MONASH
BUSINESS
SCHOOL

ISSN 1440-771X

Department of Econometrics and Business Statistics

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

A new tidy data structure to support exploration and modeling of temporal data

Earo Wang, Dianne Cook, Rob J Hyndman

January 2019

Working Paper no/yr

A new 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

19 January 2019

JEL classification: C10,C14,C22

A new tidy data structure to support exploration and modeling of temporal data

Abstract

Mining temporal data for information is often inhibited by a multitude of formats: irregular or multiple time intervals, point events that need aggregating, 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 the data to be provided, typically a matrix of numeric data with implicit time indexes. 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 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 fluid workflow for analyzing temporal data. The infrastructure of tidy temporal data has been implemented in the R package **tsibble**.

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

1 Introduction

Temporal data arrives in many possible formats, as does the time context. For example, time can carry with it 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: multiple observational units of different time lengths, multiple and heterogeneous measured variables, multiple grouping

factors. Temporal data may be occurence of events, like flight departures, that need to be induced into a regular structure.

Current time series objects, tend to be model-oriented matrices. Analysts are expected to do their own data pre-processing and take care of anything else needed to get to this stage, which leads to a myriad ad-hoc solutions and duplicated efforts.

In recent work, there has been exciting developments, that conceptually and practically streamline the pre-processing of data. Figure 1 (adapted from Wickham & Grolemund (2016)) illustrates the tidy data workflow. Time series modeling and forecasting enters the picture at the *modeling* stage. While temporal data enters at the start, there isn't a current framework for getting temporal data into the matrix format for modeling. That's the part of the workflow that the work descirbed in this paper improves.

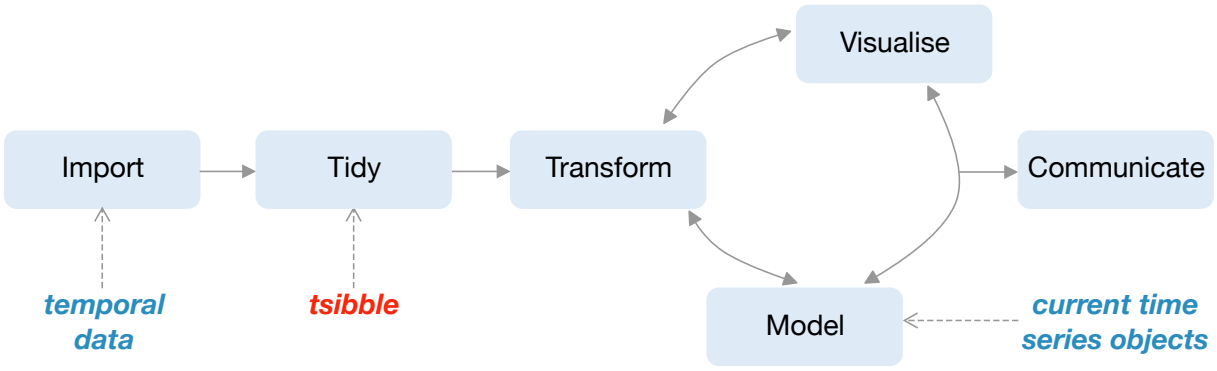

Figure 1: *Illustration of data science workflow, drawn from Wickham & Grolemund (2016), showing how current time series tools interface with the workflow and how tsibble structure and tools integrate. The new data structure, tsibble, makes the connection between temporal data input, and specialist modeling formats. It provides elements at the "tidy" step, which produce tidy temporal data for time series visualisation and modeling.*

The paper is structured as follows. Section 2 reviews temporal data structures corresponding to time series and longitudinal analysis, and discusses "tidy data" and the grammar of data manipulation. Section 3 proposes contextual semantics for temporal data, built on top of tidy data. The concept of data pipelines with respect to the time domain will be discussed in depth in Section 4, followed by a discussion of the design choices made in the software structure in Section 5. Two case studies are presented in Section 6 illustrating temporal data exploration using the newly implemented infrastructure. Section 7 discusses the future work.

2 Data structures

2.1 Time series and longitudinal data

The data problems are grouped into two types of analysis, time series and longitudinal. Despite being 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 characterizing temporal dynamics. The time series supporting modeling are represented as vectors or matrices in most statistical software. 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 into wide format which would have missing values in many cells. A drawback of this format is information unique to each individual is often repeated for all time points. The appealing feature is that data is structured in a 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.

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 by referring rows to observations and columns to variables. The principles of "tidy data" attempt to standardize the mapping of the semantics of a dataset to its physical representation. This data structure is the fundamental unit of the **tidyverse**, which is a collection of R packages designed for data science. The ubiquitous use of **tidyverse** is testament to the simplicity, practicality and general applicability of the tools. The **tidyverse** provides abstract yet functional grammars to manipulate and visualize data in easier-to-comprehend form. One of the **tidyverse** packages, **dplyr** (Wickham et al. 2018) showcases the value of a grammar as a principled vehicle to transform data for a wide range of data challenges, providing a consistent set of verbs: `mutate()`, `select()`, `filter()`, `summarise()`, and `arrange()`. Each verb focuses on a singular task. Most common data tasks can be rephrased and tackled with these five key verbs, by composing them sequentially.

The **tidyverse** largely formalizes exploratory data analysis. Many in the R community have adopted the **tidyverse** way of thinking and extended to broader domains, such as simple features for spatial data in the **sf** package (Pebesma 2018) and missing value handling in the **naniar** package (Tierney & Cook 2018). Temporal data tools need to catch up.

2.3 Existing time series standards in R

Current standards, provided by the native `ts` object in R, extended by **zoo** (Zeileis & Grothendieck 2005), and **xts** (Ryan & Ulrich 2018), assemble temporal data into matrices with implicit time indexes. These objects were designed for modelling methods. The diagram in the style of Figure 1, would place the model at the center of the analytical universe, and all the transformations, visualisations would hinge on that format. This is contrary to the tidyverse conceptualisation, which wholistically captures the full data workflow.

A new temporal data class is needed in the upstream of the workflow, which could incorporate all the downstream modules. A relatively new R package **tibbletime** proposed a data class of *time tibble* to represent temporal data in heterogeneous tabular format. It only requires an index variable to declare a temporal data object, thus placing at the import stage. However, this is a

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` *and* `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

loosely-defined structure and insufficient for time series analytics and models. Because it does not define interested subjects (i) over time and time uniqueness.

This paper describes a new tidy representation for temporal data, and a unified framework to streamline the workflow from data pre-processing to visualization and forecasting, as an integral part of a tidy data analysis.

3 Contextual semantics

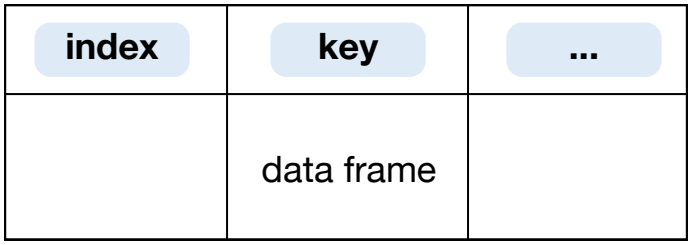

Figure 2: *The architecture of the tsibble structure is built on top of the data frame—a two-dimensional array in R, with time series semantics: index and key.*

The choice of tidy representation of temporal data arises from a data-centric perspective, which accommodates all of the operations that are to be performed on the data. Figure 1 marks where this new abstraction is placed in the tidy model, which we refer to as a "tsibble".

