fixing Overlapping Labels

- For simple date label formatting, we can take advantage of the matplotlib Figure method: `fig.autofmt_xdate()`
- However, this requires that we have the figure itself. When starting a visualization with Pandas, we only receive the Axis.
- Three ways to get the Figure:
    1. Use `ax.get_figure()` :
        - If you already have the axis.
    2. Use `plt.gcf()` #get current figure:
        - If you do not have the axis and are running the cell that creates the figure.
    3. Make the fig and ax first and then use fig, ax = plt.subplots() followed by ts.plot(ax=ax)

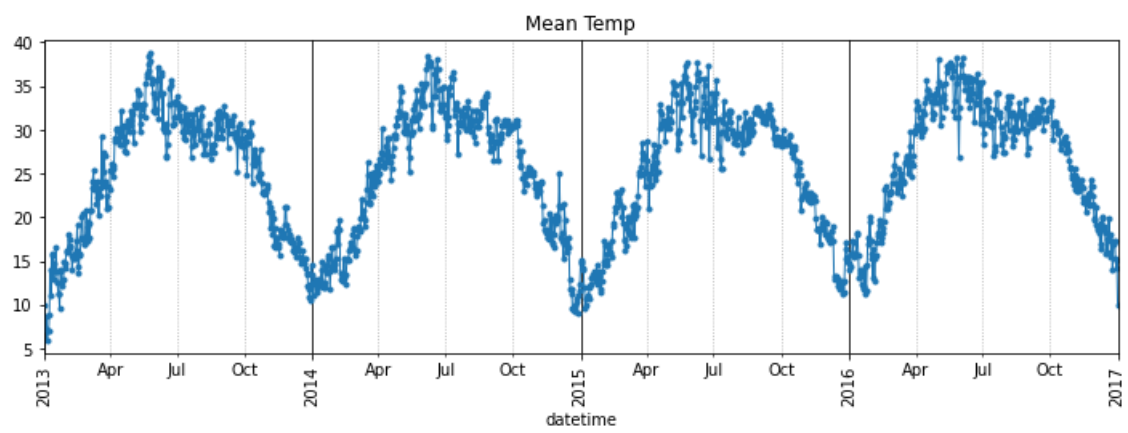
```
##  Plot the time series with pandas
ax = ts.plot(marker='.',lw=1);
ax.xaxis.set_major_locator(loc_3months)
ax.xaxis.set_major_formatter(fmt_months)
##  Using ax.get_figure()
fig = ax.get_figure()
fig.autofmt_xdate()
```


Customizing Our Ticks (advanced)

- The figure above now displays the month, but its harder to visually see the years.
- To fix this, we will apply 2 locators/formatters:
    - One for the major ticks
    - One for the minor ticks
- For the figure above, we would like to:
    - Place minor ticks at every 3 months
    - Label minor ticks with 3-letter month names
    - Place major ticks at every year.
    - Label years with 4-digit years.
    - Make the grid lines for years darker than the grid lines for months.

```
## creating our tick locators and formatters
## for minor month ticks
# create the locator to place ticks every 3 months.
loc_3months = mdates.MonthLocator(interval=3)
fmt_months = mdates.DateFormatter("%b")
## for major year ticks
loc_year = mdates.YearLocator()
fmt_year = mdates.DateFormatter("%Y")
```

```
## Make the fig and axis first
# plot the time series with pandas
fig, ax = plt.subplots(figsize=(12,4))
ts.plot(ax=ax,title='Mean Temp',marker='.',lw=1)
## customize minor ticks
ax.xaxis.set_minor_locator(loc_3months)
ax.xaxis.set_minor_formatter(fmt_months)
## customize major ticks
ax.xaxis.set_major_locator(loc_year)
ax.xaxis.set_major_formatter(fmt_year)
## Making major/minor gridlines visually distinct
ax.grid(which='minor',axis='x',ls=":")
ax.grid(which='major',axis='x',color='k')
fig.autofmt_xdate(rotation=90,ha='center')
```



