MONASH
BUSINESS
SCHOOL

ISSN 1440-771X

Department of Econometrics and Business Statistics

http://business.monash.edu/econometrics-and-business-statistics/research/publications

Tidy data structure to support exploration and modeling of temporal-context data

Earo Wang, Dianne Cook, Rob J Hyndman

June 2018

Working Paper no/yr

Tidy data structure to support exploration and modeling of temporal-context data

Earo Wang
Department of Econometrics and Business Statistics,
Monash University, VIC 3800
Australia.
Email: earo.wang@monash.edu
Corresponding author

Dianne Cook
Department of Econometrics and Business Statistics,
Monash University, VIC 3800
Australia.
Email: dicook@monash.edu

Rob J Hyndman
Department of Econometrics and Business Statistics,
Monash University, VIC 3800
Australia.
Email: rob.hyndman@monash.edu

12 June 2018

JEL classification: C10,C14,C22

Tidy data structure to support exploration and modeling of temporal-context data

Abstract

Temporal-context data is often rich with information and time formats, for example multiple observational units, different time lengths, heterogeneous data types, nested and crossed factors and etc. This work presents a cohesive and conceptual framework for organizing and manipulating temporal data, which in turn flows into visualization and forecasting routines. Tidy data principles are applied and extended to temporal data: (1) mapping the semantics of a dataset into its physical layout, (2) an explicitly declared index variable representing time, (3) a "key" comprised of single or multiple variables to uniquely identify units over time, using a syntactical and user-oriented approach in which it imposes nested or crossed structures on the data. This tidy data representation most naturally supports thinking of operations on the data as building blocks, forming part of a "data pipeline" in time-based context. A sound pipeline practice facilitates a succinct and transparent workflow for analyzing temporal data. Applications are included to illustrate tidy temporal data structure, data pipeline ideas and usage. The architecture of tidy temporal data has been implemented in the R package **tsibble**.

Keywords: temporal context, time series, data structure, R

1 Introduction

Temporal-context data consist of observational units indexed at different time points X_{jt}, where the j^{th} unit takes measurements of X on over time t, for $j = 1, \ldots, N_i$ and $1 \leq t \leq T$. Time primarily forms the contextual basis of temporal data, but it could arrive in many possible formats. For example, data recorded at fine time resolutions (hours, minutes, and seconds) are typically associated with different time zones and daylight savings. Temporal data often carries with rich information other than the time: multiple observational units of different time lengths, multiple and heterogeneous measured variables, multiple grouping factors involving nested or crossed structures, linking to other data tables, and etc.

In the literature, time series and panel (longitudinal) data are common terms referred to temporal-context data, depending on the research fields. Researchers who are concerned with modelling large T and small N would name as "time series" (serial correlation); those who are interested in modelling small T and large N as "panel data" (asymptotic). The data format is two-dimensional array, but different modelling focuses lend the data input to different representations. A matrix is used to represent multivariate time series where each row represents observations measured at a time point and each column represents a series ("wide" form). This matrix representation requires homogeneity (that is, all the columns must be of same type.), time indices implicitly inferred as attributes or meta-information, series of same time length, and explicit missingness. By contrast, panel data are organised in rectangular form of heterogeneous column types where multiple study subjects are stacked and repeated for its time indices in a single column ("long" form), due to commonly arisen unbalanced panels. This specification requires explicitly declared panel variable and index, which has been implemented in Stata's time series module and R package **plm** (Croissant, Millo & Tappe 2017). Evidently, this data organisation appears more flexible than matrix in supporting explicit time index, multiple subjects of different time lengths, and implicit missing values. [R]

Temporal data can often be aggregated in a manner that exhibits a nested or crossed structure, also known as hierarchical or grouped time series (Hyndman & Athanasopoulos 2017). For example, in a manufacturing setting, a company can add up every store's sales by region, by state and by country, which gives a strictly hierarchical time series; alternatively they can group the sales for each product together based on common attributes such as store, brand, price range and so forth, which leads to a non-hierarchical structure—a grouped time series. The R package **hts** (Hyndman et al. 2018) is the implementation of this type of time series. But it is frustrating to create and work with such data objects due to non-extensible and non-modified. Variational data aggregation (some groups no need aggregation). [R]

