`tsbootstrap`: Enhancing Time Series Analysis with Advanced Bootstrapping Techniques

Sankalp Gilda SANKALP.GILDA@GMAIL.COM
ML Collective
San Francisco, CA 98195-4322, USA

Benedikt Heidrich BENEDIKT.HEIDRICH@SKTIME.NET
sktime project

Franz Kiraly FRANZ.KIRALY@SKTIME.NET
sktime project

Abstract

In time series analysis, traditional bootstrapping methods often fall short due to their assumption of data independence, a condition rarely met in time-dependent data. This paper introduces `tsbootstrap`, a `python` package designed specifically to address this challenge. It offers a comprehensive suite of bootstrapping techniques, including Block, Residual, and advanced methods like Markov and Sieve Bootstraps, each tailored to respect the temporal dependencies in time series data. This framework not only enhances the accuracy of uncertainty estimation in time series analysis but also integrates seamlessly with the existing `python` data science ecosystem, making it an invaluable asset for researchers and practitioners in various fields.

Keywords: Time Series, Bootstrapping, Resampling, Uncertainty Quantification

1 Introduction

Time series data, characterized by their sequential observations, are the backbone of critical decision-making in various scientific and economic fields. Unlike static or cross-sectional datasets, time series data are dynamic, evolving over time and exhibiting intricate dependencies that reflect the complexities of the real world. This dynamic nature poses unique challenges, particularly when it comes to predicting future events (forecasting) and assessing the reliability of these predictions (uncertainty quantification). Traditional statistical methods, designed for independent and identically distributed (IID) data, falter in this context, as they overlook the temporal correlations that are fundamental to time series data. This gap in the statistical toolkit highlights a pressing need for methodologies that can adeptly navigate the intricacies of time series, preserving their inherent structure while facilitating rigorous analysis and inference.

Bootstrapping (Efron, 1992; Chernick, 2011) offers a powerful tool for estimating the variability of a statistic without making stringent assumptions about the underlying data distribution. However, the classic IID bootstrapping approach, which resamples data points as if they were independent, is ill-suited for time series where each observation is a chapter in an ongoing story, not an isolated event. To capture the essence of time series, bootstrapping must adapt, preserving the internal structure and dependencies within the data.

Addressing this need, various time series bootstrapping methods have emerged, each designed to respect the chronological integrity of the data while enabling robust uncertainty quantification (Kreiss and Lahiri, 2012). Yet, the software implementation of these methods presents its own set of challenges. They require not just statistical acumen but also computational finesse, demanding algorithms that can handle the complexity of time series data while being accessible to the broader scientific and industrial community.

`tsbootstrap` (Gilda, 2024) emerges as a response to the complexities and specific needs in time series analysis. It is a `python` package designed to provide a range of bootstrapping methods tailored for time series data. This package includes traditional techniques such as Block Bootstrap (Kunsch, 1989) and extends to more sophisticated methods like Markov and Sieve Bootstrap (Bühlmann, 1997). Each method within `tsbootstrap` is implemented to respect and preserve the chronological order and inherent correlations of time series data, ensuring more accurate and reliable bootstrapping.

We have designed `tsbootstrap` as a versatile and user-friendly tool to empower both academic researchers and industry practitioners, offering them new capabilities to navigate and interpret the complexities of time series data with unprecedented precision and insight. Specifically, we fill in two existing gaps:

- *Framework for Enhanced Uncertainty Quantification:* `tsbootstrap` provides a robust framework that significantly advances the practice of uncertainty quantification in time series analysis. This is particularly crucial in fields such as finance, where accurate risk assessment is essential; in meteorology, where predictive accuracy can have wide-ranging implications; and in epidemiology, where understanding uncertainty can inform public health decisions.

- *Seamless Integration with the Time Series Analysis Ecosystem:* The package is meticulously designed to complement and enhance the existing `python` data science ecosystem. By "time series analysis ecosystem", we refer to the collection of tools, libraries, and methodologies that are commonly employed in the analysis of time series data. This includes integration with popular libraries like `sktime` (sktime developers, 2024) and `pandas` (pandas development team, 2024), ensuring that `tsbootstrap` is not only a standalone tool but also a synergistic component that enhances the capabilities of other time series analysis tools.

In the rest of this paper, we delve into `tsbootstrap` in more depth, offering a comprehensive examination of its theoretical underpinnings, implementation details, and the breadth of its application. We demonstrate its utility in elevating time series analysis through case studies and comparative analyses, emphasizing its role as an essential instrument for researchers and practitioners. In subsequent sections we provide a structured exploration. In Section 2 we introduce bootstrapping and its significance. In Section 3, we focuse on `tsbootstrap`'s design principles and functionalities. In Section 4 we detail the supported bootstrap algorithms. In Section 5, we illustrate practical usage, integration with `sktime`, and extension capabilities. Finally, we conclude in Section 6, and outline the future trajectory of `tsbootstrap` in Section 7.

