International Journal of Forecasting 36 (2020) 121–128

Contents lists available at ScienceDirect

International Journal of Forecasting

journal homepage: www.elsevier.com/locate/ijforecast

Correlated daily time series and forecasting in the M4 competition

Anti Ingel *, Novin Shahroudi, Markus Kängsepp, Andre Tättar, Viacheslav Komisarenko, Meelis Kull

Institute of Computer Science, University of Tartu, J. Liivi 2, Tartu 50409, Estonia

ARTICLE INFO

Keywords:
Forecasting competitions
Time series
Correlation
Data leakage
Ensembling

ABSTRACT

We participated in the M4 competition for time series forecasting and here describe our methods for forecasting daily time series. We used an ensemble of five statistical forecasting methods and a method that we refer to as the correlator. Our retrospective analysis using the ground truth values published by the M4 organisers after the competition demonstrates that the correlator was responsible for most of our gains over the naïve constant forecasting method. We identify data leakage as one reason for its success, due partly to test data selected from different time intervals, and partly to quality issues with the original time series. We suggest that future forecasting competitions should provide actual dates for the time series so that some of these leakages could be avoided by participants.

1. Introduction

The task in the M4 time series forecasting competition was to provide forecasts for 100,000 numeric time series with different frequencies and origins (Makridakis, Spiliotis, & Assimakopoulos, 2018). The participants of the competition were free to choose their methods differently depending on the type of time series, and in what follows we will describe how we compiled our forecasts for the 4227 time series from the daily category. Each time series consisted of a numeric value for each day over a time interval ranging from 93 to 9919 days, and the task was to forecast the values for the next 14 days.

We participated in the M4 competition as a team of 24 students and one instructor, and our submission was created in a dedicated M4 seminar at the University of Tartu. We formed subteams to try out different approaches and evaluated these in an internal competition where we withheld the last available 14 days as an internal validation set. Our final submission was obtained by choosing a different ensemble of models for each category of time series: hourly, daily, weekly, monthly, quarterly and yearly. The forecasting of the daily time series was done using an ensemble of five statistical methods and one correlation-based method, which we refer to as the *correlator*. Once the organisers of the M4 competition published the full time series, we performed an analysis which revealed that the key to the good performance of our submission was the correlator.

Section 2 describes the methods that we used for forecasting the daily time series in the M4 competition. Section 3 evaluates the methods separately on the official M4 test data and demonstrates the role of the correlator in the final performance of our submission. Section 4 discusses the reasons behind the success of the correlator and its implications for forecasting competitions.

2. Methods

Our forecasts for the daily time series in the M4 competition were obtained using two methods. For a subset of

* Corresponding author.
E-mail addresses: anti.ingel@ut.ee (A. Ingel), novin@ut.ee (N. Shahroudi), markus93@ut.ee (M. Kängsepp), andre.tattar@ut.ee (A. Tättar), viacheslav.komisarenko@ut.ee (V. Komisarenko), meelis.kull@ut.ee (M. Kull).

https://doi.org/10.1016/j.ijforecast.2019.02.018
0169-2070/

the 4227 time series, we used a forecasting method which we refer to as the *correlator*. This method compares the time series in a given dataset among themselves to find highly correlated segments and uses this information to produce the forecasts. The method is used only on those time series for which the correlation is larger than a given threshold. The method is described in detail in Section 2.2.

On the rest of the time series — that is, for the time series for which the correlator did not find high correlations — an ensemble of five models was used to produce the forecasts. This ensemble is described in detail in Section 2.1.

The methods were selected based on their performances on the holdout validation dataset that we had separated from the training data. From each training time series, we removed the last 14 time points, and these removed values formed the validation data. After this, we were able to evaluate any method by calculating its performance measure on the validation set.

As a post-processing step, all of the negative predicted values were replaced with zeros. This was done because the original dataset did not contain any negative values, and thus it was deemed beneficial to avoid forecasting negative values.

2.1. Ensembling model

Our ensembling method used five models:

- a naïve model, which makes a constant forecast that is equal to the last observed value in the time series;
- an ETS model (see Appendix B for details);
- two ARIMA models (see Appendix B for details); and
- a custom model that is based on time series decomposition (see Appendix C for details).

