Data Formats and Visual Tools for Forecast Evaluation

Andrey Davydenko[1], Cuong Sai[2], Maxim Shcherbakov[2]

[1] *JCS CSBI, Saint-Petersburg, Russia, e-mail: andrey@live.co.uk*

[2] *CAD Department, Volgograd State Technical University, Volgograd, Russia, e-mail: svcuonghvktqs@gmail.com*

March 19, 2019

Abstract

Forecast evaluation is inevitably connected with the need to store actuals and forecasts for further analysis. The issue gets complicated when it comes to handling rolling-origin forecasts calculated for many series over multiple horizons. This setup can be met both in research (e.g., in forecasting competitions or when proposing a new method) and in practice (when tracking forecasting performance). In designing forecast data formats it is important to provide access to all the variables required for exploratory analysis and performance measurement. We show that existing approaches used to store forecast data are not always applicable. Here we propose flexible yet simple data schemas allowing the storage and exchange of actuals, forecasts, and additional variables of interest. We also demonstrate how various forecast evaluation tools can be implemented based on the schemas proposed.

Keywords: forecasting, forecast evaluation, forecasting accuracy, data visualization, R package

1. Introduction

Forecasting methods are now used in various fields ranging from weather prediction to supply chain management. The importance of accurate forecasts rises as companies are trying to become more efficient and more competitive.

In order to know how good a forecasting method is, we need to compare forecasts against corresponding actuals being obtained. In other words, we need empirical evaluation to assess forecasting performance.

Thus far, a number of forecasting competitions have been held to empirically evaluate alternative methods. Perhaps, the most famous one is the M3-Competition (Makridakis and Hibon, 2000) where 24 methods were tested using 3003 time series by applying different forecasting performance metrics. Recently, the M4-Competition was organised where all major machine learning and statistical methods were tested using 100,000 time series (Makridakis et al., 2018a, 2018b). Forecasting 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. Choosing a metric to compare alternative forecasts is itself a challenging and disputable issue that has been attracting attention of researches for quite some time (an overview of existing approaches can be found, for example, in Davydenko and Fildes, 2013). This paper will not focus on metrics, but instead we explore forecast evaluation workflow as a multistage process and look at technical issues of formatting and using forecast data.

The major issue we address is how to store data in order to facilitate reliable forecast evaluation. We show that formats used in existing datasets and packages do not provide some important capabilities (for example, by not allowing the storage of rolling-origin forecasts). After suggesting a more suitable format we illustrate how it can be used as a basis for building forecast evaluation tools.

The next section outlines a forecast evaluation setup where we summarize typical settings of obtaining and evaluating forecasts. Then we present our view of how forecast evaluation workflow should look like. We then introduce data formats that would meet the requirements. Then we demonstrate how the formats presented can be used to implement tools for forecast exploratory analysis and performance measurement. Data formats and tools presented can be used regardless of a scripting language or database, but our examples use R and plain csv-files.

2. Forecast Evaluation Setup and Terminology

This setup summarizes what kinds of data we want to work with, what we want to obtain as a result of the analysis, and under what requirements.

1. Suppose we have a set of time series possibly containing from one to a relatively large number of series (say, hundreds of thousands).
2. For each series we want to store actuals and to calculate and store forecasts. In particular, it may be needed to store out-of-sample forecasts produced from different origins (rolling-origin forecasts) over different horizons and, perhaps, produced using alternative 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. Both actuals and forecasts may be frequently updated as new data becomes available.

Storing each forecast produced for each horizon and each origin of interest is crucial because it is often impossible or very difficult to reproduce forecasts for evaluation purposes. For example, producing a single forecast can take a substantial time when using computationally intensive statistical method such as those used in MCMC algorithms to generate posterior probability densities (Yelland et al., 2010; Davydenko and Fildes, 2012). Judgmental forecasts or judgmental adjustments cannot be reproduced at all.

Given the above settings, we need convenient means to store and access (and, perhaps, to distribute or exchange) forecast data including actuals, forecasts, and other relevant information. We would like to find a means that would be fast in operation (applicable in industrial settings), cross-platform, and easy to learn and to implement.

Eventually, we need data structures to implement a reliable and informative forecast cross-validation and a credible comparison of alternative methods.

Some important terms we will use are clarified below.

- *Forecast origin* - the most recent historical period for which data is used to obtain a forecast.

- *Forecast horizon* - the number of periods from the forecast origin to the end of the time period being forecast (Armstrong, 2001).
- *Prediction intervals (PIs)* - the bounds within which future observed values are expected to fall, given a specified level of confidence (Armstrong, 2001).
- *Rolling-origin forecast* - this is a forecasting process in which the "origin" at which the forecast is based rolls forward in time (Hyndman, 2016).
- *Coverage probability* - the probability that the PI contains the actual value.
- *Nominal coverage probability* - the confidence level of the PI.

3. Forecast Evaluation Workflow

Any data science analysis involves not only applying some specific algorithm of interest, but also data preparation and data quality checks. Our approach is based on including these steps into the forecast evaluation process allowing it to comply with more general methodologies such as Microsoft TDSP, CRISP-DM, SEMMA, etc.

In this paper we consider the general workflow for forecast evaluation presented on Figure 1.

