Method Description

General Information

Type of Entry (*Academic, Practitioner, Researcher, Student*)	**Student**
First Name	**Konstantin**
Last Name	**Chirikhin**
Country	**Russian Federation**
Type of Affiliation (*University, Company-Organization, Individual*)	**University**
Affiliation	**Novosibirsk State University**

Team Members:

1st Member	
First Name	Boris
Last Name	Ryabko
Country	Russian Federation
Affiliation	Novosibirsk State University, Institute of Computational Technologies SB RAS

Information about the method utilized

Name of Method	**ITP (Information-Theoretic Predictor)**
Type of Method (*Statistical, Machine Learning, Combination, Other*)	**Other**
Short Description (up to 200 words)	**The method is based on applications of data compression to prediction (see B. Ryabko, J. Astola, M. Malyutov. Compression-Based Methods of Statistical Analysis and Prediction of Time Series. Springer, 2016). Combination of three following compression algorithms: zlib (https://zlib.net), re-pair (see Philip Bille, Inge Li Gørtz, and Nicola Prezza. Space-efficient re-pair compression. In *Data Compression Conference (DCC), 2017*, pages 171–180. IEEE, 2017) and ppmd (https://github.com/Shelwien/ppmd_sh) were used. For removing seasonal component STL method was used. To the end of discretized time series each possible continuation of the length**

	equal to the horizon of forecasting was written, and such "extended" series were compressed by all compressors. Distributions from different compressors were combined to a single distribution with equal weights. Pointwise predictions were obtained as means computed by the single distribution.

Extended Description:

Apart from the textual description, please consider including an informative flowchart to help researchers better understand the exact steps followed for generating the forecasts. Please also try to clarify any assumptions made, the initialization and parameterization process used, etc., to facilitate reproducibility and replicability.

Assume that we have a real-valued time series data:

$$Y(t) = y_1, y_2, \ldots, y_{n+1}.$$

If the time series is seasonal, then remove seasonal component by applying STL decomposition. Then take the first difference to obtain a series

$$X(t) = x_1, x_2, \ldots, x_n.$$

Assume that we are asked to give a forecast for the h items ahead. Do the following steps:

1. Find the minimal m and the maximal M values in the time series, then each value in the series falls into the interval $[m; M]$. Add 10% of the interval length to the upper bound M (obtaining M') and subtract that value from the lower bound m (to obtain m');

2. Split the interval $[m'; M']$ consequently to the 2, 4, 8 and 16 subintervals of the equal length. Enumerate such intervals from 0 to k-1, where k is the amount of intervals. Denote a set of interval numbers as *alphabet* $A_k = \{0,1,\ldots,k-1\}$. We enumerated intervals consequently, so any two adjacent intervals had numbers differing by 1. Then handle such partitions separately. Replace each value of the time series with coresponding number of the interval in which the value falls. So we obtain 4 different discrete time series with different alphabets ($A_2 = \{0,1\}$, $A_4 = \{0,1,2,3\}$ and $A_k = \{0,1,\ldots,k-1\}$ in the general case). If the amount of intervals is small, than corresponding discretized time series should be less noizy, and if the amount of intervals is high, then less amount of patterns will be lost due discretization. Discribed approach combines advanages of both ones;

3. Put to each of the four series every possible combination of h symbols from the corresponding alphabet A_k, $1 \le k \le 4$ (so we'll obtain $|A_k|^h$ such combinations for each series). Compress the time series with each possible combination appended to it by each of the participated compressors (zlib, rp and ppmd in our case). For each possible "continuation" we obtain coressponding code length. Note that single continuation from a lesser alphabet coresonds to several continuations from the bigger alphabet. For example, assume that we are given a time series consisting of two values from the alphabet A_2: 0,1. By spliting interval with number 0 to two new subintervals with numers 0 and 1, and, accordingly, interval with number 1 to subintervals with numbers 2 and 3, we can associate

with that series from A_2 four series from A_4: 0,2, 0,3, 1,2 and 1,3. To point out in which interval from A_4 falls specific term of the time series from A_2 enough 1 bit of infromation. In general case to perform fair comparision of code length for time series from different alphabets A_i and A_j, $i \le j$, we should add to code length of series from A_i $(log_2 i - log_2 j)\, l$ bits, where l is the full length of the series;

4. Convert code length to probability. It's well known in information theory that if we sum up values $\sum_{Y \in A^h} 2^{-|\varphi(XY)|}$, where X is a time series, Y runs all possible continuations from alphabet A of length h and $|XY|$ denotes concatenation, we obtain a value less or equals to 1 (Kraft's inequality). Replace each code length $|\phi(XY)|$ with the value $2^{-|\phi(XY)|}$;