2 Bootstrapping

Bootstrapping is a resampling technique fundamental to statistics, providing a way to estimate the distribution of a sample statistic (like the mean or variance) by resampling with replacement from the data. It allows statisticians to assess the reliability of inferential statistics, such as confidence intervals and significance tests, without relying on stringent assumptions about the underlying population distribution.

At its core, bootstrapping involves repeatedly drawing samples, typically thousands of times, from a single observed dataset. Each sample is drawn with replacement, meaning the same data point can appear multiple times in a bootstrap sample. By applying the statistical measure of interest to each resample, a distribution of these statistics is generated, offering insight into the variability and bias of the estimate.

To underscore the importance of bootstrapping in various statistical processes, consider the following key aspects:

- *Model-Free Inference:* Bootstrapping does not assume a specific parametric form for the data distribution, making it a non-parametric approach. This flexibility is crucial when the underlying distribution is unknown or complex, allowing statisticians to avoid potentially incorrect model assumptions.
- *Assessment of Uncertainty:* It provides a straightforward method to estimate the confidence intervals and standard errors of estimates, which are vital for understanding the precision of statistical inferences.
- *Validation of Models:* In model building, bootstrapping can be used to validate the stability and reliability of predictive models. It helps in assessing the model's variance and bias, offering insights into its generalizability.
- *Complex Estimators:* Bootstrapping is particularly valuable for complex estimators where the theoretical distribution is difficult or impossible to derive. It allows for the empirical examination of the estimator's distribution.

Transitioning to the specific challenges of time series data, it's crucial to recognize that the inherent dependencies within these data sets necessitate a more nuanced approach to bootstrapping. Traditional IID bootstrapping techniques, which treat each data point as independent, are not suitable for time series data due to their sequential nature (Lahiri, 2013). This misalignment highlights the need for specialized time series bootstrapping methods that respect the data's temporal structure, ensuring the integrity of the analysis.

Bootstrapping Method	Class (...`Bootstrap`)	Salient Features
Moving Block	`MovingBlock`	Suitable for series with significant intra-block dependencies; avoid for weak dependencies
Circular Block	`CircularBlock`	Ideal for seasonal/cyclical data; not recommended for non-cyclical data
Stationary	`StationaryBlock`	Best for data with varying dependency lengths; less effective for uniform dependency lengths
Non-Overlapping Block	`NonOverlappingBlock`	Effective for data with distinct segments; not suitable for continuous or highly dependent data
Tapered Block	`Bartletts`, `Hamming`, `Blackman`, `Tukey`	Mitigates edge effects using window functions; requires matching window functions to data characteristics
Residual	`WholeResidual`, `BlockResidual`	Evaluates model-based uncertainty; not intended for non-model analyses
Statistic-Preserving	`WholeStatisticPreserving`, `BlockStatisticPreserving`	Maintains key statistical properties; not for general purposes without specific preservation needs
Distribution	`WholeDistribution`, `BlockDistribution`	Leverages known distributions; inappropriate for unknown distributions
Markov	`WholeMarkov`, `BlockMarkov`	Suitable for Markovian time series; avoid for non-Markovian series
Sieve	`WholeSieve`, `BlockSieve`	Designed for autoregressive models; not recommended for non-autoregressive series

Table 1: Overview of bootstrapping classes in `tsbootstrap`.

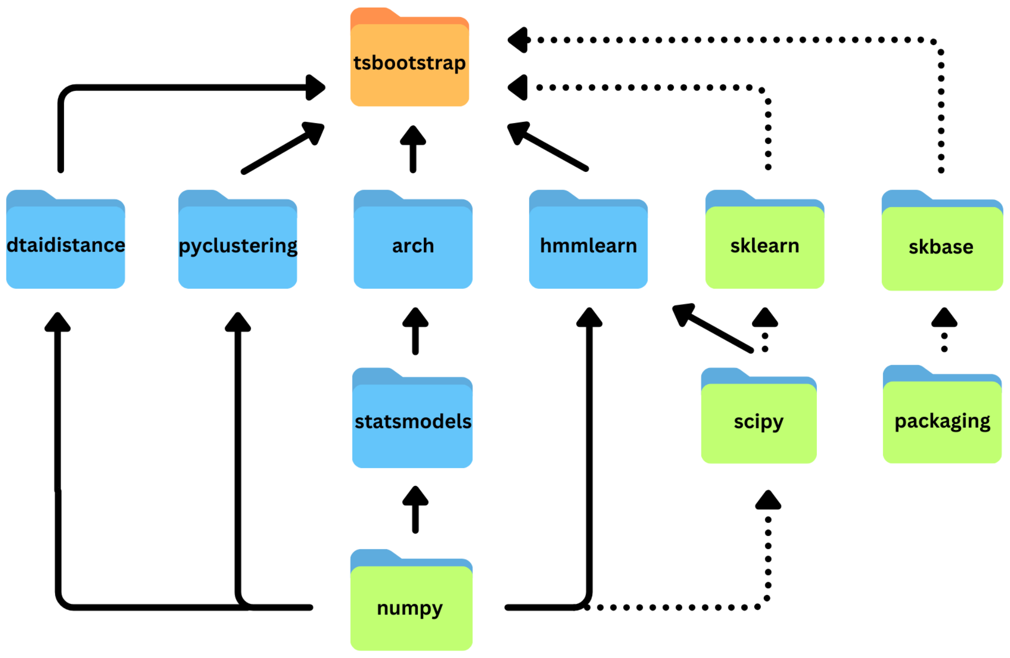

Figure 1: Optional (soft) and required (hard) dependencies for `tsbootstrap` v0.1.0. The latter are indicated via lime-colored folders and dashed arrows.

