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 data

Earo Wang, Dianne Cook, Rob J Hyndman

November 2018

Working Paper no/yr

Tidy data structure to support exploration and modeling of temporal 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

25 November 2018

JEL classification: C10,C14,C22

Tidy data structure to support exploration and modeling of temporal data

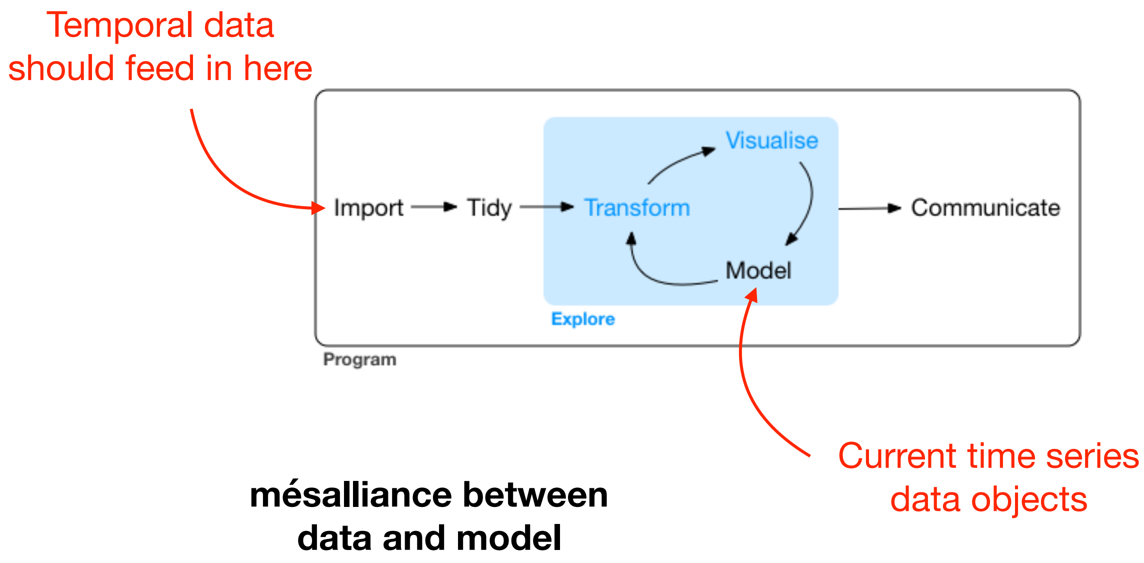
Abstract

Mining temporal 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. 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. 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 $j = 1, \dots, N_i$ 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.



The rest of the paper is structured as follows.

2 Data structures

2.1 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.

2.2 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.

3 Contextual 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. 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.

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,

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:* `country` *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

where column `year` is declared as the index variable, columns `country`, `continent` and `gender` as key. Detailed discussions about index and key specifications will be given in Section 3.1 and 3.2. 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".

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

3.1 Index

Time provides a contextual basis for temporal data. Time representation could range from date-times (such as `POSIXct` and `Date` in R) to years, to sequential numbers for simulated data for example.

A variable representing time is indispensable to a tsibble, referred to as "index". The "index" is an intact data column rather than an attribute, which makes time accessible to users. For example, one could easily extract time components, such as time of day and day of week, from the index to visualise seasonal effects of response variables. One could also join other data sources to the tsibble based on common time indices. The accessibility of tsibble index motivates data analysis towards transparency and human readability. But if the "index" is employed as attributes, analysts would write these simple queries in a programmatic manner, which can be avoidable from an analytic point of view.

How to handle time zone. Additional representations are provided, like year-month, year-quarter. Extensible. Custom index classes can be supported.