Table 1 presents 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 hosts the number of people who are diagnosed tuberculosis for each gender at three selected countries in the years of 2011 and 2012. To turn this data into a tsibble, (1) column `year` is

declared as the index variable; (2) the key can be made up of columns `country` and `gender`. The column `count` is the only measured variable in this data, but the structure is sufficiently flexible to hold other measured variables, for example, adding the corresponding population size (if known) in order to calibrate the count later.

The new data structure, tsibble, bridges the gap between raw temporal data and model inputs. Contextual semantics are introduced to tidy data in order to support more intuitive time-related manipulations and enlighten new perspectives for time series model inputs. Index, key and time interval are the three stone pillars to this new semantically-structured temporal data.

3.1 Index

Time provides a contextual basis for temporal data. A variable representing time is essential for a tsibble, and is referred to as an "index". The "index" is an intact data column rather than a masked attribute, which makes time visible and accessible to users. It is highly advantageous when manipulating time. For example, one could easily extract time components, such as time of day and day of week, from the index to visualize seasonal effects of response variables. One could also join other data sources to the tsibble based on common time indexes. The accessibility of tsibble index motivates data analysis towards transparency and human readability. When the "index" used to be employed as meta information, it created an obstacle for analysts to write these simple queries in a programmatic manner, which should be discouraged from an analytic point of view.

A variable number of time representations is spotted in the wild. A Date-time object, universally accepted across computing systems, is the most commonly used type for representing time. Date-time also typically associates with a time zone with adjustments like summer time. This diversity and time zone is acknowledged and considered the building block for *tsibble*'s index.

3.2 Key

The "key" specification is the second essential ingrdient for a tsibble. The "key" introduced uniquely identifies observations that are recorded over time in a data table. It is similar to a primary key (Codd 1970) defining each observation in a relational database. In the wide format, that multiple time series are often structured in, the columns hold a series of values, so that the column implicitly serves as identification. In long format, columns are melted with names converted to "key" values. However, the "key" provides much more flexibility. It is not constrained to a single field, but can be composed from multiple fields. The identifying

variables that the "key" is constituted from, remain the same as in the original table, with no further tweaks.

The "key" is usually known a priori 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 either of two will be inadequate, because the observations are not uniquely identified, and thus will generate a tsibble construction fail. An alternative specification of the key for this data is to include a third variable `continent`. Since `country` is nested within `continent`, it is a free variable for use. This variable brings additional information that can be used for forecasting reconciliation (Hyndman & Athanasopoulos 2017). The key needs to be explicit when multiple units exist in the data. The key can be implicit when it finds a univariate series in the table, but it cannot be absent from a tsibble.

The "key" also provides a solution to link between the data, models, and forecasts. This neatly decouples the data from models and forecasts, leaving more room for necessary model components, such as coefficients, fitted values and residuals. More details are given in the following section.

3.3 Interval

One of the cornerstones of time series data, and hence beneath a tsibble, is the 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 observations in time. A tsibble permits implicit missing time, making it difficult to distinguish regularity from the index. It relies on a 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, but would need to be processed or regularized to fit models that expect time series.

For data indexed in regular time space, the time interval is automatically calculated, by first computing absolute differences of time indexes and then the greatest common divisor. This covers all conceivable cases. The implication is that all observations 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 observation may different underlying data generating processes.

4 Data pipeline

The tsibble structure and operations support data pipelines for sequencing analysis. 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 recent data warehousing literature, outlines the workflow to prepare data for analysis, dates back to Kimball & Caserta (2011) 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 program 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. Swayne, Cook & Buja (1998), Swayne et al. (2003), Sutherland et al. (2000), 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 of transformation, visualization and modelling. Each module communicates between each other, requiring tidy input, producing tidy output, chaining a series of operations together to accomplish the analytic tasks.

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.

The time-based pipeline shepards raw temporal data through to time series analysis, and plots. It is possible and recommended to check for identical entries of key and index before analysis. Duplicates signal the data quality issue, which would likely affect succeeding analyses and hence decision making. Analysts are encouraged to gaze at data early and reason about the process of data cleaning. When the data meets the tsibble standard, it flows neatly into the analysis stage. Figure 3 illustrates the time-based pipeline.

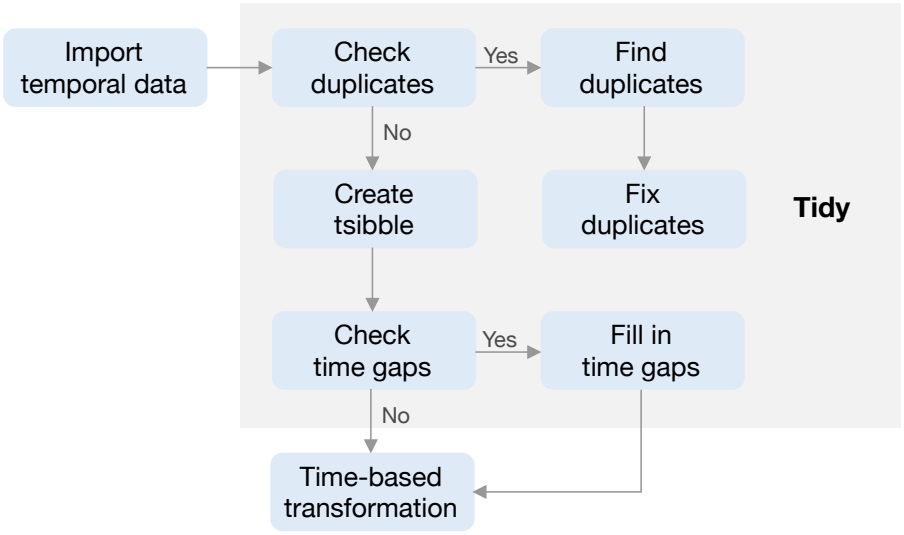

Figure 3: *A time series pipeline is different from a regular data pipeline: (1) check if the key and index uniquely identify each observation in the data whist creating a tsibble; (2) check if there are time gaps in the tsibble before analysis.*

4.1 Time-based transformation

Many time operations like lag/lead and time series models, assume an intact vector input ordered in time. To prevent inviting these errors into the analysis, it is a good practice to check and inspect any time gaps of the tsibble in the first place. Several tools are provided to understand and tackle implicit missing values in: (1) `has_gaps()` checks if there exists time gaps; (2) `scan_gaps()` reports all implicit missing observations; (3) `count_gaps()` summarises the time ranges that are absent from the data; (4) `fill_gaps()` turns them into explicit ones, along with imputing by values or functions.