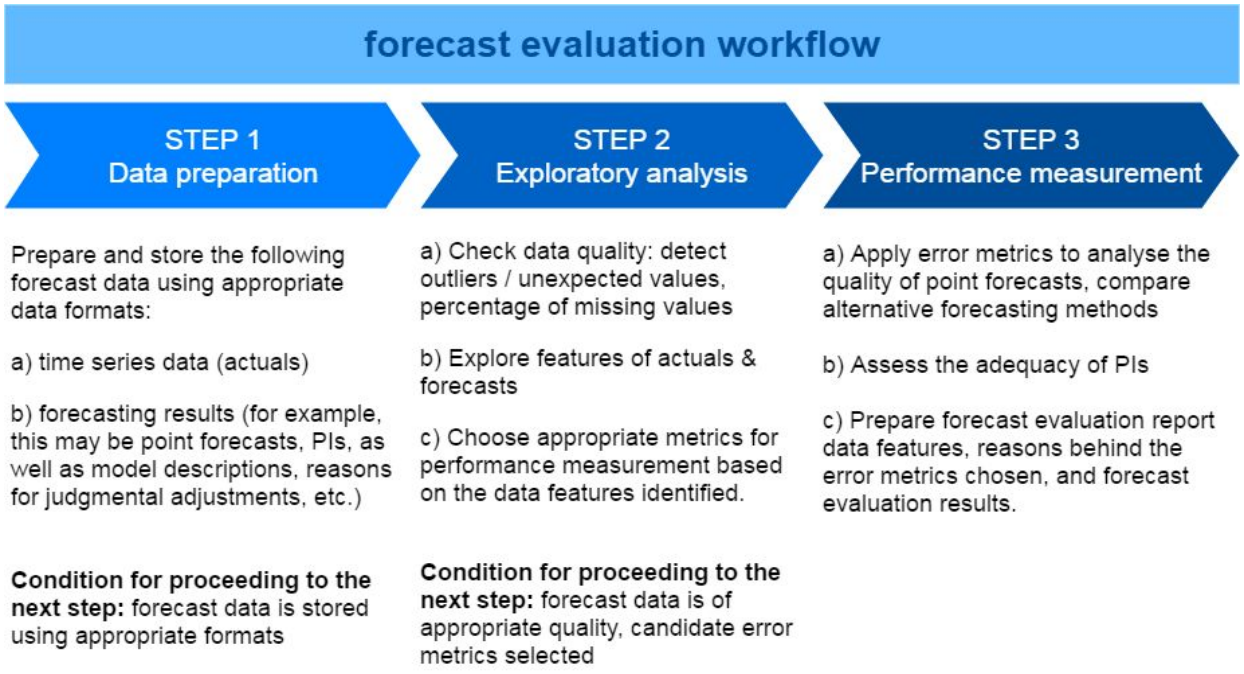

Figure 1. Forecast evaluation workflow.

The topic of how to obtain forecasts and what methods should be used is out of the scope of this paper. Instead, we look at the ways of how forecast data may be conveniently kept in order to allow access to previously submitted forecasts, to match them with actuals, and to track errors.

In fact, forecasts can be produced not only by some statistical method but also judgmentally. It is important for a forecasting system to have the capabilities of keeping all the relevant

information in order to evaluate judgmental adjustments and review and to improve the forecasting process (some examples of a GUI enabling submitting forecasts based on judgmental adjustments can be found in Davydenko and Fildes, 2008).

The next section introduces data structures allowing the implementation of the framework shown on Figure 1. Subsequent sections focus on using these structures in the exploratory analysis and performance measurement steps of the workflow.

4. New Data Formats

4.1. Existing Packages Containing Forecast Data

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

- Mcomp: Data from the M-competition and M3-competition (Hyndman, 2018a);
- Tcomp: Data from the Kaggle tourism competition (Ellis, 2018);
- tscompdata: Data from the NN3 and NN5 competitions (Hyndman, 2018b);
- M4comp2018: Data from the M4-competition (Montero-Manso et al., 2018).

The above packages use objects to store forecasts and time series actual values. The downside of this approach is that we need to use R in order to get access to data. Besides, there is no unified approach to store rolling-origin forecasts and interval forecasts.

Here we propose a general format that is based on special table schemas that can be implemented in any environment.

4.2. New Approach and Its Capabilities

This section presents our approach to store forecast data in accordance with what was said in Section 2. More specifically, we aim to have a specification with the following capabilities:

- Rolling-origin cross-validation.
- Storage of any type of forecasting results (not only point forecasts, but also interval forecasts, density forecasts, model parameters, reasons for judgmental forecasts, etc.).
- Cross-platform usage & portability.
- Ability to work with data collected for any frequency (hours, minutes, seconds, etc.) and any number of time series.
- Ease of updating actuals and forecasts, ability to store actuals separately from forecasting results, and to store forecasting results for each forecasting method separately.
- Fast access to data.
- Ease of use, understanding, and implementation.

The approach we present is based on using tables to store forecasts and actuals. This approach allows using a relational database (RDB) or portable files (e.g., .csv or Excel). As RDBs are very widely used, many companies already have an IT infrastructure for storing their data in RDB. When using database engines, SQL statements allow accessing data instantly.

