



Sophie Castel <sup>1,‡</sup> and Wesley S. Burr <sup>2,\*,‡,§</sup>

- <sup>1</sup> Applied Modelling & Quantitative Methods MSc Program, Faculty of Science, Trent University, 1600 West Bank Drive, Peterborough, ON K7L 0G2, Canada; sophiecastel@trentu.ca
- <sup>2</sup> Department of Mathematics, Faculty of Science, Trent University, 1600 West Bank Drive, Peterborough, ON K7L 0G2, Canada
- \* Correspondence: wesleyburr@trentu.ca
- Presented at the 7th International conference on Time Series and Forecasting, Gran Canaria, Spain, 19–21 July 2021.
- ‡ These authors contributed equally to this work.
- § Current address: Trent University, Department of Mathematics, ENW 335, 1600 West Bank Drive, Peterborough, ON K7L 0G2, Canada.

Abstract: Real-world time series data often contain missing values due to human error, irregular sampling, or unforeseen equipment failure. The ability of a computational interpolation method to repair such data greatly depends on the characteristics of the time series itself, such as the number of periodic and polynomial trends and noise structure, as well as the particular configuration of the missing values themselves. The interpTools package presents a systematic framework for analyzing the statistical performance of a time series interpolator in light of such data features. Its utility and features are demonstrated through evaluation of a novel algorithm, the Hybrid Wiener Interpolator.

Keywords: time series; interpolation; applied statistics; interpTools



Citation: Castel, S.; Burr, W.S. Assessing Statistical Performance of Time Series Interpolators. *Eng. Proc.* 2021, *5*, 57. https://doi.org/10.3390/ engproc2021005057

Academic Editors: Ignacio Rojas, Fernando Rojas, Luis Javier Herrera and Hector Pomare

Published: 16 July 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/).

# 1. Introduction: The Need for Interpolators

Practically-gathered time series data often contain missingness: observations are not known due to a variety of real-world causes, including instrument failure, contamination, data storage losses, or even climate and weather (e.g., for Earth-bound stellar observatories). Many of the best estimation algorithms for time series characteristics assume contiguous samples with no missingness. This contrast is the impetus behind the creation of a number of interpolation (or imputation) methods for time series data [1–5], and the previous development of a test-bench for evaluation of such methods [6].

Recently completed work on a new R package interpTools [7,8] provides an additional means of simulating particularly-structured artificial time series, imposing missing observations according to a user-specified gap structure, and repairing the incomplete series via chosen interpolation algorithms, with generous support for evaluating interpolators' statistical performance, and for generating data visualizations. In this paper we discuss the framework developed, and present some results comparing the Hybrid Wiener Interpolator (HWI) [1] to a number of other standard algorithms.

A significant practical challenge when determining the effectiveness of a given interpolator on a particular time series is that the true value of a missing data point at a given index of a stochastic time-ordered process is generally unknown. Performance metrics, such as those described by [2], typically assume the form:

$$C(\hat{x}_i, x_i) = C(\hat{x}_i - x_i), \quad i = 1, ..., I,$$
(1)

where *C* is some function of the deviation between the interpolated data point,  $\hat{x}_i$ , and the true data point,  $x_i$ . Without knowing  $x_i$  or at least its probabilistic structure, exactly, it is impossible to determine such a measure of accuracy.



The statistical performance of interpolators depends greatly on the structural nuances of the dataset chosen. Some algorithms are better-suited for time series with high numbers of embedded periodicities, whereas others are more suitable for low-frequency data. Interpolators' performance may also depend on any particular pattern of missing values present in the data, e.g., cubic splines fail exponentially as the gap width increases [8]. Other methods, such as the HWI [1], are more resilient to longer sequences of consecutive missing observations. As such, there are trade-offs to any chosen interpolator, and careful consideration of the research objectives and parameters of the study should be the first step in the selection of an interpolator in any practical setting.

### 2. Framework for Interpolation Using interpTools

The interpTools package allows a user to simulate a 'mock' time series containing similar features to a real-world dataset of interest, such that the original data points are known, and performance metrics of the form  $C(\hat{x}_i, x_i)$  can be calculated, following the application of a specified pattern of gaps and a particular interpolation algorithm (Figure 1). Using simulated data enables the user to benchmark performance and make an informed choice regarding which interpolator would be most suitable for use on similar time series outside of the laboratory setting.