A suite of verbs, and corresponding functions, are used to flatten the lumpy path from temporal data to an object that can be directly modeled in the tsibble framework, and transformed for visualization. A tsibble is an object, conceptually considered as a noun, and hence an action

performed on the object can be phrased as a verb. 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. Attention has been paid to error handling. If a tsibble cannot be maintained in the output of a pipeline module, for example the index is dropped by aggregation, an error informs users of the problem and suggests alternatives. This avoids surprising users and reminds them of the time context.

Each verb focuses on one oepration to achieve its goal. The verb should be self-explanatory to advise what it is supposed to do or fail, for example `filter()` picks observations, `select()` picks variables, and `left_join()` joins two tables. These general-purpose verbs are made available in the **tidyverse** suite. When used in the temporal context, these verbs are adapted to the 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 (Table 2 are provided to expand the **tidyverse** vocabulary. 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.

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. Chaining operations is achieved with the pipe operator `%>%` introduced in the **magrittr** package. A sequence of functions can be composed in a way that can be naturally read from left to right, which improves the readability of the code. It consequently generates a block of code without saving intermediate values. Most importantly, a new ecosystem for tidy time series analysis has been undertaken, using the tsibble framework, and is called "tidyverts", a play on tidyverse that acknowledges the time series analysis purpose.

4.2 Time series visualization

As a special case of the data frame, a tsibble pipes directly into the grammar of graphics. It is easy to create and extend specialist time-related plotting methods based on tsibble structure, such as autocorrelation plots and calendar-based graphics (Wang, Cook & Hyndman 2018).

Table 2: *A list of table verbs working with tsibble. Functions in bold originating from the tidyverse and are adapted for tsibble.*

	Verb	Description
Time gaps	`has_gaps()`	Test if a tsibble has gaps in time
	`scan_gaps()`	Report implicit missing entries
	`count_gaps()`	Summarize time gaps
	`fill_gaps()`	Fill in gaps by values and functions
Row-wise	**`filter()`**	Pick rows based on conditions
	`filter_index()`	Provide a shorthand for time subsetting
	`slice()`	Select rows based on row positions
	`arrange()`	Sort the ordering of row by variables
Column-wise	**`select()`**	Pick columns by variables
	`mutate()`	Add new variables
	`transmute()`	Drops existing variables
	`summarise()`	Aggregate values over time
Group-wise	`index_by()`	Group by index candidate
	`group_by()`	Group by one or more variables
	`group_by_key()`	Group by key variables
Reshape	**`gather()`**	Gather columns into long form
	`spread()`	Spread columns into wide form
	`nest()`	Nest values in a list-variable
	`unnest()`	Unnest a list-variable
Join tables	**`left_join()`** **`right_join()`** **`full_join()`** **`inner_join()`** **`semi_join()`** **`anti_join()`**	Join two tables together

4.3 Time series models

Modeling is crucial to explanatory and predictive analytics, but often imposes stricter assumptions on tsibble data. The verbs listed in Table 2 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 relates to each "key" value in tsibble. This avoids expensive data copying and reduces 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 a similar spirit. The modeling module is a current endeavor.

5 Software structure and design decisions

5.1 Data first

The prime force that drives the software's design choices is "data". All functions in **tsibble** start with `data` or its variants as the first argument, namely "data first". They work naturally with the pipe operator `%>%`, read as "then". This not only lays out a consistent interface but also addresses the significance of the data throughout the software.

Beyond the tools, the print display provides a quick and comprehensive glimpse of data in temporal context, particularly useful when handling a large collection of data. The contextual summary provided by the print function shown below, on the data from Table 1, contains (1) data dimension with its shorthand time interval, alongside time zone if date-time, (2) variables that constitute the "key" with the number of series. These details aid users in understanding their data better.

```
#> # A tsibble: 12 x 5 [1Y]
#> # Key:       country, gender [6]
#>   country     continent gender  year count
#>   <chr>       <chr>     <chr>  <dbl> <dbl>
#> 1 Australia   Oceania   Female  2011   120
#> 2 Australia   Oceania   Female  2012   125
#> 3 Australia   Oceania   Male    2011   176
#> 4 Australia   Oceania   Male    2012   161
#> 5 New Zealand Oceania   Female  2011    36
#> # ... with 7 more rows
```

5.2 Functional programming

Rolling window calculations are widely-used techniques in time series analysis, but are not restricted to temporal applications. These operations are dependent on having an ordering, particularly time ordering for temporal data. Three common types of variations for sliding window operations are:

1. **slide**: sliding window with overlapping observations.

2. **tile**: tiling window without overlapping observations.
3. **stretch**: fixing an initial window and expanding to include more observations.

Figures 4, 5 and 6 show the animations of rolling windows for sliding, tiling and stretching, respectively, on annual tuberculosis cases for Australia. A block of consecutive elements with a window size of 5 are initialized and started rolling sequentially till the end of series.

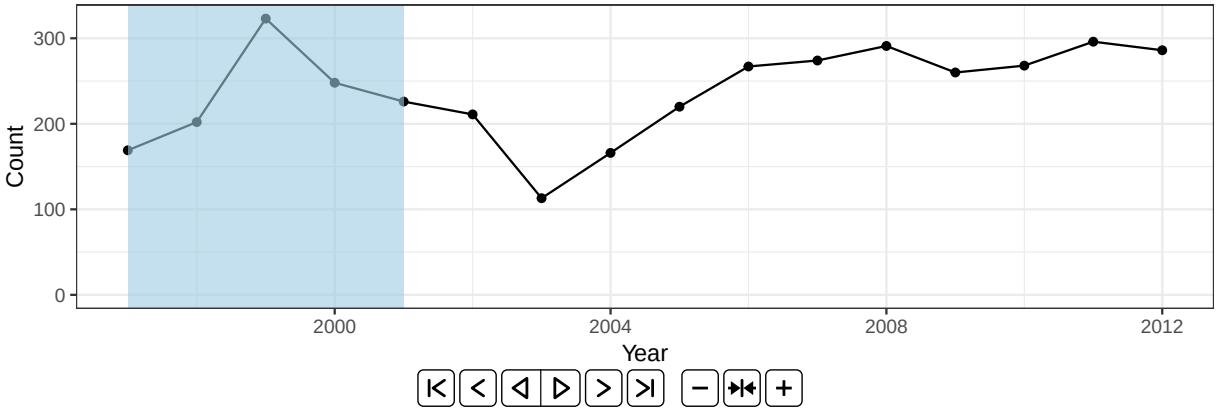

Figure 4: *An illustration of window of size 5 sliding over annual tuberculosis cases in Australia. (Animation needs to be viewed with Adobe Acrobat Reader.)*