5. Denote as X_k real-valued time series X, transformed to the alphabet A_k. Merge such "probability distributions" (actually it is not probablitity distributions because the sum of probabilities may be less than 1) from different alphabets:

$$\mu(XY_{16}) = \sum_{k \in \{2,4,8,16\}} \omega_k \, 2^{-|\phi(XY_k)|},$$

where ω_k- weights, $\sum_k \omega_k = 1$, and in the calculations ω_k were computed by the formula:

$$\omega_k = \begin{cases} \frac{1}{k} - \frac{1}{k+1}, k \neq 4, \\ \frac{1}{k+1}, k = 4. \end{cases}.$$

So now we have single distribution from each compressor with the largest alphabet (16 in our case).

6. Merge "probability distributions" from different compressors into single distribution by the formula:

$$\hat{\mu}(XY_{16}) = \frac{1}{3}\mu_{zlib}(XY_{16}) + \frac{1}{3}\mu_{rp}(XY_{16}) + \frac{1}{3}\mu_{ppmd}(XY_{16}).$$

Now we have one "distribution" for all possible continuations.

7. Normalize probabilities to sum up to the 1:

$$\tilde{\mu}(Y_{16}) = \frac{\hat{\mu}(XY_{16})}{\sum_{Z \in A_{16}^h} \hat{\mu}\,(XZ_{16})}.$$

8. Replace in each continuation interval's numbers with interval's centers. For example, if min value $m' = 1$, max value $M' = 17$ and the interval was splitted to the 16 subintervals, then replace 0 with 1.5, 1 with 2.5, 15 with 16.5;

9. To obtain pointwise predictions, compute mean value of the marginal distribution for each step. For example, if $m' = 2$, $M' = 4$, $A_2 = \{0,1\}$ and the following distribution of continuations of length two is obtained:

Continuation (interval's numbers)	Continuation (interval's means)	Probability
0 0	2.5 2.5	0.2
0 1	2.5 3.5	0.5
1 0	3.5 2.5	0.2
1 1	3.5 3.5	0.1

Then we'll compute prediction for step 1 as (0.2+0.5)*2.5 + (0.2+0.1)*3.5 = 1.75 + 1.05 = 2.8 and for step 2 as (0.2+0.2)*2.5 + (0.5+0.1)*3.5 = 1+2.1=3.1.

10. Integrate predictions and add seasonal component (assuming that seasonal component is constant over time).

To speed up the computations for large k, the following approach was employed. Split original series to the k subseries:

$$Y_1(t) = y_1, y_{1+k}, y_{1+2k}, \dots,$$
$$Y_2(t) = y_2, y_{2+k}, y_{2+2k}, \dots,$$
$$\dots$$
$$Y_k(t) = y_k, y_{k+k}, y_{k+2k}, \dots.$$

and forecast each series separately to the $\lceil h/k \rceil$ steps. Let's denote forecasts of the i-th series as $\hat{Y}_i(1), \dots, \hat{Y}_i(\lceil h/k \rceil)$. Then forecast for the original time series can be obtained as $\hat{Y}_1(1), \hat{Y}_2(1), \hat{Y}_k(1), \hat{Y}_1(2), \hat{Y}_2(2), \dots, \hat{Y}_k(\lceil h/k \rceil)$. To improve accuracy, first $\lceil h/k \rceil$ values were predicted by the full-length time series. Then each Y_i was considered as ordinary full-length time series.

How to reproduce results

Installation. The program was tested on Linux Ubuntu 16.04 and 18.04. To build the program R, Anaconda and MPICH should be installed on the machine. All other necessary packages for Python will be installed automatically.
Install g++, CMake and MPICH:

```
sudo apt-get install g++ cmake mpich
```

Install Anaconda:

```
wget https://repo.anaconda.com/archive/Anaconda3-5.2.0-Linux-x86_64.sh
cd Downloads
./Anaconda3-5.2.0-Linux-x86_64.sh
```

Install R:

```
sudo apt-get install r-base
```

Go to the directory with the program and type:

```
~/anaconda3/bin/pip install itp
```

If anaconda was installed to other than the home directory, path should be changed. Now the program should be installed.
Forecasts generation. To generate forecasts for each group of time series (yearly, monthly and so on) separate Python script was provided. The scripts were placed in the gen_forecasts directory. The forecasts will be printed to the standard output. For example, to generate forecasts for daily data, type

```
~/anaconda3/bin/python forecast_daily.py
```

Name of the file with time series data by default is m4c_daily_data.txt for daily data, m4c_yearly_data.txt for yearly data and so on. You can change the name in the corresponding script. The format of input file is the same in the which the data were given except the first column – it should be numeric. In the software implementation parameter k was called as `sparse`.