Our approach is based on using the these two major table schemas:

- Time Series Table Schema (TSTS) to store time series actuals;
- Forecast Table Schema (FTS) to store forecasting results including point forecasts, prediction intervals, and other variables of interest.

Forecasts and actuals are stored in separate tables. In order to slice-and-dice forecast data easier, we may need a table containing both actuals and forecasts. This is done using Actual and Forecast Table Schema (AFTS). Such data is obtained using a simple SQL query. The AFTS format also allows you to slice-and-dice forecast data effectively.

4.3. Time Series Table Schema (TSTS)

In this schema each actual observation is stored in a table as a separate single row. The table to store such rows has the columns specified by Table 1.

Table 1. Time Series Table Schema (TSTS)

Column	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. We recommend the use of the ISO 8601 standard	"1997" in case of yearly data, "1997-01-20" in case of daily data, "1997-11" in case of monthly data, "1997-W03" in case of weekly data, "2018-Q2" in case of quarterly data
value	The value observed	100

* 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 columns in the table or an additional table specifying features of series (e.g., describing time series details, units, frequency, etc.). However, the schema specified by Table 1 includes the columns that are always necessary for forecast evaluation across many time series. In this specification we do not impose restrictions on the types of the data, but we advise that timestamps should be stored as strings.

We recommend to use the ISO 8701 (“ISO 8601”, n.d.) standard as it allows adequate sorting of rows and correct comparison of strings containing timestamps. For example, “1997-01-20” is less than “1998-01-19” but in case of using another format this may not be the case for strings: “20.01.1997” > “19.01.1998”.

Here is how the M3-Competition data can look like in the TSTS format:

series_id	value	timestamp
Y1	3103.96	1984
Y1	3360.27	1985
Y1	3807.63	1986
Y1	4387.88	1987
Y1	4936.99	1988
Y1	5379.75	1989

For missing observations the corresponding row can be omitted or the value can be denoted as NA. Sometimes it is necessary to store indications of censored data or out-of-stock events, etc., which can also be represented by additional agreements, which we will not address in this paper. The purpose of the current schema is only to set out a general approach.

4.4. Forecast Table Schema (FTS)

This schema is needed to store forecasting results. Each row contains forecasting results relating to some given time series produced using some given method for a given horizon at a given origin. Table 2 specifies the columns required. Columns containing prediction intervals can be added or excluded.

Table 2. Forecast Table Schema (FTS)

Column	**Description**	**Examples**
series_id*	Time series identifier for which the forecast was calculated	“Y1”
timestamp*	Any representation of the period to which the observation relates. We recommend the use of the ISO 8601 standard	"1997" in case of yearly data, "1997-01-20" in case of daily data, "1997-11" in case of monthly data, "1997-W03" in case of weekly data, "2018-Q2" in case of quarterly data
origin_timestamp*	Origin of the forecast (provided in the same format as the timestamp)	"2000" in case of yearly data, "1997-01-23" in case of daily data, etc.
horizon*	Forecast horizon	3
method_id*	Method identifier -- a unique name that identifies a method by which the forecasting result was produced	"ARIMA"
forecast	Point forecast	234
lo95	The lower limit for the 95% prediction interval	178
hi95	The upper limit for the 95% prediction interval	273

lo90	The lower limit for the 90% prediction interval	
hi90	The upper limit for the 90% prediction interval	
,,,	,,,	,,,

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

The FTS table may be extended by adding columns to represent additional types of forecasting results (e.g., this may be textual info to store reasons for judgmental forecasts or arrays containing density forecasts). Also, if needed, we can have columns to store structured info using JSON or XML formats. Table 2 specifies typical results used in forecast evaluation.

Here is how forecasts for the M3-Competition data can look like in the FTS format:

series_id	method_id	forecast	horizon	timestamp	origin_timestamp
Y1	NAIVE2	4936.99	1	1989	1988
Y1	NAIVE2	4936.99	2	1990	1988
Y1	NAIVE2	4936.99	3	1991	1988
Y1	NAIVE2	4936.99	4	1992	1988
Y1	NAIVE2	4936.99	5	1993	1988
Y1	NAIVE2	4936.99	6	1994	1988

4.5. Actual and Forecast Table Schema (AFTS)

It is often convenient to work with a table having both actuals and forecasts in one row. By joining columns from TSTS and FTS tables based on the TSTS key fields, the Actual and Forecast Table Schema (AFTS) is obtained.

4.6. Data Preparation Process and Scenario Examples

Since the formats proposed above (TSTS and FTS) assume a table structure, the most efficient way (in terms of updating the data and accessing relevant slices) is to store forecast data within a RDBMS. When TSTS and FTS tables are stored in a RDBMS, relevant pieces of data are obtained through SQL queries. Tables containing both actuals and forecasts (formatted using the AFTS) then can be obtained using a simple INNER JOIN SQL query.

Another scenario is to use csv files. Then actuals are stored separately from forecasts. Moreover, it is possible to store forecasts from each method in a separate file and then merge the tables for further analysis.

Of course, forecast data can be stored inside an R-package, but the idea behind our approach is still to store data as a table and not as a list of language-specific data structures.

5. The Forvision Package

The forvision package for R (Sai et al., 2019) implements tools to support some steps of the workflow shown on Figure 1. The tools are implemented assuming that forecast data is stored using the TSTS and FTS formats. Our further illustrations will be based on this package, but similar functionality and API design can be implemented in other environments (e.g., in Python).

