Data Schemas for Forecasting (with examples in R)

Sai Van Cuong
CAD Department
Volgograd State Technical University
Volgograd, Russia
svcuonghvktqs@gmail.com

Dr. Andrey Davydenko
JSC CSBI
Saint-Petersburg, Russia
andrey@live.co.uk

Prof. Maxim Shcherbakov
CAD Department
Volgograd State Technical University
Volgograd, Russia
maxim.shcherbakov@gmail.com

***Abstract*—Evaluation of forecasting performance using real-world data is inevitably connected with the question of how to store actuals and forecasts in a convenient way. The issue gets complicated when it comes to working with rolling-origin out-of-sample forecasts calculated for many time series. This setup can be met in both research tasks (such as forecasting competitions or when some new method is proposed) and in practical settings. When designing data schemas for forecasting it is important to provide access to the information needed for exploratory time series analysis and accuracy evaluation. We found that existing approaches to store forecasting data often cannot be applied efficiently as they are either not flexible enough or they require too much resources to implement and maintain the data storage. Here we propose a flexible yet simple way of keeping forecasting data allowing the storage and exchange of actuals, forecasts, and other relevant information. We also present an R package that helps perform exploratory data analysis and accuracy evaluation based on the data schemas proposed.**

Keywords—Forecasting, forecasting methods, forecasting accuracy, forecasting competition, data visualization, R packages

I. Introduction

Nowadays advanced forecasting methods are used in different fields ranging from weather forecasting to inventory control. Advances in hardware, software, and mathematical methods have made it possible to use forecasting algorithms in various industries. The importance of accurate forecasts rises as companies are trying to become more efficient and competitive.

In order to know how good a forecasting method is we need to compare forecasts being produced against actuals being obtained. The aim is to see how well a method can reproduce actuals. Thus, we need empirical evaluation in order to assess the applicability and the effectiveness of a forecasting method.

Thus far, various forecasting competitions have been held to empirically evaluate forecasting performance of different methods. The most famous competition at present is, perhaps, the M3 competition [6, 9] where the accuracy of various methods has been assessed for different categories of time series and different forecasting horizons using various metrics. The question of choosing a good metric for forecast evaluation is itself still a difficult one [8] and has been attracting the attention of researches for quite some time. But we here will focus not on the metrics, but on the technical issues of keeping actuals and forecasts in a convenient way so that proper metrics could be easily applied.

The issue we address here is how to store available data in order to facilitate the evaluation of forecasts.

We look at some existing approaches that implement forecasting data storage and show that some improvements are needed. In particular, it's important to find efficient ways of how to store rolling-origin forecasts with different horizons. Plus, sometimes we need additional info such as confidence intervals, or, perhaps, textual data describing the reasons for adjustments, etc.

We start with describing typical settings of obtaining and evaluating forecasts and then switch to how to organize a data structure that would meet the requirements we set above. We then describe some examples and show how such structures can be used implemented and used. The data structures we describe can be used in different programming environments regardless of a database management system or scripting language.

II. Tasks and Terms

We consider the following setup.

1) Suppose we have a set of time series. In general, the set can contain from one to a relatively large number of series (say, tens of thousands).

2) For each time series we want to store actuals and to calculate and store forecasts. In particular, it is needed to store out-of-sample forecasts produced from different origins (rolling-origin forecasts) and with different horizons and, perhaps, using different methods. We also may want not only to store point forecasts, but prediction intervals (PIs), density forecasts, and additional information related to forecasting

process (such as model coefficients, reasons for judgmental adjustments, etc.).

3) We assume that both actuals and forecasts may be frequently updated as new data becomes available.

Given the above settings, we need to have a convenient means to store and access (and, perhaps, to distribute or exchange) forecasting data including actuals, forecasts, and the related information. We would like to find a means that would be fast, cross-platform, easy to learn and to implement.

Eventually, the storage of forecasting data is needed to perform adequate out-of-sample evaluation of forecasting accuracy. In the case of forecasting competitions, a well-defined approach to store forecasting data should enable a credible approach for forecasting accuracy comparisons.

Some important terms we will be using are clarified below.

