Method Description

General Information

Type of Entry (*Academic, Practitioner, Researcher, Student*)	Academic
First Name	Fotios
Last Name	Petropoulos
Country	United Kingdom
Type of Affiliation (*University, Company-Organization, Individual*)	University
Affiliation	University of Bath

Team Members (*if applicable*):

1st Member	
First Name	Ivan
Last Name	Svetunkov
Country	United Kingdom
Affiliation	Lancaster University
2nd Member	
First Name	
Last Name	
Country	
Affiliation	

Information about the method utilized

Name of Method	SCUM (Simple Combination of Univariate Models)
Type of Method (*Statistical, Machine Learning, Combination, Other*)	Combination
Short Description (up to 200 words)	Our approach is a simple combination of four models: exponential smoothing, complex exponential smoothing, automatic ARIMA and dynamic optimized theta method. The median operator was used to combine the outputs of the four models. More details are provided in the following document.

Extended Description:

For an extended description, please see the attached document.

A **S**imple **C**ombination of **U**nivariate **M**odels (SCUM)

Fotios Petropoulos[a,*], Ivan Svetunkov[b]

[a] *School of Management, University of Bath, UK*
[b] *Centre for Marketing Analytics and Forecasting, Lancaster University Management School, Lancaster, UK*

Abstract

This document describes the approach implemented to produce our point forecasts and prediction intervals for the M4-competition submission.

Keywords: M4-competition, ETS, ARIMA, Theta method, Complex exponential smoothing, median combination

1. The approach

Our approach is a simple combination of four models. All model were applied using existing implementations in the `R` statistical software. The following four models were considered:

- Exponential smoothing (ETS, Hyndman et al., 2002), which selects the best of the fifteen exponential smoothing models based on the minimisation of a pre-specified information criterion. For time series with frequencies lower or equal to 24 (yearly, quarterly, monthly and daily), we used the `ets()` function of the *forecast* package (Hyndman et al., 2017); for higher frequencies (weekly and hourly series), we used the `es()` function of the *smooth* package (Svetunkov, 2018), which selects the best model out of all the possible 30 combinations. Both functions use by default the corrected Akaike information criterion (AICc) for the model selection procedure.
- Complex exponential smoothing (CES, Svetunkov and Kourentzes, 2016), which sidesteps the ETS taxonomy and produces non-linear trends with a slope depending on the data characteristics. We used the `auto.ces()` function of the *smooth* package, which makes a selection between two seasonal and one non-seasonal models.
- Automatic Autoregressive Integrated Moving Average model (AutoARIMA, Hyndman and Khandakar, 2008), which identifies the best ARIMA model. We used the `auto.arima()` function of the *forecast* package.

*Correspondence: Fotios Petropoulos, School of Management, University of Bath, Claverton Down, Bath, BA2 7AY, UK.

Email addresses: `f.petropoulos@bath.ac.uk` (Fotios Petropoulos), `i.svetunkov@lancaster.ac.uk` (Ivan Svetunkov)

May 31, 2018

- Dynamic Optimised Theta Model (DOTM, Fiorucci et al., 2016b), which is an extension of the theta method for forecasting (Assimakopoulos and Nikolopoulos, 2000). DOTM optimises the θ value of the theta line that focuses on the short-term curvatures of the seasonally adjusted data. We used the `dotm()` function of the *forecTheta* package (Fiorucci et al., 2016a). Note that for the series that are longer than 5000 observations (D2047, D2194 and D4099), DOTM is applied only on the last 5000 observations.

Table 1 summarises the `R` packages and functions used for each model and frequency.

Table 1: Forecasting models and the corresponding `R` functions.

Model	Frequency	`R` package	Function
ETS	≤ 24	*forecast* 8.2	`ets()`
	> 24	*smooth* 2.3.1	`es()`
CES	All	*smooth* 2.3.1	`auto.ces()`
AutoARIMA	All	*forecast* 8.2	`auto.arima()`
DOTM	All	*forecTheta* 2.2	`dotm()`

The median operator was used to combine the outputs of the four models. While more complicated methods for averaging the prediction intervals information from the different models could have been considered, Lichtendahl et al. (2013) claim that a simpler method of averaging the quantiles seems to work quite well. As such, we opt for simplicity and average the lower and upper prediction intervals similarly to the point forecasts using the median operator.