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)

3.2 Key

The "key" specification contributes to creating a tsibble alongside the index. The concept of "key" is introduced to identify units or subjects that are observed over time in a data table. In the wide format, each column holds a series of values, so that the column itself serves for the sake of identification. In the long format, all column names are melt to the corresponding "key" values. But the "key" is much more flexible than simply column names. Because it is not constrained to a single field, but can be comprised of multiple fields. The identifying variables that the "key" is constituted of, stay the same as they are in the original table, with no further tweaks.

Each tsibble must hold a "key". It is normally a priori known by analysts. For example, Table 1 describes the number of tuberculosis cases for each gender across the countries every year. This data description suggests that columns `gender` and `country` have to be declared as the key, similar to a panel variable for longitudinal data. Lacking in either of two will be inadequate for the key and thus fail to construct a tsibble. The key is explicit when multiple units exist in the data. Key can be implicit when it finds a univariate series in the table, but it cannot be absent from a tsibble.

Not only pinpoints a "key" observational units in the tsibble but also provides a solution to seamlessly link between the data, models, and forecasts. This neatly separates the data from models and forecasts, leaving more room for compulsory model components, such as coefficients, fitted values and residuals. More details are given in the following section.

3.3 Interval

One of the cornerstones beneath a tsibble is time interval. This information plays a critical role in computing statistics (e.g. seasonal unit root tests) and building models (e.g. seasonal ARIMA). The principal divide is regularly or irregularly spaced in time. Since a tsibble permits implicit missing time, it is impossible to distinguish regularity from the index. It relies on user's specification by switching the `regular` argument off, when the data involves irregular intervals. This type of data can flow into event-based data modelling, not models that expect time series.

As for data indexed in regular time space, the time interval is derived by first computing absolute differences of time indices and then the greatest common divisor, which covers all conceivable cases. This implies that all subjects in a tsibble have one and the only interval. Data collected at different intervals should be organized in separate tsibbles, encouraging well-tailored analysis and models, because each subject may underly different data generating process.

4 Data pipeline

There has been a long history of pipeline discussions and implementation centering around data in various aspects. A data pipeline describes the general flow of data through an analysis, and can generally assist in conceptualising the process as it might be applied to a variety of problems. The Extract, Transform, and Load (ETL), from the data warehousing literature, outlines the workflow to prepare data for analysis, dates back to (XXX Caserta, Joe, 1965? REFS needed) can be considered a data pipeline. Building a data pipeline can be technically difficult, to make it sufficiently general for various data, with many implementation decisions on the interface, input and output objects and functionality. It is useful to articulate the data pipeline induced by new data tools.

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) describes a viewing pipeline for interactive statistical graphics, that takes control of the transformation from data to plot. **xgobi**, **ggobi**, **orca**, Wickham et al. (2010) and Xie, Hofmann & Cheng (2014) implemented data pipelines for the interactive statistical software **xgobi**, **ggobi**, **orca**, **plumbr** and **cranvas**, respectively. The pipeline is typically described with a one way flow, from data to plot. For interactive graphics, where all plots need to be updated when a user interacts with one plot, the events typically trigger the data pipeline to be run. Xie, Hofmann & Cheng (2014) uses a reactive programming framework, to implement the pipeline, in which user's interactions trigger a sequence of modules to update their views, that is, practically the same as running the data pipeline producing each plot.

The tidy data abstraction lays a pipeline infrastructure for data analysis modules, transformation, visualization and modelling. Each module communicates between each other, requiring tidy input, producing tidy output, and consequently chains a series of operations together to accomplish the analytic tasks at hand.

What is notable about an effective implementation of a data pipeline is that it coordinates a user's analysis making it cleaner to follow, and permits a wider audience to focus on the data analysis without getting lost in a jungle of computational intricacies. A fluent and fluid pipeline glues tidy data and the grammar of data manipulation together. It helps (1) break up a big problem to into manageable blocks, (2) generate human readable analysis workflow, (3) avoid introducing mistakes, at least making it possible to trace them through the pipeline.

