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

28 August 2019

JEL classification: C88, C81, C82, C22, C32

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, and heterogeneous data types. On the other hand, the software supporting time series modeling and forecasting, makes strict assumptions on the data to be provided, typically requiring a matrix of numerical 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, modeling and forecasting routines. Tidy data principles are extended to temporal data by: (1) mapping the semantics of a dataset into its physical layout; (2) including an explicitly declared index variable representing time; (3) incorporating a "key" comprising 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 contexts. A sound data pipeline facilitates a fluent 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, with many different time contexts. For example, time can have various resolutions (hours, minutes, and seconds), and can be associated with different time zones with possible adjustments such as daylight saving time. Time can be regular (such as quarterly economic data or daily weather data), or irregular (such as 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, and multiple grouping factors. Temporal data may comprise the occurrence of time-stamped events, such as flight departures.

Despite this variety and heterogeneity, little organization or conceptual oversight on how one should get the wild data into a tamed state is provided for temporal data. Analysts are expected to do their own data preprocessing and take care of anything else needed to allow further analysis, which leads to a myriad of ad hoc solutions and duplicated efforts.

Wickham & Grolemund (2016) proposed the tidy data workflow, which provides a conceptual framework for exploring data (as described in Figure 1). In temporal domain, data with time information arrives at the "import" stage. A new abstraction, *tsibble*, is the gatekeeper at the "tidy" stage, to verify if the raw temporal data is appropriate for downstream analytics. The exploration loop will be aided with declarative grammars, yielding more robust and accurate analyses.

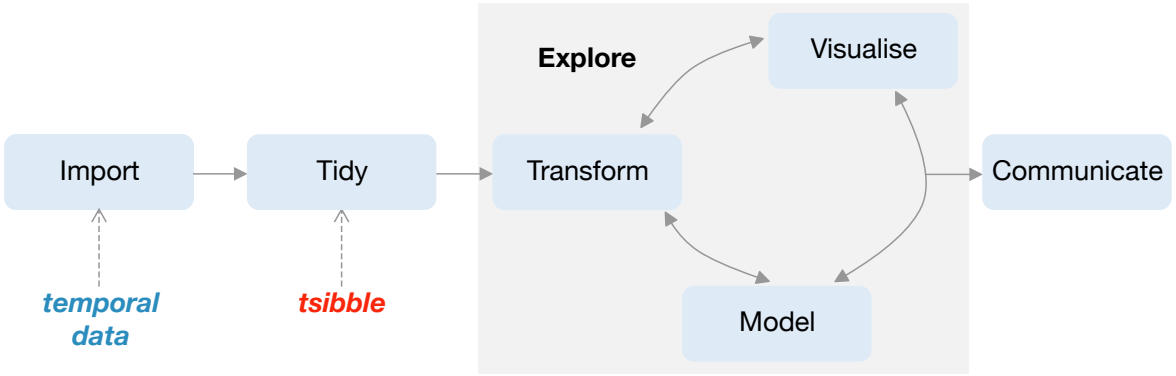

Figure 1: *Annotation of the data science workflow regarding temporal data, drawn from Wickham & Grolemund (2016). The new data structure, tsibble, makes the connection between temporal data input, and downstream analytics. It provides elements at the "tidy" step, which produce tidy temporal data for temporal visualization 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 future work.

2 Data structures

2.1 Comparing time series and longitudinal data

Temporal data problems often fall into two types of analysis, time series and longitudinal. Both of these may have very similar data input, but the representation for modelling is typically different. Time series analysis tends to focus on the dependency with series, and the cross-correlation between series. Longitudinal analysis tends to focus on overall temporal patterns across demographic or experimental treatment strata, that incorporates within subject dependency.

Time series can be univariate or multivariate, and require relatively long lengths (i.e., large T) for modeling. With this large T property, the series can be handled as stochastic processes for the primary purpose of forecasting, and characterizing temporal dynamics. Due to an expectation of regularly spaced time, and equal lengths across series, multivariate time series are typically assumed to be in the format where each column contains a single time series, and time is specified implicitly. This also implies that data are columns of homogeneous types: either all numeric or all non-numeric. To wrestle data, in whatever format that it arrives in, to this modeling format can be frustrating. The format could be considered to be model-centric, rather than data-centric, and thus throws the analyst into the deep end of the pool, rather than allowing them to gently wade to the modeling stage from the shallow end. The expectation is that the "model" is at the center of the analytical universe. This is contrary to the **tidyverse** conceptualization (Figure 1), which holistically captures the data workflow. More support needs to be provided, in the form of consistent tools and data structures, to transform the data into the analytical cycle.