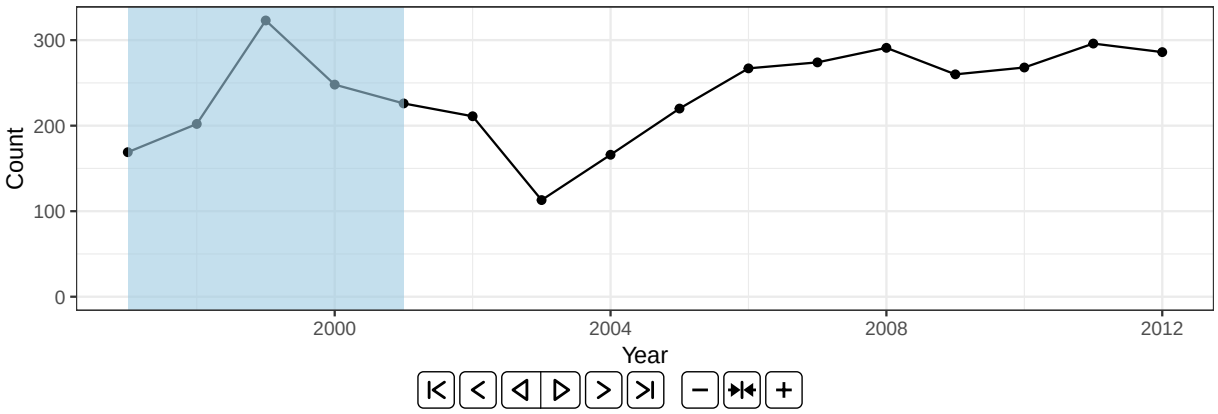

Figure 5: *An illustration of window of size 5 tiling over annual tuberculosis cases in Australia. (Animation needs to be viewed with Adobe Acrobat Reader.)*

Rolling window uses a programming paradigm—functional programming, which is different from those table verbs listed in Table 2. Table verbs expect and return a tsibble, and does something that the function name suggests. On the contrary, these rolling window functions could accept arbitrary input types and would return arbitrary sorts of output, depending on which method is put into the rolling window. For example, computing moving averages requires numerics and a function like `mean()`, and produce averaged numerics; however, rolling window regression takes a data frame and a linear regression method like `lm()`, and generate a complex object that contains coefficients, fitted values, and etc. In addition, the same function will be called and applied to each window of elements independently, which means programming is

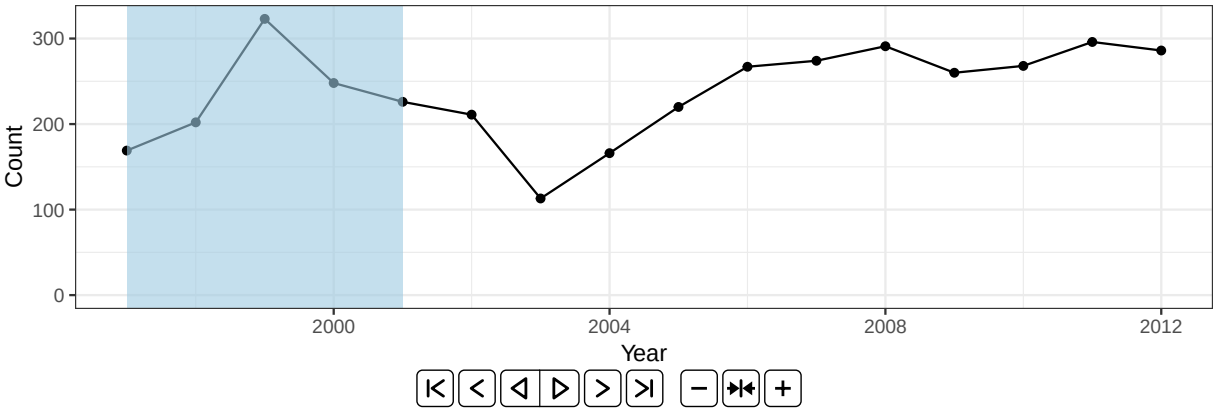

Figure 6: *An illustration of window of size 5 stretching over annual tuberculosis cases in Australia. (Animation needs to be viewed with Adobe Acrobat Reader.)*

done with for-loop statements. But functional programming emphasizes on expressions instead of statements.

The **purrr** package sets a great example of functional programming in R. It provides a complete and consistent set of tools to iterate each element of a vector with a function. Rolling window not just iterates but roll over a sequence of elements, with `slide()`, `tile()` and `stretch()`. `slide()` expects one input, `slide2()` two inputs, and `pslide()` multiple inputs. For type stability, the functions always return lists. Other variants including `*_lgl()`, `*_int()`, `*_dbl()`, `*_chr()` return vectors of the corresponding type, as well as `*_dfr()` and `*_dfc()` for row-binding and column-binding data frames respectively. Their multiprocessing equivalents prefixed by `future_` enable rolling in parallel. This family of functions empowers users to incorporate window-related operations in their workflows.

5.3 Modularity

Modular programming is adopted while designing the **tsibble** package. Modularity benefits users with variety and flexibility and developers with easy maintenance.

All user-facing functions can be roughly organized into three major chunks according to their functionality: vector functions (1d), table verbs (2d), and window family. Each chunk is an independent module, but works interdependently. Vector functions in the package mostly deal with time. When collapsing a tsibble to less granular interval, these atomic functions can be combined with the `index_by()` table verb to accomplish this. A different function results in easily switching to aggregation of different time resolution. Since these functions are not exclusive to a tsibble, they can be used in a variety of applications in conjunction with other packages. On the other hand, these tsibble verbs can incorporate many third-party vector

functions to step out of current tsibble zone. It is generally easier to trace back the errors users encounter from separating 1d and 2d functions. (lost in a web of functions)

5.4 Extensibility

As a fundamental infrastructure, extensibility is a design decision that is focused on from the start of **tsibble**'s development. Contrary to the "data first" principle for end users, extensibility is developer focused and would be mostly used in dependent packages, which heavily relies on S3 classes and methods in R (Wickham 2018). It can be extended in two major aspects: custom index and new tsibble class.

Time representation could be arbitrary, for example R's native `POSIXct` and `Date` for versatile date-times, nano time for nanosecond resolution implemented in **nantotime** (**R-nanotime**), and pure numbers in simulations. Yet ordered factors can also be a source of time, such as month names from January to December and weekdays from Monday to Sunday. Tsibble supports an extensive range of index types from numerics to nano time, but there might be custom indexes used for some occasions, for example school semesters. These academic terms vary from one institute to another within an academic year, which is defined differently from a calendar year. New index would be immediately recognized by the software upon defining `index_valid()`, as long as it can be ordered from past to future. The interval regarding semesters is further outlined through `pull_interval()`. As a result, the rest software methods such as `has_gaps()`, `count_gaps()` and `fill_gaps()` will have instant support for data that contains this new index.

The class of tsibble is an underlying basis of temporal data, and there is a demand for sub-classing a tsibble. For example, a fable is actually an extension to a tsibble, mentioned in Section 4.1. A low-level constructor `new_tsibble()` provides a vehicle to easily create a new subclass. First of all, this new object itself is a tsibble. It perhaps needs more metadata than those of a tsibble, that gives rise to a new data extension, like prediction distributions to a fable. Tsibble verbs are also S3 generics. Developers will be able to implement these verbs for the new class if needed.

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 (Bureau of Transportation

Statistics 2018). 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 tools to easily locate the duplicates in the data with `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 7 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 8 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 7.