Forecast origin – the most recent historical period for which data is used to build a forecasting model. The next time period is the first forecast period [2].

Forecast horizon – the number of periods from the *forecast origin* to the end of the time period being forecast [1].

Prediction intervals (PIs) – the bounds within which future observed values are expected to fall, given a specified level of confidence. For example, a 95% PI is expected to contain the actual forecast 95% of the time. Some researchers have found that estimated PIs are typically too narrow for quantitative and judgmental forecasting methods [1].

III. EXISTING APPROACHES USED IN FORECASTING COMPETITIONS

A number of well-known forecasting competitions have been conducted up to this moment (including M1, M2, M3, M4, and others) [4]. These competitions have had an enormous influence on the field of forecasting focusing on what models produced good forecasts, rather than on the mathematical properties of those models.

For some of the competitions the data is available in the form of R packages:

Mcomp: Data from the M-competition and M3-competition.

M4comp2018: Data from the M4-competition.

Tcomp: Data from the Kaggle tourism competition.

tscompdata: Data from the NN3 and NN5 competitions.

The above packages use the following approach to store forecasting data:

1) Time series are provided as a list of objects. Each series within this list is of class Mdata withh the following structure shown in Table I.

TABLE I. TIME SERIES STRUCTRE USED IN AVAILABLE R PACKAGES

Field name	Description
sn	Name of the series
st	Series number and period. For example, "Y1" denotes first yearly series, "Q20" denotes 20th quarterly series and so on
n	The number of observations in the time series
h	The number of required forecasts
period	Interval of the time series. Possible values are "YEARLY", "QUARTERLY", "MONTHLY" & "OTHER"
type	The type of series. Possible values are "DEMOGR", "INDUST", "MACRO1", "MACRO2", "MICRO1", "MICRO2" & "MICRO3"
description	A short description of the time series
x	A time series of length n (the historical data)
xx	A time series of length h (the future data)

2) Forecasts are provided as a list of dataframes. Each list element is the result of one forecasting method. The dataframe then has the following structure: Each row is the forecast of one series. Rows are named accordingly. For example, if 18 forecasts are prodiced, there are 18 columns, and, if fewer forecasts than 18 exist, the row is filled up with NA values.

The major problem with this approach is that we cannot keep rolling-origin forecasts and it is not cross-platform, i.e., it assumes the use of the R programming environment.

IV. NEW DATA SCHEMAS FOR FORECASTING TASKS

Here were describe our approach to store forecasting data including actuals and forecasts in accordance with what was said in Section II ('Tasks and Terms').

The approach we describe below is convenient when we want to store forecasting data in a relational database (RDB) or as a portable table file (e.g., 'csv' or Excel). RDBs are very widely used, many companies already have an IT infrastructure for storing their data in RDB. Thus, this format is most likely to be adopted in practice (compared to alternatives, such as JSON/XML, etc.).

A. *Time Series Table Schema (TSTS)*

Here we assume each observation is stored in a table as a separate record (line). The table to store such records has the following fields (Table II).

TABLE II. TIME SERIES TABLE SCHEMA (TSTS)

Field name (column name)	Description	Examples
*series_id	Time series identifier - a unique name that identifies a time series	"Y1"
*timestamp	Any representation of the period to which the observation relates.	"01.01.1997" in case of daily data "Sep 1997" in case of monthly data

Field name (column name)	Description	Examples
		"Week 49, 1997" in case of weekly data
value	The value observed	1000

* the key (the unique value that should not duplicated) for this table schema is <series_id, timestamp>. In other words, we cannot have two (or more) records in a table relating to the same time series and the same period of observation (timestamp).

We may have additional fields (columns) in this table or additional table specifying the features of time series. However, the above schema includes the fields that are necessary for further processing of time series data. Here we do not impose restrictions on data types.

If some observation is missing, the corresponding table line can be omitted or corresponding value can be denoted as 'NA'. Observation can also contain censored data, etc., which can also be represented by additional agreements, but here we will not look at the details of such cases. Here we aim to set out a general approach for storing and handling forecasting data.