**Figure 1.** Example interpolation using the Exponential Weighted Moving Average method on the first dataset (k = 1) of a series of simulated data with 20% missing values at a minimum gap length of 2, with absolute deviations  $|\hat{x}_i - x_i|$  highlighted in red.

The default package model simulates time series  $x_t$  based on the classic additive model for time series:

$$x_t = m_t + t_t + \xi_t, \ t = 0, \cdots, n-1,$$
 (2)

where  $m_t$  is the mean function,  $t_t$  is the trend function, and  $\xi_t$  is the noise function. The following section provides a brief description of each component, along with its set of defining parameters. The package also supports arbitrary user-generated series for extension beyond this particular model: in particular, it is simple to generate multiplicative models if that is more relevant to the interests of the user (the reason the additive model was used in the development of this algorithm and package is that many astrophysics time series datasets are more accurately modeled using additive structure, with limited or no seasonality to speak of, and the second author has an interest in data of this type).

The mean component,  $m_t$  is comprised of a constant, non-varying mean element (a 'grand mean'),  $\mu$ , and a varying polynomial trend element,  $\mu_t$ :

$$m_t = \mu + \mu_t, \ t = 0, ..., n - 1$$
 (3)

$$=\mu+\sum_{i=1}^{\phi}a_i\left(\frac{t-c}{n}\right)^i,\tag{4}$$

where  $\mu \sim u(\frac{-n}{100}, \frac{n}{100})$  and *c* is a randomly sampled integer in the range [1, n]. The polynomial coefficients  $a_i \sim n(0, \frac{n}{20i})$  and are sampled in this way to facilitate the desirable property that the coefficients 'scale down' (i.e.,  $a_i \rightarrow 0$ ) as  $i \rightarrow \phi$ . The parameter  $\phi$  is chosen by the user and represents the degree of the polynomial. This could be re-expressed without the  $\mu$  by allowing the summation to run from i = 0, but structurally the code implementation assumes a static non-time-varying mean and time-varying polynomial mean.

The trend component,  $t_t$  is considered to be a finite linear combination of sinusoids. The interptools package simulates the trend component according to the construction:

$$t_t = \sum_{i=1}^{\psi} b_i \sin(\omega_i t), \tag{5}$$

where  $b_i \sim n(1, \frac{n}{200})$  (to allow for variation in relative Signal-to-Noise for individual periodic components, with the normal distribution ensuring extremely high-coefficient sinusoids will be rare) and  $\omega = [\omega_1, \omega_2, ..., \omega_{\psi}]$  with  $\omega_i$  defining the period of the *i*th sinusoid. The default is to sample  $\psi = 20$  unique values for each  $\omega_i \in [\frac{2\pi}{N}, \pi]$  (using the fundamental Fourier frequency, and bounded by Nyquist). This is user-controllable, and is intended to allow the user a degree of influence over the relative signal-to-noise in the simulations, where "signal" is considered as overall periodic components, and "noise" is the background. Many scientific datasets have dozens to thousands of such periodic signals present (e.g., astrophysics, helioseismology, seismology, oceanography), and they are the object of interest in such fields and analyses, so control of this parameter is of importance for generalizability of the algorithm and package.

The noise component,  $\xi_t$  is assumed to be an ARMA( $P^*$ ,  $Q^*$ ) stochastic process:

$$\xi_t = \alpha_1 X_{t-1} + \dots + \alpha_{P^*} X_{t-P^*} + Z_t + \beta_1 Z_{t-1} + \dots + \beta_{Q^*} Z_{t-Q^*}, \tag{6}$$

with variance  $\sigma_{\xi_t}^2$ , where  $P^*$  is the autoregressive (AR) order,  $Q^*$  is the moving-average (MA) order, and  $Z_t$  is a white noise process.

Each of these components can be generated independently, or simultaneously. Metadata regarding information about the features of each component, such as the polynomial equation of  $m_t$ , or the exact frequencies contained in  $t_t$ , are saved to memory in list objects of class simList. We reiterate here that this is simply the default structure for simulation, and the user is able to specify their own model of interest, and generate their own synthetic time series for testing with ease.

