Notes on sktime Pull Request

Eric Berger

2024-05-06

Table of contents

1 Addition of 27 FPP3 datasets to sktime/datasets

We attempt to convert to `sktime mtypes` the 27 tsibble objects defined in "Forecasting: Principles and Practice" by Rob J Hyndman and George Athanasopoulos. (Hyndman and Athanasopoulos 2021) The third edition of this book (and associated packages) are referred to as `fpp3`.

The R `tsibble` class defined in the R `tsibble` package is a derived class of the R `tibble` class which is itself derived from the R `data.frame` class. The `tsibble` class in its full generality is comparable to the `sktime pd-multiindex` and `pd_multiindex_hier mtypes`. The terminology is slightly different. For the `tsibble`, the non-timepoints of the multi-index are referred to as the `key`, and the timepoints comprise the `index`. Also, the `tsibble` term `regular` corresponds to the `sktime` term `equally-spaced`.

Our understanding of the `sktime` terms `scitype` and `mtype` comes from the `sktime` tutorial notebook (sktime 2024).

1.1 Description of the 27 R tsibble Datasets

Table 1: R Tsibble Attributes for the 27 Datasets

dataset	#rows	#cols	#series	#keys	#key cols	index_type	period	regular
ansett	7407	4	30	30	2	yearweek	1 week	TRUE
aus_accommodation	592	5	24	8	1	yearquarter	1 quarter	TRUE
aus_airpassengers	47	2	1	1	0	numeric	1 year	TRUE
aus_arrivals	508	3	4	4	1	yearquarter	1 quarter	TRUE
aus_livestock	29364	4	54	54	2	yearmonth	1 month	TRUE
aus_production	218	7	6	1	0	yearquarter	1 quarter	TRUE
aus_retail	64532	5	304	152	2	yearmonth	1 month	TRUE
bank_calls	27716	2	1	1	0	POSIXct	5 minute	TRUE
boston_marathon	265	5	15	5	1	integer	1 year	TRUE
canadian_gas	542	2	1	1	0	yearmonth	1 month	TRUE
gafa_stock	5032	8	24	4	1	Date	NA	FALSE
global_economy	15150	9	1841	263	1	numeric	1 year	TRUE
guinea_rice	42	2	1	1	0	numeric	1 year	TRUE
hh_budget	88	8	24	4	1	numeric	1 year	TRUE
insurance	40	3	2	1	0	yearmonth	1 month	TRUE
nyc_bikes	4268	12	100	10	1	POSIXct	NA	FALSE
olympic_running	312	4	14	14	2	integer	4 year	TRUE
PBS	67596	9	1344	336	4	yearmonth	1 month	TRUE
pedestrian	66037	5	12	4	1	POSIXct	1 hour	TRUE
pelt	91	3	2	1	0	numeric	1 year	TRUE
prices	198	7	6	1	0	numeric	1 year	TRUE
souvenirs	84	2	1	1	0	yearmonth	1 month	TRUE
tourism	24320	5	304	304	3	yearquarter	1 quarter	TRUE
us_change	198	6	5	1	0	yearquarter	1 quarter	TRUE
us_employment	143412	4	296	148	1	yearmonth	1 month	TRUE
us_gasoline	1355	2	1	1	0	yearweek	1 week	TRUE
vic_elec	52608	5	4	1	0	POSIXct	30 minute	TRUE

1.2 The sktime mtypes of the FPP3 Datasets

Table 2: Datasets Sorted by sktime mtype

pd.Series
aus_airpassengers
bank_calls
canadian_gas
guinea_rice
souvenirs
us_gasoline
pd.DataFrame
aus_production
insurance
pelt
prices
us_change
vic_elec
univariate pd-multiindex
aus_arrivals
univariate pd_multiindex_hier
ansett
aus_livestock
olympic_running
tourism
multivariate pd-multiindex
aus_accommodation
boston_marathon
gafa_stock
global_economy
hh_budget
nyc_bikes
pedestrian
us_employment
multivariate pd_multiindex_hier
aus_retail
PBS