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, and in different quantities. The primary format required for modeling is stacked data, blocks of measurements for each individual, with columns indicating panels, times of measurement and the measurements themselves. An appealing feature is that data is structured in a semantic manner with reference to observations and variables, with panel and time explicitly stated.

2.2 Existing data standards

In R (R Core Team 2018), time series and longitudinal data are of different representations. The native `ts` object and the enhancement of **zoo** (Zeileis & Grothendieck 2005) and **xts** (Ryan & Ulrich 2018), assemble time series into wide matrices with implicit time indexes. If there are multiple sub-groups, like country or product type, these would be kept in different data objects. A relatively new R package **tibbletime** (Vaughan & Dancho 2018b) proposed a data class of *time tibble* to represent time series in heterogeneous long format. It only requires an index variable to be declared. However, this is insufficient, and a more rigid data structure is required for temporal analytics and modeling. The **plm** (Croissant & Millo 2008) and **panelr** (Long 2019) packages both manage longitudinal data in long format.

Stata (StataCorp 2017) provides two commands, `tsset` and `xtset`, to declare time series and panels respectively, both of which require explicit panel id and time index specification. Different variables would be stored in multiple columns. The underlying data arrangement is only long form, for both types of data. Both groups of functions can be applied interchangeably to whether the data is declared for time series or longitudinal data. The SAS software (SAS Institute Inc. 2018) also handles both types of data in the same way as Stata.

2.3 Tidy data

Wickham (2014) coined the term "tidy data", to standardize the mapping of the semantics of a dataset to its physical representation. In tidy form, rows correspond to observations and columns to variables. Tidy data is a rephrasing of the second and third normal forms from relational databases, but the explanation in terms of observations and variables is easier to understand because it uses statistical terminology.

Multiple time series, with each column corresponding to a measurement is tidy data when the time index is explicitly stored in a column. The stacked data format used in longitudinal data storage is tidy, and accommodates explicit identification of sub-groups.

The tidy data structure is the fundamental unit of the **tidyverse**, which is a collection of R packages designed for data science. The ubiquitous use of the **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. 2019), 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()`, `summarize()`, and `arrange()`. Each verb focuses

on a singular task. Most common data tasks can be rephrased and tackled with these five key verbs, in conjunction with `group_by()` to perform grouped operations.

The **tidyverse** largely formalizes exploratory data analysis. Many in the R community have adopted the **tidyverse** way of thinking and extended it 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). This paper with the associated **tsibble** R package (Wang, Cook & Hyndman 2019) extends the tidy way of thinking to temporal data.

For temporal data, the tidy definition needs additional criteria, that assist in handling the time context. This is addressed in the next section, and encompasses both time series and longitudinal data. It provides a unified framework to streamline the workflow from data preprocessing to visualization and modeling, as an integral part of a tidy data analysis.

3 Contextual semantics

The choice of tidy representation of temporal data arises from a data- and model-oriented perspective, which can accommodate all of the operations that are to be performed on the data in time-based contexts. Figure 1 marks where this new abstraction is placed in the tidy model, which is referred to as a "tsibble". Adapting the "tidy data" principles, in tsibble:

1. Index is a variable with inherent ordering from past to present.
2. Key is a set of variables that define observational units over time.
3. Each observation should be uniquely identified by index and key.
4. Each observational unit should be measured at a common interval, if regularly spaced.

Figure 2 sketches out the data form required for a tsibble, an extension of the tidy format to the time domain. Beyond the layout, tsibble gives the contextual meaning to variables in order to construct the temporal data object, as newly introduced "index" and "key" semantics stated in the definitions 1 and 2 above. Variables other than index and key are considered as measurements. The definitions of 3 and 4 imply that tsibble is *tidier* than tidy data, positioning itself as a model input that gives rise to more robust and reliable downstream analytics.