These five methods were chosen based on empirical results on the holdout validation data. In addition, we also considered Holt-Winters (Hyndman, Koehler, Snyder, & Grose, 2002), BATS (Livera, Hyndman, & Snyder, 2011), LSTM networks (Kong et al., 2019) and a combination of ARIMA and a neural network (Zhang, 2003). We trained each of the chosen methods on each time series, to obtain five models. The ensemble forecast was obtained by taking the median of the five single-method-based forecasts for each time point in the forecasting horizon. A flowchart of the method can be seen in Fig. 1.

2.2. Correlator

The idea behind the correlator is to look for highly correlating patterns between different time series at different times. If such a correlation is found between the last 14 days of the current time series and some other 14-day period that continues for at least 14 days in the training set, then the correlator uses these subsequent 14 days to produce forecasts for the current series. The practical applications of this idea are limited, because it relies on the existence of correlating patterns between different times in different time series. However, this idea worked surprisingly well in the M4 competition, as we show and discuss in the following sections.

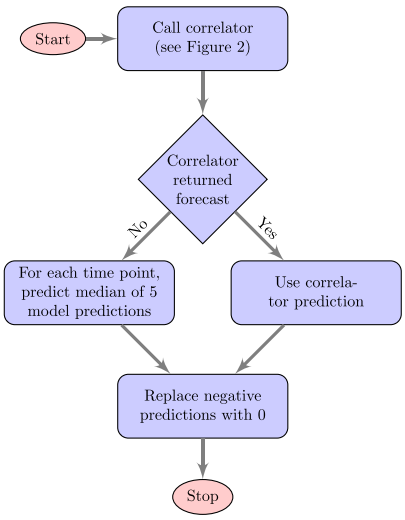

Fig. 1. Flowchart of the prediction process.

There are 4227 time series in the daily category, with varying lengths and starting times. Let us denote the jth time series of the daily dataset by y^j and its data points by $y^j_1, y^j_2, \ldots, y^j_{n_j}$, where n_j denotes the length of the jth time series. Furthermore, for convenience, we will use the notation $[i_1 : i_2]$ for interval indexing; that is, we denote the data points of time series y^j from index i_1 until index i_2 as $y^j_{[i_1:i_2]} := (y^j_{i_1}, \ldots, y^j_{i_2})$.

The correlator produces forecasts for the jth time series $y^j_1, \ldots, y^j_{n_j}$ by first taking its last 14 values $y^j_{[n_j-13:n_j]}$ and calculating the Pearson correlation coefficient between this vector of 14 values and every non-terminal segment of the same length across all time series, where by non-terminal we mean segments which have at least 14 subsequent data points. We denote the Pearson correlation coefficient between the end segment of the jth time series and a segment of the kth time series $y^k_1, \ldots, y^k_{n_k}$ ending at $\tau \in \{14, \ldots, n_k - 14\}$ by

$$r_{jk}(\tau) := \text{pearson}\left(y^j_{[n_j-13:n_j]}, y^k_{[\tau-13:\tau]}\right), \tag{1}$$

where, for any length n and any vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ of real numbers, pearson(a, b) is the Pearson correlation coefficient

$$\text{pearson}(a, b) = \frac{\sum_{i=1}^{n}(a_i - \text{mean}(a))(b_i - \text{mean}(b))}{\text{std}(a) \cdot \text{std}(b)}, \tag{2}$$

where

$$\begin{aligned} \text{mean}(a) &= \frac{1}{n}\sum_{i=1}^{n} a_i, \\ \text{std}(a) &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \text{mean}(a))^2}. \end{aligned} \tag{3}$$

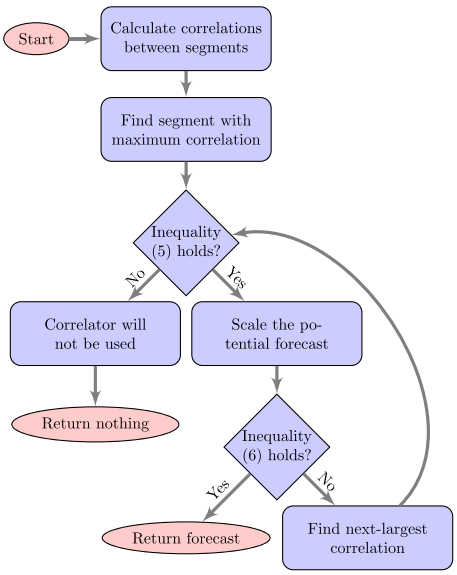

Fig. 2. Flowchart of the correlator.