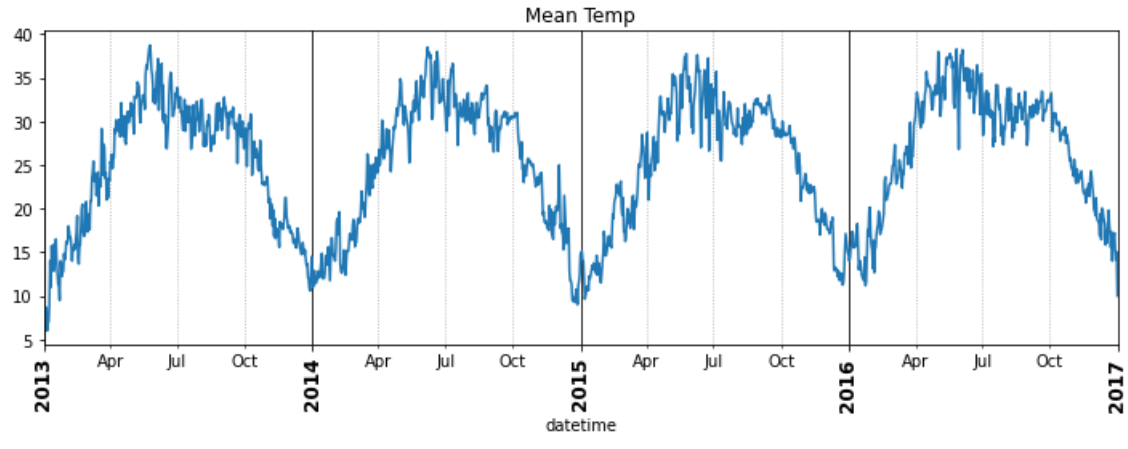
Some Additional Customizations

Increasing Font Size of Major Tick Labels

- Finally, we are going to change the formatting of our major xticklabels to use a large fontsize and bold fontweight.

```
##  Make the fig and axis first
#  plot the time series with pandas
fig, ax = plt.subplots(figsize=(12,4))
ts.plot(ax=ax,title='Mean Temp');
##  saving current xticks
ax.set_xticklabels(ax.get_xticklabels(), fontsize='large',fontweight='bold' )
##  customize minor ticks
ax.xaxis.set_minor_locator(loc_3months)
ax.xaxis.set_minor_formatter(fmt_months)
##  customize major ticks
ax.xaxis.set_major_locator(loc_year)
ax.xaxis.set_major_formatter(fmt_year)
##  Making major/minor gridlines visually distince
ax.grid(which='minor',axis='x',ls=":")
ax.grid(which='major',axis='x',color='k')
fig.autofmt_xdate(rotation=90,ha='center')
```

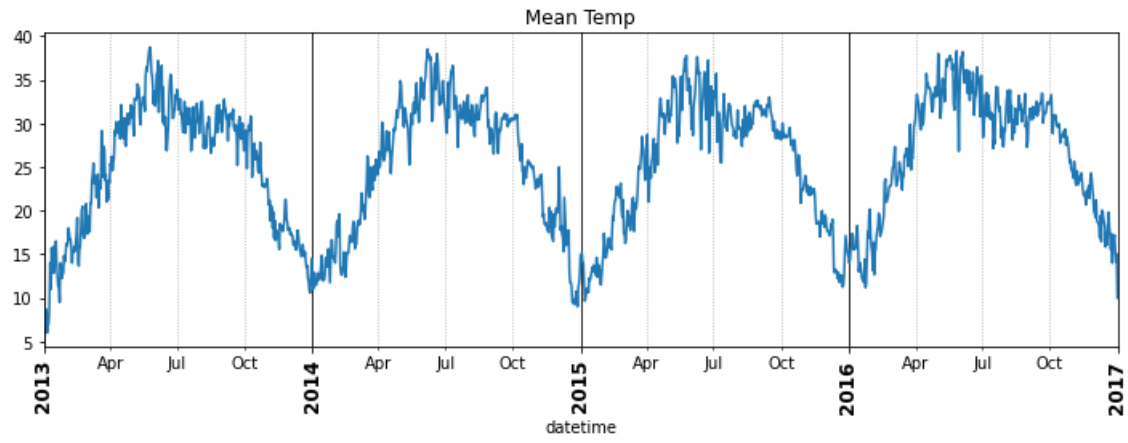
```
/var/folders/rf/vw4r41jd7vd95x1w0dth7v9h0000gp/T/ipykernel_3230/1277064622.py:8:
UserWarning: FixedFormatter should only be used together with FixedLocator
ax.set_xticklabels(ax.get_xticklabels(), fontsize='large',fontweight='bold' )
```



- The warning message we receive is a bit of a red-herring. There are some advanced scenarios where it may cause an issue, but for our use we will likely not run into the problem.
- To make the warning disappear though, we can also set the xticks themselves when we set the xticklabels.

> *NOTE: when combining tick formatters/locators & ax.set_xticklaebs(): - you need to use ax.set_xticklabels() before applying the formatters, or you may lose your labels!*

```
##  Make the fig and axis first
#  plot the time series with pandas
fig, ax = plt.subplots(figsize=(12,4))
ts.plot(ax=ax,title='Mean Temp');
##  Set xticks and xticklables
ax.set_xticks(ax.get_xticks())
ax.set_xticklabels(ax.get_xticklabels(), fontsize='large',fontweight='bold' )
##  customize minor ticks
ax.xaxis.set_minor_locator(loc_3months)
ax.xaxis.set_minor_formatter(fmt_months)
##  customize major ticks
ax.xaxis.set_major_locator(loc_year)
ax.xaxis.set_major_formatter(fmt_year)
##  Making major/minor gridlines visually distince
ax.grid(which='minor',axis='x',ls=":")
ax.grid(which='major',axis='x',color='k')
fig.autofmt_xdate(rotation=90,ha='center')
```



Creating a Function for your Code

Once you have decided on a style that you like, you can work more efficiently by defining your plot formats and styles within a function. Take some time to read this function! Do you understand it? Think about how you could modify this function to achieve different results!

```
## we've done enough customization that it makes sense to bundle this in a
function
def plot_dates(ts,figsize=(12,4),xtick_fontsize='large', xtick_fontweight='bold',
               title= None):## Make the fig and axis first

    if title is None:
        title = ts.name
    # plot the time series with pandas
    fig, ax = plt.subplots(figsize=figsize)
    ts.plot(ax=ax,title=title);

    ## creating our tick locators and formatters
    ## for minor month ticks
    # create the locator to place ticks every 3 months.
    loc_3months = mdates.MonthLocator(interval=3)
    fmt_months = mdates.DateFormatter("%b")
    ## for major year ticks
```

Previous

Next