To materialize the abstraction of the tsibble, a subset of tuberculosis cases (World Health Organization 2018), as presented in Table 1, is used as an example. It contains 12 observations and 5 variables landing in a tidy data form. Each observation comprises the number of people who

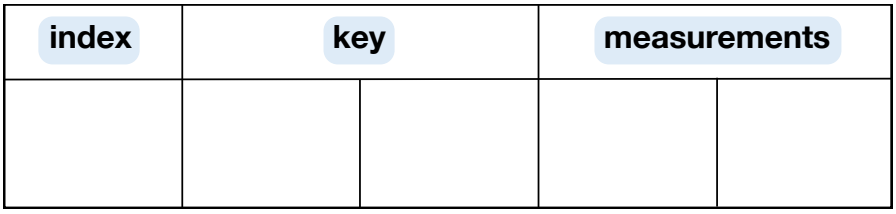

Figure 2: *The architecture of the tsibble structure is built on top of the "tidy data" principles, with temporal semantics: index and key.*

are diagnosed with tuberculosis for each gender at three selected countries in the years of 2011 and 2012. From tidy data to tsibble data, index and key should be declared: column `year` as the *index* variable, and column `country` together with `gender` as the *key* variable forming the observational units. Column `count` is the only measured variable in this data, but the data structure is sufficiently flexible to hold more measurements; for example, slotting the corresponding population size (if known) into the data column for normalizing the count later. Note, this data further satisfies the must-have for the distinct rows determined by index and key, and regularly spans over one-year intervals.

Table 1: *A small subset of estimates of tuberculosis burden collected by World Health Organization 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

The new tsibble structure bridges the gap between raw data and the rigorous state of temporal data analysis. The proposed contextual semantics is the new add-on to tidy data in order to support more intuitive time-related manipulations and enlighten new perspectives for time series and panel model inputs. Index, key and time interval build the three stone pillars to this new semantically structured temporal data. Each is now described in more detail.

3.1 Index

Index is a variable with inherent ordering from past to present.

Time provides the contextual basis for temporal data. Time can be seen in numerous representations, from sequential numerics to the most commonly accepted date-times. Regardless of this diversity, time should be inherently ordered from past to present, so should be the index variable to a tsibble.

Index is an explicit data variable rather than a masked attribute (such as in the `ts` and `zoo` classes), exposing a need for more accessible and transparent time operations. It is often necessary to visualize and model seasonal effects of measurements of interest, meaning that time components, such as time of day and day of week, should be easily extracted from the index. When the index is available only as meta information, it provides an obstacle for analysts by complicating the writing of even simple queries, often requiring special purpose programming. From an analytical point of view this should be discouraged.

3.2 Key

Key is a set of variables that define observational units over time.

What subjects/entities are to be observed over time, leads to the second component of a tsibble–key. The key can consist of empty, single, or multiple variables identifying units measured along the way. When only a single observational unit is present in the table, no key needs to be specified. However, when multiple units exist in the data, the key should be supplied by identifying variables to sufficiently define the units. In longitudinal data, the key can be thought of as "panel" (such as in the Stata) but constrained to a single variable in existing data structures. In tsibble, the key allows for multiple variables of nesting, crossing, or union relations (Wilkinson 2005), that can be useful for forecasting reconciliation (Hyndman & Athanasopoulos 2017; Hyndman et al. 2018) and richer visualization. For example, Table 1 describes the number of tuberculosis cases for each gender across the countries every year. This suggests that the key comprises at least columns `gender` and `country`. Since `country` is nested within `continent`, `continent` can be included in the key specification, but is not compulsory.

Each observation should be uniquely identified by index and key.

Inspired by a "primary key" (Codd 1970), a unique identifier for each observation in a relational database, the tsibble key also uniquely identifies each observational unit over time. When

constructing a tsibble, any duplicates of key-index (i.e. key-value) pairs will fail, because duplicates signal a data quality issue, which would likely affect subsequent analyses and hence decision making. For example, either `gender` or `country` alone is not enough to be the key for the tuberculosis data. Analysts are encouraged to better understand the data, or reason about the process of data cleaning when handling duplicates. Figure 3 peels the tidy module with clear routes required for a tsibble. The rigidity of tsibble, as the fundamental data infrastructure, warrants the validity of temporal data analysis for the later stage.