5.1. Downloading and Installation

To install and use the package, run this code:

```
install.package(devtools)
devtools::install_github("forvis/forvision", build_vignettes = TRUE)
library(forvision)
```

5.2. Datasets Used for Illustrations

The package has several example datasets available as data frames. For further illustrations, we will use the following datasets:

- `m3_yearly_ts` - yearly actuals from the M3-competition (format: TSTS),
- `m3_yearly_fc` - yearly forecasts from the M3-competition (format: FTS),
- `m3_quaterly_ts` - quarterly actuals from the M3-competition formatted (format: TSTS),
- `m3_quaterly_fc_pis` - quarterly forecasts containing prediction intervals calculated for the M3-competition data (format: FTS).

The package has a special function to obtain a data frame containing both actuals and forecasts (a table in the AFTS format):

```
m3_yearly_af <- createAFTS(m3_yearly_ts, m3_yearly_fc)
```

5.3. Visual Tools

The forvision package implements functions to construct visual tools for exploratory analysis and forecast performance measurement. These functions produce objects created with ggplot2 (Wickham, 2016) and dygraphs (Vanderkam at. al., 2018) packages. This allows the aesthetics of graphs to be adjusted with high flexibility. The dygraph objects also allow some degree interactivity (i.e., the use of sliders for time series and showing relevant values on mouse hovering).

6. Exploratory Analysis

When forecast data is stored in the appropriate format, we can define APIs and algorithms to query, slice-and-dice, and visualise forecast data.

This section presents some exploratory tools defined using the data structures proposed above.

6.1. Prediction-Realization Diagram

The prediction-realization diagram is a scatterplot showing how forecasts correlate with actuals (Ostrom, 1990). Here we will plot point forecasts and actuals relating to different series, origins, and horizons on the same graph. The objective is to explore the distribution of errors, to identify outliers and biases, to compare alternative forecasts.

To plot the diagram we need forecast data in the AFTS format. We can use any subset of the initial data set if needed. This R-code shows the function call for constructing the diagram, Figure 2 shows the result:

```
plotPRD(m3_yearly_af)
```

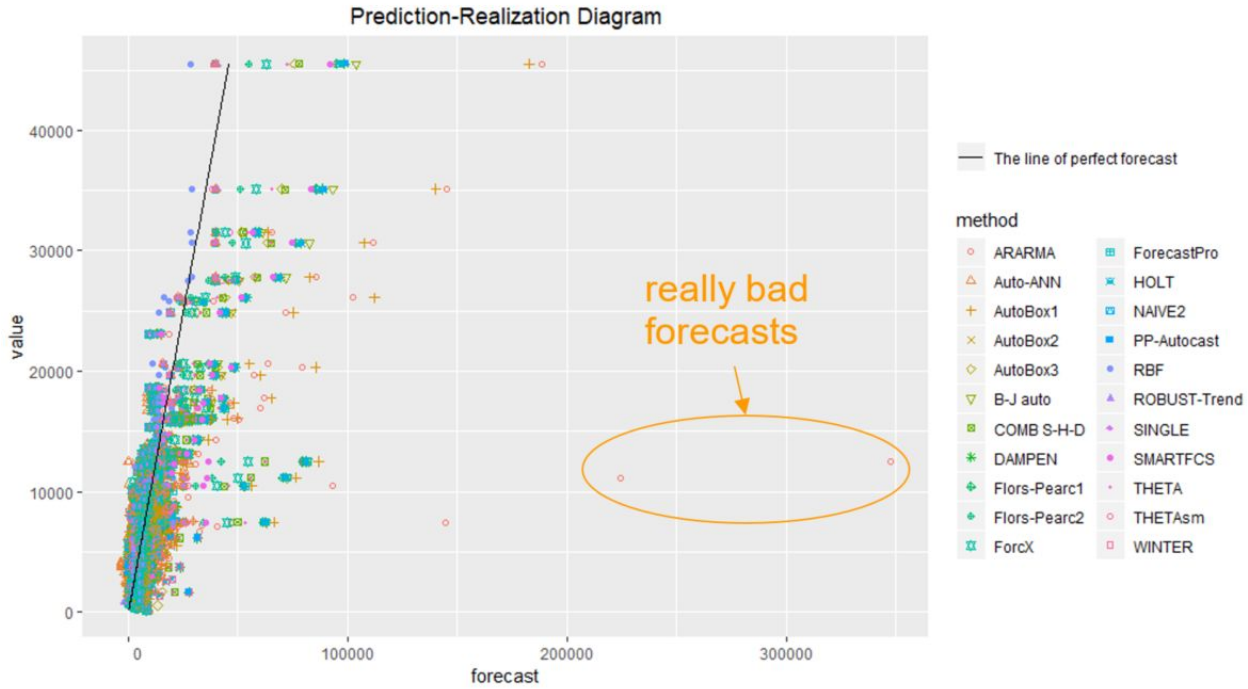

Figure 2. Prediction-realization diagram for M3 yearly data. Different colors and marks to show forecasts relating to different forecasting methods. The Y=X line represents perfect forecasts.