B. *Forecasts Dynamic Table Schema (FDTS)*

One possible approach to store forecasts is to use the schema shown in Table III. Each forecasting result (be it a point forecast or a limit of a prediction interval) produced with a forecasting method is stored as a separate record (line) in a table. The advantage of this approach is that we can use any number of forecast result attributes without the need to change the fields of the table. The disadvantage is, however, that such tables will be more difficult to handle compared with the alternative schema described below.

TABLE III. FORECASTS DYNAMIC TABLE SCHEMA (FDTS)

Column name	Description	Examples
*series_id	Time series ID for which the forecast was calculated (see Table 1, 'RDB Time Series Schema')	"Y1"
*method_id	Method identifier - a unique name that identifies a method by which the forecasting result was calculated	"auto.arima"
*period_time stamp	Any representation of the period to which the forecast relates.	"01.01.1997"
*origin_time stamp	Origin of the forecast (provided in a timestamp format)	"29.12.1996"
*horizon	Forecast horizon	"3"
*variable	The name of the variable that describes the forecasting result.	forecast" for point forecast "lo95" for the lower limit for the 95% prediction interval "hi95" for the upper limit for the 95% prediction interval "model name" to describe the model used when finding the best model accoriding to Akaike's Information Criterion "error" to store messages describing if anything went wrong "warnings" etc.
value	The value of the variable	"1000.55" for [variable] = "forecast" "ARIMA(1,0,0)" for [variable] = "model name" "Not enough observations" for [variable] = "error" etc.

* the key (the unique value that should not duplicated) for this table schema is <series_id, method_id, forecast_timestamp, origin_timestamp, horizon, variable>.

Here, for simplicity, we assume all the fields being character data or text. Just as was said for the time series table, we may have additional fields for the forecasts table.

The two schemas described above assume that

- We need to ensure that there are no two or more records in a table having the same key
- Values in "timestamp" field of the Time Series Schema are constructed using the same rules as the values in "origin_timestamp" and "period_timestamp" fields.
- Adding or deleting records to tables should be treated as a single transaction, so it is advisable to use stored procedures to implement such operations.

Examples:

M3 competition data represented using the TSTS:

series_id	category	value	timestamp
Y1	MICRO	940.66	1975
Y1	MICRO	1084.86	1976
Y1	MICRO	1244.98	1977
Y1	MICRO	1445.02	1978
Y1	MICRO	1683.17	1979
Y1	MICRO	2038.15	1980
Y1	MICRO	2342.52	1981
Y1	MICRO	2602.45	1982
Y1	MICRO	2927.87	1983
Y1	MICRO	3103.96	1984

M3 competition forecasts represented using the FDTS:

series	method	timestamp	origin_timestamp	variable	value
Y1	ARIMA	1989	1988	forecast	5486.10
Y1	ARIMA	1990	1988	forecast	6035.21
Y1	ARIMA	1991	1988	forecast	6584.32
Y1	ARIMA	1992	1988	forecast	7133.43
Y1	ARIMA	1993	1988	forecast	7682.54
Y1	ARIMA	1994	1988	forecast	8231.65
Y2	ARIMA	1989	1988	forecast	4230.00
Y2	ARIMA	1990	1988	forecast	4230.00
Y2	ARIMA	1991	1988	forecast	4230.00
Y2	ARIMA	1992	1988	forecast	4230.00

Btw, we can expand it for rolling-origin forecasts and for CIs.

C. *Forecasts Table Schema (FTS)*

If we want our data to be easier to read, one possible format is to re-shape the FDTS in such way that each line corresponds to all the forecasting results obtained for one series using one method for one horizon and for one specified origin. The output table to store forecasting result can contain the fields shown in Table IV.

TABLE IV. FORECAST TABLE SCHEMA

series_id*	period_timestamp*	origin_timestamp*	horizon*	method*	forecast	lo95	hi95

* the key (the unique value that should not duplicated) for this table schema is <series_id, method_id, forecast_timestamp, origin_timestamp, horizon>.

This format has its advantages and disadvantages. One advantage is that it allows choosing different types for different variables (e.g., 'double' for forecasts and 'text' for method_id. However, this approach is not as flexible as the one we described earlier: when adding new types of variables (say, 'lo80' and 'hi80'), adding new columns to the table will be required.