Next, for each time series $y^j_1, \ldots, y^j_{n_j}$ we find the values k and τ which maximise $r_{jk}(\tau)$. In other words, the end segment of a time series y^j is compared to all given time series and a time series y^k is found with a segment that is most similar to the end segment of time series y^j and continues for at least 14 data points thereafter. The next 14 data points $y^k_{[\tau+1:\tau+14]}$ from the time series y^k are then scaled and translated using a function f, defined as

$$f(x) = \left(x - \text{mean}\left(y^k_{[\tau-13:\tau]}\right)\right) \frac{\text{std}\left(y^j_{[n_j-13:n_j]}\right)}{\text{std}\left(y^k_{[\tau-13:\tau]}\right)} + \text{mean}\left(y^j_{[n_j-13:n_j]}\right), \tag{4}$$

where x is a real number. The function f maps the values $y^k_{[\tau-13:\tau]}$ approximately to the values $y^j_{[n_j-13:n_j]}$. The obtained result is considered as the forecast for the time series y^j. We denote this result as $\hat{y}^j_{n_j+1} := f(y^k_{\tau+1}), \ldots, \hat{y}^j_{n_j+14} := f(y^k_{\tau+14})$.

The resulting forecast from the correlator is used as the actual forecast only if two conditions are met. The first condition that must hold is that the correlation must be at least 0.9999; that is,

$$r_{jk}(\tau) \geq 0.9999. \tag{5}$$

This condition was used to avoid spurious correlations. As the segment over which the correlation is calculated is only 14 data points long, a high threshold had to be used.

The second condition is

$$\text{std}\left(\hat{y}^j_{[n_j+1:n_j+14]}\right) \leq 2.5 \cdot \text{std}\left(y^j_{[n_j-13:n_j]}\right), \tag{6}$$

which means that the standard deviation of the forecast $\hat{y}^j_{[n_j+1:n_j+14]}$ must not exceed 2.5 times the standard deviation of the end segment of the time series y^j. Without this condition, the forecast might have a sudden change in standard deviation relative to the end segment of the time series; e.g., the forecast could contain large jumps. Thus, the correlator skips forecasts that do not satisfy Eq. (6) in order to find forecasts that are more similar to the end segment of current time series.

If Eq. (6) does not hold, new values k' and τ' are found such that they maximise $r_{jk'}(\tau')$ and differ from the previously found k and τ. This is iterated until suitable values are found or it is shown that no suitable values exist, in which case the correlator is not used. The flowchart of the correlator can be seen in Fig. 2.

The only changeable parameters for this method are the number of values used from the end of the time series, in this work 14, the correlation cut-off threshold in Eq. (5), and the multiplier for the standard deviation comparison in Eq. (6). All of these were chosen in this work based on empirical results on the validation dataset. However, our retrospective analysis showed that the threshold of 0.9999 was not optimal and Eq. (6) was not needed (see Table 2).

2.3. Differences from submission

After our forecasts had been submitted to the M4 competition, we found two bugs in our implementation of the correlator. Thus, there are two differences between the method used for producing the submission and the description given in Section 2.2.

First, in the original submission, the correlator was used only on time series D1–D2138, not on any of the time series D2139–D4227. This was probably caused by an unseen memory error when applying the correlator. We will refer to this error as Bug 1.

The second difference is that, due to human error, the condition

$$\text{std}\left(\hat{y}^j_{[n_j+1:n_j+14]}\right) \leq 2.5 \cdot \text{std}\left(y^k_{[\tau-13:\tau]}\right) \tag{7}$$

was used instead of Eq. (6). We will refer to this error as Bug 2.

3. Results

This section discusses the performances of all models presented in Section 2.1 individually and compares them to the performances of the ensemble with and without the correlator, to determine which part of our method was most successful. We also compare the performances of the correlator with different parameter values. Tables 1–3 show our post-submission analysis of the data, using the actual test set that was made public by the organisers after the competition.

Our experiments use the correlator's parameter values mentioned in Section 2.2 unless stated otherwise. The results were evaluated using MASE, sMAPE and the overall weighted average (OWA) of the relative MASE and the relative sMAPE (M4 Team, 2018). The relative MASE and relative sMAPE are obtained by dividing the MASE and sMAPE of the forecast by the MASE and sMAPE of the naïve forecast, respectively, and OWA is the mean of the obtained values. Thus, an OWA of less than one is better than Naïve and a value over one is worse than Naïve. In

Table 1
OWA for the single models, the ensemble and the benchmark method Comb on the daily dataset.

