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

November 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

1 November 2018

JEL classification: C10,C14,C22

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

Abstract

Mining temporal-context data for information is often inhibited by a multitude of time formats: irregular or multiple time intervals, multiple observational units or repeated measurements on multiple individuals, heterogeneous data types, nested and crossed factors indicating hierarchical sub-groups. Time series models, in particular, the software supporting time series forecasting makes strict assumptions on data to be provided, typically a matrix of numeric data with an implicit time index. Going from raw data to model-ready data is painful. 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) including an explicitly declared index variable representing time, (3) incorporating a "key" comprised of single or multiple variables to uniquely identify units over time, using a syntax-based 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 data pipeline facilitates a fluent and transparent workflow for analyzing temporal data. Applications are included to illustrate tidy temporal data structure, data pipeline structure and usage. The infrastructure of tidy temporal data has been implemented in the R package **tsibble**.

Keywords: temporal data, time series, data structures, data wrangling, tidy data, R, forecasting, data science, exploratory data analysis

1 Introduction

Temporal-context data, X_{jt}, consist of $j = 1, \ldots, N_i$ observational units indexed at different time points, $1 \leq t \leq T$. Time forms the contextual basis of temporal data, but it can arrive in many possible formats, which makes preparing it for modeling and forecasting painful. For example,

data can be recorded at various time resolutions (hours, minutes, and seconds), and they are typically associated with different time zones with adjustments like summer time. It could be irregularly recorded, which is particularly true with longitudinal measurements like patient visits to a doctor's office. Temporal data also often contains rich information in addition to time: multiple observational units of different time lengths, multiple and heterogeneous measured variables, multiple grouping factors involving nested or crossed structures. The data problems are grouped into two types of analysis, time series and longitudinal.

1.1 Time series and longitudinal data

Despite of exactly the same data input, the representation of time series and longitudinal data diverges due to different modelling approaches.

Time series can be univariate or multivariate, and for modelling requires relatively large in length (i.e. large T). Time series researchers and analysts who are concerned with this large T property, are mostly concerned with stochastic processes, for the primary purpose of forecasting, and characterising temporal dynamics. The time series supporting modeling are represented as vectors or matrices in most statistical software, with the standards being provided by the R (R Core Team 2018) packages `ts`, **zoo** (Zeileis & Grothendieck 2005), and **xts** (Ryan & Ulrich 2018) (*Why include MATLAB (The MathWorks, Inc. 2017)??*). Multivariate time series are typically assumed to be in the format where each row is assumed to hold observations at a time point and each column to contain a single time series. (The tidy data name for this would be **wide format**.) This implies that data are columns of homogeneous types: numeric or non-numeric, but there are limited supporting methods for non-numeric variables. In addition, time indexes are stripped off the data and implicitly inferred as attributes or meta-information. It strictly requires that the number of observations must be the same across all the series. Data wrangling from the form that data arrives in, to this specialist format can be frustrating and difficult, inhibiting the variety of downstream tasks such as analytics.

For longitudinal analysis, researchers and analysts are primarily interested in explaining trends across and variations among individuals, and making inference about a broader population. Longitudinal data or panel data typically assumes fewer measurements (small T) over a large number of individuals (large N). It often occurs that measurements for individuals are taken at different time points, resulting in an unbalanced panel. Thus, the primary format required for modeling is stacked series, blocks of measurements for each individual, with columns indicating individual, time of measurement and the measurements. (The tidy data name for this would

be **long format**.) Evidently, this data organisation saves storage space for many sparse cells, compared to structuring it in that wide format which would have missing values in many cells. A detriment of this format is that demographic information for subjects is often repeated for each time point. However, appealing feature is that data is structured in a concise and semantic manner with reference to observations and variables, with the time index stated explicitly. This opens the door to easily operating on time to make calculations and extract different temporal components, such as month and day of the week. It is conducive to examining the data in many different ways and leading to more comprehensive exploration and forecasting.

Longitudinal data representation has been implemented in Stata's time series module (StataCorp 2017) and R package **plm** (Croissant & Millo 2008). The underpinning data structure is a two-dimensional column-homogeneous array, as other tabular data. Specifying a longitudinal data set from a generic array needs explicitly declaring time and individuals (or panel variable in Stata's `tsset` command). Therefore, each row of the data can be identified by a specific individual and time point. Individuals, however, can be only declared through a single variable, not multiple.