A closer examination of some big airports across the US will give an indication about how well the busiest airports manage the outflow traffic on a daily basis. A subset that contains observations for Houston (IAH), New York (JFK), Kalaoa (KOA), Los Angeles (LAX) and Seatle (SEA) airports is obtained first. The succeeding operations compute delayed percentages every day at each airport, which are framed as grey lines in Figure 9. Winter months tend to fluctuate a lot compared to the summer across all the airports. 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 operation with large window size smoothes out the fluctuations and gives a stable trend around 25% over the year. The LAX airport has seen a gradual decline in delays over the year, whereas the SEA airport has

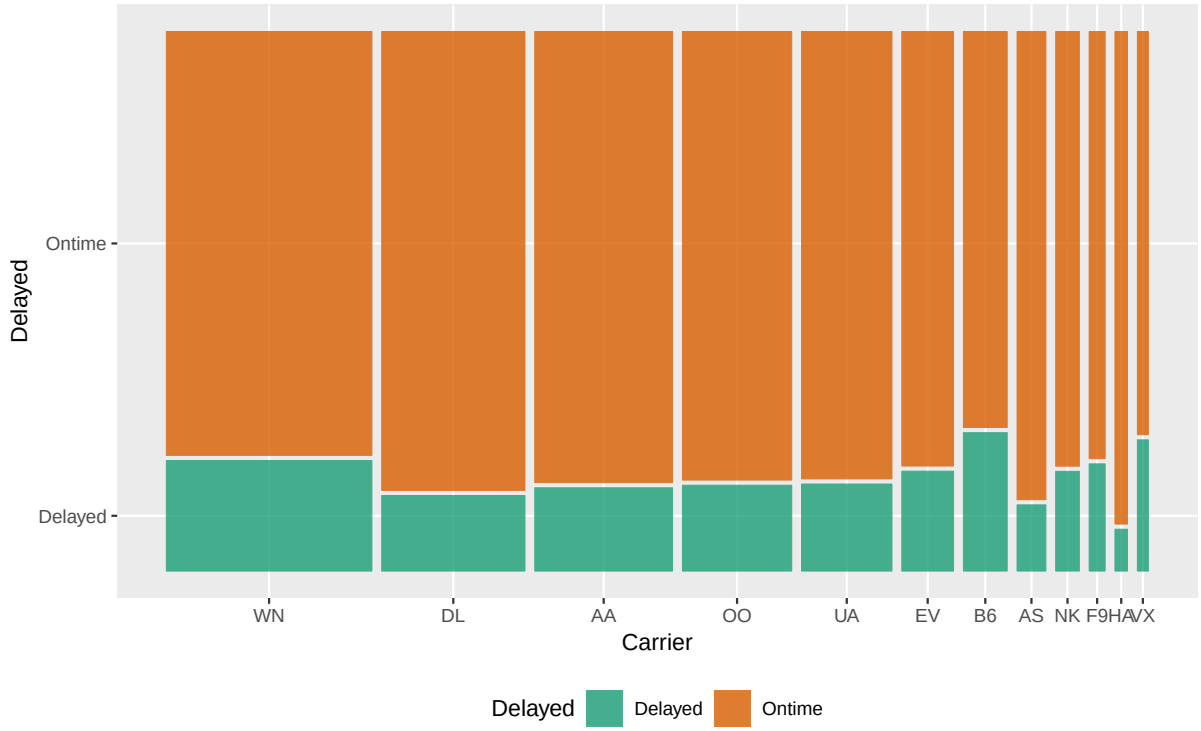

Figure 7: *Mosaic plot showing the association between the size of airline carriers and the delayed proportion of departure in 2017. Southwest Airlines is the largest operator, but does not operate as efficient as Delta. Hawaiian Airlines, also as a small operator, outperforms the rest.*

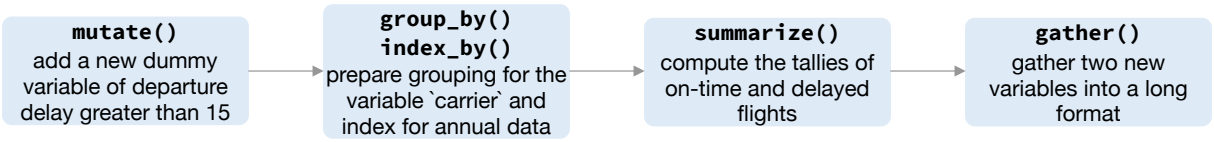

Figure 8: *Flow chart illustrates the pipeline that pre-processes the data for creating Figure 7.*

a steady control of delays over time. The IAH and JFK airports have undergone more delays in the middle of year, while the KOA has the inverse pattern with higher delay percentage in both ends of the year.

What time of day and day of week should we travel to avoid suffering from horrible delay? Figure 11 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.