Table 3: Datasets Sorted Alphabetically

Dataset	**sktime mtype**
ansett	univariate pd_multiindex_hier
aus_accommodation	multivariate pd-multiindex
aus_airpassengers	pd.Series
aus_arrivals	univariate pd-multiindex
aus_livestock	univariate pd_multiindex_hier
aus_production	pd.DataFrame
aus_retail	multivariate pd_multiindex_hier
bank_calls	pd.Series
boston_marathon	multivariate pd-multiindex
canadian_gas	pd.Series
gafa_stock	multivariate pd-multiindex
global_economy	multivariate pd-multiindex
guinea_rice	pd.Series
hh_budget	multivariate pd-multiindex
insurance	pd.DataFrame
nyc_bikes	multivariate pd-multiindex
olympic_running	univariate pd_multiindex_hier
PBS	multivariate pd_multiindex_hier
pedestrian	multivariate pd-multiindex
pelt	pd.DataFrame
prices	pd.DataFrame
souvenirs	pd.Series
tourism	univariate pd_multiindex_hier
us_change	pd.DataFrame
us_employment	multivariate pd-multiindex
us_gasoline	pd.Series
vic_elec	pd.DataFrame

1.3 Issues Encountered

- The **ansett** series metadata returned by **check_is_mtype()** reports equally_spaced as False. This took a while to track down but it is accurate. For example, the csv file with the data shows that the key SYD-PER, First Class has no entry for week "1990 W01". (Which begs the question why Table 1 reports the **ansett** dataset as regular.)
- The **boston_marathon** series fails the **check_is_mtype()** because the DataFrame contains two descriptive columns - Champion and Country. These columns are not part of the multiindex. The test passes if these columns are removed.
- The **global_economy** series fails the **check_is_mtype()** because the DataFrame contains one descriptive column - Code (the country code). This column is not part of the multiindex. The test passes if this column is removed.
- The **nyc_bikes** series fails the **check_is_mtype()** because the DataFrame contains 3 problematic columns - stop_time, type, gender. These columns are not part of the multiindex. Type and gender are descriptive. The stop_time column could be converted into a numerical type, but this was not done.
- The **us_employment** series fails the **check_is_mtype()** because the DataFrame contains one descriptive column - Title. This column is not part of the multiindex. The test passes if this column is ignored.
- The **aus_retail** series fails the **check_is_mtype()** because the DataFrame contains one descriptive column - 'Series ID'. This column is not part of the multiindex. The test passes if this column is ignored.

1.4 Accessing the FPP3 Datasets

Example:

```python
from sktime.datasets import load_fpp3

aus_retail = load_fpp3("aus_retail")

# checking mytpe
from sktime.datatypes import check_is_mtype
isnot = ['not','']
ret = check_is_mtype(aus_retail, mtype="pd_multiindex_hier")
print(f"aus_retail is {isnot[ret]} a pd_multiindex_hier")

# drop redundant column and check mtype
aus_retail = aus_retail.drop('Series ID', axis=1)
ret = check_is_mtype(aus_retail, mtype="pd_multiindex_hier")
print(f"aus_retail with redundant col dropped is {isnot[ret]} a pd_multiindex_hier")
```