1.2 Time series of nested and crossed factors

A collection of time series can often be intrinsically arranged along multivariate data of nested and crossed factors, which is also known as hierarchical and grouped time series (Hyndman & Athanasopoulos 2017). For example, from an operational point of view, a manufacturer can add up every store's sales by region, by state, and by country along a geographical tree; alternatively, one can group the sales for each product together based on common attributes such as store, brand, price range and so forth, resulting in a non-hierarchical structure. The former gives an example of hierarchical time series, and the latter shows grouped time series. And even when collections of time series are absent from hierarchical structure, it is possible to arrange them empirically into a hierarchy such as regression tree (Sankaran & Holmes 2017).

The R package **hts** (Hyndman et al. 2018) is the implementation of structured collections of time series, and provides tools for visualizing, analyzing, and forecasting such data. But it is a pain to configure node list or grouping matrix as the increase in hierarchic depth and grouping factors. This is resulted from the fact that the structure specification is separated from data itself and represented numerically, which is rather counterintuitive from the user's perspective.

1.3 Tidy data and the grammar of data manipulation

Wickham (2014) coined the term "tidy data", which is a rephrasing of the second and third normal forms in relational databases but in a way that makes more sense to data scientists. A suite of paired verbs such as `gather` and `spread` (Wickham & Henry 2018) were used to describe the reshape processing from messy data to tidy data. Tidy data, which standardized the mapping from the semantics of a dataset to its physical representation, serves remarkably as a fundamental unit of data analysis. R package **ggplot2** (Wickham 2009) pioneered developing the grammar of graphics based on tidy data, thus making it possible mapping from data space to visual elements. **dplyr** (Wickham et al. 2018) generalised a coherent and consistent set of verbs 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 represents time and a "key" that uniquely identifies each unit that measurements take place on over time. The "key" serves a similar purpose 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. (The matrix representation also implies that every observation is determined by column variable and time-indexed row.)

Table 1 shows a "tidy" temporal data example using a subset of tuberculosis cases estimated by World Health Organization (2018). It contains 12 observations and 5 variables arranged in a "long" tabular form. Each observation uniquely records the number of people, who are diagnosed tuberculosis for each gender at three selected countries in the years of 2011 and 2012, where column `year` is declared as the index variable, columns `country`, `continent` and `gender` as key. To be more specific, the key is passed via `id(country | continent, gender)`, which produces a geographic hierarchy of country nested within continent, crossed with a demographic factor. Detailed discussions about index and key specifications will be given in Section 2.1 and 2.2. Although column `count` is the only measure in this case, it is sufficiently flexible to hold other measured variables, for example, adding the corresponding population size (if known) in order to calibrate the count later.

Given the nature of temporal ordering, a temporal data set must be sorted by time index as a result. If a key is explicitly declared, the key will be sorted first depending on the ordering of factor levels and the ordering of variables, and followed by arranging time in past-to-future order. The tuberculosis example shows the key is sorted in alphabetic order, followed by ascending years by default, as in Table 1. Since time-based operations, for example leading, lagging, or differencing series, require a vector in its time order, the arranged data format assures its validity for these operations without returning unexpected results. There are other possible arrangements in the key, but the baseline is the index (i.e. `year`) ascends within each key element. If any operations on the tidy temporal data results in a deviation from this baseline, it will issue a warning to the users.

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

Table 1: *A small subset of estimates of tuberculosis burden generated by World Health Organisation in 2011 and 2012, with 12 observations and 5 variables. The index refers to column* `year`*, the key to multiple columns:* `contry` *nested under* `continent` *crossed with* `gender`*, and the measured variable to column* `count`*.*

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. For example, subsetting a certain time period of data, extracting time components (like 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. This argument is also applicable to the creation of year-week and year-quarter. (coercion)

Since Table 1 possesses complete observations for each category, the interval obtained from the greatest common factor is 1 year.

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 is typically a priori known structure 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 (for example, `country` nested within `continent`); 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 (for example, `country` crosses with `gender`). Nesting is a special case of crossing. The former exhibits a hierarchical ordering with lower-level category (children) nested within higher-level category (parent) (more categories in lower level); 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 (`|`) (can be interpreted probabilistically as the conditional distribution for its parent) is used to indicate nesting variables (reflect depths), 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.

It encourage users to look into the data as earlier as possible, examining the data quality issues.

3 Data pipeline

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.

What is notable about data pipeline is that it not only frees users from a tedious and error-prone analysis but also empowers a wider audience 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.