Example:

series	method	timestamp	origin_timestamp	forecast	Lo95	Hi95
Y1	ARIMA	1989	1988	5486.10	5298.756	5673.444
Y1	ARIMA	1990	1988	6035.21	5616.295	6454.125
Y1	ARIMA	1991	1988	6584.32	5883.342	7285.298
Y1	ARIMA	1992	1988	7133.43	6107.303	8159.557
Y1	ARIMA	1993	1988	7682.54	6293.158	9071.922
Y1	ARIMA	1994	1988	8231.65	6444.500	10018.80
Y2	ARIMA	1989	1988	4230.00	2786.439	5673.561
Y2	ARIMA	1990	1988	4230.00	2188.496	6271.504
Y2	ARIMA	1991	1988	4230.00	1729.678	6730.322
Y2	ARIMA	1992	1988	4230.00	1342.877	7117.123

It is possible to make this format more flexible if some of the columns will contain a JSON or XML representation of a list of variables. E.g., we can have a column named "method params" containing an XML representation of a list of parameters.

V. EXAMPLES IN R

Here we show how we can use the new data schemas in order to easily filter/evaluate accuracy and perform exploratory data analysis.

Let's assume our data is loaded into two dataframes:

ts - time series data provided as a data frame using the Time Series Table Schema (TSTS)

fc - forecasts data provided as a data frame using the Forecasts Table Schema (FTS)

A. *Exploratory analysis of Forecasts*

a) Prediction-Realization Diagram

We can use the following code to see how forecasts correlate with actuals:

```
plotPRD( ts, fc)
```

This function produces a plotly graph shown in Fig. 1. This graph can help identify outliers and check the correctness of the data including actuals and forecasts.

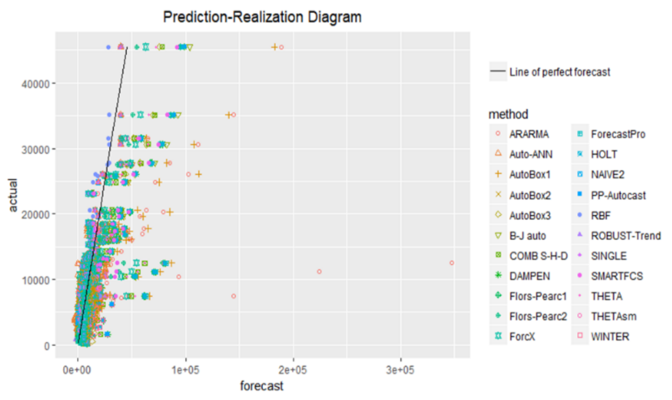

Fig. 1. Prediction-Realization Diagram of forecasts from different forecasting methods

b) Fanchart

Using this function we can see how forecasts made for a specified origin correspond to actuals (Fig. 2)

```
plotSeries(ts, fc, series_id="M1", origin="Jun 1989")
```