4.1 Time-based pipeline

XXX may be good to have a diagram here showing how the time-based pipeline builds on a regular data pipeline

XXX Put comments in above each paragraph, with the topic of each paragraph. its not clear how this section flows yet. the topics would help.

The time-based pipeline shepards raw temporal data through to time series analysis, and plots. It is advised to scrutinize identical entries of key and index in the right beginning. Duplicates signal the data quality issue, which would likely affect succeeding analyses and hence decision making. Analysts are encouraged to gaze at data as earlier as possible and reason about the process of data cleaning. When the data meets the tsibble standard, it flows into the exploration stage.

Time series models typically assume that the input is a complete and regularly-spaced series, which most temporal data can barely satisfy. Temporal data becomes more granular in time resolution and more disaggregated to individual level. This inevitably goes with implicit missing values and noisiness. Data needs transformation to some extent for modelling. A suite of verbs are introduced to flatten the lumpy path from temporal data to the object that directs to modelling under the tsibble framework, as well as to facilitate transforming tsibble in various kinds of shapes for analysis and visualization. The principle that underpins most verbs is a tsibble in and a tsibble out, thereby striving to retain a valid tsibble by automating index and key updates under the hood. If a tsibble cannot hold, for example swiping index off, an error prompts users to complain and suggest alternatives. This warrants users least surprise and reminds them of time awareness.

Transformation assembles a set of modules. A unit that makes up a module is function or preferably "verb". A tsibble is an object, conceptually considered as a noun, and hence an action performed on the object can be phrased as a verb. Each verb focuses on one thing and achieve its goal. The verb should be self-explanatory to advise what it is supposed to do or fail, for example `filter()` picking observations, and `select()` picking variables. These general-purpose verbs are made available in the **tidyverse** suite. When manipulating in temporal context, these verbs are adapted to time domain. A perceivable difference is summarizing variables between data frame and tsibble. The former will reduce to a single summary, whereas the latter will obtain the index and their corresponding summaries. New tsibble-specific verbs are proposed to expand the **tidyverse** vocabulary. We believe that users, who are already familiar with **tidyverse**, will experience a gentle learning curve for mastering tsibble verbs and glide into temporal data analysis with low cognitive load.

A table to list the verbs and what they do.

- **implicit missingness**: `has_gaps()`, `count_gaps()`, `fill_gaps()`
- **row-wise**: `filter()`, `slice()`, `arrange()`
- **column-wise**: `mutate()`, `select()`, `summarize()`
- **group-wise**: `index_by()`, `group_by()`
- **reshape**: `gather()`, `spread()`, `nest()`, `unnest()`

Friedman & Wand (2008) asserted "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." Each verb works with other transformation family members in harmony. This set of verbs can result in many combinations to prepare tsibble for a broad range of visualization and modeling problems. Most importantly, the ecosystem for tidy time series analysis has been undertaking on the basis of tsibble in R, known as "tidyverts".

As a special case of data frame, a tsibble pipes into the grammar of graphics straight way, making most use of this powerful graphical system. It should be easy to create and extend some specialist plotting methods based on tsibble structure, such as autocorrelation plots (**tsibblestats**) and calendar-based graphics (**sugrrants**).

Modeling is crucial to explanatory and predictive analytics, but often imposes stricter assumptions on tsibble data. The verbs listed in Table **??** ease the transition to a tsibble that suits modeling. A tidy forecasting framework built on top of tsibble is under development, which aims at promoting transparent forecasting practices and concise model representation. A tsibble usually contains multiple time series. Batch forecasting will be enabled if a univariate model, such as ARIMA and Exponential Smoothing, is applied to each time series independently. This yields a "mable" (short for model table), where each model only tags to each "key" value in tsibble to avoid expensive data copying and reduce model storage. The mable is further supplied to forecasting methods, to produce a tsibble in which each "key" along with its future time holds predictions. It also underlines the advantage of tsibble's "key" in linking between data inputs, models and forecasts. Advanced forecasting techniques, such as vector autocorrelation, hierarchical reconciliation, and ensembles, can be developed in alike spirit. The modeling module will be fulfilled and integrated eventually.