3.1 Time-based pipeline

This "tidy" data 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, by contrast to data cleaning. We shall see how this data abstraction breeds a consistent data process.

All the individual operations strive to keep the time context so that they will implicitly preserve the index variable.

3.2 Modularity and clarity

An individual operation that performs on a specific task can be phrased as a verb. This verb is self-explanatory enough to express what it is supposed to do or fail. We shall adapt **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()`
- **reshape**: `gather()`, `spread()`
- **time-wise**: `lead()`, `lag()`, `difference()`
- **rolling window**: `slide()`, `tile()`, `stretch()`

3.3 Composition and transparency

- "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.

Every verb generates the output of the same type of the input, making chaining possible. Once the initial setup is done, the program takes care of updating the index and key, and empowers users concentrate on the substance of data analysis. The one who write the code would easily reason about the code and every one else would find the code fairly easy to comprehend.

4 Case studies

4.1 On-time performance for domestic flights in U.S.A

The dataset of 2017 on-time performance for US domestic flights represents event-driven data caught in the wild, sourced from US Bureau of Transportation Statistics (reference). It contains 5,548,445 operating flights with many measurements (such as departure delay, arrival delay in minutes, and other performance metrics) and detailed flight information (such as origin, destination, plane number and etc.) in a tabular format. This kind of event describes each flight scheduled for departure at a time point in its local time zone. Every single flight would be uniquely identified by the flight number and its scheduled departure time, from a passenger's point of view. In fact, it fails to pass the tsibble hurdle due to duplicates in the original data. An error is immediately raised when attempting to convert this data into a tsibble, and a closer inspection has to be carried out to locate the issue. The **tsibble** package provides the tool to easily find the duplicates in the data with `find_duplicates()`. Below shows the problematic entries.

```
#>   flight_num  sched_dep_datetime  sched_arr_datetime dep_delay arr_delay
#> 1      NK630 2017-08-03 17:45:00 2017-08-03 21:00:00       140       194
#> 2      NK630 2017-08-03 17:45:00 2017-08-03 21:00:00       140       194
#>   carrier tailnum origin dest air_time distance origin_city_name
#> 1      NK  N601NK    LAX  DEN      107      862      Los Angeles
#> 2      NK  N639NK    ORD  LGA      107      733          Chicago
#>   origin_state dest_city_name dest_state taxi_out taxi_in carrier_delay
#> 1           CA         Denver         CO       69      13             0
#> 2           IL       New York         NY       69      13             0
#>   weather_delay nas_delay security_delay late_aircraft_delay
#> 1             0       194              0                   0
#> 2             0       194              0                   0
```

The issue is perhaps introduced when updating or entering the data into a system. The same flight is scheduled at exactly the same time, together with the same performance statistics but different flight details, which is very unlikely. Flight NK630 is usually scheduled at 17:45 from Chicago to New York, searching into the whole records. A decision is made on removal of the first row from the duplicated entries before proceeding to the tsibble creation.

This dataset is intrinsically heterogeneous, encoding in numbers, strings, and date-times. The tsibble framework, as expected, incorporates this type of data, without the loss of data richness and heterogeneity. To declare the flight data as a valid tsibble, column `sched_dep_datetime` is specified as "index", column `flight_num` as "key" via `id(flight_num)`. As a result of event data, this data is irregularly spaced, and hence switching to irregular option is necessary. The program internally validates if the key and index produce the distinct rows, and then sort the key and the index from past to recent. When the tsibble creation is done, the print display is data-oriented and contextually informative, such as dimensions, irregular interval (`5,548,444 x 22 [!]`) and the number of time-based observational units (`flight_num [22,562]`).

```
#> # A tsibble: 5,548,444 x 22 [!]
#> # Key:       flight_num [22,562]
```

Transforming tsibble for exploratory data analysis with a suite of time-specific and general-purpose manipulation verbs can result in well-constructed pipelines. From the perspective of a passenger, one need to travel smart, by choosing an efficient carrier to fly with and the time of day to avoid congestion, for example. We take a drill-down approach to exploring this data, starting with annual carrier performance and followed by disaggregating to finer time resolutions.

Figure 1 visually presents the end product of aggregating the number of on-time and delayed flights to the year interval by carriers. This pipeline is initialized defining a new variable if the flight is delayed, and involves summarizing the tallies of on-time and delayed flights for each carrier annually. To prepare the summarized data for a mosaic plot, it is further manipulated by melting new tallies into a single column. The flow chart shown as Figure 2 demonstrates the operations undertaking in a data pipeline. The input to this pipeline is a tsibble of irregular interval, and the output ends up with a tsibble of unknown interval. The final data is each carrier along with a single year, thereby the interval undetermined. It in turn feeds into the

mosaic display. Note that Southwest Airlines, as the largest carrier, operates less efficiently compared to Delta, in Figure 1.