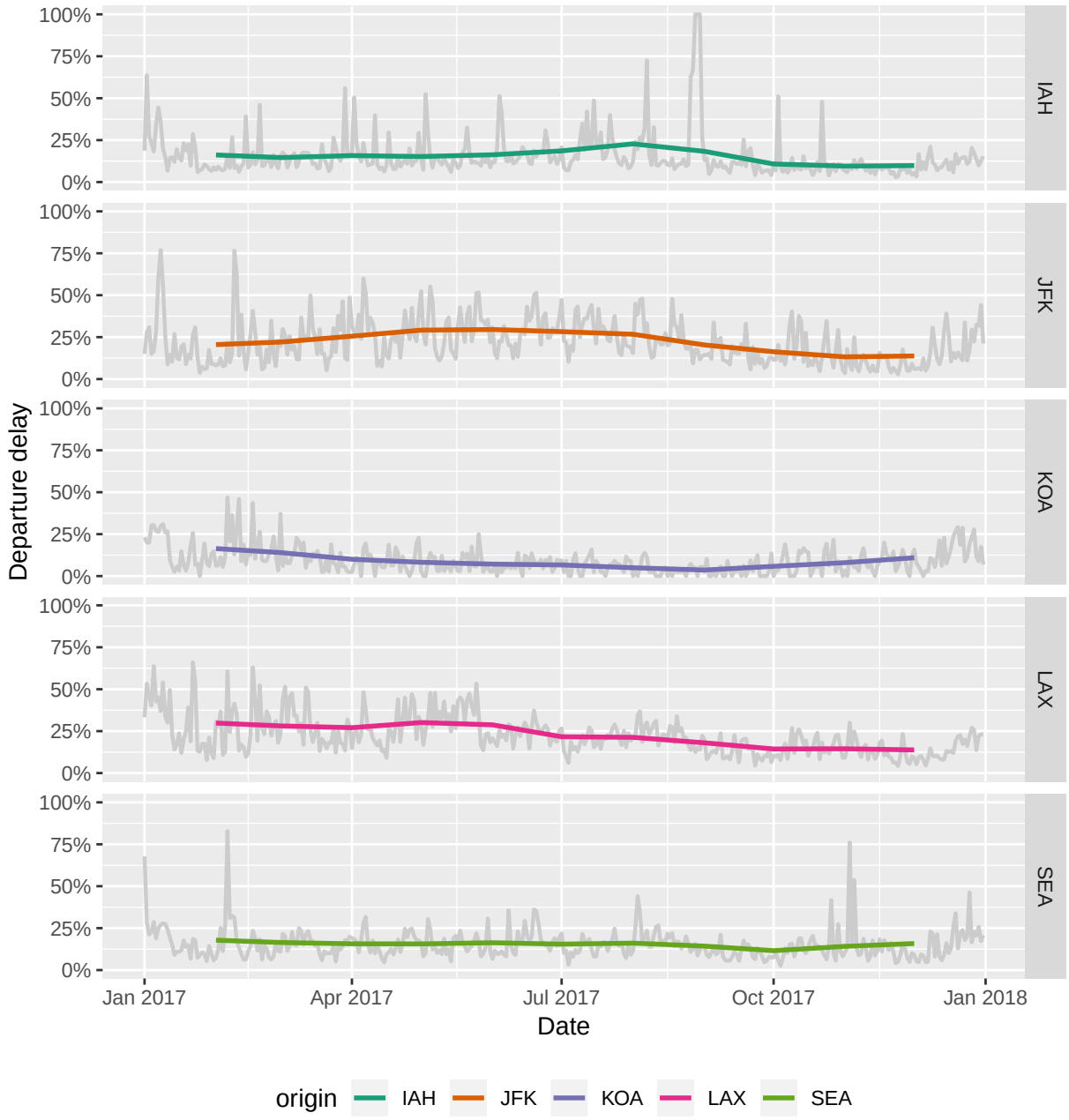

Figure 9: *Daily delayed percentages for departure with two-month moving averages overlaid at five international airports. There are least fluctuations and relatively fewer delays observed at KOA airport. The estimates of temporal trend are around 25% across the other four airports, but highlight different time periods of severe delays.*

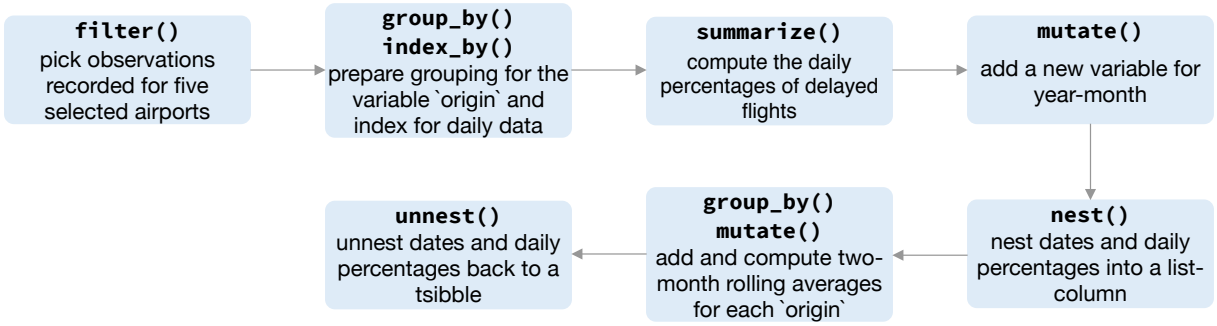

Figure 10: *Flow chart illustrates the pipeline that pre-processes the data for creating Figure 9.*

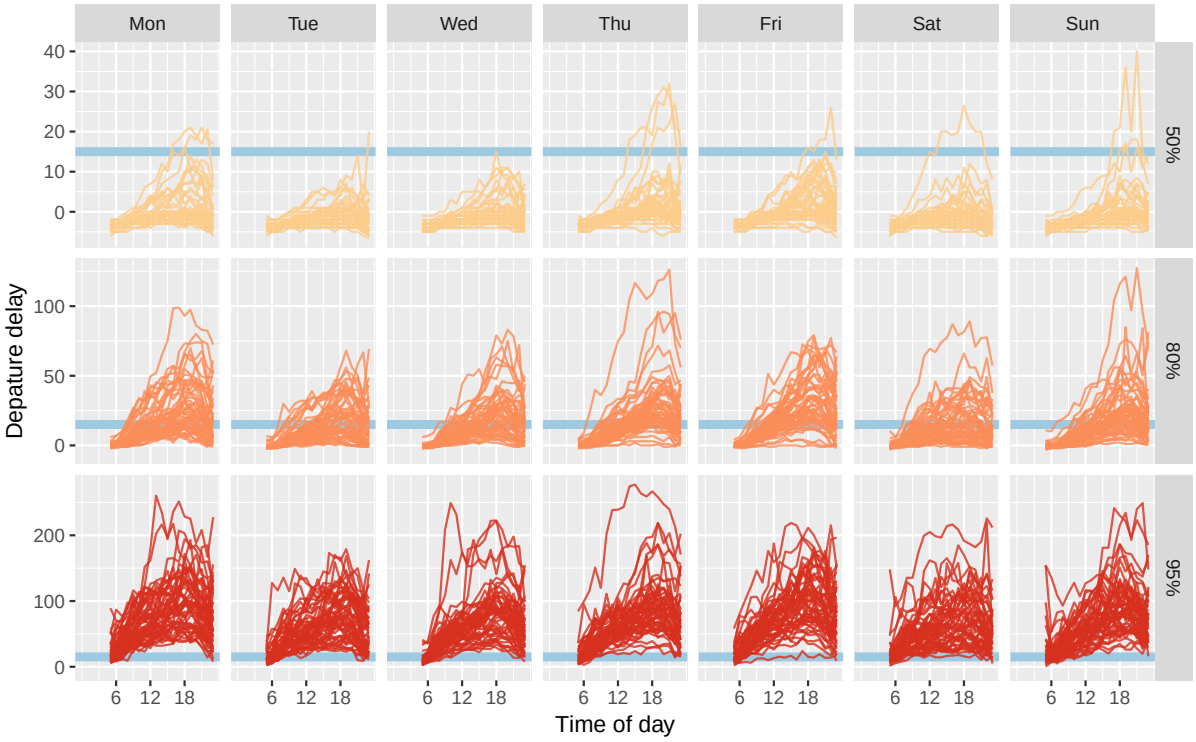

Figure 11: *Small multiples of lines about departure delay against time of day, faceting day of week and 50%, 80% and 95% quantiles. A blue horizontal line indicates the 15-minute on-time standard to help grasp the delay severity. Passengers are apt to hold up around 18 during a day, and are recommended to travel early. The variations increase substantially as the upper tails.*

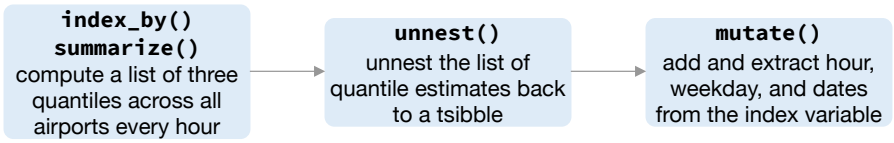

Figure 12: *Flow chart illustrates the pipeline that pre-processes the data for creating Figure 11.*

6.2 Smart-grid customer data in Australia

Sensors have been put up to collect data for the project of smart city across major cities in Australia. One of the trials is monitoring households' electricity usage through installed smart meters in the area of Newcastle over 2010–2014 (Department of the Environment and Energy 2018). Year 2013 has been sliced to examine temporal patterns of customer's energy consumption with **tsibble** in this paper. Half-hourly general supply in kwH have been recored for 2,924 customers in the data set, resulting in 46,102,229 observations in total. Customer's demographic data provides explanatory variables other than time in a different data table. Two data tables might be joined to explore different sources that contribute to daily electricity use when needed.