We go through the whole exploration circle, and keep iterating and refining until data insights gained. The tsibble data structure substantially lubricates between these modules for time-based pipelines. The cohesive and coherent framework results in cleaner, clearer and more expressive code.

4.2 Rolling window in functional programming

- **rolling window**: `slide()`, `tile()`, `stretch()`

5 Software structure

- Extensibility

6 Case studies

6.1 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 [!] <UTC>`) 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.

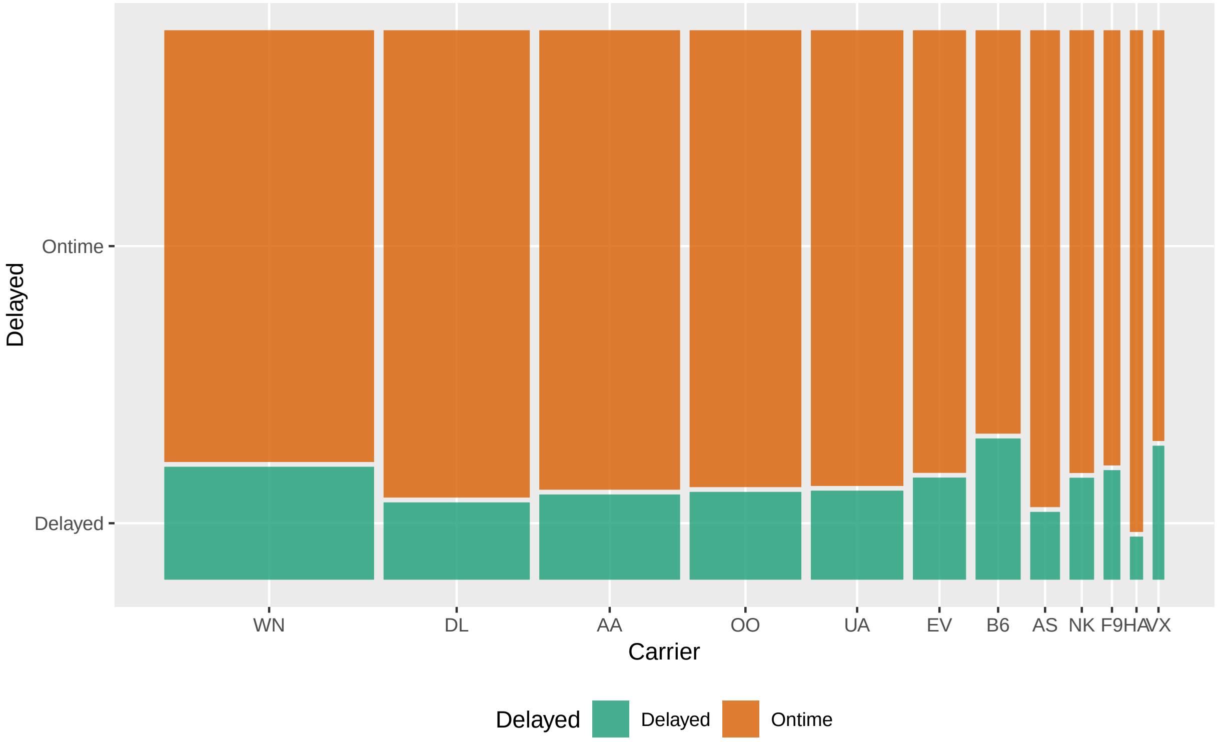


Figure 1: *Mosaic plot showing the association between the size of airline carriers and the delayed proportion of departure in 2017. Southwest Airlines is the largest operator, but does not operate as efficient as Delta. Hawaiian Airlines, also as a small operator, outperforms the rest.*

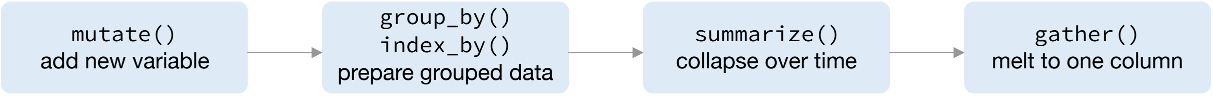


Figure 2: *Flow chart illustrates the pipeline that pre-processes the data for creating Figure 1.*

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.