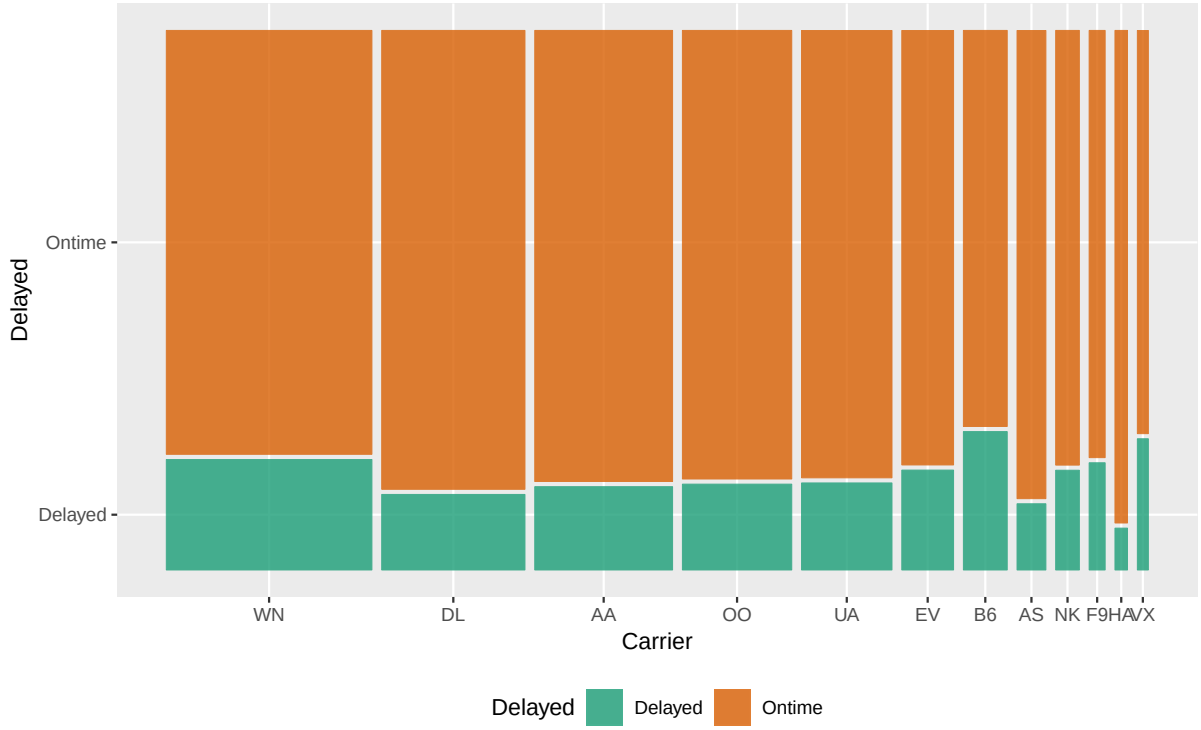

Figure 1: *Mosaic plot showing the association between the size of airline carriers and the delayed proportion of departure in 2017. Southwest Airlines is the largest operator, but does not operate as efficient as Delta. Hawaiian Airlines, also as a small operator, outperforms the rest.*

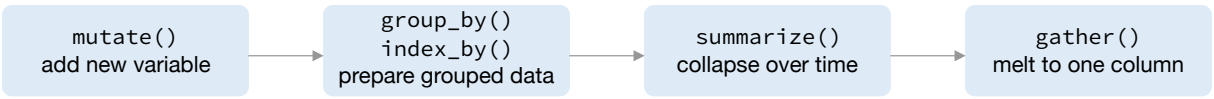

Figure 2: *Flow chart illustrates the pipeline that pre-processes the data for creating Figure 1.*

[Change New York Airports to others to be able to see seasonal effects]

A closer examination of New York airports will give an indication about how well the busiest airports manage the outflow traffic on a daily basis. A subset that contains observations for EWR, JFK and LGA airports is obtained first. The succeeding operations compute delayed percentages every day at each airport, which are framed as grey lines in Figure 3. LGA fluctuates a lot compared to the other two. What superimposes on these lines is two-month moving averages so that a temporal trend is more visible. The number of days for each month is variable. Moving averages for two months call for computing weighted mean. But this can also be accomplished using a pair of commonly used verbs–`nest()` and `unnest()` to handle

list-columns, without worrying weights specification. The sliding with large window size smoothes out the fluctuations and gives a stable trend around 25% over the year.