The graph on Figure 2 spots some really unwanted cases (when forecast seriously overestimated actuals). For example, having a forecast close to 350,000 units we had actual of only about 11,000 units. We also observe some negative forecasts, but actuals are always non-negative. Let's take a closer look at the cases spotted. With the AFTS format we can query forecast data in order to get a table with details:

```
subset(m3_yearly_af, forecast > 100000)
```

By looking at query results we found out that the unwanted cases related to series_id="Y113". Below we illustrate how to show these forecasts on a time series graph.

6.2. Fixed Origin and Fixed Horizon Graphs

The fixed origin graph shows point forecasts produced for the same time series from the same origin, but for different horizons. It is possible to show forecasts from several methods on the same graph.

When actuals table is given in the TSTS format and forecasts table in the FTS format, we can define the following algorithm for producing the fixed origin graph. Firstly, we need to select (from the forecasts table) all point forecasts produced by the methods of interest and relating to the specified origin and specified time series. Then plot a time series graph using actuals stored in the actuals table, then plot the forecasts selected, use different colors for different methods.

Figure 3 shows the cases of poor forecasts we identified based on the prediction-realization diagram (these cases relate to series Y113 of the M3 yearly data set).

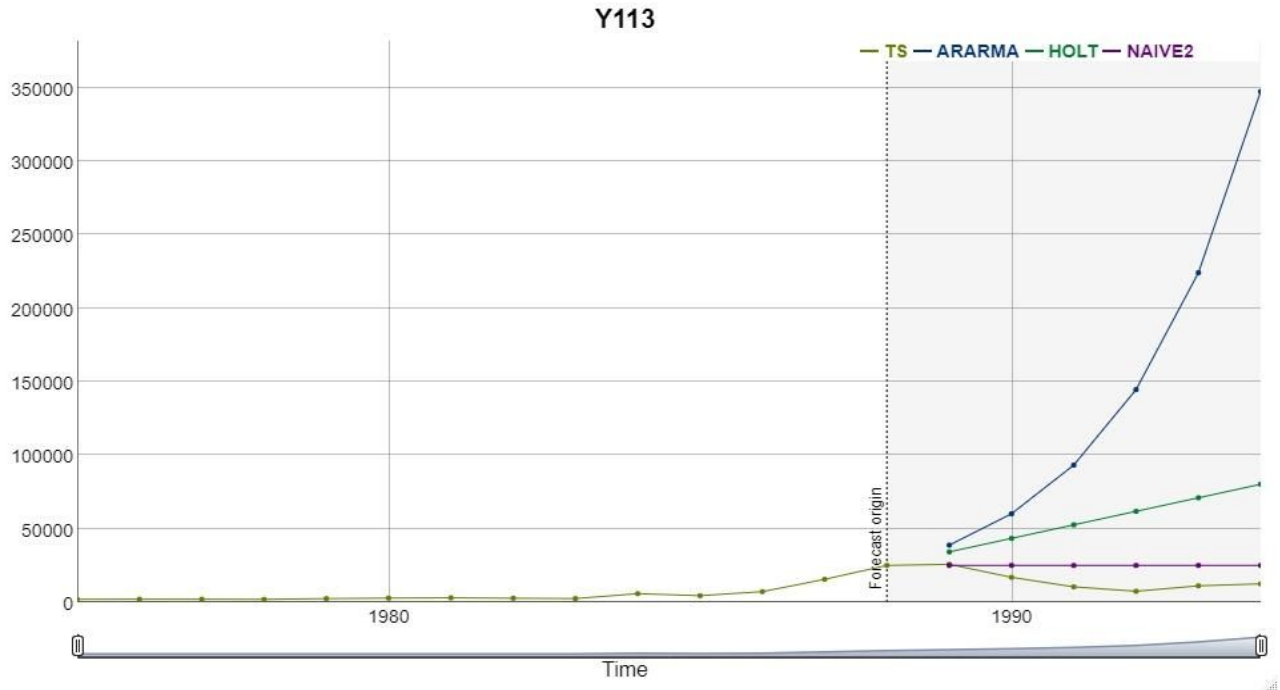

Figure 3. Fixed origin graph.

Code used to obtain the graph:

```
# First, in order visualization tools to work correctly, we must prepare
# appropriate time-based object timestamp columns:
library(zoo)
m3_yearly_ts$timestamp_dbo <- as.yearmon(m3_yearly_ts$timestamp, format = '%Y')
m3_yearly_fc$timestamp_dbo <- as.yearmon(m3_yearly_fc$timestamp, format = '%Y')

# plot fixed origin graph
plotFixedOrigin(m3_yearly_ts, m3_yearly_fc, "Y113",  1988, c("ARARMA", "HOLT",
"NAIVE2"))
```

Method "ARARMA" sometimes performs badly as it tends to extrapolate trends that do not hold. Thus, in this example the prediction-realization diagram together with the fixed origin graph helped identify the risks of using ARARMA.

If we have a data set containing rolling-origin forecasts, The fixed horizon graph can be constructed as well. The fixed horizon graph shows point forecasts produced for the same time series, with the same horizon, but from various rolling origins. The graph is constructed in a similar manner to what we described above.