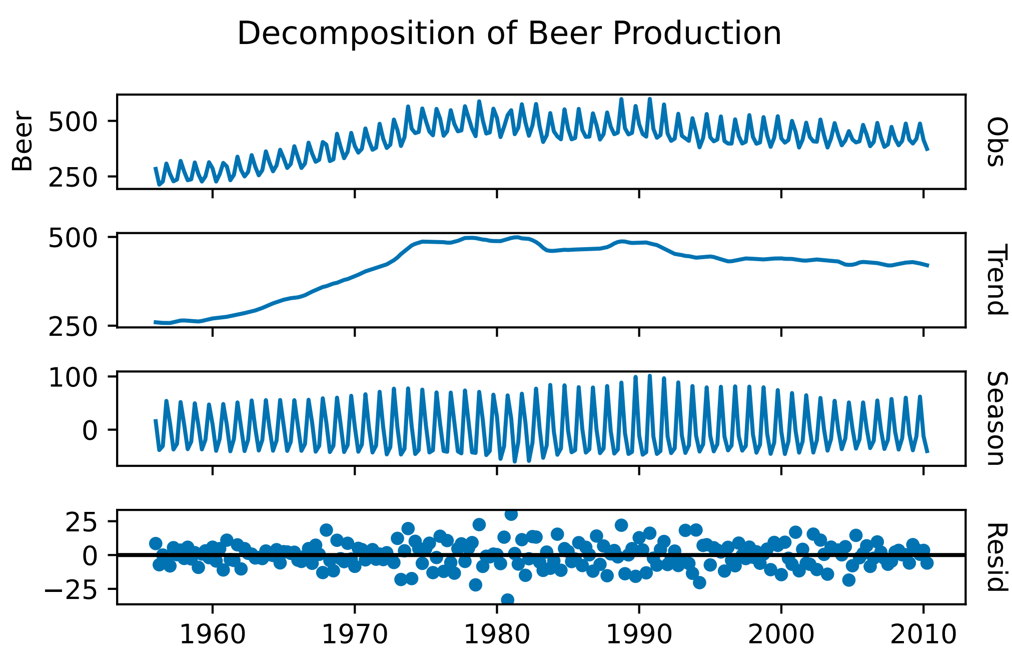
```
aus_retail is not a pd_multiindex_hier
aus_retail with redundant col dropped is  a pd_multiindex_hier
```

To get the csv files I benefited from the GitHub repository `zgana/fpp3-python-readalong`.

2 New method for STLForecaster to Plot STL Decomposition: plot_components()

Example:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sktime.datasets import load_fpp3
from sktime.forecasting.trend import STLForecaster