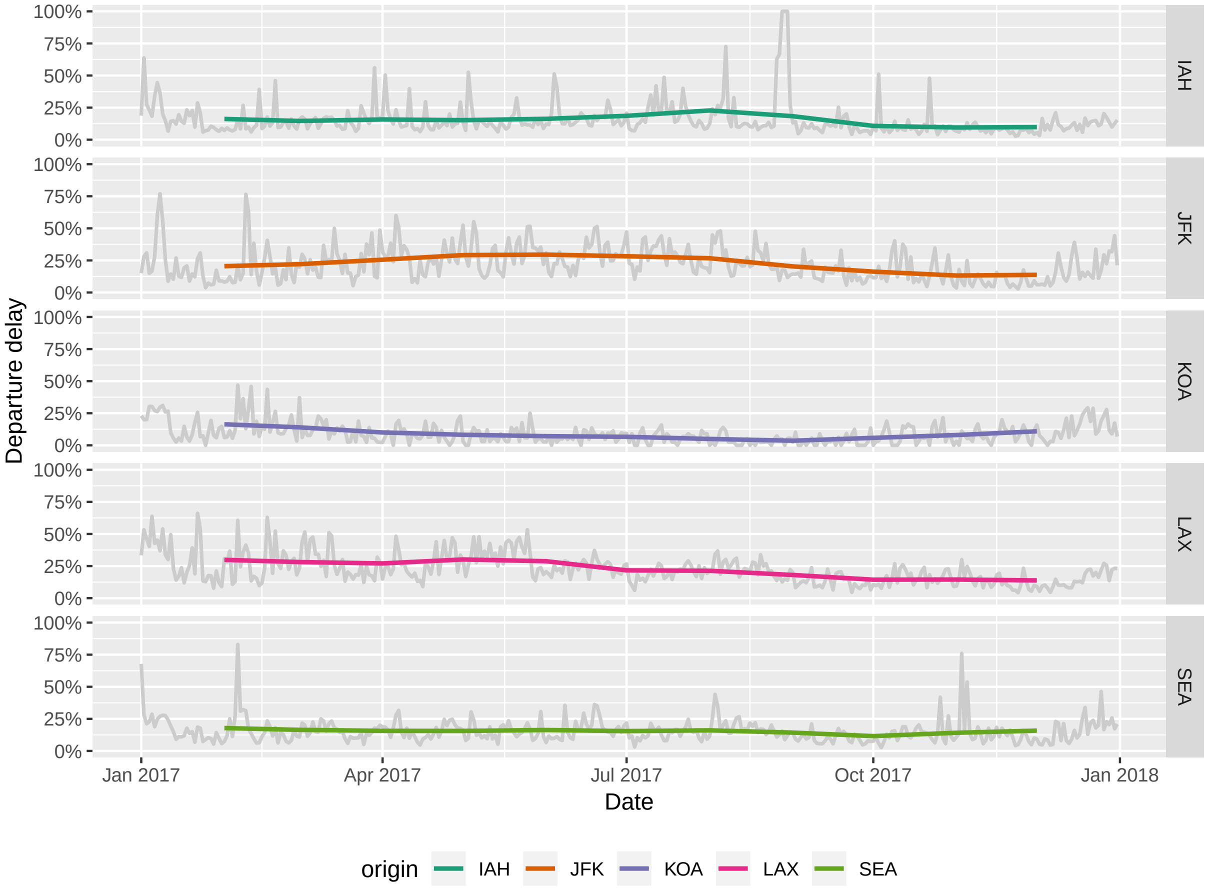


Figure 3: *Daily delayed percentages for departure with two-month moving averages overlaid at five international airports. There are many fluctuations observed at LGA airport. The estimates of temporal trend are around 25% across three airports, highlighting relatively less delay in Fall.*

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.