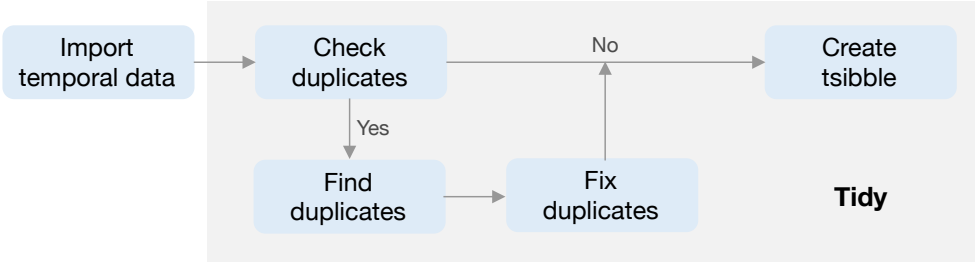

Figure 3: *Details about the tidy stage for a tsibble. Built on top of "tidy data", each observation should be uniquely identified by index and key, thereby no duplicated key-index pairs.*

Since observational units are embedded, modelling and forecasting across units and time in a tsibble will be simplified. The tsibble key plays a role of the central transit hub in connecting multiple tables managed by the data, models, and forecasts. This neatly decouples the expensive data copy from downstream summaries, significantly reducing the storage space.

3.3 Interval

Each observational unit should be measured at a common interval, if regularly spaced.

The principal divide of temporal data is regularly versus irregularly spaced data. Event data typically involves irregular time intervals, such as flight schedules or customer transactions. This type of data can flow into event-based data modeling, but would need to be processed, or regularized, to fit models that expect data with a fixed-time interval.

There are three possible interval types: a number, unknown, and irregular. To determine the interval for regularly spaced data, tsibble computes the greatest common divisor, and is a number. If only one observation is available for each unit, which may occur after aggregating data, the interval is reported as unknown. When the data arrives with irregular time, like event data, the interval would be specified as irregular, to prevent the tsibble creator attempting to guess an interval.

To abide by the "tidy data" rules – "Each type of observational units should form a table" – in a tsibble each observational unit shares a common interval. This means that a tsibble will report one single interval, whether the data has a fixed or mixed set of intervals. To handle mixed interval data, it should be organized into separate tsibbles for a well-tailored analysis.

This tiny piece of information, the interval, is carried over for tsibble-centric operations. For example, this makes implicit missing time handling convenient, and harmoniously operates with statistical calculations, and models, on seasonal periods.

4 Temporal data pipelines

A data pipeline describes the flow of data through an analysis, and can generally assist in conceptualizing the process for a stream of problems. Mcilroy, Pinson & Tague (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 based on their standard streams, so that the output of each program becomes the input to another. The Extract, Transform, and Load (ETL) process from recent data warehousing literature dating back to Kimball & Caserta (2011) outlines the workflow to prepare data for analysis, and can also be considered a data pipeline. Buja et al. (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.

A fluent data pipeline anticipates a standard data structure. The tsibble data abstraction lays the plumbing for data analysis modules of transformation, visualization and modeling in temporal contexts. It provides a data infrastructure to a new ecosystem, **tidyverts**, a play on tidyverse that acknowledges the time series analysis purpose, in the spirit of "tidy data" to the **tidyverse**.

4.1 Transformation

The **tsibble** package not only provides a "tsibble" data object but also a domain specific language in R for transforming temporal data. It takes advantage of the wrangling verbs implemented in the **dplyr** package, and develops a suite of new tools for facilitating temporal manipulation with ease primarily in two aspects: implicit missingness handlers and time-aware aggregations.

Implicit missingness are values that should be present but are absent. Regarding regularly-spaced temporal data, data entries that should be available at certain time points but are missing, leaving gaps in time. It will be a problem when temporal models and operations like lag/lead are applied. A family of verbs help to explore implicit missing values and turn them into the state of explicitness to eliminate potential concerns, listed as follows:

- `has_gaps()` checks the existence of time gaps.
- `scan_gaps()` reveals all implicit missing observations.
- `count_gaps()` summarizes the time ranges that are absent from the data.
- `fill_gaps()` turns them into explicit ones, along with imputing by values or functions.

These verbs are evocative and of simple interface. They by default look into gaps for each individual time period. Switching on the option `.full = TRUE` will fill in the full-length time span, creating fully balanced panels with respect to longitudinal study.

The other key function, an adverb `index_by()`, is the counterpart of `group_by()` in **dplyr**, grouping and partitioning the index only. It is most useful to be used in conjunction with `summarize()`, and thus aggregations to upper-level time resolutions will take place. This combination automatically produces new index and interval, and helps to regularize data of irregular interval.