3 `tsbootstrap`

3.1 Design Principles

1. **Modularity and Extensibility**: The package's architecture is highly modular, allowing users to easily incorporate custom bootstrapping algorithms. This flexibility makes it an adaptable tool for a wide range of time series data applications. `tsbootstrap`'s design embraces the strategy pattern and composition, ensuring that new bootstrapping strategies can be easily added, promoting a collaborative environment for the community to contribute and extend the library.

2. **User-Centric API**: The API is crafted with user experience in mind, ensuring simplicity in performing complex bootstrapping tasks. This design principle is evident in the package's comprehensive documentation and illustrative examples provided on the GitHub repository. Following `scikit-learn`-like conventions, `tsbootstrap` provides a consistent API , making it accessible and user-friendly for those accustomed to `scikit-learn`'s patterns.

3. **Integration with the `python` Ecosystem**: `tsbootstrap` is built to seamlessly integrate with popular `python` libraries like `numpy` and `pandas`, enhancing its utility in the prevalent data science ecosystem. Moreover, its design facilitates easy integration with other `python` libraries, particularly `sktime`, allowing `tsbootstrap` to be used within existing pipelines, enhancing the robustness and reliability of time series analysis.

4. **Alignment with `scikit-learn`'s Cross-Validation**: Mirroring `scikit-learn`'s cross-validation class, `tsbootstrap` makes a slight adjustment: it focuses on bootstrapping for time series data, replacing the `split` method with a `bootstrap` method. This approach is particularly suited for time series data, respecting and preserving their temporal structure during the resampling process, and providing more reliable estimates of model performance.

3.2 Core Functionalities

1. **Diverse Bootstrapping Methods**: We include various methods such as Block (Kunsch, 1989; Politis and Romano, 1994; Paparoditis and Politis, 2001), Residual, Statistic-Preserving, and Distribution Bootstrap. Each method is fine-tuned for specific types of time series data, offering users a tailored approach to their analytical needs.

2. **Customization and Flexibility**: Users can customize various parameters like block size, sampling methods, and random seed settings. This level of customization is instrumental in conducting precise and controlled bootstrapping analyses.

3. **Efficient Data Handling and Processing**: We employ advanced data handling techniques, ensuring efficient processing of large time series datasets, input data and parameter validation, and fine-grained logging.

3.3 Integration into the `python` ML Universe

`tsbootstrap` is designed to integrate seamlessly with established and widely used libraries like `sklearn` (Pedregosa et al., 2011; Grisel et al., 2024) and `sktime` (sktime developers, 2024).

`tsbootstrap` adopts the task-specific unified interface principle, and composable specification language. It also provides task independent interface points common to `sklearn` and `sktime`, such as `get_params`, `set_params`, `get_fitted_params`.

This high-level interoperability between `tsbootstrap`, `sklearn`, and adjacent framworks such as `sktime` is mediated by the `skbase` library (RNKuhns et al., 2024), providing shared interface points and base classes. This API integration also allows `tsbootstrap` to leverage `sktime`'s handling of time series specific issues such as forecasting and time-dependent splitting.

Interoperability is in-principle possible both ways, and methodologically advantageous: `sktime` can leverage `tsbootstrap` to make, or improve, probabilistic forecasts, via custom bootstrapping schemas. Conversely, `tsbootstrap` can use `sktime` forecasters or transformations to construct bootstrap schemas with specific model assumptions, e.g., forecast or model residual bootstraps.

Finally, this design of `tsbootstrap` emphasizes extensibility and community engagement, encouraging contributions that expand its range of bootstrapping techniques and enhance its integration with the broader `python` data science landscape. Using an extensible, `sklearn`-like interface mediated by `skbase`, users can easily create interface compatible bootstrap algorithms, to add to the core library, or in their own code bases. Details on extension are presented in Section 4.3.

4 Supported Bootstrap Algorithms

`tsbootstrap` offers an extensive array of bootstrapping techniques, specially designed for time series data to preserve their inherent temporal structures during the resampling process. Below we provide a detailed overview of the supported bootstrap algorithms within the package:

4.1 Block Bootstrap Methods

For time series bootstrapping, Block bootstrap methods stand out as critical tools. These methods are indispensable because they allow the preservation of dependency structures within the data, a fundamental aspect that distinguishes time series from cross-sectional data. By resampling blocks of observations rather than individual data points, these methods maintain the chronological order and inherent dependencies, which are crucial for accurate time series analysis.