During a power outage, electricity usage for some households may become unavailable, thus resulting in implicit missing values in the database. Gaps in time occur to 17.9% of the households in this dataset. It would be interesting to explore these missing patterns as part of preliminary analysis. Since the smart meter has been installed at different dates for each household, it is reasonable to assume that the records are obtainable for different time lengths for the households, instead of full lengths across the households. Figure 13 reveals the gaps for the top 49 households and the rest collapsed into a single batch arranged in rows from high to low in tallies. Missing values can be found at any time point during the entire time span. A small number of customers have undergone energy unavailability in consecutive hours, indicated by a line range in the plot. On the other hand, the majority suffer occasional breakdown with more frequent happenings in January.

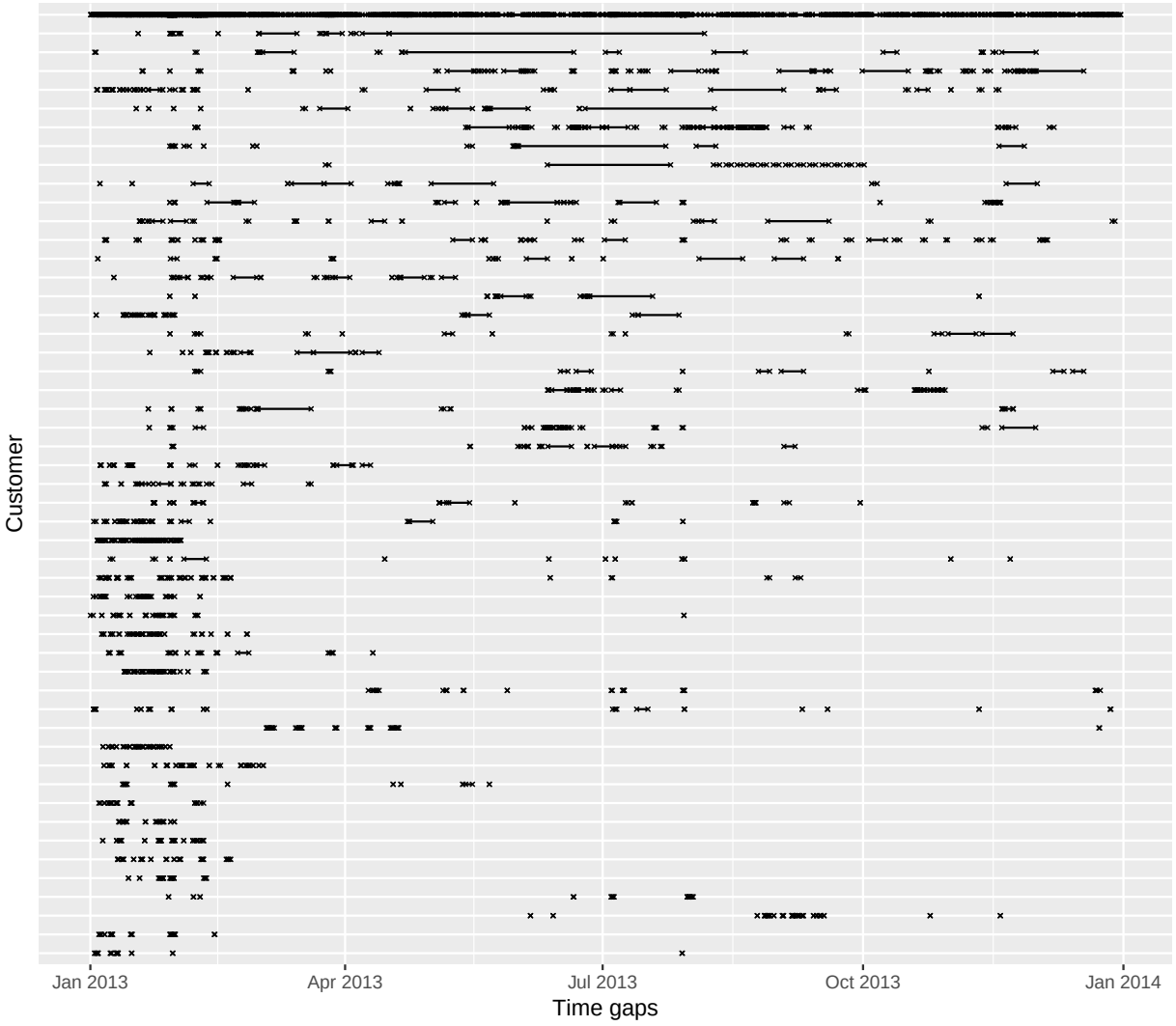

Figure 13: *Time gap plots for the top 49 customers with most implicit missing values and the rest grouped into the one. Each cross represents the missing in time and a line between two dots shows continuous missingness over time. Each row corresponds to one customer describes the incidences of time gaps.*

Aggregation across all individuals helps to sketch a big picture about if the representative behaviors change over time, laid out in the calendar display shown as Figure 14. Each glyph represents the daily pattern of residential electricity usage every thirty minutes with higher consumption in the time when human are more active. The daily snapshots diverge in traces, depending on the season in the year. During the summer months (December and January), the late-afternoon peak becomes predominant driven by the use of air conditioning, especially in the hot days with daily average temperature greater than 25 degrees. However, the winter time (July and August) sees two peaks in a day, which is probably due to heating in the morning and evening. This also illustrates how the tsibble data can easily integrate with other tools and graphics.