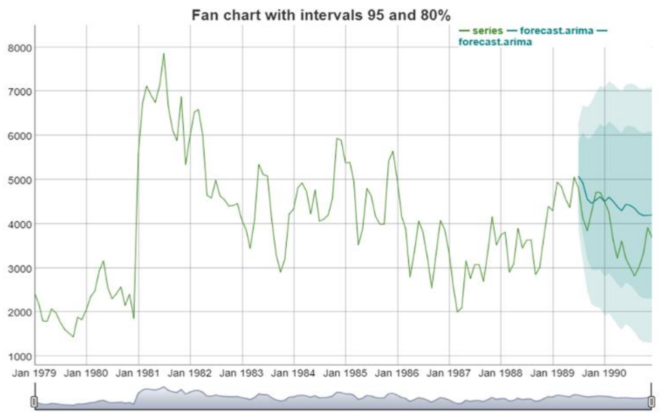

Fig. 2. Fanchart of time series with intervals 95 and 80%

B. *Accuracy*

calculateMAPEs(ts, fc)

```
$`MAPE`
              horizon = 1 horizon = 2 horizon = 3 horizon = 4 horizon = 5  horizon
AutoBox2         7.951192    18.21996    20.24227    21.65581    24.46921    27.176
ForcX            8.495870    18.75417    20.60045    22.70112    24.23433    26.428
RBF              8.146542    18.86758    21.67786    22.58918    25.20706    26.928
THETAsm          7.907310    18.26210    21.41826    23.33240    25.61775    27.892
NAIVE2           8.360053    19.23712    21.70531    23.45871    25.17578    27.351
SINGLE           8.426719    19.53460    21.70985    23.59725    25.35748    27.934
Auto-ANN         8.956602    19.67521    21.76107    24.36152    26.41399    29.817
ROBUST-Trend     7.606495    18.64720    22.39440    24.83567    27.61491    30.665
ForecastPro      8.426093    18.77205    22.10483    25.87735    27.74920    30.459
COMB S-H-D       7.964892    19.02728    22.76000    25.56244    28.63649    30.248
Flors-Pearc1     8.561016    19.38149    22.80052    25.34184    27.62398    30.955
THETA            8.172273    19.38538    22.36993    25.85993    28.69015    31.019
B-J auto         8.638050    19.71086    22.78263    26.77603    27.99026    30.821
PP-Autocast      8.141452    19.19054    22.75382    26.17481    30.09973    31.094
DAMPEN           8.161127    19.23165    22.88949    26.32286    30.25410    31.274
Flors-Pearc2    10.903332    21.38609    23.17941    24.91399    27.72512    31.299
SMARTFCS         9.796722    20.29223    23.64564    25.85210    28.55908    31.991
AutoBox3        10.698830    21.89010    25.29647    28.45540    29.57899    33.621
HOLT             8.504891    20.57738    26.74072    30.80756    34.94463    37.946
WINTER           8.504891    20.57738    26.74072    30.80756    34.94463    37.946
ARARMA           9.091266    20.68177    25.10429    30.14883    34.99774    40.380
AutoBox1        10.119198    22.51186    27.07629    31.31042    34.37756    40.084
```

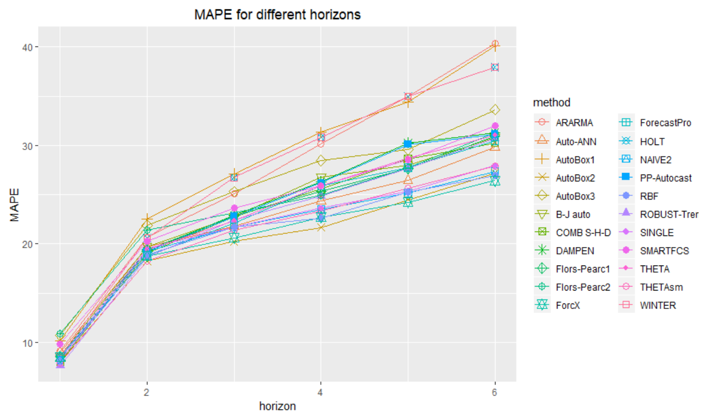

Fig. 3. Mean absolute percentage error for different forecasting methods and different horizons

C. *Validation of PIs*

validatePIs(ts, fc)

```
                   category cases total percentage  Lo 95  Hi 95
18 ETS_with_horizon = 18  1238  1428     86.695 84.823 88.414
4   ETS_with_horizon = 4  2604  3003     86.713 85.447 87.908
17 ETS_with_horizon = 17  1241  1428     86.905 85.044 88.612
6   ETS_with_horizon = 6  2615  3003     87.080 85.827 88.259
16 ETS_with_horizon = 16  1247  1428     87.325 85.487 89.007
5   ETS_with_horizon = 5  2623  3003     87.346 86.104 88.515
7   ETS_with_horizon = 7  2078  2358     88.126 86.751 89.404
15 ETS_with_horizon = 15  1261  1428     88.305 86.524 89.927
3   ETS_with_horizon = 3  2661  3003     88.611 87.421 89.726
8   ETS_with_horizon = 8  2091  2358     88.677 87.328 89.928
13 ETS_with_horizon = 13  1282  1428     89.776 88.087 91.299
1   ETS_with_horizon = 1  2709  3003     90.210 89.091 91.250
2   ETS_with_horizon = 2  2709  3003     90.210 89.091 91.250
14 ETS_with_horizon = 14  1290  1428     90.336 88.685 91.819
10 ETS_with_horizon = 10  1302  1428     91.176 89.584 92.597
12 ETS_with_horizon = 12  1325  1428     92.787 91.320 94.075
11 ETS_with_horizon = 11  1328  1428     92.997 91.548 94.266
9   ETS_with_horizon = 9  1330  1428     93.137 91.700 94.394
```