6.3. Fan Charts

Fan chart shows point forecasts and prediction intervals produced for the same time series from some given fixed origin, with different horizons. Fan chart relates shows forecasts produced by only one selected method. The objectives is to visually explore PIs, to identify outliers/unexpected results, to assess the uncertainty around forecasts and adequacy of the PIs. Examples of unacceptable PIs: too wide for the practical settings, lower limit is below zero for non-negative time series, the actual coverage does not correspond to the nominal coverage, etc.

Similarly to the point forecasts graphs presented above, fan charts can be constructed given data in the TSTS and FTS formats. Figure 4 shows a fan chart produced using the forvision package.

The code below illustrates the API design based the use of the TSTS and FTS formats.

```
# prepare appropriate time-based object timestamp columns
library(zoo)
m3_quarterly_ts$timestamp_dbo <- as.yearqtr(m3_quarterly_ts$timestamp, format
= '%Y-Q%q')
m3_quarterly_fc_pis$timestamp_dbo <- as.yearqtr(m3_quarterly_fc_pis$timestamp,
format = '%Y-Q%q')

# plot a fan chart
plotFan(m3_quarterly_ts, m3_quarterly_fc_pis,  "Q1", "1992-Q4", "ARIMA")
```

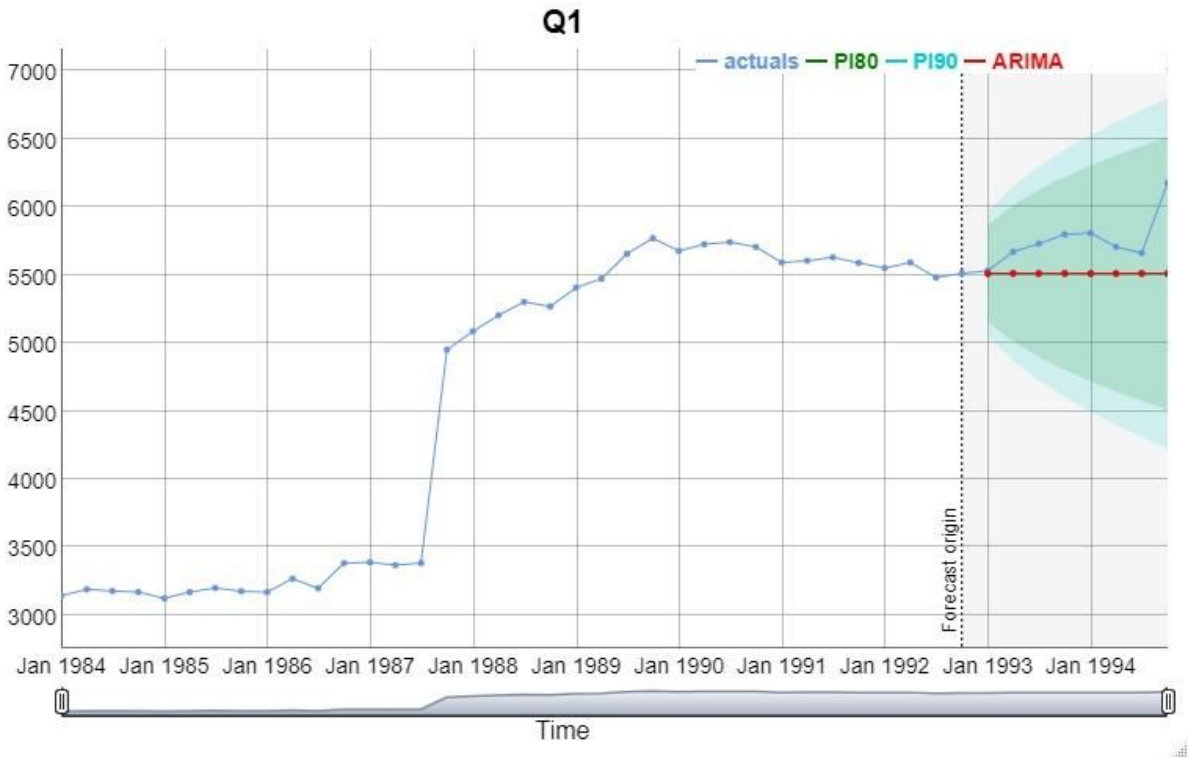

Figure 4. Fan chart.

7. Measuring Performance

When measuring forecast performance we can look at the accuracy of point forecasts and assess the adequacy of prediction intervals. In this section we show some tools that can be used to accomplish these tasks.

7.1. Assessing the Accuracy of Point Forecasts

In the exploratory analysis section we explored the distribution of actuals and forecasts for the M3 yearly data. We spotted some zero actuals and negative forecasts, which makes some popular metrics such as MAPE difficult to use. But even having only positive actuals and forecasts, MAPE can be unreliable or even misleading (Davydenko and Fildes, 2013). Nonetheless, MAPE remains very popular. So the examples below well relate to MAPE.