### 2.1. Imposing a Gap Structure

Once the artificial data (*K* time series) has been generated, the user can remove observations according to a gap structure defined by parameters *p*, the proportion of data missing, and *g*, the gap width, where each observation (except the endpoints) has the same Bernoulli probability of missingness,  $p_{omit} = 1/(N-2)$ . These inputs are vectorized such that the user can test any number of different (*p*, *g*) combinations, where *P* and *G* represent the total number of options for the removal percentage and gap width, respectively. The number of unique gap structures to randomly generate under a particular (*p*, *g*) parameterization is specified with *K*, which gives the user control over the number of iterations: higher *K* means more replicates, which tends to give more stable estimates, as

with most statistics (for the analyses shown here, K = 100, which empirically gave stable results on repeated runs and random seeds). An example is shown in Figure 2.



**Figure 2.** An example of the relationship between the number of holes and gap width for N = 40: (a) a time-plot of the original series; (b) a time-plot of a gappy series with p = 0.10 and g = 1; (c) a time-plot of a gappy series with p = 0.10 and g = 4.

The result is *K* 'gappy' series, each with a total of  $I = p \cdot N$  missing observations, appearing structurally as  $\leq \frac{p \cdot N}{g}$  randomly-spaced non-overlapping holes of width *g*, where a 'hole' is defined as a sequence of adjacent missing observations. Note that since holes may be placed adjacent to one another, this quantity describes the number of holes visible, at most. Classic Missing Completely At Random (MCAR) can be simulated by setting the gap length to 1.

## 2.2. Performing Interpolation

Once a set of gap-imposed data has been generated, the user may test interpolation algorithms on those data using parInterpolate(), which executes in parallel for efficiency. The package provides a number of built-in interpolation algorithms, though a user may also choose to provide any developed algorithm for flexibility and extension (e.g., [3,5]). The output is a highly nested list with every combination of (m, p, g) having a list of K interpolated time series, each of length N, where m represents the particular interpolation method used, and each  $\hat{\mathbf{x}}_k$  for k = 1, ...K approximates the original time series.

#### 2.3. Evaluating Statistical Performance

A definition of statistical performance at any time index will be given by some function of the deviation, *C* (Equation (1)). Generally speaking, *C* quantifies how well the interpolated series,  $\hat{\mathbf{x}} = {\{\hat{x}_t\}}_{t=0}^{N-1}$ , captures the essence of the original series. The package contains 18 such performance metrics, and it is also possible for the user to define their own custom performance statistics. For every  $(\mathbf{x}, \hat{\mathbf{x}})$  pair, statistical performance can be calculated via the function performance(), with resulting output a list of class pf and dimension  $M \times P \times G \times K$ , where *M* represents the total number of algorithms tested, and *P* and *G* are the total number of different proportion missing and gap width parameterizations applied, respectively. The terminal node of any (m, p, g) branch in this nested list is a vector of all the performance criteria for a particular combination of experimental conditions. Consider that for any (m, p, g) branch, there are a set of *K* values for each criterion. Thus, each performance metric has a sampling distribution containing *K* elements. The performance matrices can be condensed by aggregating sample statistics over *K* to reduce dimensionality, via the function aggregate\_pf().

The Hybrid Wiener Interpolator (HWI) [1] is a novel iterative interpolation algorithm based on estimation of sub-components of a time series using robust frequency-domain spectral methods. The essence of the algorithm is the estimation of periodic components and time-varying mean elements using multitaper methods [9], enveloping an embedded Wiener covariance interpolation step [10] for the approximately stationary noise background. This pair-wise estimation proceeds iteratively until converged, in approximately an Estimation-Maximization model.

As a visual example, imagine a path through such an above list, where p = 15%, g = 10, and m = HWI, such that we end with a collection { $\hat{\mathbf{x}}_k : k \in 1, ..., K$ } (Figure 3a). Imagine applying some performance metric formula, say, C = MSE, to each ( $\hat{\mathbf{x}}_k, \mathbf{x}_k$ ) pair, such that we have a collection of K values. The sampling distribution of { $C_k$ } is shown in Figure 3b. Then, a sample statistic f (e.g., the sample median) can be calculated for this distribution, such that f = median({ $C_k, k \in 1, ..., K$ }) (Figure 3b). This value represents the aggregated (median) performance of the HWI for a gap structure of (p = 15%, g = 10), denoted by the value f(15%, 10) (indicated in Figure 3c).

$$p = 15\%, g = 10, m = HWI$$



**Figure 3.** (a) Structure diagram of *K* interpolations performed using the Hybrid Wiener Interpolator for (p, g) = (15%, 10). (b) Example sampling distribution of the median of a selected performance metric  $C_k$ . (c) An example surface plot *f* over  $\{(p, g) : p \in 1, ..., P, g \in 1, ..., G\}$ , indicating the aggregated performance at (p, g) = (15%, 10).