Here is a deeper look into the Block Bootstrap methods provided by `tsbootstrap`:

- Moving Block Bootstrap (MBB): This method is particularly effective for time series with significant autocorrelation within contiguous segments. In MBB, overlapping blocks of data are resampled, ensuring preservation of the temporal dependence. This technique is especially useful in analyzing financial time series, where capturing the short-term dependencies is crucial for accurate forecasting and risk assessment.
- Circular Block Bootstrap (CBB): CBB extends the concept of MBB by treating the data as if it were circular, meaning that the end of the series connects back to the beginning. This method is particularly beneficial for handling seasonal or cyclical time series, as it allows the bootstrapping process to maintain the inherent cyclical properties, such as those found in meteorological or economic seasonal patterns.
- Stationary Block Bootstrap (SBB): This approach introduces a random element to the block length, providing a more dynamic resampling process. Unlike MBB or

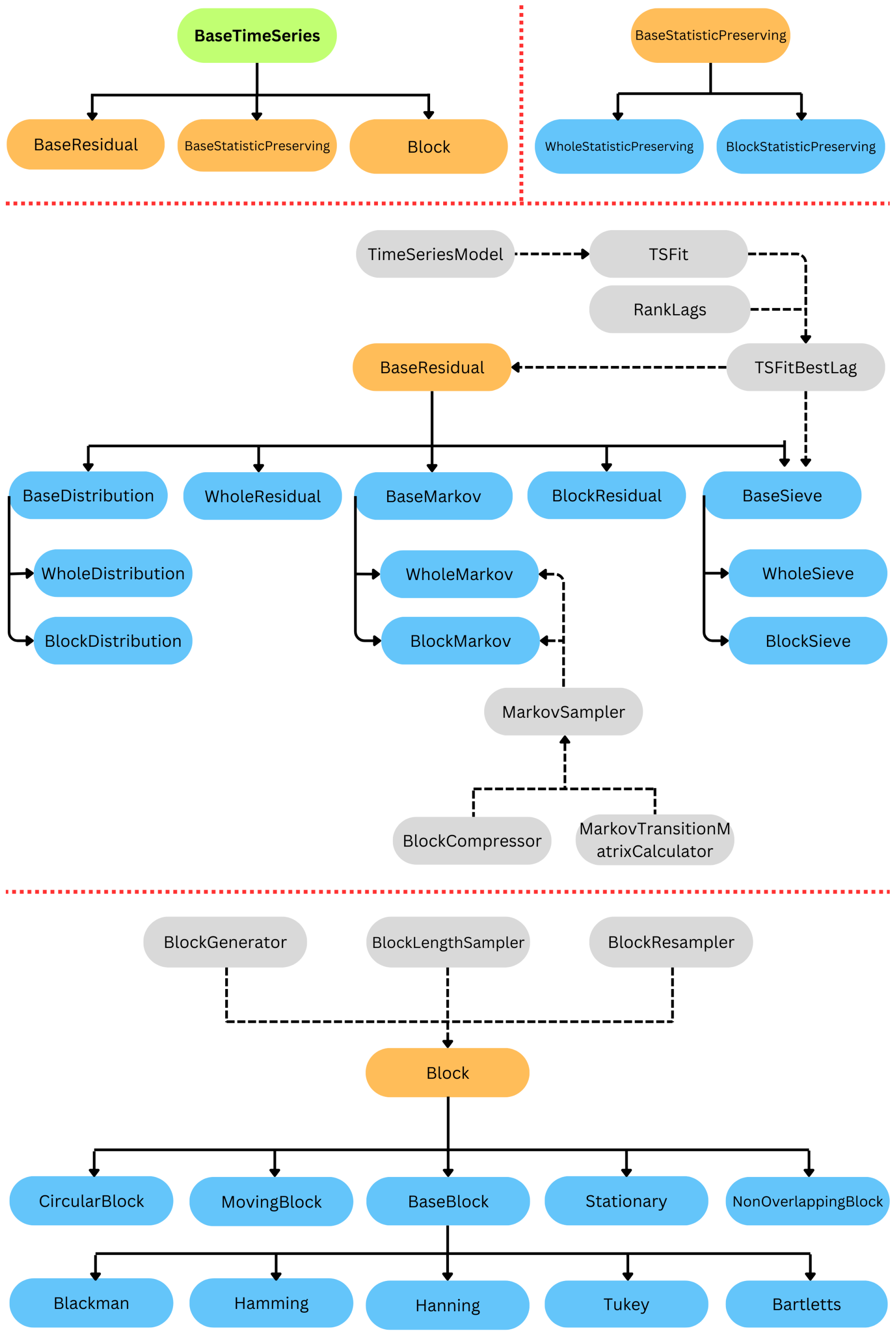

Figure 2: *Top left:* `BaseTimeSeriesBootstrap` is the base-class, and has three main child classes, indicated in Orange. *Top right, middle, and bottom:* Class structure for the three main child classes depicted from the top-left figure. gray blocks/classes are helper classes that are not exposed to the user. All classes/blocks in green, orange, and blue have `Bootstrap` as the suffix in the package.

CBB, where the block size is fixed, Stationary Bootstrap (Politis and Romano, 1994) randomly varies the length of each block according to a predetermined probability distribution. This method is adept at capturing both short- and long-term dependencies within the time series, offering a versatile approach that can adapt to various structures within the data.