		Macro	Micro	Demographic	Industry	Finance	Other	All
Single model	ARIMA 1	1.160	1.023	1.142	1.016	1.052	1.038	1.041
	ARIMA 2	1.123	1.009	1.144	1.031	1.035	1.051	1.033
	ETS	1.022	0.989	0.983	1.012	0.991	1.006	0.996
	Custom method	1.336	1.360	1.050	1.342	1.380	1.304	1.354
	Naïve	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Ensemble	No correlator	1.020	0.990	0.999	1.008	0.987	1.011	0.995
	Correlator with Bugs 1 and 2	0.805	0.785	0.849	1.007	0.985	1.011	0.930
	Correlator with Bug 2	0.805	0.785	0.849	1.007	0.944	0.981	0.908
	Correlator	0.776	0.778	0.814	1.007	0.939	0.979	0.903
Benchmark	Comb	1.015	0.980	0.978	1.005	0.958	1.005	0.979

Notes: The benchmark method Comb is the arithmetic average of the Simple, Holt and Damped exponential smoothing models that is also used by Makridakis et al. (2018). We evaluated the performance of Comb using its implementation that is available in the M4 code repository.[1]

Table 2
Comparison of the correlator's performance on the daily dataset for different values of the STD ratio threshold in Eq. (6).

	Correlation threshold 0.9999				Correlation threshold 0.999				Correlation threshold 0.99			
STD ratio threshold	2	2.5	3	–	2	2.5	3	–	2	2.5	3	–
Correlator used (#)	480	518	541	566	743	794	827	855	1704	1772	1827	1867
Correlator used (%)	11.4	12.3	12.8	13.4	17.6	18.8	19.6	20.2	40.3	41.9	43.2	44.2
Correlator MASE	0.191	0.183	0.196	0.254	0.227	0.214	0.221	0.275	2.176	2.134	2.140	2.143
Correlator sMAPE	0.310	0.298	0.334	0.386	0.299	0.284	0.306	0.356	2.485	2.411	2.411	2.408
Correlator OWA	0.098	0.092	0.098	0.115	0.109	0.101	0.104	0.121	0.729	0.709	0.704	0.699
Full OWA	0.913	0.903	0.897	0.891	0.875	0.863	0.854	0.848	0.890	0.876	0.870	0.864

Notes: A dash means that Eq. (6) was not used. The correlator MASE, sMAPE and OWA values are calculated from the forecasts of the correlator; that is, only a subset of the daily dataset is considered. The full OWA is calculated from the full dataset using forecasts obtained from a combination of the ensemble and the correlator.

Table 3
Performance of the correlator on all categories.

	Hourly	Daily	Weekly	Monthly	Quarterly	Yearly
Time series (#)	414	4227	359	48000	24000	23000
Correlator used (#)	25	518	7	1357	450	178
Correlator used (%)	6.0	12.3	1.9	2.8	1.9	0.8
Correlator MASE	0.124	0.183	1.732	4.972	0.552	2.929
Correlator sMAPE	0.560	0.298	0.632	8.144	1.396	5.321
Correlator OWA	0.028	0.092	0.360	0.580	0.128	0.444

Notes: The correlator MASE, sMAPE and OWA values are calculated from the forecasts of the correlator; that is, only a subset of the daily dataset is considered. The correlator used a 14-time-point window for calculating correlations for all categories except for yearly, for which 13 was used, because some of the time series in the yearly dataset had only 13 data points.

what follows, the values presented for MASE, sMAPE and OWA are the average values over all forecasts considered.

As can be seen from Table 1, the main improvement over Naïve was achieved due to the use of the correlator. Of the single models, only ETS achieved a better performance than Naïve, with an OWA of 0.996. The ensemble of five models improved the result only slightly, to 0.995. As can be seen from Table 2, the use of the correlator would have made it possible to obtain an OWA of 0.863. However, our submission achieved only 0.930 due to our use of a higher correlation threshold (0.9999 instead of 0.999) and the presence of two bugs in the code, as was described in Section 2.3. An additional analysis showed that using only one of the two ARIMA models resulted in a better OWA: 0.980 when only ARIMA 1 was used and 0.982 when only ARIMA 2 was used.

It can be seen from Table 2 that using a value of 2.5 for the constant in Eq. (6) gives the smallest OWA for the correlator's forecasts (see the row 'Correlator OWA' in the table). However, the OWA for the combination of ensemble and correlator is the smallest when Eq. (6) is not used at all (see the row 'Full OWA'). Without Eq. (6), an OWA of 0.848 could have been achieved by the proposed method.