y = load_fpp3('aus_production')['Beer']
TITLE = "Decomposition of Beer Production"
fig, ax = STLForecaster(sp=4).fit(y).plot_components(TITLE)
plt.show()
```



3 xticks and xticklabels in `sktime.utils.plot_series()`

This release replaces the way that xticks and xticklabels are created in `sktime.utils.plot_series()`. The previous version does a custom calculation for this. The new version uses matplotlib's default choice for xticks and xticklabels.

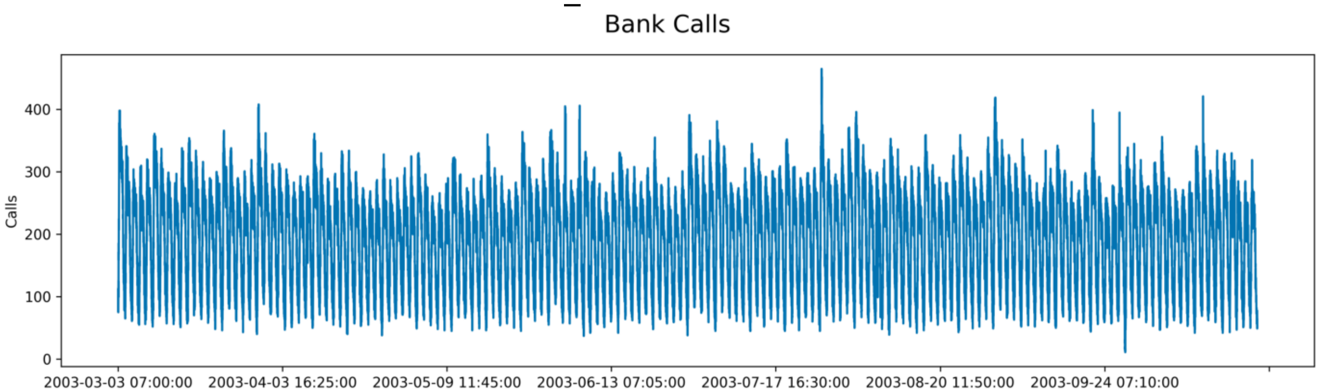
3.1 Visual differences - full series

In many cases the previous and new versions produce similar output. Here is a case where they don't. The previous version chooses xticklabels that show the full time stamp YYYY-MM-DD HH:MM:SS. The new version uses only YYYY-MM for the xticklabel.

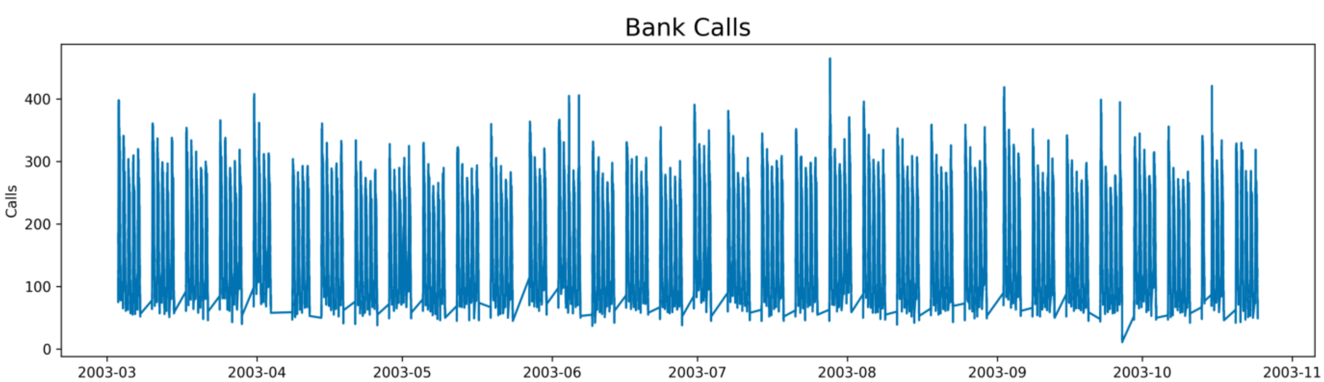
The following code was run for the previous and new versions:

```
y = load_fpp3('bank_calls')
fig, ax = plot_series(y, markers=[''], title='Bank Calls')
```

bank_calls - Old Kernel



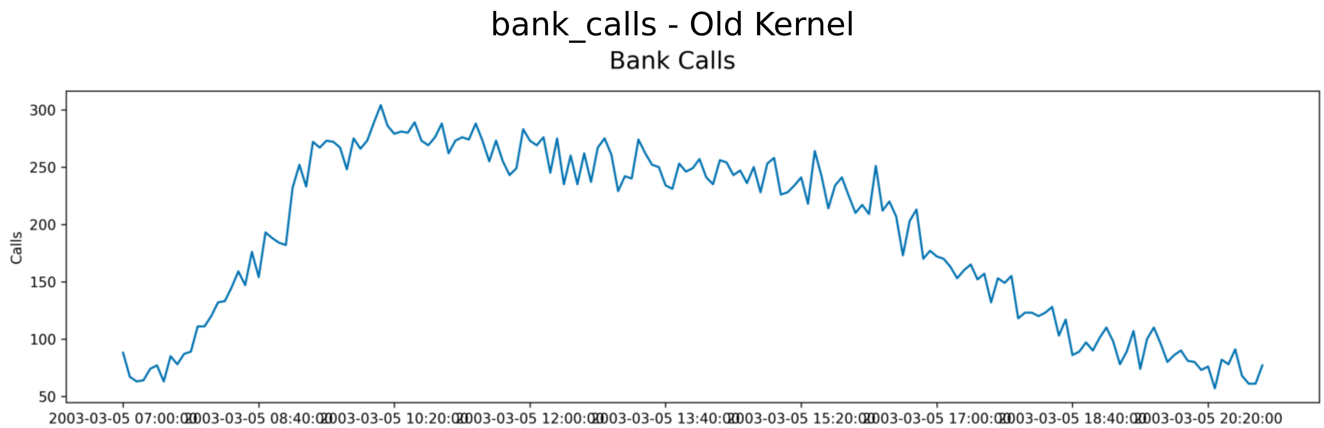
bank_calls - New Kernel

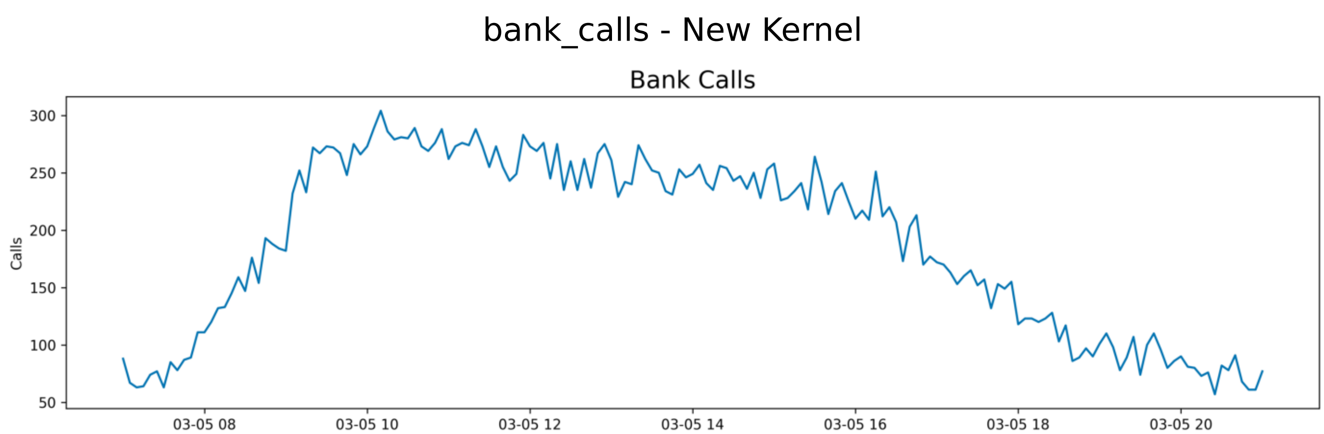


3.2 Visual differences - zoom in on series to one calendar day

We zoom into a single calendar day. This example shows that matplotlib's default behaviour is much better.