- Non-Overlapping Block Bootstrap (NBB): NBB is a simpler variant where blocks of data are sampled in a non-overlapping manner. This method ensures that each data point is included only once in each bootstrap sample, reducing redundancy. NBB can be particularly effective for time series data with clear and distinct segments where the independence between blocks can be reasonably assumed.
- Tapered Block Bootstrap (TBB): TBB (Paparoditis and Politis, 2001) is an advanced variant that incorporates window functions to taper the blocks, reducing the potential discontinuities at the block edges. This method can be applied on top of the existing block bootstrap methods – such as MBB, CBB, SB, and NBB – as a base. The tapering process involves applying window functions like Hamming, Hanning, Blackman, or Tukey to the data blocks, which helps mitigate edge effects and improve the representativeness of the bootstrap sample. This technique is particularly useful in scenarios where the edge effects might introduce bias or when the time series exhibits strong seasonal components or trends. By employing different window functions, users can fine-tune the tapering to match the specific characteristics and requirements of their time series data, enhancing the flexibility and effectiveness of the bootstrapping process.

4.2 Advanced Bootstrap Methods

`tsbootstrap` includes a comprehensive suite of time series bootstrapping techniques, each designed for specific types of time series data:

- *Residual Bootstrap:* This method is particularly adept at handling model-based uncertainties (Kreiss and Lahiri, 2012; Imon and Ali, 2005). After fitting a time series model to the data, the residuals (the differences between the observed values and the model's predictions) are bootstrapped. This approach allows for the assessment of the variability and uncertainty of the model predictions, accommodating a range of models from simple linear regressions to complex ARIMA models. The Residual Bootstrap is invaluable for validating the stability and accuracy of the predictions made by these time series models.
- *Statistic-Preserving Bootstrap:* This technique ensures that the bootstrapped samples retain a user-defined key statistical property of the original dataset, be it mean, variance, or something else. It is particularly beneficial in applications where maintaining the original data's distributional characteristics is crucial for accurate analysis and

inference. This method is essential for studies where the underlying distributional properties play a critical role in the analysis outcomes or model validations.

- *Distribution Bootstrap:* When the underlying distribution of the time series is known or can be estimated, the Distribution Bootstrap method leverages this information to generate resampled data. This approach is ideal for datasets where specific distributional assumptions hold true, allowing for more precise and theoretically grounded inferences; in situations where the time series is believed to follow a known distributional pattern, aiding in more accurate modeling and forecasting.

- *Markov Bootstrap:* Tailored for time series exhibiting Markovian properties, this method is suitable for data where the future state is dependent only on the current state and not on the sequence of events that preceded it. Commonly used in financial time series analysis, the Markov Bootstrap (Härdle et al., 2003) can capture the stochastic processes underlying the financial markets, offering insights into the dynamics and potential future behaviors of these markets.

- *Sieve Bootstrap:* This method is specifically designed for time series that can be modeled using autoregressive structures. By active on the coefficients of an autoregressive model, the Sieve Bootstrap provides a robust framework for analyzing and forecasting time-dependent data. This approach is particularly powerful in scenarios where the data exhibits autoregressive behavior, enabling analysts to make informed forecasts and understand the underlying dynamics of the time series (Bühlmann, 1997).

Each of these methods demonstrates `tsbootstrap`'s capability to address the unique challenges presented by different types of time series data, making it a versatile tool in the field of time series analysis.

4.3 Extension Template

`tsbootstrap` is meant to be extensible, designed to make it easy for power users to add new bootstrap algorithms, to the main code base, or (potentially closed or differently licensed) third party extensions.

For this, `tsbootstrap` follows the combined strategy/template design pattern also seen in `sktime` or `sklearn`, using the foundations of `skbase` for extension and testing:

- *Extension template:* implementers of new algorithms are provided with an extension template, i.e., a fill-in `python` file with step-by-step instructions on filling in a new algorithm.

- *Extension contract:* boilerplate is abstracted away in the extension template - only a private `_bootstrap` needs to be implemented, with simplified inputs, while class inheritance and boilerplate layering ensure public interface compliance.

- *Contract checker:* extenders are provided with a compliance checker method, `check_estimator`, which allows testing at runtime, and can be used in 3rd party testing setups for continuous API compliance checks.

Further details, including a 90 minute general-purpose tutorial to extending `skbase`-templated packages, can be found in the `skbase` documentation (RNKuhns et al., 2024).

5 Usage and Integration with `sktime`

5.1 Basic Usage

In Figure 1, we detail the various building blocks and user-facing classes in `tsbootstrap`. Below, in Example Code 1 and Figure 1, we demonstrate a simple usage of `MovingBlockBootstrap` using simulated data.