Besides new verbs, the **dplyr** vocabulary has been adapted and expanded to facilitate temporal transformations. The **dplyr** suite showcases the general-purpose verbs for effectively manipulating tabular data. But these verbs need handling with care due to the context switch. A perceivable difference is summarizing variables between normal data and tsibble using `summarize()`. The former will reduce to a single summary, whereas the latter will obtain the index and their corresponding summaries.

Attention has been paid to warning and error handling. The principle that underpins most verbs is a tsibble in and a tsibble out, thereby striving to maintain a valid tsibble over the course of transformation. If the desired temporal ordering is changed by row-wise verbs (such as `arrange()` and `slice()`), a warning is broadcast. If a tsibble cannot be maintained in the output of a pipeline module (likely occurring to column-wise verbs), for example the index is dropped by `select()`-ing, an error informs users of the problem and suggests alternatives. This avoids surprising users and reminds them of the time context. In general, users who are already familiar with the **tidyverse**, should have less resistance to learning the new semantics and verbs.

4.2 Visualization

The **ggplot2** package (as the implementation of grammar of graphics) builds a powerful graphical system to declaratively visualize data. The data underpinning of **ggplot2** is tidy data, and in turn tsibble integrates well with **ggplot2**. The integration encourages more flexible graphics for exploring temporal structures via index, and individual or group differences via key.

Line charts are universally accepted for ordered data, such as time series plots or spaghetti plots, depending on the fields. But they end up with exactly the same grammar: chronological time mapped to the horizontal axis, and the interested measurement on the vertical axis, for each unit. Many specialist plots centering around time series or longitudinal data, hence can be described and re-created under the umbrella of the grammar and **ggplot2**.

4.3 Model

Modeling is crucial to explanatory and predictive analytics, where time series and longitudinal data analysis start to diverge. Nevertheless, tsibble as a model-oriented object can flow into both types of models, with new semantics (index and key) to be internally utilized, to accelerate modelling.

Most time series models are univariate, such as ARIMA and Exponential Smoothing, modelling temporal dynamics for each series independently. The **fable** package (O'Hara-Wild, Hyndman & Wang 2019) provides a tidy forecasting framework built on top of tsibble, which is under development, with the goal of promoting transparent and human-centered forecasting practices. With the presence of the key, tsibble holds many series. Since models are fundamentally scalable, the `model()` and `forecast()` generics will take care of fitting and forecasting univariate models to each series across time in a tsibble at once.

Panel data models, however, put emphases on overall, within, and between variations both across individuals and time. Fixed and random effects models can be developed in line with the **fable** design.

4.4 Summary

To sum it up, the tsibble abstraction provides a formal organization of forwarding tidy data to model-oriented temporal data. The supporting operations can be chained for sequencing analysis, articulating a data pipeline. As Friedman & Wand (2008) stated, "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." A mini snippet below glues transformation and forecasting together, to envision the fluent pipeline.

```
pedestrian %>%
  fill_gaps() %>% # turn implicit missingness to explicit
  filter(year(Date_Time) == 2016) %>%  # subset data of year 2016
  model(arima = ARIMA(Count)) %>% # fit arima to each sensor
  forecast(h = days(2)) # forecast 2 days ahead
```

The `pedestrian` dataset (City of Melbourne 2017) contains hourly tallies of pedestrians at four counting sensors in 2015 and 2016 in inner Melbourne, available in the **tsibble** package. The pipe operator `%>%` introduced in the **magrittr** package (Bache & Wickham 2014) chains the verbs, read as "then". A sequence of functions are composed in a way that can be naturally read from left to right, which improves the code readability.

It coordinates a user's analysis making it cleaner to follow, and permits a wider audience to document the data analysis through code without getting lost in a jungle of computational intricacies. 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.

5 Software structure and design decisions

The **tsibble** package development follows closely to the **tidyverse** design principles (Tidyverse Team 2019).

5.1 Data first

The primary force that drives the software's design choices is "data". All functions in the package **tsibble** start with `data` or its variants as the first argument, namely "data first". This lays out a consistent interface and 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 contexts, particularly useful when handling a large collection of data. The contextual information provided by the `print()` function, shown below from Table 1, contains (1) data dimension with its shorthand time interval, alongside time zone if date-times, (2) variables that

constitute the "key" with the number of units. These summaries 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, and often apply to other 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.