After averaging the point forecasts and prediction intervals, we set any negative values to zero, as the M4-competition data set consists of non-negative values.

2. Reproduction information

We provide two `R` scripts that allow the reproduction of the point forecasts and prediction intervals.

"M4-FPIS-create-datasets-20180523.R" file requires as input the comma-separated-value (csv) files provided by the organisers at the beginning of the competition ("Daily-train.csv", "Hourly-train.csv", etc.) and saves the data in RData format. Six RData files are produced, one for each data frequency.

"M4-FPIS-submission-code-20180525.R" file requires as input the RData files produced as output from the previous script and outputs the point forecasts and prediction intervals in three csv files ("all-ffcs.csv", "all-lower.csv", "all-upper.csv"). The `R` code to reproduce the submission values (point forecasts and prediction intervals) for one time series is provided in the Appendix. It is possible to reproduce the point forecasts and prediction intervals for

all or specific series. For example, the following code will reproduce of the submission values for all series:

```
ids = array(NA, c(2, sum(tsns[1:6])))
ids[1,] = c(rep(1, tsns[1]), rep(2, tsns[2]), rep(3, tsns[3]),
            rep(4, tsns[4]), rep(5, tsns[5]), rep(6 ,tsns[6]))
ids[2,] = c(1:tsns[1], 1:tsns[2], 1:tsns[3], 1:tsns[4], 1:tsns[5], 1:tsns[6])
```

As another example, the following code will reproduce the submission values for the 13th yearly series, the 45th and the 10001th quarterly, the 30102th monthly and the 11th weekly series:

```
ids = array(NA, c(2, 5))
ids[1,] = c(1, 2, 2, 3, 4)
ids[2,] = c(13, 45, 10001, 30102, 11)
```

Generally speaking, one should define a two-dimensional array called `ids` with exactly two rows and as many columns as the number of series required to be reproduced. The first row of `ids` should contain the identifiers for the datasets (1: yearly; 2: quarterly; 3: monthly; 4: weekly; 5 daily; 6: hourly) while the second row should contain the specific series identified within each subset.

At this stage we should note that we used `R` version 3.4.3 (2017-11-30), *forecast* version 8.2, *smooth* version 2.3.1, *forecTheta* version 2.2. Moreover, a parallel implementation was adopted in our code, where the following packages and versions were used: *doSNOW* (1.0.16), *foreach* (1.4.4), *snow* (0.4-2), *iterators* (1.0.9). The output of the `sessionInfo()` function that gives details of all packages and versions used is provided in the Appendix.

All forecasts and prediction intervals were produced using the Bath Balena High Performance Computing (HPC) Service at the University of Bath. More specifically, we used Intel Skylake nodes that are dual socket nodes clocked at 2.6GHz. Each Skylake node has 2× Intel Xeon Gold 6126 CPUs providing 24 cores and 192GB memory (8GB per core) via 6 memory channels per CPU socket.

References

Assimakopoulos, V., Nikolopoulos, K., 2000. The Theta model: a decomposition approach to forecasting. International journal of forecasting 16 (4), 521–530.

Fiorucci, J. A., Louzada, F., Yiqi, B., 2016a. forecTheta: Forecasting Time Series by Theta Models. R package version 2.2.
URL https://cran.r-project.org/package=forecTheta

Fiorucci, J. A., Pellegrini, T. R., Louzada, F., Petropoulos, F., Koehler, A. B., Oct. 2016b. Models for optimising the theta method and their relationship to state space models. International journal of forecasting 32 (4), 1151–1161.