Wickham (2014) coined the term "tidy data" and formalized the processing from messy data to "tidy data". These principles attempt to standardize the mapping from the semantics of a dataset to its structure and facilitate data analysis in a coherent way. Based on the systematic structuring principles, the grammar of data manipulation, as implemented in the R package **dplyr** (Wickham et al. 2018), abstracts an coherent and consistent set of methods to handle a wide range of data transformation tasks. Wickham & Grolemund (2016) argues that 80% of data analysis tasks can be solved with tidy tools while the remaining 20% requires other tools. [R]

This paper proposes a unified data representation of temporal-context data, blending time series of nested and crossed factors into a two-dimensional column-homogeneous array in a long format. By leveraging the "tidy data" principles, observations and variables position and bridge their meanings in both physical and internal structures. Data manipulation involves in transforming either observations or variables, or both, which can be described and achieved with a collection of shorthand operators. A chain of data transformations lend itself to a data pipeline.

The rest of the paper is structured as follows.

2 Data semantics

The choice of representation of temporal-context data is made from a data-centric perspective, which is taken in the light of the operations that are to be performed on the data. This data abstraction reflects most common problems of incoming data and transformation in reality. Firstly, a data set must be structured in a "tidy" rectangular layout each row has the same schema and each field has a single value. Secondly, declaring the data set to contain temporal observations is determined by an "index" that represent time and a "key" that uniquely identifies each unit that measurements take place on over time. The "key" works similarly as the panel variable in the Stata's **tsset** command to define the units or subjects, but it is expanded to include multiple variables rather than a single one. A syntax is introduced to express a key consisting of nested and crossed factors. The composition of index and key uniquely defines each observation in a data table, which is equivalent to a primary key (Codd 1970) in a relational database.

Given the nature of temporal ordering, a temporal data set must be sorted by time index. If a key is explicitly declared, the key will be sorted first and followed by arranging time in ascending order.

This high-level data abstraction is semantically structured, which shed lights on time index and what we call "key".

2.1 Time index and interval

Time forms an integral component and a contextual basis of temporal data. A variable representing time needs explicitly declared at the process of constructing a temporal data, referred to as "index". This accessibility of index promotes transparency and unambiguity while manipulating time. For example, subsetting a certain time period of data, extracting time components (like

Table 1: *A subset of estimates of tuberculosis burden generated by World Health Organisation in 2011 and 2012.*

country	continent	gender	year	count
Australia	Oceania	Female	2011	120
Australia	Oceania	Female	2012	125
Australia	Oceania	Male	2011	176
Australia	Oceania	Male	2012	161
New Zealand	Oceania	Female	2011	36
New Zealand	Oceania	Female	2012	23
New Zealand	Oceania	Male	2011	47
New Zealand	Oceania	Male	2012	42
United States of America	Americas	Female	2011	1170
United States of America	Americas	Female	2012	1158
United States of America	Americas	Male	2011	2489
United States of America	Americas	Male	2012	2380

time of day and day of week), or converting time zones are directly dealt with index. It is also often to join other data tables based on the common time indices. Tools are provided to work with time itself, rather than wrappers to work with a whole data set. This promotes transparency and visibility to make inspection easier. This sets the tone on methods design philosophy: well-structured, expressive and unambiguous workflow.

For data indexed in regular time space, the time interval is obtained by computing the greatest common divisor based on the non-negative differences of a given time index, although the data may include implicit time gaps. This implies that all the units in the table must share a common interval. It is relatively easy to determine if the data is daily or sub-daily since most statistical packages have built-in time representations for date-times and dates, for example their corresponding classes `POSIXct` and `Date` in R. However these turn out inappropriate for weekly, monthly and quarterly data. For example, the smallest time unit of monthly data is months in a year, which is equispaced points in a year. However, using the first day of the month can range from 28 to 31 days, which is irregularly spaced. A genuine year-month representation should avoid associating days with its format. Displaying the year and month components suffice to signal its monthly aggregated values instead of observed values at a particular timestamp.