Finally, Table 3 shows the performances of the correlator on the other categories of the M4 dataset. We can see that the correlator was the most successful on the daily and hourly datasets, where it was used on 12% and 6% of the time series, respectively, and achieved OWA values of less than 0.1. On the rest of the categories, the correlator was used on under 3% of time series, and in most cases the OWA was considerably larger than on the daily and hourly datasets. This suggests that the similarities between time series were easiest to exploit in the daily and hourly categories.

[1] https://github.com/M4Competition/M4-methods.

4. Discussion

As can be seen from Table 1, most of the improvement over Naïve was achieved by using the correlator. This implies that the daily dataset contained various time series that had short segments that were highly correlated with segments of other series. However, it turns out that actually the correlated regions are often much longer. We investigated the nature of correlations in this dataset more closely by performing the following further analysis.

Similarly to the correlator of Section 2.2, we cross-correlated each time series y^j with every other time series y^k by sliding one time series along the other. However, instead of calculating the Pearson correlation coefficient after each shift between 14 data points, we calculated it globally on the whole overlapping region. Formally, this means that, for every $\tau \in \{14, \ldots, n_k - 14\}$, we calculated the following correlation:

$$r'_{jk}(\tau) := \text{pearson}\left(y^j_{[n_j-(\min(n_j,\tau)-1):n_j]}, y^k_{[\tau-(\min(n_j,\tau)-1):\tau]}\right). \tag{8}$$

As in Section 2.2, we found the values k and τ which maximise $r'_{jk}(\tau)$ for each time series y^j. As a result, we found k and τ with $r'_{jk}(\tau) \geq 0.995$ for 1083 time series. A detailed inspection revealed that some of those high correlations were due to a single big jump in both time series which resulted in a high correlation after alignment, even though the regions before the jump and after the jump in the two series were not correlated, respectively. We discarded such cases with jumps manually (the detailed list is available in the code repository, see Appendix A).

The remaining set C of 1004 pairs of highly correlated time series (y^j, y^k) with the respective shifts τ constitute cases with global correlations, not just short correlated segments. Note that the length of the overlapping segment between time series y^j and y^k in Eq. (8) is $\min(n_j, \tau)$. Fig. 3 shows the histogram over the overlap lengths $\min(n_j, \tau)$ of the pairs in set C. We can see that the daily dataset even contains time series that have highly correlated segments with lengths of up to around 4000 data points. However, highly correlated segments of less than 1000 data points are the most frequent. In what follows, we split the 1004 cases in set C into four categories.

T1: **Self-correlations** (cases where $(y^j, y^j) \in C$). These are time series where the end is highly correlated with the beginning. While short self-correlations could be explained by periodicity, our identified cases can have very long repeats, potentially pointing at problems in the data; see Fig. 4 for an example.

T2: **Mutual correlations** (cases where $(y^j, y^k) \in C$ and $(y^k, y^j) \in C$ for $j \neq k$). These are pairs of time series where the end of one series correlates with the beginning of the other, and vice versa. For an example, see Fig. 5.

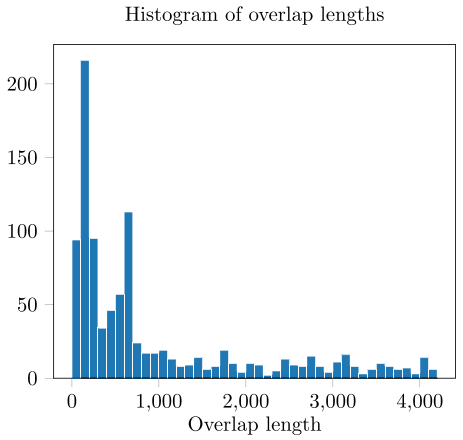

Fig. 3. Histogram of the lengths of highly correlated regions, as identified by our analysis.

T3: **Synchronised correlations** (cases where $j \neq k$ and the correlated region corresponds to the same dates in both time series). In such cases, using the end of y^k to forecast the continuation of y^j would be using data from the future (forecasting earlier dates using data from later dates), which is of course not possible in practical applications and demonstrates data leakage (Kaufman, Rosset, Perlich, & Stitelman, 2012) in the M4 data. For an example, see Fig. 6. Note that these cases could not be identified at the time of the competition, as the actual starting dates of the time series were only published by the M4 organisers after the competition (Makridakis et al., 2018).