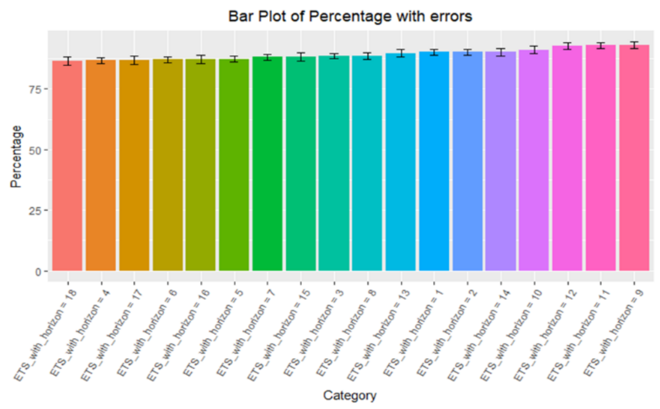

Fig. 4. Bar plot of percentage with errors for different horizons

VI. CONCLUSIONS

Having forecasting data stored in a well-defined way is crucial for monitoring and evaluating forecasting accuracy. In spite of the fact that a number of large-scale forecasting competitions have been conducted, at present there is no unified approach of how to store forecasting data. In this paper we aimed to present a data schema that is suitable for keeping forecasting data in a table as a part of an RDB or as a portable file.

We also showed how to implement various algorithms for accuracy evaluation based on the data structures proposed. We provided some examples in R, but, analogously, other existing languages (such as Python) can also be used to perform tasks such as data exploratory analysis and accuracy evaluation. Hopefully, the solutions presented will be flexible enough to be applied by academics and researchers and also by practitioners. One aim of the paper is to highlight the need of separating the forecasting data from the algorithms and tools for handling data (such as tools for viewing time series and forecasting results).

ACKNOWLEDGMENT

The reported study was supported by RFBR research projects 16-37-60066 mol_a_dk.

REFERENCES

[1] J. Scott Armstrong, “Principles of forecasting: A handbook for Researchers and Practitioners,” University of Pennsylvania, USA, 2001.

[2] http://www.moneycontrol.com/glossary/trading-terms/forecast-origin_2465.html.

[3] Shcherbakov, M. V., Brebels, A., Shcherbakova, N. L., Tyukov, A. P., Janovsky, T. A., & Kamaev, V. A. (2013). A survey of forecast error measures. World Applied Sciences Journal, 24, 171–176.

[4] https://en.wikipedia.org/wiki/Makridakis_Competitions.

[5] Rob J. Hyndman, Anne B. Koehler, “Another look at measures of forecast accuracy,” in International Journal of Forecasting, 2006, pp. 679-688.

[6] https://forecasters.org/resources/time-series-data/m-competition/

[7] Rob J. Hyndman, Yeasmin Khandakar, “Automatic Time Series Forecasting: The forecast pacakge for R,” Journal of statistical software, vol. 27, 2008.

[8] Andrey Davydenko, Robert Fildes, “Measuring forecasting accuracy: The case of judgmental adjustments to SKU-level demand forecast,” International Journal of Forecasting, 2013, pp. 510-522.

[9] Spyros Makridakis, Michele Hibon, “The M3-Competition: results, concluions and implications,” International Journal of Forecasting, 2000, pp. 451-476.

[10] Owoeye. D, M. Shcherbakov and V. Kamaev, “A photovoltaic output backcast and forecast method based on cloud cover and historical data,” In the proceedings of the The Sixth IASTED Asian Conference on Power and Energy Sysstems, 2013, pp. 28-31.