```
y = load_fpp3('bank_calls')
y = y.loc['2003-03-05']
fig, ax = plot_series(y, markers=[''], title='Bank Calls')
```

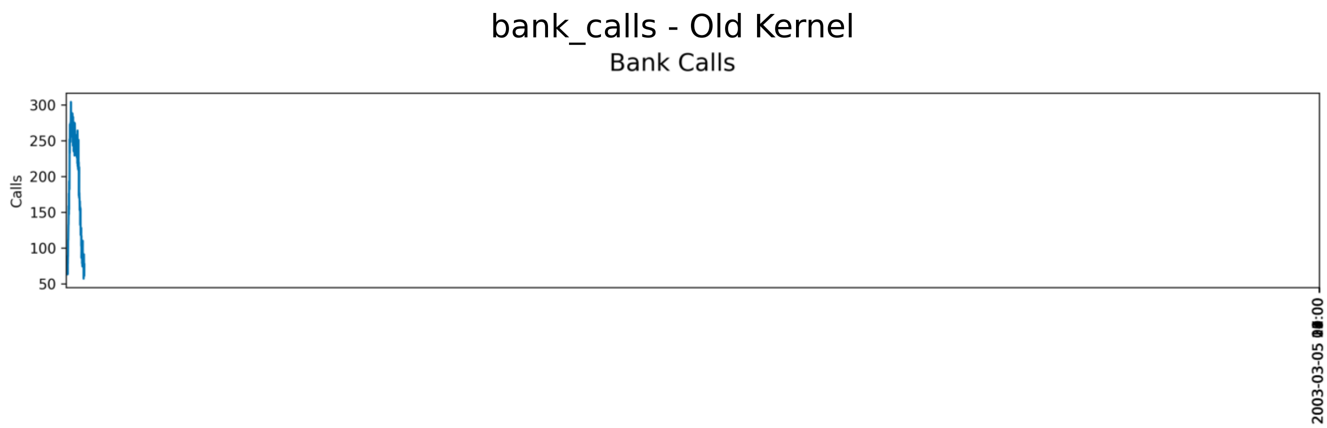

bank_calls - Old Kernel
Bank Calls
Calls
300
250
200
150
100
50
2003-03-05 07:00:00
2003-03-05 08:40:00
2003-03-05 10:20:00
2003-03-05 12:00:00
2003-03-05 13:40:00
2003-03-05 15:20:00
2003-03-05 17:00:00
2003-03-05 18:40:00
2003-03-05 20:20:00

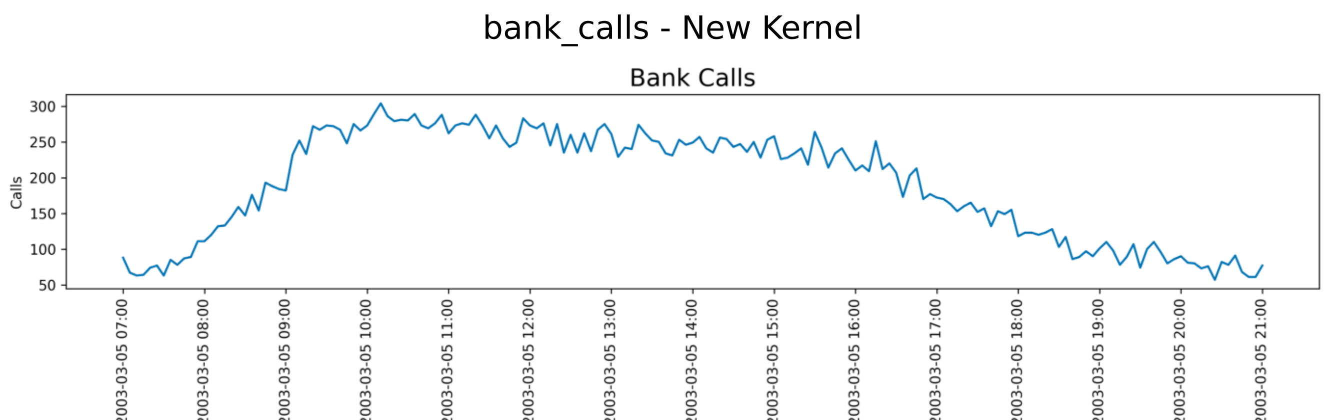

bank_calls - New Kernel
Bank Calls
Calls
300
250
200
150
100
50
03-05 08
03-05 10
03-05 12
03-05 14
03-05 16
03-05 18
03-05 20

3.3 Visual differences - one calendar day with rotated ticklabels

Rotating the xaxis xticklabels is a disaster for the previous version. This is because the previous version of `plot_series` does not conform to matplotlib's conventions for setting the x-axis scale. The new version works as expected.