Hyndman, R., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., Petropoulos, F., Razbash, S., Wang, E., Yasmeen, F., 2017. forecast: Forecasting functions for time series and linear models. R package version 8.2.
URL https://cran.r-project.org/package=forecast

Hyndman, R. J., Khandakar, Y., Jul. 2008. Automatic time series forecasting: The forecast package for R. Journal of statistical software 27 (3), 1–22.
Hyndman, R. J., Koehler, A. B., Snyder, R., Grose, S., Jul. 2002. A state space framework for automatic forecasting using exponential smoothing methods. International journal of forecasting 18 (3), 439–454.
Lichtendahl, K. C., Grushka-Cockayne, Y., Winkler, R. L., Mar. 2013. Is it better to average probabilities or quantiles? Management science 59 (7), 1594–1611.
Svetunkov, I., 2018. smooth: Forecasting Using Smoothing Functions. R package version 2.3.1. URL https://github.com/config-i1/smooth
Svetunkov, I., Kourentzes, N., 2016. Complex exponential smoothing for seasonal time series.

Acknowledgments

This project made use of the Balena High Performance Computing (HPC) Service at the University of Bath.

Appendix: R code and session information

Producing point forecasts and prediction intervals for a single series (y)

```
# Define structures to hold forecasts and prediction intervals for one series
fcs = array(NA, c(4, horizon))
pis = array(NA, c(4, horizon, 2))

# ETS
if (frequency(y) > 24){
  # condition for W and H
  fcs1 = es(y, h=horizon, intervals="p", level=0.95)
  fcs[1,] = as.vector(fcs1$forecast)
} else {
  fcs1 = forecast(ets(y), h=horizon, level=95)
  fcs[1,] = as.vector(forecast(fcs1, h=horizon)$mean)
}
pis[1,,1] = fcs1$lower
pis[1,,2] = fcs1$upper

# CES
fcs2 = auto.ces(y, h=horizon, intervals="p", level=0.95)
fcs[2,] = as.vector(fcs2$forecast)
pis[2,,1] = fcs2$lower
pis[2,,2] = fcs2$upper

# AutoARIMA
fcs3 = forecast(auto.arima(y), h=horizon, level=95)
fcs[3,] = as.vector(forecast(fcs3, h=horizon)$mean)
pis[3,,1] = fcs3$lower
pis[3,,2] = fcs3$upper

# DOTM
if (length(as.vector(y)) > 5000){
  # condition for D2047, D2194 and D4099
  fcs4 = dotm(ts(tail(as.vector(y), 5000), frequency = frequency(y)),
     h=horizon, level=95)
} else {
  fcs4 = dotm(y, h=horizon, level=95)
}
fcs[4,] = as.vector(fcs4$mean)
pis[4,,1] = fcs4$lower
pis[4,,2] = fcs4$upper

# bind forecasts and prediction intervals in a singe vector
fcspis = c(apply(fcs, 2, median), apply(pis[,,1], 2 ,median), apply(pis[,,2],
   2 ,median))

# set negative values to zero
fcspis[fcspis < 0] = 0
```

Session information (output of `sessionInfo()`)

```
R version 3.4.3 (2017-11-30)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Scientific Linux release 6.9 (Carbon)

Matrix products: default
BLAS/LAPACK:
    /apps/intel/parallel_studio_xe_2018/compilers_and_libraries_2018.0.128/
linux/mkl/lib/intel64_lin/libmkl_intel_lp64.so

locale:
 [1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8 LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8 LC_NAME=C
 [9] LC_ADDRESS=C         LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] parallel stats graphics grDevices utils datasets methods
[8] base

other attached packages:
[1] doSNOW_1.0.16 snow_0.4-2 iterators_1.0.9 foreach_1.4.4
[5] forecTheta_2.2 tseries_0.10-42 smooth_2.3.1 forecast_8.2

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.14 magrittr_1.5 munsell_0.4.3 colorspace_1.3-2
 [5] lattice_0.20-35 rlang_0.1.6 quadprog_1.5-5 TTR_0.23-2
 [9] plyr_1.8.4   xts_0.10-1    nnet_7.3-12   quantmod_0.4-12
[13] grid_3.4.3   timeDate_3042.101 gtable_0.2.0 lazyeval_0.2.1
[17] lmtest_0.9-35 tibble_1.4.1 nloptr_1.0.4 ggplot2_2.2.1
[21] codetools_0.2-15 curl_3.1 fracdiff_1.4-2 compiler_3.4.3
[25] pillar_1.1.0 scales_0.5.0 zoo_1.8-1
```