Method Description

General Information

Type of Entry (*Academic, Practitioner, Researcher, Student*)	Practitioner
First Name	Alex
Last Name	Roubinchtein
Country	USA
Type of Affiliation (*University, Company-Organization, Individual*)	Company-Organization
Affiliation	Washington State Employment Security Department

Team Members (*if applicable*):

1st Member	
First Name	
Last Name	
Country	
Affiliation	
2nd Member	
First Name	
Last Name	
Country	
Affiliation	

Information about the method utilized

Name of Method	Optimum combination of models
Type of Method (*Statistical, Machine Learning, Combination, Other*)	Statistical
Short Description (up to 200 words)	We used R-software and attached code includes comments. The core concept of our approach is the use of optimum combinations of forecasts for three model types: 1-ets or tbats, 2-auto-arima and 3-simple regression from trend and seasonality. These main codes are from forecast package (Author Rob J Hyndman, et al. To find optimum combination we use function nmkb (Nelder-Mead optimization algorithm for derivative-free optimization) from the dfoptim package (Author Ravi Varadhan, Johns Hopkins University, and Hans W. Borchers, ABB Corporate Research. The weights are defined based on performance on training time-frame. Two options were used for different

	frequencies (types) of the data: 1- performance on full sample and 2- combined results of performance on full sample and hold-out sample. Hold-out periods are equal to forecast periods. Models have slightly different specifications for different data sets, due to the handling of seasonality, criteria's reflecting training options, the handling of "problematic" series, etc. Since forecast processing takes a long time, we used R's parallel processing for all calculations. This enabled us to view process controls, e.g. convergence for optimization. We produced intermediate outputs for each data type and then combined them in a unified output for posting forecasts and intervals.

Extended Description:

The main tools used in our code were developed by Rob J Hyndman and his colleagues in framework of package forecast and related publications. Our main idea is weighting forecasts from different models, based on performance on training data. We use training on full sample for annual and weekly forecasts and combine training on full sample and hold out sample for other four types of the data. Hold-out periods are equal forecasting periods.

Our optimization process used the Nelder-Mead optimization algorithm for derivative-free optimization. The nmkb function from the dfoptim package (Author Ravi Varadhan, Johns Hopkins University, and Hans W. Borchers, ABB Corporate Research) performed much better than any other optimization option, which we tested. Convergence to optimum was achieved for all series.

We avoided the use of constrained optimizations, due to performance issues and convergence problems. Since we still want the totals of weights be close to 1, we introduced penalties for differences between totals of weights and 1 in our criteria's. For very few cases (total of 6) the penalties were not enough and rather than increase penalties we included in our code a 'safety option'. The safety option criteria were defined as follows: when absolute differences between totals of weights and 1 was more than 5%, we used simple averages between auto-arima and ets forecasts. Another safety option in our code is to use seasonal naïve forecast for cases when combined output from the model has any negative numbers. That was also very seldom.

Frequencies in our models reflect the expected seasonal periods. For four of the six types of data, frequencies were the same as those declared by organizers. For weekly and daily data we allowed seasonal variation within models. The logic behind this is the following: If a series naturally non-seasonal, the allowance of seasonality has a very limited negative impact, if any, as long as the seasonal model was good. Seasonal variations just will be insignificant in estimations. Missing seasonality, by assuming data to be non-seasonal, has the potential for greater damage to estimations (even based on some established threshold for seasonal variation in separate testing).
We assumed seasonal periods of 52 for weekly data and 7 for daily. This is pretty much in line with Hyndman's approach to weekly and daily forecasts. Following his approach, we also use Fourier regressors to reflect seasonality. In our office we found this technique proven to be useful for seasonal adjustments of weekly data for unemployment claims. Classical ARIMA seasonal adjustment models, used by the U.S. Bureau of Labor Statistics, for seasonal adjustments of monthly data are not applicable to weekly series.

Instructions for using code:

Code is written in R and uses packages freely available in CRAN.

First, copy all training data into a working directory. For example, create a working directory folder named ‘m4comp’ on your C drive.

Connect the R script to the directory by running code line 27: setwd (“C: /m4comp”).
After connecting the script to your directory, you have two basic options for using code:

I. You may run the entire code from line 32 through line 582. This will result in the creation of all intermediate outputs and final files for point forecasts, upper and lower bounds formatted per delivery requirements. This option takes a very long time. It is hard accurately estimate total processing time since we produced and compared a few forecast options for each data type using significantly different computers: from most powerful with 32 cores to very limited desktops with just 4 cores. Our more powerful 16 and 32 core computers were network servers and speed was significantly impacted by network conditions and other users. Our latest test, involving only 100 series for each type of time series, on a network server with 16 cores (15 cores available for processing code), gave us an estimated total time of about 80 hours, with about 70% of total time used for monthly series. Only parallel processing making processing time reasonable. The approximate time saving is about 0.6-0.7 multiplied by proportion of used cores. For instance, let’s assume that we have a processing time on a 4 core computer (only 3 cores used) of 3 hours. Than the processing time on 16 core computer (15 core used) should not be more than 1 hour (3/ (15/3)/0.6). Processing times will probably be far less if you replace 0.6 with a more realistic value of 0.7.

II. Run separate code-blocks for each data type and combine results.
Annual and weekly series have standalone blocks. For quarterly and monthly

data we use the same forecasting function and just change parameters. Different function is used for daily and hourly data. Below are more detailed instruction about how to split code in separate blocks and combine results.

1. Run lines 27-67 as one block. This first block sets the working directory, creates basic functions and prepares core-clusters for parallel processing. This first block of code should be run each time after quitting and reopening R.
2. Run lines 71-158 as a stand-alone block for annual data. Line 161 can be run if you want to continue down through the script without quitting R.
3. Lines 169-254. Makes function for quarterly and monthly data.
4. Lines 258-264 Makes quarterly forecasts with the function made in block 3. Line 265 can be run if you want to continue without quitting R.
5. Lines 270-276. Makes monthly forecasts with the function made in block 3. Line 277 can be run if you want to continue without quitting R. Since this is the most time consuming part of the script, the block can be run in several chunks. If you choose to separate 48,000 monthly series into four parts, make sure to combine them in the 'month_f.csv' file.
6. Lines 280-357. Stand-alone code for weekly forecast.
7. Lines 361-462. Makes function for daily and hourly forecasts.
8. Lines 466-472. Makes daily forecasts with the function made in block 7. Line 473 can be run if you want to continue without quitting R.
9. Lines 477-483. Makes daily forecasts with the function made in block 7. Line 485 can be run if you want to continue without quitting R.
10. Line 489-579. Combines all outputs. This part of the code can be run independently and does not require block 1.

Flow chart displaying a case when all forecasts are run in separate code-blocks and R is quit after each forecast without saving R-Environments.