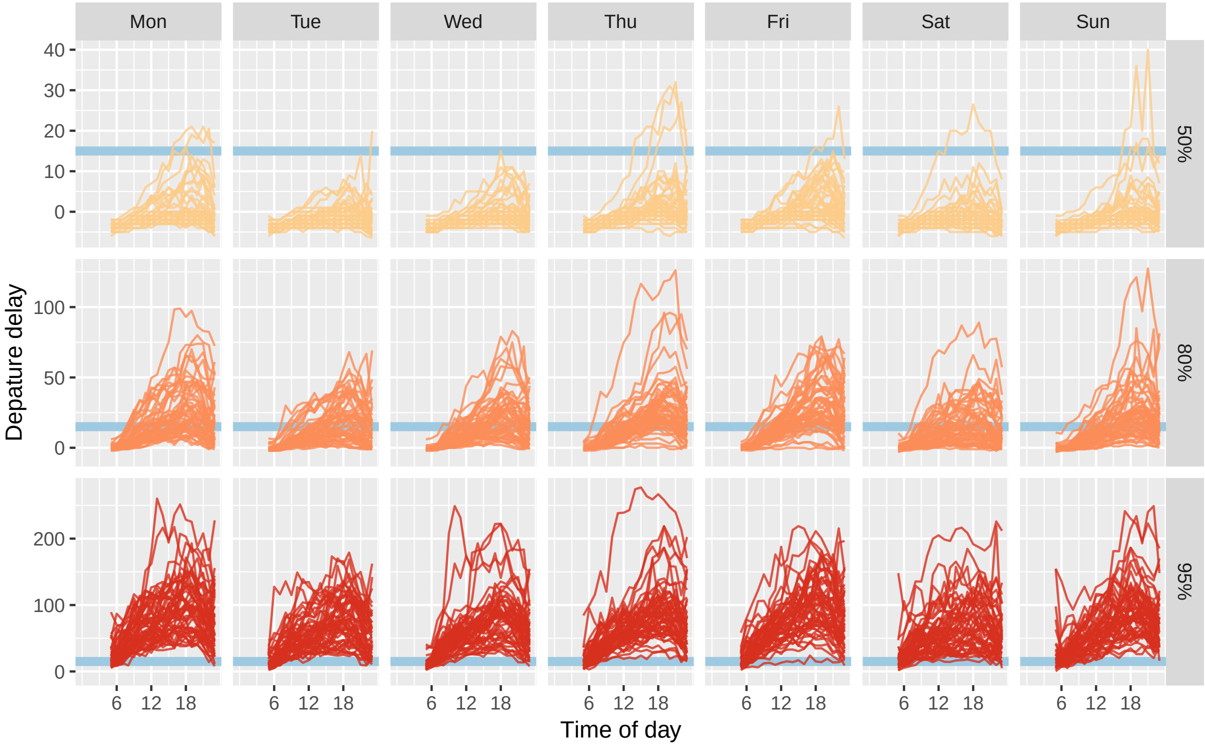


Figure 4: *Small multiples of lines about departure delay against time of day, faceting day of week and 50%, 80% and 95% quantiles. A blue horizontal line indicates the 15-minute on-time standard to help grasp the delay severity. Passengers are apt to hold up around 18 during a day, and are recommended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

6.2 Smart-Grid Customer Data in Australia

https://data.gov.au/dataset/smart-grid-smart-city-customer-trial-data

7 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.

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.

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.nclude 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.

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.

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 0.7.8. 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.