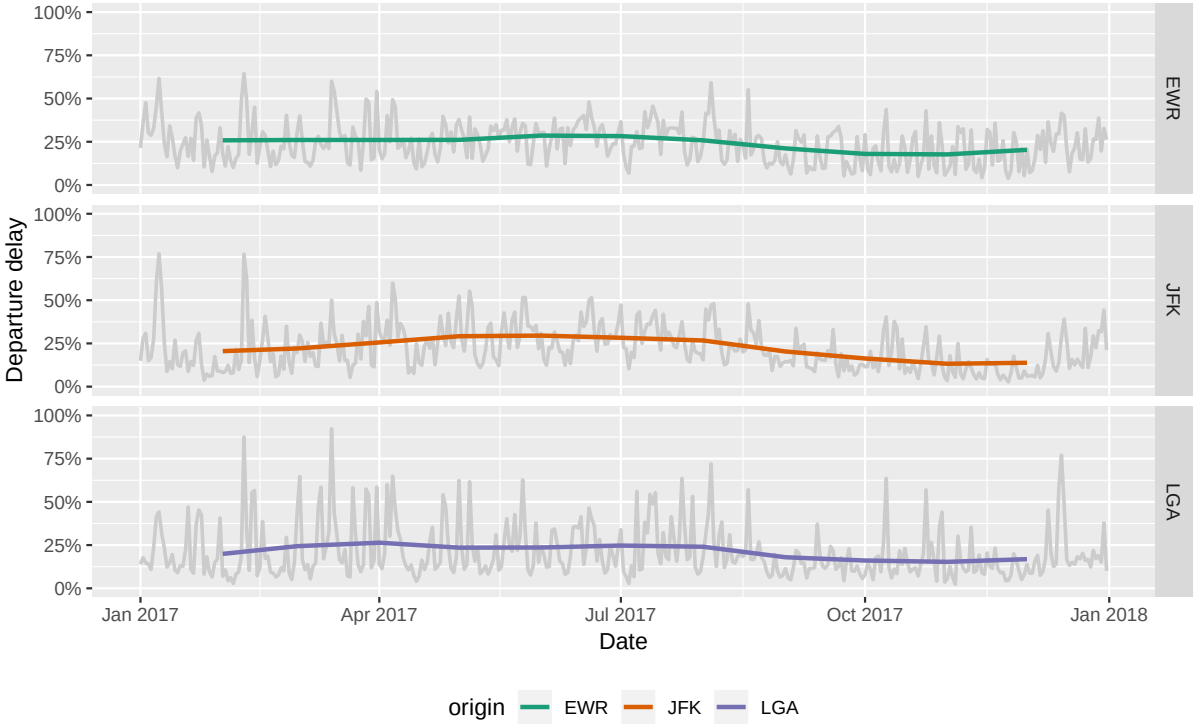

Figure 3: *Daily delayed percentages for departure with two-month moving averages overlaid at New York airports. There are many fluctuations observed at LGA airport. The estimates of temporal trend are around 25% across three airports, highlighting relatively less delay in Fall.*

What time of day and day of week should we travel to avoid suffering from horrible delay? Figure 4 plots hourly quantile estimates across day of week in the form of small multiples. The upper-tail delay behaviors are of primary interest, and hence 50%, 80% and 95% quantiles are shown. To reduce the likelihood of delay suffering, it is recommended to avoid the peak hour at 18. As moving towards the upper extremes, the variations considerably increase, making departure time unpredictable.

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.