```
y = load_fpp3('bank_calls')
y = y.loc['2003-03-05']
fig, ax = plot_series(y, markers=[''], title='Bank Calls')
xticks = pd.date_range(start=y.index.min(), end=y.index.max(), freq='h')
ax.set_xticks(xticks)
plt.xticks(rotation=90)
ax.set_xticklabels([tick.strftime('%Y-%m-%d %H:%M') for tick in xticks])
plt.subplots_adjust(bottom=0.4)
```

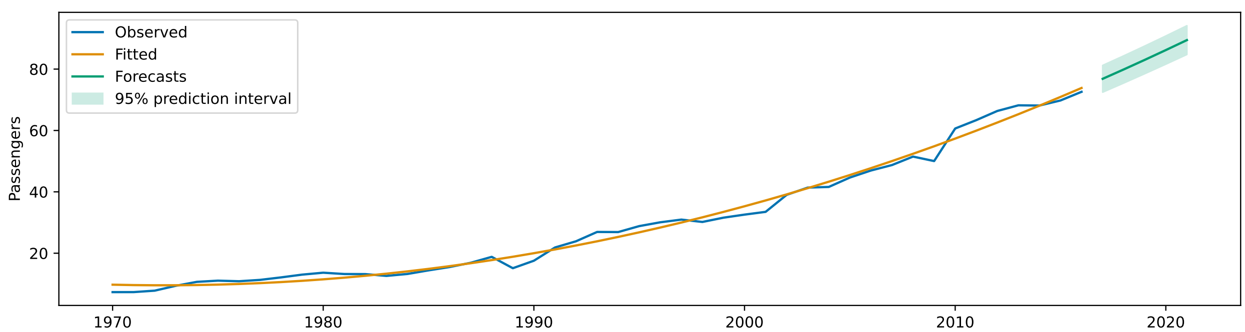

bank_calls - Old Kernel
Bank Calls
Calls
300
250
200
150
100
50


bank_calls - New Kernel
Bank Calls
Calls
300
250
200
150
100
50
2003-03-05 07:00
2003-03-05 08:00
2003-03-05 09:00
2003-03-05 10:00
2003-03-05 11:00
2003-03-05 12:00
2003-03-05 13:00
2003-03-05 14:00
2003-03-05 15:00
2003-03-05 16:00
2003-03-05 17:00
2003-03-05 18:00
2003-03-05 19:00
2003-03-05 20:00
2003-03-05 21:00

4 New method for PolynomialTrendForecaster: predict_interval()

Example

```
import numpy as np
import matplotlib.pyplot as plt
from sktime.datasets import load_fpp3
from sktime.forecasting.trend import PolynomialTrendForecaster
from sktime.forecasting.base import ForecastingHorizon
from sktime.utils import plot_series

y = load_fpp3("aus_airpassengers")