T4: **Unsynchronised correlations** (cases where $j \neq k$ and the correlated region corresponds to different dates in the two time series). Depending on which time series is earlier, this might or might not be a leakage. The example in Fig. 7 shows series that are claimed to start at the same date but become almost perfectly correlated after shifting one of them, possibly suggesting that the start date information might not be correct.

Table 4 shows the number of time series pairs in each of the categories T1, T2, T3 and T4. Note that we made the categories mutually exclusive by excluding those that belong to the narrower categories T1 and T2 from the wider categories T3 and T4.

Note that above we have only categorised correlations in which one of the segments is at the end and another at the beginning of some time series. We did also identify other, more peculiar kinds of correlations, such as the example shown in Fig. 8, where one time series can be split into three large segments and rearranged almost perfectly into another time series.

The above results raise the question of whether the success of the correlator in the M4 competition could be explained fully by leakages in forecasting using information from the future. Our additional analysis revealed that the correlator used future data in only 26% of the

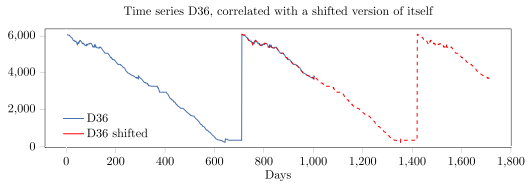

Fig. 4. Time series that repeats itself.

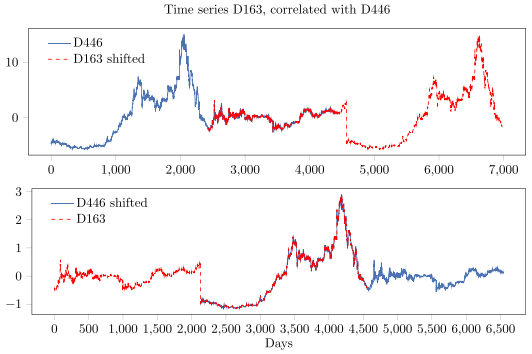

Fig. 5. Time series such that the beginning of one is correlated with the ending of the other, and vice versa.

Table 4
Numbers of time series in categories T1–T4.

	T1	T2	T3	T4	All
Time series (#)	12	202	23	767	1004
MASE	1.615	0.077	0.702	0.471	0.410
sMAPE	1.418	0.084	0.540	0.424	0.370
OWA	0.482	0.030	0.202	0.166	0.144

Notes: The table shows how many time series were assigned to each of the categories T1–T4, and the performances of forecasts when using these similarities.

cases in which it was applied. Allowing the correlator to use only past values resulted in OWA values of 0.895 and 0.917 with correlation thresholds of 0.999 and 0.9999 respectively. These results lead us to the hypothesis that there may be further data quality issues, including duplications of data within one time series, explaining some of the correlations of type T1; rearrangement errors of time series segments, explaining some of the T2 cases; and wrong starting dates, explaining cases in T4.

Importantly, the evidence that we have presented here suggests strongly that the success of the correlator in forecasting the future from the past was due only to data quality issues in the M4 data, most probably going back to issues in the databases from which the M4 data originated.

5. Conclusions

This article has described our method for obtaining forecasts for the daily time series within the M4 competition. We used an ensemble of five statistical forecasting methods and a method that we refer to as the correlator. The correlator was applied to a time series only if its last 14 days were found to be correlated strongly with

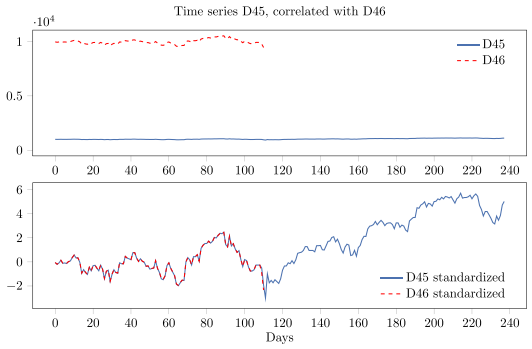

Fig. 6. Time series that match after standardising.

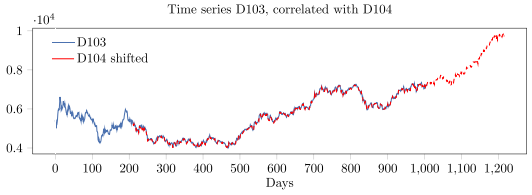

Fig. 7. Time series that are shifted versions of one another.