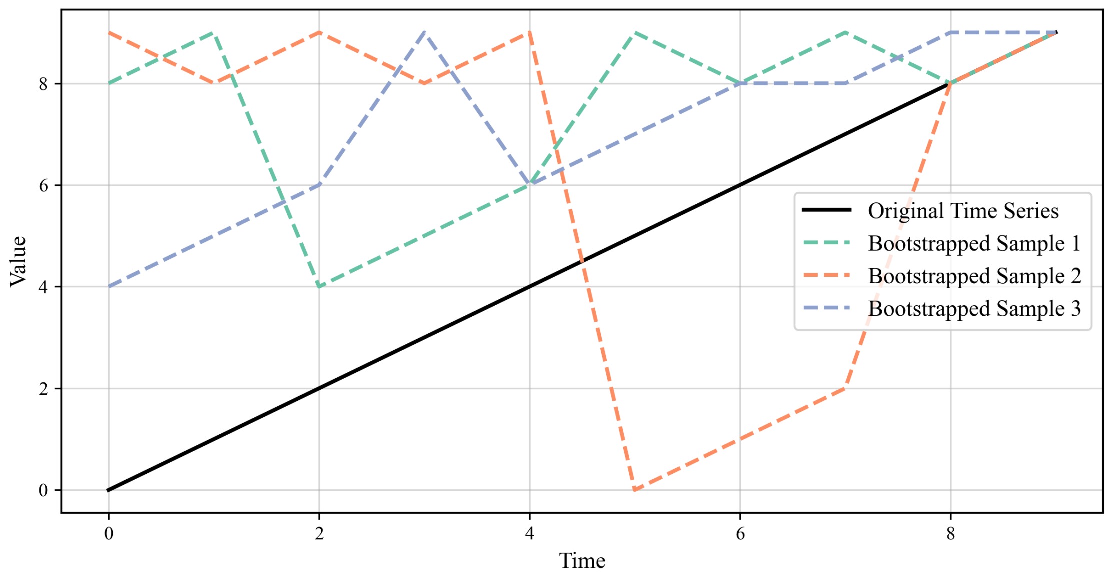

Figure 3: Output of Example Code 1.

5.2 Integration with `sktime`

`sktime` is a `python` library for time series providing unified interfaces for various tasks, such as classification, regression, or forecasting. For all of these tasks, bootrapping can improve the results. Thus, in the following, we first describe how `tsbootstrap` is integrated with

```python
from tsbootstrap import MovingBlockBootstrap
import numpy as np
import matplotlib.pyplot as plt


# Create custom time series data. While below is for univariate data,
# "tsbootstrap" can handle multivariate time series as well.
n_samples = 10
X = np.arange(n_samples)

# Instantiate the bootstrap object
n_bootstraps = 3
block_length = 3
rng = 42
mbb = MovingBlockBootstrap(
         n_bootstraps=n_bootstraps,
         rng=rng,
         block_length=block_length
      )


# Generate bootstrapped samples
return_indices = False
bootstrapped_samples = mbb.bootstrap(
    X, return_indices=return_indices)

# Collect bootstrap samples
X_bootstrapped = []
for data in bootstrapped_samples:
    X_bootstrapped.append(data)

X_bootstrapped = np.array(X_bootstrapped)

# Plot the bootstrapped samples and the original time series
plt.figure()
plt.plot(X, label="Original Time Series", color="black")
for i in range(n_bootstraps):
    _ = plt.plot(X_bootstrapped[i].reshape(-1,),
                 label=f"Bootstrapped Sample {i+1}",
                 ls="--")

plt.legend()
plt.xlabel("Time")
plt.ylabel("Value")
plt.show()
```

Example Code 1: Example code snippet demonstrating the use of tsbootstrap `sktime`. We then show the combined usage of `sktime` and `tsbootstrap` with a probabilistic forecasting example.

5.2.1 SKTIME'S TSBOOTSTRAPADAPTER

`sktime`'s `TSBootstrapAdapter` follows the unified API design as the other transformers implemented in `sktime`. In other words, these methods provide `fit`, `transform`, `get_params`, `set_params` methods, and `__init__` as the constructor. Below we briefly present these methods.

__init__ is the constructor. It takes as parameters, the `tsbootstrap` object that should be used within `sktime`.

fit fits the transformer. It is empty for this adapter.

transform creates the bootstraps. It takes as arguments, the time series that should be bootstrapped as X.

get_params returns the current parameters of the adapter.

set_params allows to set the different parameters.

This interface enables the usage of all `tsbootstrap` methods within `sktime` since it requires that an arbitrary `tsbootstrap` object is passed to the adapter and that all of these classes have the same interface. This interface consistency guarantees that the classes are exchangeable.

5.2.2 USAGE: PROBABILISTIC FORECASTING

Point forecasts comprises one value per time step. Thus, such forecasts do not quantify uncertainty. To quantify the uncertainty in forecasting, often probabilistic forecasts are generated. However, many forecasting algorithms do not support probabilistic forecasting directly. Thus, in `sktime`, there exist different methods to provide uncertainty quantification for point forecasts. One of these approaches that smoothly interacts with bootstrapping is the `BaggingForecaster`. The `BaggingForecaster` fits one model per bootstrap. During inference, each fitted model makes a forecast. Finally, out of this set of forecasts, the uncertainty can be derived.