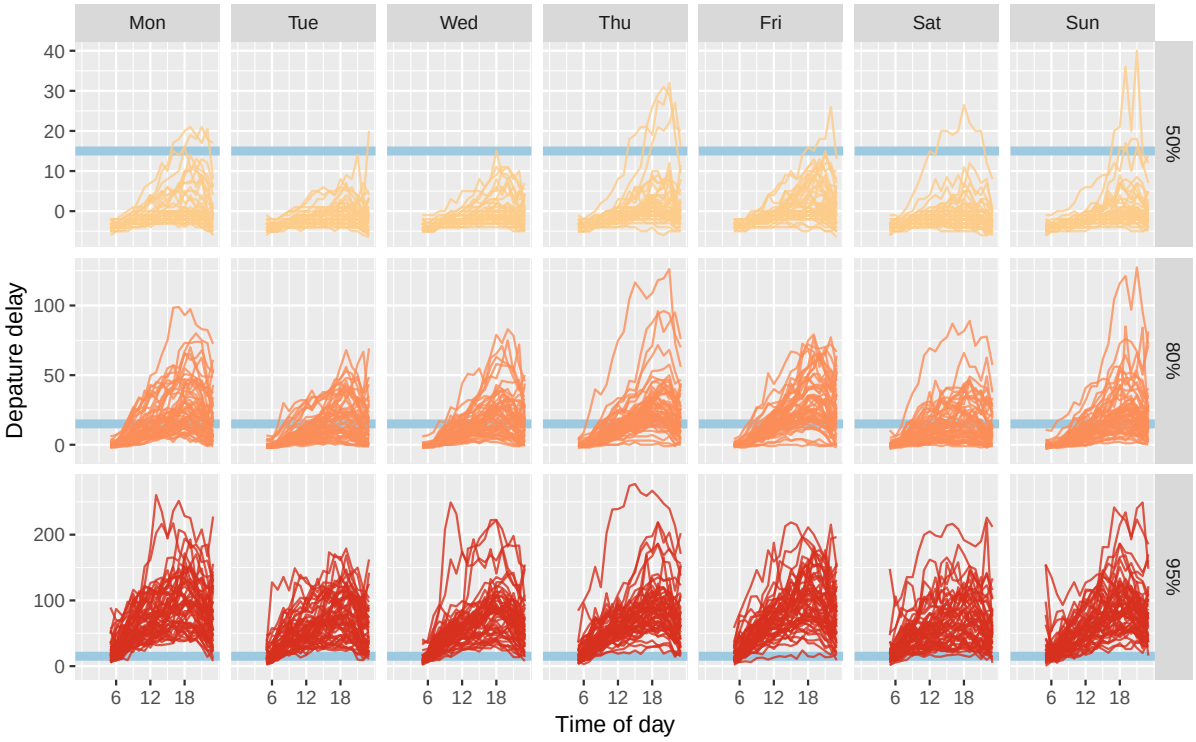

Figure 4: *Small multiples of lines about departure delay against time of day, faceting day of week and 50%, 80% and 95% quantiles. A blue horizontal line indicates the 15-minute on-time standard to help grasp the delay severity. Passengers are apt to hold up around 18 during a day, and are recommended to travel early. The variations increase substantially as the upper tails.*

A new visualization framework for handling time series with nesting and crossing structure is under development. The proposed data structure proves a natural input (unit time series and parent-children structure) that can be piped into a visualization program. As long as the program understands the embedded structure, it will produce sensible layout and interactivity.

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 (2008). Panel Data Econometrics in R: The plm Package. *Journal of Statistical Software, Articles* **27**(2), 1–43.

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.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/.

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

Ryan, JA & JM Ulrich (2018). *xts: eXtensible Time Series*. R package version 0.11-0. https://CRAN.R-project.org/package=xts.

Sankaran, K & S Holmes (2017). Interactive Visualization of Hierarchically Structured Data. *Journal of Computational and Graphical Statistics* **0**(ja), 0-0. eprint: https://doi.org/10.1080/10618600.2017.1392866.

StataCorp (2017). *Stata Statistical Software: Release 15*. StataCorp LLC. College Station, TX, United States. https://www.stata.com.

The MathWorks, Inc. (2017). *MATLAB and Statistics Toolbox Release 2017b*. The MathWorks, Inc. Natick, Massachusetts, United States. http://www.mathworks.com.

Wickham, H (2009). *ggplot2: Elegant Graphics for Data Analysis*. New York, NY: Springer-Verlag New York.

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 0.7.7. 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 & L Henry (2018). *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions*. R package version 0.8.2. https://CRAN.R-project.org/package=tidyr.

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.

World Health Organization (2018). *Tuberculosis Data*. Block 3510, Jalan Teknokrat 6, 63000 Cyberjaya, Selangor, Malaysia. http://www.who.int/tb/country/data/download/en/ (visited on 06/05/2018).

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

Zeileis, A & G Grothendieck (2005). zoo: S3 Infrastructure for Regular and Irregular Time Series. *Journal of Statistical Software, Articles* **14**(6), 1–27.