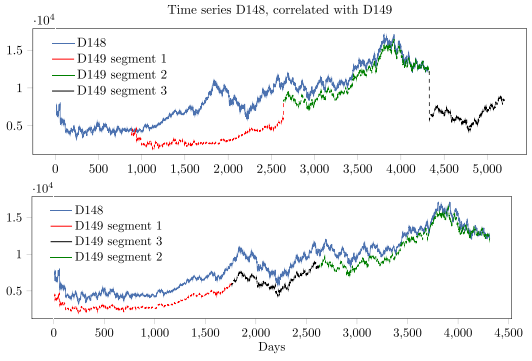

Fig. 8. Time series that can be split into large segments and rearranged into one another.

a 14-day period in another time series. Our retrospective analysis using the full time series published by the M4 organisers after the competition demonstrated that the correlator was responsible for most of our gains over the naïve constant forecasting method. We found that the correlator actually used data from the future in 26% of cases in which it was applied. However, we had no way of filtering those cases out in our submission because the actual starting dates were not published by the M4 organisers until after the competition. We suggest that the starting dates be provided explicitly in future competitions in order to avoid this problem. This analysis has revealed some potential data quality issues with the M4 data, including potential duplications and rearrangement errors within particular time series and potential errors in starting dates. We cannot rule out the possibility that the success of the correlator in forecasting the future from the past was due only to data quality issues in the time series.

Acknowledgments

Our submission to the M4 time series forecasting competition was prepared during a seminar dedicated to M4 at the University of Tartu. The authors would like to thank all other participants of the seminar for useful discussions: Alina Vorontseva, Anton Potapchuk, Aytaj Aghabayli, Basar Turgut, Diana Grygorian, Gunay Abdullayeva, Jayasinghe Arachchilage Sriyal Himesh Jayasinghe, Joonas Puura, Kaur Karus, Maksym Semikin, Markus Loide, Martin Liivak, Martin Valgur, Mikhail Papkov, Märten Veskimäe, Olha Kaminska, Prabhant Singh, Saumitra Bagchi and Yevheniia Kryvenko.

Appendix A. Implementation of the algorithm

The code for producing the forecasts using our method can be found in the Github repository.[2] Detailed instructions on how to run the code are provided in the `README.md` file in the repository.

Appendix B. ARIMA and ETS models

Two ARIMA models (Box, Jenkins, Reinsel, & Ljung, 2015) and an ETS model (Hyndman et al., 2002) were fitted using the corresponding functions in the `forecast` package, version 8.3 (Hyndman & Khandakar, 2008), in the R software version 3.2.4. ETS was fitted using the function `ets` with the default parameter settings. The ARIMA models were fitted using the `auto.arima` function. In what follows, we discuss the differences between the two ARIMA models.

The first ARIMA model was obtained using the stepwise algorithm (Hyndman & Khandakar, 2008) with the default settings except for the starting values for the order of the autoregressive model and the moving average model, which were set to zero. We refer to this method as ARIMA 1.

The second model was obtained using a non-stepwise algorithm that searches through all of the (S)ARIMA models, with the maximum allowed order for the model set to eight. The fitting of the ARIMA models was done using maximum likelihood estimation instead of using approximation with the conditional sum of squares, as in the ARIMA 1 method. We refer to this method as ARIMA 2.

Our retrospective analysis showed that using only one ARIMA model in the ensemble instead of two gave better results, as was outlined in Section 3. Thus, using only ARIMA 1 would have been beneficial, since ARIMA 2 is much more computationally expensive.

Appendix C. Custom method

Our custom time series forecasting method[3] was inspired by *Forecasting with decomposition* from Hyndman and Athanasopoulos (2014). Our method differs from that described by Hyndman and Athanasopoulos (2014) in two main ways. First, we use the classical decomposition (Macaulay, 1931) instead of STL decomposition (Cleveland, Cleveland, McRae, & Terpenning, 1990), and, second, we use either linear extrapolation (Shumway & Stoffer, 2017) or past values from the decomposition for forecasting the trend and residuals.

More specifically, a classical additive decomposition is performed on the time series, with the frequency set to two instead of one as in the daily dataset. This means that a moving average filter with weights $(0.25, 0.5, 0.25)$ is applied to the time series in order to obtain the trend, and then the seasonal component with period two is calculated from that; see Hyndman and Athanasopoulos (2014) for details. However, in most cases the seasonal component is very small relative to the other components, since the time series does not actually have a frequency of two. Thus, setting the frequency to two mainly serves to reduce the noise in the time series by applying a moving average filter. For the decomposition, we used the `seasonal_decompose` function in the `StatsModel` package version 0.8.0 (Seabold & Perktold, 2010).