Example Code 2 shows an exemplary usage `tsbootstrap` and `sktime` for forecasting. First, the required modules are imported. Afterwards, the time series is loaded and split into train and test set. Third, the forecaster is initialised. In this example we use a `BaggingForecaster`. This forecaster takes as arguments a bootstrapper and a base forecaster. For the bootstrapper, we use `BlockResidualBootstrap` from `tsbootstrap`, which is wrapped using `TSBootstrapAdapter`, and as base forecaster, we use a SARIMA model. Fourth, we fit the forecaster using the training data. Finally, in the last step, we forecast the time series and also forecast intervals.

```
# Example code snippet demonstrating the use of tsbootstrap together with
# sktime
from sktime.forecasting.arima import ARIMA
from sktime.transformations.bootstrap import TSBootstrapAdapter
from sktime.forecasting.compose import BaggingForecaster
from sktime.transformations.series.detrend import Detrender, Deseasonalizer
from tsbootstrap import MovingBlockBootstrap
from sktime.utils.plotting import plot_series

from sktime.datasets import load_airline
from sktime.forecasting.model_selection import temporal_train_test_split

import matplotlib.pyplot as plt


# Load time series data with sktime or custom data with pandas
data = load_airline()
train, test = temporal_train_test_split(data, test_size=12)


# Initialize the bootstrapper with custom settings
adapter = TSBootstrapAdapter(
            BlockResidualBootstrap(
               MovingBlockBootstrap(20)
            )
         )

# Create a BaggingForecaster using the bootstrap adapter and a SARIMA model
forecaster = Deseasonalizer(sp=12, model="multiplicative") * Detrender() *
             BaggingForecaster(adapter,
                               ARIMA(order=(1, 1, 0),
                               seasonal_order=(0, 1, 0, 12)),)

# Fit the forecaster
forecaster.fit(train)


# Perform predictions
pred = forecaster.predict(range(1, len(test) + 1))
pred_intervals = forecaster.predict_interval(range(1, len(test) + 1))

# Plot
plt.close("all")
fig, ax = plot_series(data, pred, pred_interval=pred_intervals, labels=["y"
                                        , "y_pred"])
ax.set_xlabel("Time")
plt.show()
```

Example Code 2: Exemplary usage of `tsbootstrap` together with `sktime` to perform a forecast on the airline dataset.

The resulting forecast together with the intervals are shown in Figure 4.

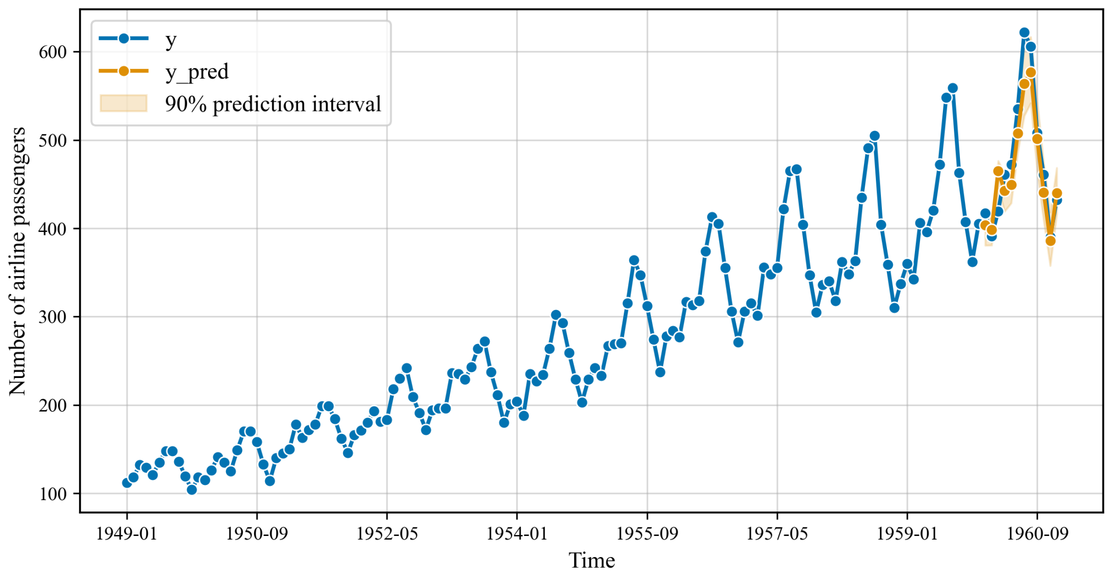

Figure 4: Exemplary usage of `tsbootstrap` and `sktime`; output from Example Code 2.

6 Conclusion

We have demonstrated that `tsbootstrap` advances the field of time series analysis by offering a robust and user-friendly `python` package designed for sophisticated bootstrapping techniques. Its comprehensive suite of methods not only addresses the specific needs of time series data but also enhances their analysis with a focus on usability and integration. This tool can play a crucial role in fields where accurate uncertainty quantification is essential, such as finance, meteorology, and epidemiology. Through `tsbootstrap`, researchers and practitioners can better predict and manage risks, ultimately leading to more informed decision-making.