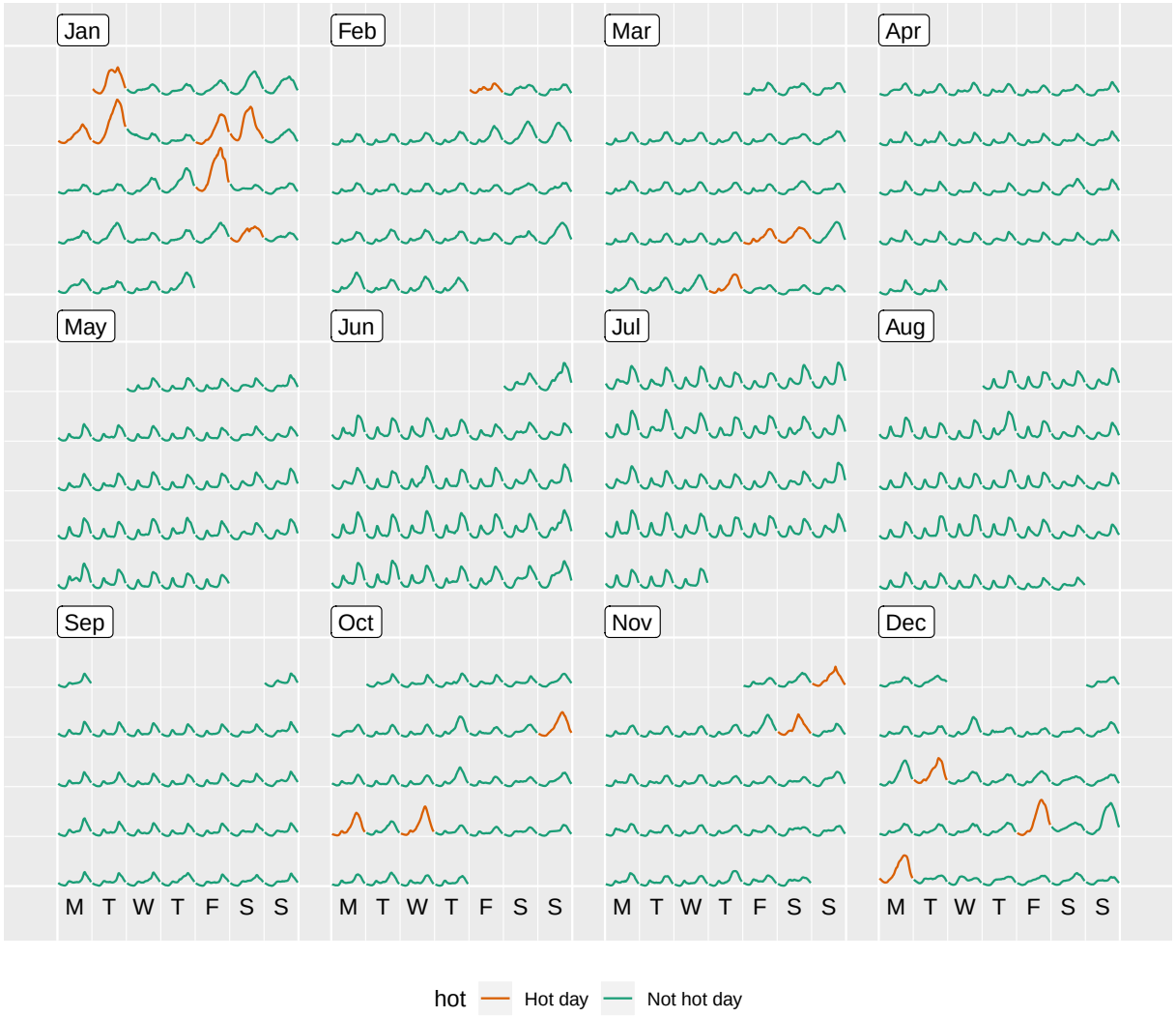

Figure 14: *Half-hourly average electricity use across all customers in the region, organized in the calendar format. Distinctive patterns between summer and winter months occur to daily residential energy consumption. Air conditioning drives the peak in the late afternoon in January and December, however, peaks in the morning and evening are more pronounced in the winter months due to heating use.*

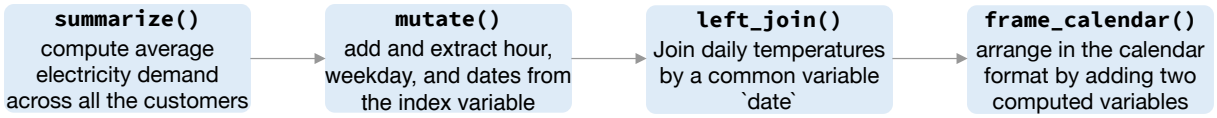

Figure 15: *Flow chart illustrates the pipeline that pre-processes the data for creating Figure 14.*

7 Conclusion and future work

A new data abstraction representing temporal data named as "tsibble" has been proposed, spotlighting the "tidy data" principles brought to time domain. Tidy data begins to take shape in the state of time with the introduction of contextual semantics: index and key. Declared index provides direct support to time variable; variables that comprise the key defines study subjects over time. These semantics further determines unique data entries required for a valid tsibble. No matter how temporal data arrives, tsibble respects time index and keeps data richness. A tsibble frictionlessly pops into transformation, visualization, modelling and smoothly shifts amongst, allowing for rapid iterations in gaining data insights.

A missing piece of the *tsibble* data is to enable user-defined calendars and respect structural missing observations. For example, a call center operates only between 9:00am and 5:00pm on week days and stock trading resumes on Monday straight after Friday. No data available outside trading hours would be labelled as structural missingess, which *tsibble* currently disregards. However, few R packages provide functionality to create and manage many sorts of calendars, including market-specific business calendar. This delays the implementation. Generally, custom calendars would be easily embedded into the tsibble framework. Consequently these tsibble operators, like `fill_gaps()`, would work out of box; forecasts would be generated within its definable time range.

The **tsibble** package provides the grammar of temporal data manipulation, regardless of how the data is stored. Currently, it works for managing and manipulating temporal data frames in memory locally. But it is possible to work with remote tables stored in databases, such as SQLite and MySQL, using exactly the same tsibble code. This is the future work.

Acknowledgements

The authors thank Mitch O'Hara-Wild for many discussions on the software design.

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.

Bureau of Transportation Statistics (2018). *Carrier On-Time Performance*. 1200 New Jersey Avenue, SE Washington, DC 20590. https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236 (visited on 09/26/2018).

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

Department of the Environment and Energy (2018). *Smart-Grid Smart-City Customer Trial Data*. Australian Government, Department of the Environment and Energy. https://data.gov.au/dataset/4e21dea3-9b87-4610-94c7-15a8a77907ef (visited on 11/19/2018).

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.

Kimball, R & J Caserta (2011). *The Data WarehouseÂ ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. John Wiley & Sons.

Pebesma, E (2018). Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal* **10**(1), 439–446.

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.

Sutherland, P, A Rossini, T Lumley, N Lewin-Koh, J Dickerson, Z Cox & D Cook (2000). Orca: A Visualization Toolkit for High-Dimensional Data. *Journal of Computational and Graphical Statistics* **9**(3), 509–529.

Swayne, DF, D Cook & A Buja (1998). XGobi: Interactive Dynamic Data Visualization in the X Window System. *Journal of Computational and Graphical Statistics* **7**(1), 113–130.

Swayne, DF, D Temple Lang, A Buja & D Cook (2003). GGobi: evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis* **43**, 423–444.

Tierney, NJ & DH Cook (2018). *Expanding tidy data principles to facilitate missing data exploration, visualization and assessment of imputations*. eprint: arXiv:1809.02264.

Wang, E, D Cook & RJ Hyndman (2018). *Calendar-based graphics for visualizing people's daily schedules*. eprint: arXiv:1810.09624.

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

Wickham, H (2018). *Advanced R*. 2nd ed. Chapman & Hall. https://adv-r.hadley.nz/.

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

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.