### 3. Data Visualization

## 3.1. 3D Surfaces

Considering the full set of possible gap combinations  $\{(p_i, g_j)\}$ , for  $i \in 1, ..., P$ ,  $j \in 1, ..., G$  as a discrete mesh in  $\mathcal{R}^2$ , mapping these aggregated statistics traces out a surface f(p, g) in  $\mathcal{R}^3$ , where the height of the surface at a point (p, g) represents an interpolator's aggregated performance when subjected to a specific proportion of data missing and gap width. Visualizing performance as a surface helps the user to understand the behavior of an interpolator in light of changes to gap structure. Extreme points on the surface represent gap structures at which performance is exemplified: either optimal, or worst-case. For cross-comparison across interpolation methods, multiple performance surfaces can be graphically layered on top of one another, where the 'best' interpolator for a particular gap structure will be at an extremum of the surfaces at the corresponding (p, g) coordinate point.

This visualization is generated by the plotSurface() function, where the user can specify any number of algorithms, the sample statistic to be represented by f(p,g), and a performance metric of choice. It is also possible to select an algorithm to highlight via the argument highlight, as well as the colour palette. As the implementation is dynamic, the user can also interact with the surface plot widget by manipulating the camera perspective, adjusting the zoom, and hovering over data points for more precise numerical information. A static export of such a surface is shown in Figure 3c.

## 3.2. Heatmaps

For more static (and printable) representations of the 3D surface plot, heatmaps are a nice equivalent. Using heatmapGrid(), a three-dimensional surface can be collapsed into a heatmap through conversion of the third dimension to colour, to which the value of the metric is proportional. The function multiHeatmap() function enables the user to arrange multiple heatmaps into a grid to facilitate cross-comparison between multiple criteria or methods. Demonstration examples are provided in Figure 4.

## 3.3. Collapsed Cross-Section Plots

Changing the perspective angle on a given surface plot can offer further insights on the relationship between interpolator performance and the gap pattern parameters. Imagine rotating such a surface such that it is viewed perpendicular to either the p - f or g - f plane: this allows a user to examine performance with respect to changes in one variable across all values of the other. The sampling distribution can be visualized as a ribbon, where the upper and lower bounds are the largest and smallest values observed across the set of sample statistics contained in the collapsed variable (the highest and lowest points on the corresponding surface plot), and the central line is the median value of this set. The user can generate these collapsed cross-section plots using the plotCS() function.

When cross\_section = 'p', the *g*-axis is collapsed, and the resulting plot is a crosssection of 'proportion missing' (Figure 5, left). Here, we can see how the performance of an algorithm on a particular dataset changes as the total number of missing observations increases  $(p \rightarrow P)$ . When cross\_section = 'g', the *p*-axis is collapsed, and the resulting plot is a cross-section of gap width (Figure 5, right): we can observe how the overall performance of an algorithm on a particular dataset changes as the width of the gaps increases  $(g \rightarrow G)$ . The widths of the ribbons may indicate the sensitivity of an algorithm to a particular gap parameter, where 'thicker' ribbons indicate a greater disparity between the best and worst interpolations, and 'thinner' ribbons correspond to algorithms that seem to perform similarly regardless of the value of the defining axis.



**Figure 4.** Example of heatmaps generated comparing two interpolation algorithms (Replace with Constant and Nearest Neighbour) on a common set of data. The bottom axis is the proportion of missing points, and the side axis is the designated width of each individual gap, with the colours representing the median value of the Mean Squared Error (MSE) metric.



**Figure 5.** Representation of collapsed cross-section plots derived from a parent performance surface (**center**). The shown bands extract information not shown in the 3D plot, indicating designated percentile intervals across the simulation replicates.

#### 3.4. Uncollapsed Cross-Section Plots

The collapsed cross section plots can be further deconstructed using  $plotCS_un()$  such that the performance can be assessed with respect to changes in one variable (*p* or *g*) across each individual value of the other.