Extensible. Custom index classes can be supported.

2.2 Key

The "key" identifies units or subjects that are observed over time in a data table, which are typically known in advance by users. The absence of key only occurs to a univariate case where the key is implicit. Multiple units must associate with an explicit "key" to allow for identification. The key is not constrained to a single column, but is comprised of multiple columns. When involving multiple variables, one need to distinguish nesting and crossing data variables. "Nesting" refers to a scenario where a variable is nested within another when each category of the former variable co-occurs with only one category of the latter; on the other hand, "crossing" means that a variable is crossed with another when every category of one variable co-occurs with every category of the other. Nesting is a special case of crossing. The former exhibits a hierarchical ordering with lower-level category (children) nested within higher-level category (parent); whereas the ordering of the latter makes no difference. Whilst constructing tidy temporal data, users also need to specify the key that identifies the structure on the data. Rather than creating a separate object that contains the data structure, like what the **hts** does, we propose a syntax-based interface to directly deal with variables, so that it takes advantage of data semantics. The expression of a vertical bar (`|`) is used to indicate nesting variables, while a comma (`,`) for crossing variables. Wilkinson (2005) discussed the distinction between nesting and crossing with respect to facets in graphics, and he used `/` and $*$ instead for nested and crossed expressions. A crossing structure displays all unique and possible combinations of crossed variables, including those not found in the data, in a graphical layout. However, tidy temporal data finds only the combinations that occur in the data and intentionally respects the structure of incomplete combinations.

3 Data pipeline

This representation choice will prove suitable for almost all conceivable applications of temporal data. But the data abstraction cannot live alone with an additional toolchain in order to be able to acquire insight and knowledge about the data itself. A desirable toolbox chains transformation, visualisation, and modelling as a data science pipeline. Each component itself is also a self-contained pipeline that can be decomposed into many individual operations. This paper will focus on part of the general pipeline—data transformation, for example filtering observations, selecting and summarising variables. We shall see how this data abstraction breeds a consistent data process.

3.1 Literature review

There has been a long history of pipeline discussions and implementation centering around data in various aspects. The Extract, Transform, and Load (ETL) process is most commonly deployed pipeline in data science. However, building a sound data pipeline can be technically difficult, which involves many implementation decisions, including interface, input/output, functionality and so on, to be made.

Doug McIlroy (1978) coined the term "pipelines" in software development, while developing Unix at Bell Labs. In Unix-based computer operating systems, a pipeline chains together a series of operations on the basis of their standard streams, so that the output of each programme becomes the input to another. This shapes the Unix toolbox philosophy: "each do one simple thing, do it well, and most importantly, work well with each other" (Raymond 2003).

Andreas Buja & McDonald (1988) proposed a viewing pipeline for data analysis in interactive statistical graphics and generalised a collection of elements required for the pipeline, which takes control of transformation from data to plot. Wickham et al. (2010) and Xie, Hofmann & Cheng (2014) implemented the data pipeline in interactive statistical software **plumbr** and **cranvas** respectively, using a reactive programming framework, in which user's interactions trigger a sequence of modules to update views accordingly.

3.2 Time-based pipeline analysis

What is notable about data pipeline is that it not only frees users from a tedious and error-prone analysis but also empowers them to focus on the statistical analysis tasks at hand without concerning the details of computational implementation. A fluent and coherent pipeline glues the grammar of data manipulation and tidy data as a fundamental unit of data analysis together. It helps (1) break up a big problem to into manageable blocks, (2) generate human readable analysis workflow, (3) avoid introducing mistakes as many as possible.

An individual operation that performs on a specific task can be phrased as a verb. This verb is self-explanatory enough to express what the specific task is. We shall apply **dplyr** key verbs to time domain and expand its vocabulary a little more to handle time-related analysis problems.