Having input data in the AFTS format lets us construct accuracy vs horizon graphs and tables. Fig. 5 shows the accuracy vs horizon graph for the M3 early data and the listing below shows the corresponding the R-code.

```r
# exclude inappropriate cases
m3_yearly_af2 <- subset(m3_yearly_af, value > 0 & forecast > 0)
# calculate MAPEs
acc <- calculateMAPE(m3_yearly_af2)
acc$plot
```

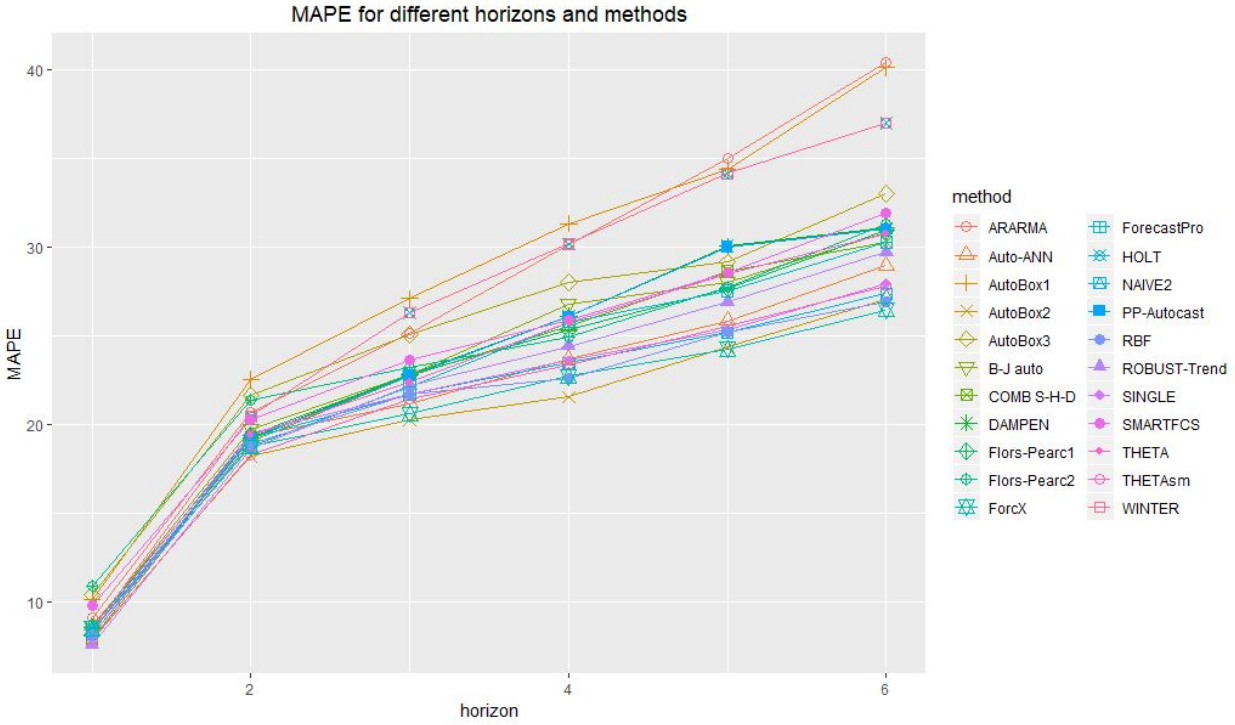

Figure 5. Accuracy vs horizon graph.

7.2. Validation of Prediction Intervals

Assessing the adequacy of prediction intervals is based on calculating coverage probabilities and comparing them with the nominal probability. If the confidence limits for the coverage probability are to wide, we can conclude that there's not enough observations to draw conclusions about the validity of PIs. Ideally, the confidence limits should be relatively narrow and include the nominal coverage. Surprisingly, very little research has been conducted in the area of validating prediction intervals for well-known methods. Perhaps, most well-known attempt is (Athanasopoulos, et al., 2011), but it still did not provide evidence on confidence bounds for empirical coverage. Here we propose a visual tool showing both empirical coverage and corresponding error bounds.

Using the AFTS format we can implement the coverage chart shown on Figure 6. The corresponding code is:

```r
# show coverage chart for ARIMA forecast method
m3_quarterly_af <- createAFTS(m3_quarterly_ts, m3_quarterly_fc_pis)
plotCoverage(m3_quarterly_af, pi = 90, methods = "ARIMA")
```