Interpretable as 'slices' of the surface plot generated by plotSurface(), these 'uncollapsed' cross-section plots will give insight as to whether there are specific combinations of (p, g) for which the performance of a particular method is particularly sensitive. Each ribbon represents the distribution of a metric across the *K* simulations, where the upper and lower bounds represent the (e.g., 2.5% and 97.5%) quantiles, respectively, and the interior line is formed by the set of sample statistics from corresponding points on the parent surface plot. An advantage of viewing the data in this way is that it allows us to view the error bars at each (p, g) coordinate point without creating an over-crowded and hard-to-read surface plot visualization. An example is shown in Figure 6.



**Figure 6.** Example of cross-section plots, for the MSE metric, with *p* and *g* as shown, showing the algorithms Nearest Neighbours and Replace-With.

## 4. Analysis of the HWI for d = 5 Test Cases

The motivation behind the development of interpTools was to test the HWI [1], originally developed to correct missingness in structured astrophysical data. The objective of the research was to audit its statistical performance on various classes of time series following the application of different gap structure parameterizations, and to complete a full comparative analysis against more classically-used interpolation algorithms. The following will present some results from that study demonstrating the robustness of the HWI as well as showcasing the utility of the interpTools package.

The analysis was conducted on an (arbitrary) five artificial time series (d = 1, 2, ..., 5) of length N = 1000, where the mean component  $m_t$  was a cubic polynomial function ( $\phi = 3$ ) and was fixed  $\forall d$ . The periodic trend component  $t_t$  was set to vary, where the number of embedded sinusoids,  $\psi$ , increased by a factor of ten with each new dataset ( $\psi_d = d \times 10$ ) so as to scale the effective "signal" presence against the "noise" presence. Recall from above that many real-world astrophysical time series have thousands of periodic components present. All five time series were generated against a background of ARMA(0,0) white noise.

The inclusion of a time-varying mean component allowed for the simulation of a particular form of nonstationarity; a property that, in most other conventional interpolation algorithms, would first need to be corrected through estimation and removal of underlying

monotonic trends. Often in practice, this correction is either overlooked, or done using crude techniques (such as differencing) that do not preserve the integrity of the data, and are prone to increasing statistical error [8]. One of the major advantages of the HWI is that it can be applied to certain classes of nonstationary data, such that no prior manipulations are necessary [1]. Gap structures were applied with combinations of missingness proportions (*p*) up to 30% and gap widths (*g*) up to 25 observations wide, of which each (*p*, *g*) parametrization contained K = 1000 replicates. In addition to the HWI, other interpolation algorithms were explored: the Kalman filter (KAF), Exponential Weighted Moving Average (EWMA), and cubic splines, with further discussion of other approaches detailed in the appendices of [8].

In assessing the performance metrics, it was clear that in this case the HWI led to significantly more accurate estimation of the missing values, was the most consistent in its estimation, and the most stable when subjected to increasing missingness and data complexity [8]. The HWI maintained its rank, even when compared to the more robust KAF and EWMA algorithms. The cubic splines performed comparably at modest gap structures, but quite poorly when p and g were large, showing particular sensitivity to gap width. Figure 7 provides a summary of the statistical performance of each algorithm (excluding the cubic splines) on the fifth dataset, according to the Normalized Root Mean Squared Deviation (NRMSD) metric (optimal when minimized), which was median-aggregated across the 1000 replicates. The corresponding surface values for the HWI are shown in Table 1.



**Figure 7.** Median Normalized Root Mean Squared Deviation (NRMSD) values across the *K* simulated interpolations for each (p, g) gap structure imposed on the fifth dataset, with colour proportional to value, and scaled across the set. Each surface and its corresponding heatmap define the performance of a particular interpolation method (HWI, KAF, EWMA). The HWI outperformed the other methods over all parameters.

Note that this should not be considered to be a complete or robust examination of the performance of the HWI against other algorithms. Further analysis was done in [8], but as with many computational algorithms, demonstration of improvements can only be done in limited test cases due to resources. The HWI provides a number of theoretical advantages over other classic algorithms, especially in highly structured time series with large numbers of readily-detected periodic components, which was why it was developed, and these results seem to reinforce that the design was effective for series of this type. The algorithm is being used "in the wild" by several national science agencies for imputation of scientific data sets, with good results.