- **row-wise**: `filter()`, `slice()`, `arrange()`, `fill_na()`
- **column-wise**: `mutate()`, `select()`, `summarise()`
- **group-wise**: `index_by()`, `group_by()`

- **time-wise**: `lead()`, `lag()`, `difference()`
- **rolling window**: `slide()`, `tile()`, `stretch()`

- "No matter how complex and polished the individual operations are, it is often the quality of the glue that most directly determines the power of the system" (Friedman & Wand 2008)

- The resulting code is well-structured, easier to read and comprehend. A way of chaining together simple operations to perform complex tasks in an efficient way.

Individual operations can talk to each other.

A pipeline exhibits a hierarchy of data operations: (1) atomic (1-dimensional) vectors (`mean(variable)`) –> (2) (2-dimensional) data table (`summarise()`) –> repeat step (1) and (2) to form a chain. Reversely, a data pipeline is decomposed into rectangular blocks, and then into atomic strips.

The index and key will automate the update.

4 Application: U.S.A domestic flights on-time performance (2016-2017)

A dataset of on-time performance of domestic flights in U.S.A from 2016 to 2017 is studied and explored for illustration of tidy data and data pipeline.

5 Conclusion and future work

A tidy representation of time series data, and data pipelines to facilitate data analysis flow have been proposed and discussed. It can be noted that tidy temporal data gains greater flexibility in keeping data richness, making data transformation and visualisation easily. A set of verbs provides a fluent and fluid pipeline to work with tidy time series data in various ways.

The ground of time series modelling or forecasting is left untouched in this paper. The future plan is to bridge the gap between tidy data and model building. Currently, it is required to casting to matrix from tidy data and therefore building a model. But time series models should be directly applied to tidy data as other wrangling tools do, without such an intermediate step. In particular, a univariate time series model, like arima and exponential smoothing, can be

applied to multiple time series independently. A tidy format to represent model summaries and forecasting objects will be developed and implemented in the future. Model summaries include coefficients, fitted values, and residuals; forecasting objects include future time path and distributions generating prediction intervals.

References

Andreas Buja Daniel Asimov, CH & JA McDonald (1988). "Elements of a Viewing Pipeline for Data Analysis". In: *Dynamic Graphics for Statistics*. Ed. by WS Cleveland & ME McGill. Belmont, California: Wadsworth, Inc.

Codd, EF (1970). A relational model of data for large shared data banks. *Communications of the ACM* **13**(6), 377–387.

Croissant, Y, G Millo & K Tappe (2017). *plm: Linear Models for Panel Data*. R package version 1.6-6. https://CRAN.R-project.org/package=plm.

Doug McIlroy E. N. Pinson, BAT (1978). Unix Time-Sharing System Forward. *The Bell System Technical Journal*, 1902–1903.

Friedman, DP & M Wand (2008). *Essentials of Programming Languages, 3rd Edition*. 3rd ed. The MIT Press.

Hyndman, RJ & G Athanasopoulos (2017). *Forecasting: Principles and Practice*. Melbourne, Australia: OTexts. OTexts.org/fpp2.

Hyndman, R, A Lee, E Wang & S Wickramasuriya (2018). *hts: Hierarchical and Grouped Time Series*. R package version 5.1.5. https://CRAN.R-project.org/package=hts.

Raymond, ES (2003). *Basics of the Unix Philosophy: Chapter 1. Philosophy*. http://www.faqs.org/docs/artu/ch01s06.html.

Wickham, H (2014). Tidy Data. *Journal of Statistical Software* **59**(10), 1–23.

Wickham, H, R François, L Henry & K Müller (2018). *dplyr: A Grammar of Data Manipulation*. R package version 0.7.5. https://CRAN.R-project.org/package=dplyr.

Wickham, H & G Grolemund (2016). *R for Data Science*. O'Reilly Media. http://r4ds.had.co.nz/.

Wickham, H, M Lawrence, D Cook, A Buja, H Hofmann & DF Swayne (2010). The Plumbing of Interactive Graphics. *Computational Statistics*, 1–7.

Wilkinson, L (2005). *The Grammar of Graphics (Statistics and Computing)*. Secaucus, NJ: Springer-Verlag New York, Inc.

Xie, Y, H Hofmann & X Cheng (2014). Reactive Programming for Interactive Graphics. *Statistical Science* **29**(2), 201–213.