Figure 4 shows 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 by computing average counts.

Rolling window adapts to functional programming, which the **purrr** package (Henry & Wickham 2019a) sets a good example. These functions accept and return arbitrary inputs and outputs, with arbitrary methods. For example, moving averages anticipate numerics and produce averaged numerics via `mean()`. However, rolling window regression feeds a data frame into a linear regression method like `lm()`, and generates a complex object that contains coefficients, fitted values, and etc.

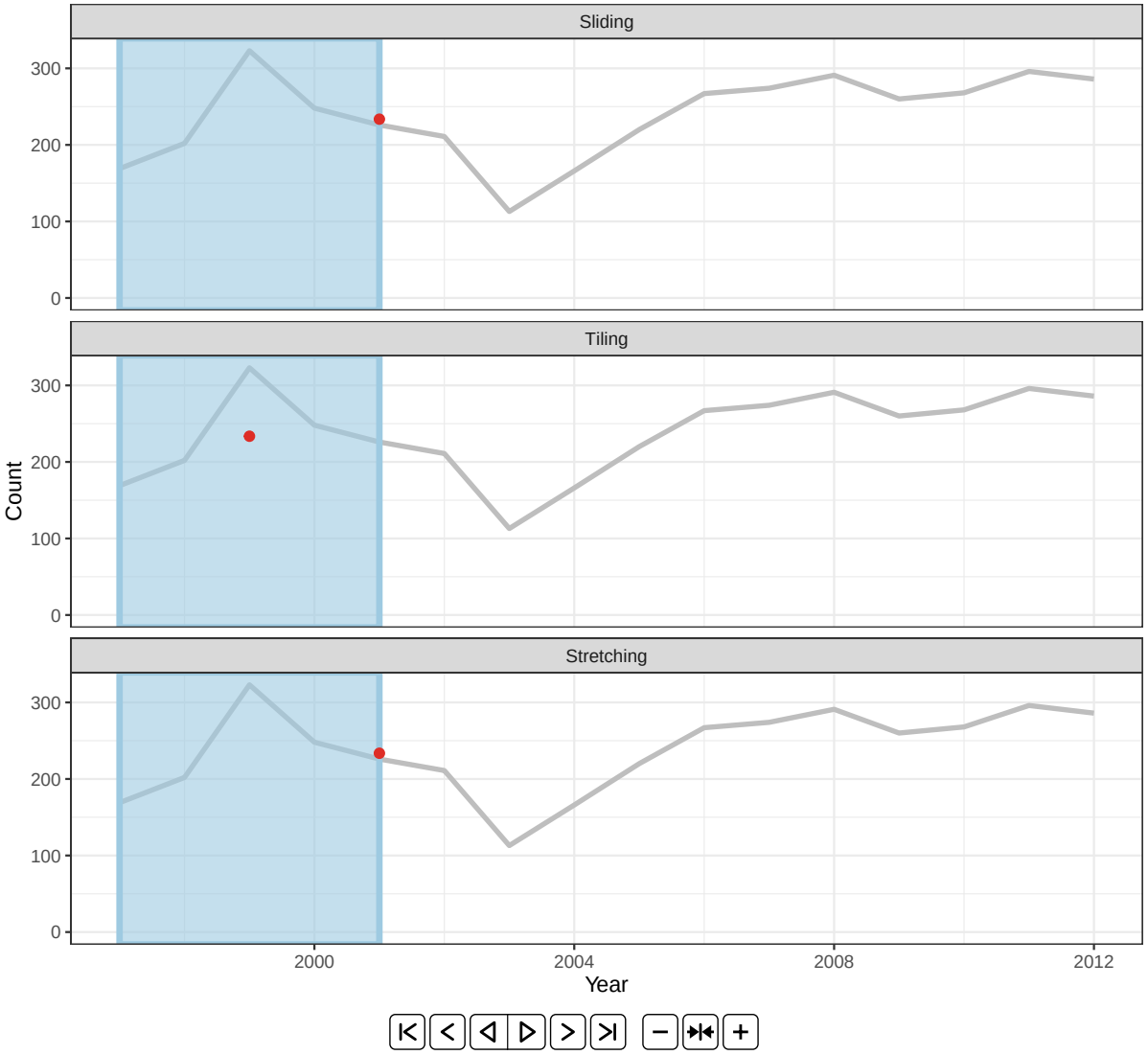

Figure 4: *An illustration of window of size 5 computing rolling averages over annual tuberculosis cases in Australia with respect to sliding, tiling and stretching. (Animation needs to be viewed with Adobe Acrobat Reader.)*

Rolling window not just iterates but rolls over a sequence of elements of a fixed window. A complete and consistent set of tools are available for facilitating window-related operations, a family of `slide()`, `tile()`, `stretch()`, and their variants. `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 types, 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 (Bengtsson 2019; Vaughan & Dancho 2018a).