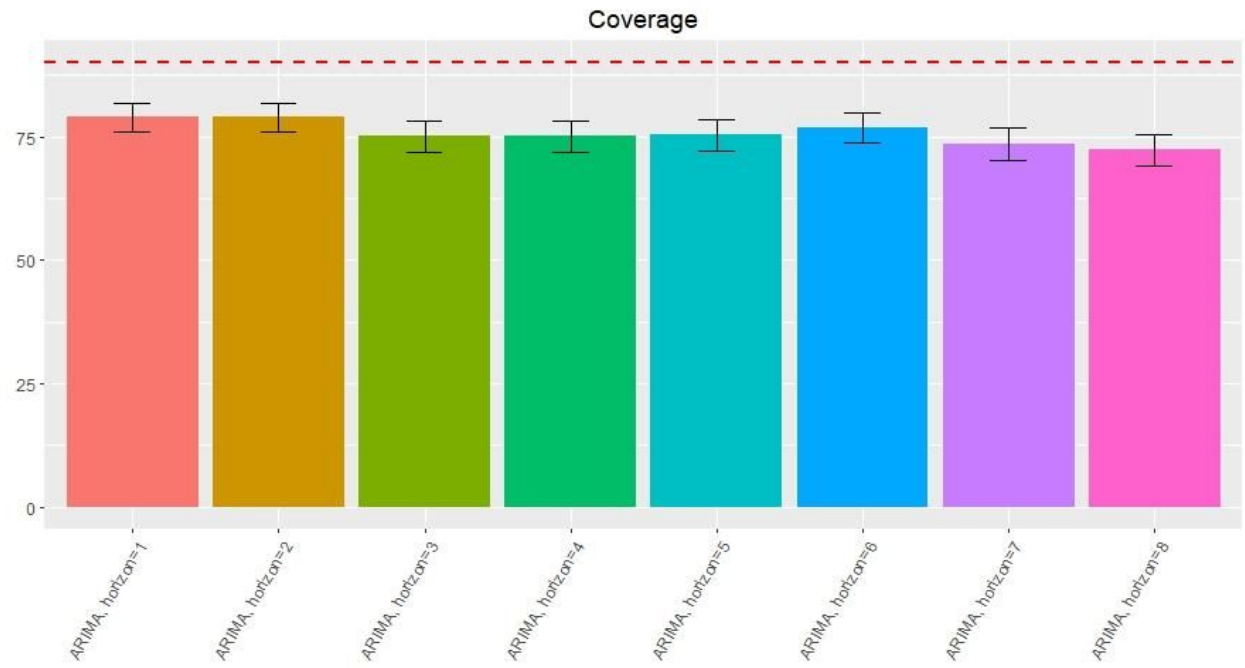

Figure 6. Coverage diagram. Error bars indicate 90% confidence intervals for coverage probabilities. Dashed line shows nominal coverage probability.

By looking at the coverage chart on Figure 6, it can be seen that the method (we used auto.arima function from the 'forecast' R-package to obtain PIs) tends to underestimate the uncertainty associated with the forecasts produced.

8. 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 a 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, any other language (e.g., Python) can be used to perform exploratory analysis and accuracy evaluation. Hopefully, the approach presented is flexible enough to be applied by academics/researchers, and practitioners. One important aim of the paper was to highlight the need to separate forecast data itself from the algorithms and tools used for data analysis.

References

Armstrong, J. S. (2001). *Principles of forecasting: a handbook for researchers and practitioners*. Boston, MA: Kluwer Academic.

Athanasopoulos, G., Hyndman, R. J., Song, H., & Wu, D. C. (2011). The tourism forecasting competition. *International Journal of Forecasting*, 27(3), 822-844.

Davydenko, A., & Fildes, R. (2012). *A joint Bayesian forecasting model of judgment and observed data* (LUMS Working Paper 2012:4). Lancaster University.

Davydenko, A., & Fildes, R. (2008). *Models for product demand forecasting with the use of judgmental adjustments to statistical forecasts*. Paper presented at the 28th international symposium on forecasting (ISF2008), Nice. Retrieved on 20 Sep 2018 from https://pdfs.semanticscholar.org/eb4e/393e629b803d6802a823825a0b97c9963116.pdf

Davydenko, A., & Fildes, R. (2013). Measuring forecasting accuracy: The case of judgmental adjustments to SKU-level demand forecasts. *International Journal of Forecasting*, 29(3), 510-522.

Ellis, P. (2018). *Tcomp: Data from the 2010 Tourism Forecasting Competition*. R package version 1.0.1. URL: CRAN.R-project.org/package=Tcomp

Hyndman, R. (2016, December 5). *Cross-validation for time series* [Blog post]. Retrieved from https://robjhyndman.com/hyndsight/tscv/

Hyndman, R. (2018a). *Mcomp: Data from the M-Competitions*. R package version 2.8. URL: CRAN.R-project.org/package=Mcomp

Hyndman, R. (2018b). *tscompdata: Time series data from various forecasting competitions*. R package version 0.0.1. URL: github.com/robjhyndman/tscompdata

Makridakis, S., & Hibon, M. (2000). The M3-Competition: results, conclusions and implications. *International journal of forecasting*, 16(4), 451-476.

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

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018b). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PloS One*, 13(3), 1-26.

Montero-Manso, P., Netto, C., & Talagala, T. (2018). *M4comp2018: Data from the M4-Competition*. R package version 0.1.0. URL: github.com/carlanetto/M4comp2018

Ostrom, C. W. (1990). *Time series analysis: Regression techniques.* Newbury Park, CA: Sage.

Sai, C., Davydenko, A., & Shcherbakov, M. (2019). *Forvision: Tools for forecast visualisation and evaluation*. R package version 0.0.1. URL: github.com/forvis/forvision

Vanderkam, D., Allaire, J., Owen, J., Gromer, D., & Thieurmel, B. (2018). *dygraphs: Interface to "Dygraphs" Interactive Time Series Charting Library*. R package version 1.1.1.6. URL: CRAN.R-project.org/package=dygraphs

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag, New York

Wikipedia contributors. (2019, March 1). ISO 8601. In *Wikipedia, The Free Encyclopedia*. Retrieved March 14, 2019, from https://en.wikipedia.org/w/index.php?title=ISO_8601&oldid=885599346

Yelland, P. M., Kim, S., & Stratulate, R. (2010). A Bayesian model for sales forecasting at Sun Microsystems. *Interfaces*, 40(2), 118-129.