forecaster = PolynomialTrendForecaster(degree=2)
forecaster.fit(y)
fitted_values = forecaster.predict(ForecastingHorizon(y.index, is_relative=False))
fh = ForecastingHorizon(np.arange(1,6), is_relative=True)
pred_values = forecaster.predict(fh)
pred_interval = forecaster.predict_interval(fh, coverage=[0.95])
fig, ax = plot_series(y, fitted_values, pred_values, pred_interval=pred_interval,
                      markers=['']*3, labels=['Observed', 'Fitted', 'Forecasts'])
plt.show()
```



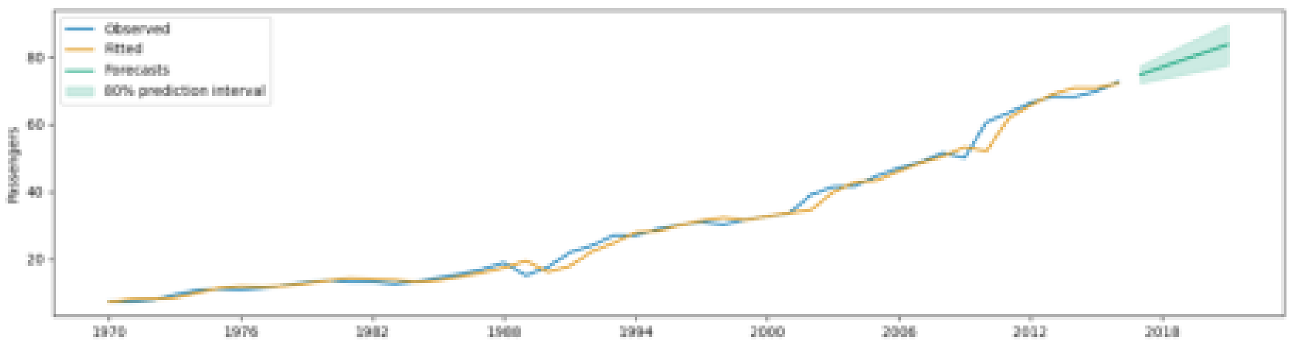
5 Multiple Prediction Intervals in sktime.utils.plot_series()

This may be a bug or a missing feature. The `plot_series()` function should support the plotting of multiple confidence levels when prediction intervals are passed. This is now working.

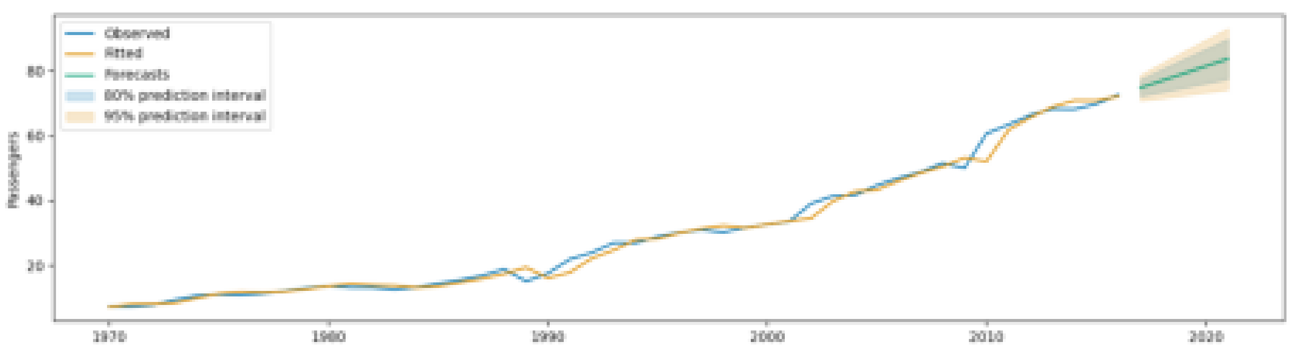
The following code was run for the previous and new versions:

```python
import numpy as np
import matplotlib.pyplot as plt
from sktime.datasets import load_fpp3
from sktime.forecasting.ets import AutoETS
from sktime.forecasting.base import ForecastingHorizon
from sktime.utils import plot_series

y = load_fpp3("aus_airpassengers")

forecaster = AutoETS(error="add", trend="add", damped_trend=False)
forecaster.fit(y)
fitted_values = forecaster.predict(ForecastingHorizon(y.index, is_relative=False))
fh = ForecastingHorizon(np.arange(1,6), is_relative=True)
pred_values = forecaster.predict(fh)
pred_interval = forecaster.predict_interval(fh, coverage=[0.95, 0.8])
fig, ax = plot_series(y, fitted_values, pred_values, pred_interval=pred_interval,
                      markers=['']*3, labels=['Observed', 'Fitted', 'Forecasts'])
plt.show()
```

confidence_intervals_ets - Old Kernel



confidence_intervals_ets - New Kernel



References

Hyndman, R. J., and G. Athanasopoulos. 2021. *Forecasting: Principles and Practice.* 3rd ed. Melbourne, Australia: OTexts. https://otexts.com/fpp3/.

sktime. 2024. "AA Datatypes and Datasets." Online Python Notebook. https://github.com/sktime/sktime/blob/main/examples/AA_datatypes_and_datasets.ipynb.