		Gap Width (g)			
		1	5	10	25
Proportion missing $(p)$	5%	11.94	10.62	10.34	11.2
	10%	11.02	9.57	9.51	9.91
	15%	10.42	9.17	9.17	9.42
	20%	10.27	9.09	9.08	9.36
	25%	10.29	9.00	9.01	9.39
	30%	10.31	9.14	9.09	9.52

**Table 1.** Median NRMSD statistics of the HWI interpolations on the fifth dataset, aggregated across the K = 1000 simulations in each (p, g) gap specification. Compare with Figure 6.

# 5. Conclusions

The R package interpTools provides a robust set of computational tools for scientists and researchers alike to evaluate interpolator performance on artificially-generated time series data in the presence of various gap structure patterns. Investigating these relationships in the safety of a lab setting with synthetic data allows researchers to benchmark performance and make informed decisions about which interpolation algorithm will be most suitable for a real-world dataset with comparable features. The package also provides a number of data visualization tools that allow a user to distill the resulting copious amounts of performance data into sophisticated, customizable, and interactive graphics.

Through use of this package, we have demonstrated that the Hybrid Wiener Interpolator demonstrates robust performance in the presence of large numbers and lengths of gaps for a selected set of periodic signals against background noise (a relatively broad class of time series often encountered in physical science applications). It is our hope is that by using the framework presented in this paper, interested users will be able to better understand the relationships between interpolators and time series, and minimize the harmful implications of making erroneous inferences from poorly-repaired gappy time series data. The framework also allows for comparison of novel algorithms to accepted standard approaches, novel metrics, and novel time series structural inputs, allowing for a very general support in the development of targeted methods.

**Author Contributions:** Conceptualization, W.S.B.; methodology, W.S.B. and S.C.; software, S.C.; validation, S.C. and W.S.B.; resources, W.S.B.; writing—original draft preparation, S.C.; writing—review and editing, W.S.B.; visualization, S.C.; supervision, W.S.B.; funding acquisition, W.S.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** Wesley S. Burr is funded by the National Sciences and Engineering Research Council (NSERC) of Canada, under the Discovery Grant Program. Sophie Castel received an Ontario Graduate Scholarship for the pursuit of this work.

**Data Availability Statement:** The R package used to develop the simulation results in this paper is available on GitHub [7].

Conflicts of Interest: The authors declare no conflict of interest.

# Abbreviations

The following abbreviations are used in this manuscript:HWIHybrid-Wiener InterpolatorNRMSDNormalized Root Mean Squared DeviationKAFKalman Filter algorithmEWMAExponential Weighted Moving Average algorithm

### References

- 1. Burr, W.S. Air Pollution and Health: Time Series Tools and Analysis. Ph.D. Thesis, Queen's University, Kingston, ON, Canada, 2012.
- 2. Lepot, M.; Aubin, J.B.; Clemens, F. Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. *Water* 2017, *9*, 796. [CrossRef]
- 3. Andiojaya, A.; Demirhan, H. A bagging algorithm for the imputation of missing values in time series. *Expert Syst. Appl.* **2019**, *129*, 10–26. [CrossRef]
- 4. Hippert-Ferrer, A.; Yan, Y.; Bolon, P. EM-EOF: Gap-filling in incomplete SAR displacement time series. *IEEE Trans. Geosci. Remote. Sens.* **2020**, 1–18. [CrossRef]
- 5. Savarimuthu, N.; Karesiddaiah, S. An unsupervised neural network approach for imputation of missing values in univariate time series data. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e6156. [CrossRef]
- 6. Beck, M.W.; Neeraj, B.; Asencio-Cortés, G.; Kulat, K. R Package imputeTestbench to Compare Imputation Methods for Univariate Time Series. *R J.* 2018, 10, 218–233. [CrossRef] [PubMed]
- 7. Castel, S.; Burr, W.S. interpTools: Tools for Systematic Testing and Evaluation of Interpolation Algorithms. R Package Version 0.1.0. Available online: https://github.com/castels/interpTools (accessed on 10 July 2021).
- 8. Castel, S. A Framework for Testing Time Series Interpolators. Master's Thesis, Trent University, Peterborough, ON, Canada, 2020.
- 9. Thomson, D.J. Spectrum estimation and harmonic analysis. *Proc. IEEE* **1982**, *70*, 1055–1096. [CrossRef]
- 10. Wiener, N. Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications; MIT Press: Cambridge, UK, 1950.