We invite the community to participate in the ongoing development of `tsbootstrap`. We are particularly keen on collaborations that explore new domains of application or enhance our understanding of time series analysis challenges. By joining our efforts, you can help shape the future of bootstrapping techniques in time series analysis. The continuous development and enhancement of `tsbootstrap`, as highlighted in its GitHub repository, and its close integration with `sktime`, ensures its relevance and utility in the evolving landscape of time series analysis. We are excited to see how `tsbootstrap` will continue to evolve and how you, the community, will be a part of this journey.

7 Future Work

We have a clear yet evolving roadmap for future enhancements, including the integration of more sophisticated bootstrapping methods and improved performance optimizations. These are delineated in Issue #144 in the `tsbootstrap` repository.

1. **Performance and Scaling:** Improve handling of large datasets, by integrated with established distributed backends such as Dask, Spark, and Ray. Explore ways to optimize and profile the code for improvements in both time- and space-complexity.

2. **Tuning and Automation:** Implement adaptive algorithms that dynamically adjust block sizes based on autocorrelation, and integrate fractional block lengths for precise resampling (Bühlmann and Künsch, 1999). Additionally, implement automated feedback mechanisms to refine the resampling process by analyzing real-time dataset characteristics and iteratively adjusting to minimize discrepancies between original and bootstrapped data.

3. **Real-time and Streaming Data:** Process data in real-time by incorporating event-driven programming or reactive frameworks that handle data streams efficiently.

4. **Enhanced integration with `sktime`:** Develop standardized evaluation tools that utilize `sktime`'s metrics for detailed performance comparisons between original and bootstrapped time series data. This includes establishing a repository of shared datasets for benchmarking across both libraries, and creating extensive documentation and examples that demonstrate the combined use of `tsbootstrap` and `sktime`. Additionally, introduce functionalities to generate distribution or sampler-like objects for advanced probabilistic forecasting.

5. **API Extension:** Support `pandas DataFrame`s, and expand API capabilities to handle panel and hierarchical time series. At the same time, improve handling of exogenous data within bootstraps for complex models, and refine model state management to accommodate both fittable and pretrained models.

References

Peter Bühlmann. Sieve bootstrap for time series. *Bernoulli*, pages 123–148, 1997.

Peter Bühlmann and Hans R Künsch. Block length selection in the bootstrap for time series. *Computational Statistics & Data Analysis*, 31(3):295–310, 1999.

Michael R Chernick. *Bootstrap methods: A guide for practitioners and researchers*. John Wiley & Sons, 2011.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, pages 569–593. Springer, 1992.

Sankalp Gilda. tsbootstrap, March 2024. URL `https://doi.org/10.5281/zenodo.10866090`.

Olivier Grisel, Andreas Mueller, Lars, Alexandre Gramfort, Gilles Louppe, Thomas J. Fan, Peter Prettenhofer, Mathieu Blondel, Vlad Niculae, Joel Nothman, Guillaume Lemaitre, Arnaud Joly, Loïc Estève, Jérémie du Boisberranger, Jake Vanderplas, manoj kumar, Hanmin Qin, Nicolas Hug, Nelle Varoquaux, Robert Layton, Adrin Jalali, Rajagopalan (Venkat) Raghav, Johannes Schönberger, Julien Jerphanion, Roman Yurchak, Lucy Liu, Christian Lorentzen, Tom Dupré la Tour, Wei Li, and Chiara Marmo. scikit-learn/scikit-learn: Scikit-learn 1.4.2, April 2024. URL `https://doi.org/10.5281/zenodo.10951361`.

Wolfgang Härdle, Joel Horowitz, and Jens-Peter Kreiss. Bootstrap methods for time series. *International Statistical Review*, 71(2):435–459, 2003.

AHM Imon and M Masoom Ali. Bootstrapping regression residuals. *Journal of the Korean Data and Information Science Society*, 16(3):665–682, 2005.

Jens-Peter Kreiss and Soumendra Nath Lahiri. Bootstrap methods for time series. In *Handbook of statistics*, volume 30, pages 3–26. Elsevier, 2012.

Hans R Kunsch. The jackknife and the bootstrap for general stationary observations. *The annals of Statistics*, pages 1217–1241, 1989.

Soumendra Nath Lahiri. *Resampling methods for dependent data*. Springer Science & Business Media, 2013.

The pandas development team. pandas-dev/pandas: Pandas, April 2024. URL `https://doi.org/10.5281/zenodo.10957263`.

Efstathios Paparoditis and Dimitris N Politis. Tapered block bootstrap. *Biometrika*, 88 (4):1105–1119, 2001.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Dimitris N Politis and Joseph P Romano. The stationary bootstrap. *Journal of the American Statistical association*, 89(428):1303–1313, 1994.

RNKuhns, Franz Király, Tomas Pereira de Vasconcelos, Anirban Ray, LGTM Migrator, MBristle, and Sagar Mishra. sktime/skbase: v0.7.7, April 2024. URL `https://doi.org/10.5281/zenodo.10984314`.

sktime developers. sktime/sktime: v0.28.1, April 2024. URL `https://doi.org/10.5281/zenodo.7117735`.