Two approaches are considered for predicting the trend and the residuals, and all four combinations of these approaches are evaluated on the internal validation set, with the approach that performs the best according to MASE evaluation measure (Hyndman & Koehler, 2006) being used on the actual dataset. One approach considered is a linear extrapolation from the last 14 values, which was performed using `curve_fit` from the `Scipy` package version 1.0.1.[4] The other approach is to use the previous 14 values of the corresponding component as the forecasts.

Linear functions were chosen for the extrapolation because the method was developed mainly for yearly data, in which training samples were limited and the trend was not changing dramatically in many cases. Thus, the poor performance on the daily dataset was unsurprising. For the yearly dataset, our custom method achieved an OWA of 0.967. We decided to include the custom method in the daily ensemble in order to introduce some variability.

The seasonal component is predicted as a seasonal naïve, which means that the forecast for the current period is the corresponding value for the last period. The final forecast is the sum of the predicted trend, residual and seasonal components. In our original submission, this method was used for all daily time series except for time series D3160, which had an incorrect forecast due to a bug in our code.

References

Box, G., Jenkins, G., Reinsel, G., & Ljung, G. (2015). Time series analysis: Forecasting and control. In *Wiley series in probability and statistics*. Wiley.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, *6*(1), 3–73.

Hyndman, R., & Athanasopoulos, G. (2014). *Forecasting: principles and practice*. OTexts, URL: https://otexts.org/fpp2/.

[2] https://github.com/antiingel/correlator.

[3] Originally named *fearless* in our submission.

[4] See http://www.scipy.org/.

Hyndman, R., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, *27*(3), 1–22.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*(4), 679–688.

Hyndman, R. J., Koehler, A. B., Snyder, R. D., & Grose, S. (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting*, *18*(3), 439–454.

Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *6*(4), 15:1–15:21.

Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y., & Zhang, Y. (2019). Short-term residential load forecasting based on LSTM recurrent neural network. *IEEE Transactions on Smart Grid*, *10*(1), 841–851.

Livera, A. M. D., Hyndman, R. J., & Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, *106*(496), 1513–1527.

M4 Team (2018). *M4 competitor's guide: prizes and rules*. URL: https://www.m4.unic.ac.cy/wp-content/uploads/2018/03/M4-Competitors-Guide.pdf. [Online]. (Accessed September 27, 2018).

Macaulay, F. R. (1931). *The smoothing of time series*. NBER Books.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 competition: results, findings, conclusion and way forward. *International Journal of Forecasting*, *34*(4), 802–808.

Seabold, S., & Perktold, J. (2010). Statsmodels: econometric and statistical modeling with Python. In *Proceedings of the 9th python in science conference* (pp. 57–61). SciPy society Austin.

Shumway, R. H., & Stoffer, D. S. (2017). *Time series analysis and its applications: with R examples* (4th ed.). Springer International Publishing.

Zhang, G. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*, 159–175.

Anti Ingel is a PhD student of computer science at the University of Tartu, Estonia, where he specialises in computational neuroscience and machine learning. He obtained his Master's degree at University of Tartu. During his studies, he collaborated in Computational Neuroscience Lab at the University of Tartu where he worked on classification algorithms for brain-computer interfaces.

Novin Shahroudi is a MSc student of Computer Science at the University of Tartu, Estonia. He obtained his BSc from Qazvin Azad University, Iran. During his bachelor studies, he was in Mechatronics Research Laboratory (MRL) as an undergraduate research programmer collaborating on the RoboCup SPL project.

Markus Kängsepp is a PhD student of computer science at the University of Tartu, Estonia. He obtained his bachelor's and master's degree from the Institute of Computer Science at the University of Tartu.

Andre Tättar is a PhD student of computer science at the University of Tartu, Estonia. He obtained his bachelor's and master's degree from the institute of Computer Science at the University of Tartu.

Viacheslav Komisarenko is a MSc of computer science student at the University of Tartu, Estonia. He obtained his BSc from the Institute for Applied System Analysis of the National Technical University of Ukraine "Kiev Polytechnic Institute".

Meelis Kull is an associate professor of data mining at the University of Tartu, Estonia. He obtained his PhD from the University of Tartu and worked as a post- doctoral senior research associate at the University of Bristol, UK.