5.3 Modularity

Modular programming is adopted in the design of the **tsibble** package. Modularity benefits users by providing small focused and cleaner chunks, and provides developers with simpler 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 operate on time. The atomic functions (such as `yearmonth()` and `yearquarter()`) can be embedded in the `index_by()` verb to collapse a tsibble to a less granular interval. Since they are not tied to a tsibble, they can be used in a broader range of data applications not constrained to tsibble. On the other hand, the table verbs can incorporate many other vector functions from a third party, like the **lubridate** package.

5.4 Extensibility

As a fundamental infrastructure, extensibility is a design decision that was employed 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). The package 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 in **nanotime** (Eddelbuettel & Silvestri 2018), and numerics in simulation. Ordered factors can also be a source of time, such as month names, January to December, and weekdays, Monday to Sunday. The **tsibble** package 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 institution to another, within an academic year which is defined differently from a calendar year. A new index would be immediately recognized upon defining `index_valid()`, as long as it can be ordered from past to future. The interval regarding semesters is further outlined through `interval_pull()`. As a result, all tsibble methods such as `has_gaps()` and `fill_gaps()` will have instant support for data that contains this new index.

The class of tsibble is an underpinning for temporal data, and sub-classing a tsibble will be a demand. A low-level constructor `new_tsibble()` provides a vehicle to easily create a new subclass. 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, for example prediction distributions to a forecasting tsibble.

5.5 Tidy evaluation

The **tsibble** packages leverages the **tidyverse** grammars and pipelines through tidy evaluation (Henry & Wickham 2019c) via the **rlang** package (Henry & Wickham 2019b). In particular, the table verbs extensively use tidy evaluation to evaluate computation in the context of tsibble data and spotlights the "tidy" interface that is compatible with the **tidyverse**. This not only saves a few keystrokes without explicitly repeating references to the data source, but the resulting code is typically cleaner and more expressive, when doing interactive data analysis.

6 Case studies

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

The dataset of on-time performance for US domestic flights in 2017 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 should 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()`. The problematic entries are shown below.

```
#>   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 was 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. As flight NK630 is usually scheduled at 17:45 from Chicago to New York (discovered by searching the full database), a decision is made to remove the first row from the duplicated entries before proceeding to the tsibble creation.

This dataset is intrinsically heterogeneous, encoded in numbers, strings, and date-times. The tsibble framework, as expected, incorporates this type of data without any loss of data richness and heterogeneity. To declare the flight data as a valid tsibble, column `sched_dep_datetime` is specified as the "index", and column `flight_num` as the "key". This data happens to be irregularly spaced, and hence switching to the irregular option is necessary. The software internally validates if the key and index produce distinct rows, and then sorts the key and the index from past to recent. When the tsibble creation is done, the print display is data-oriented and contextually informative, including dimensions, irregular interval with the time zone (`5,548,444 x 22`) and the number of observational units (`NA`).

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

Transforming a tsibble for exploratory data analysis with a suite of time-aware and general-purpose manipulation verbs can result in well-constructed pipelines. A couple of use cases are described to show how to approach the interested questions by wrangling the tsibble while maintaining its temporal context.

What time of day and day of week should passengers travel to avoid suffering from horrible delay? Figure 5 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 computed. This pipeline is initialized by regularizing and reshaping the list of the upper quantiles of departure delays for each hour. To visualize the temporal profiles, the time components (for example hours and weekdays) are extracted from the index. The flow

chart (Figure 6) demonstrates the operations undertaking in the data pipeline. The input to this pipeline is a tsibble of irregular interval for all flights, and the output ends up with a tsibble of one-hour interval by quantiles. To reduce the likelihood of suffering a delay, it is recommended to avoid the peak hour around 6pm (18) from Figure 5.

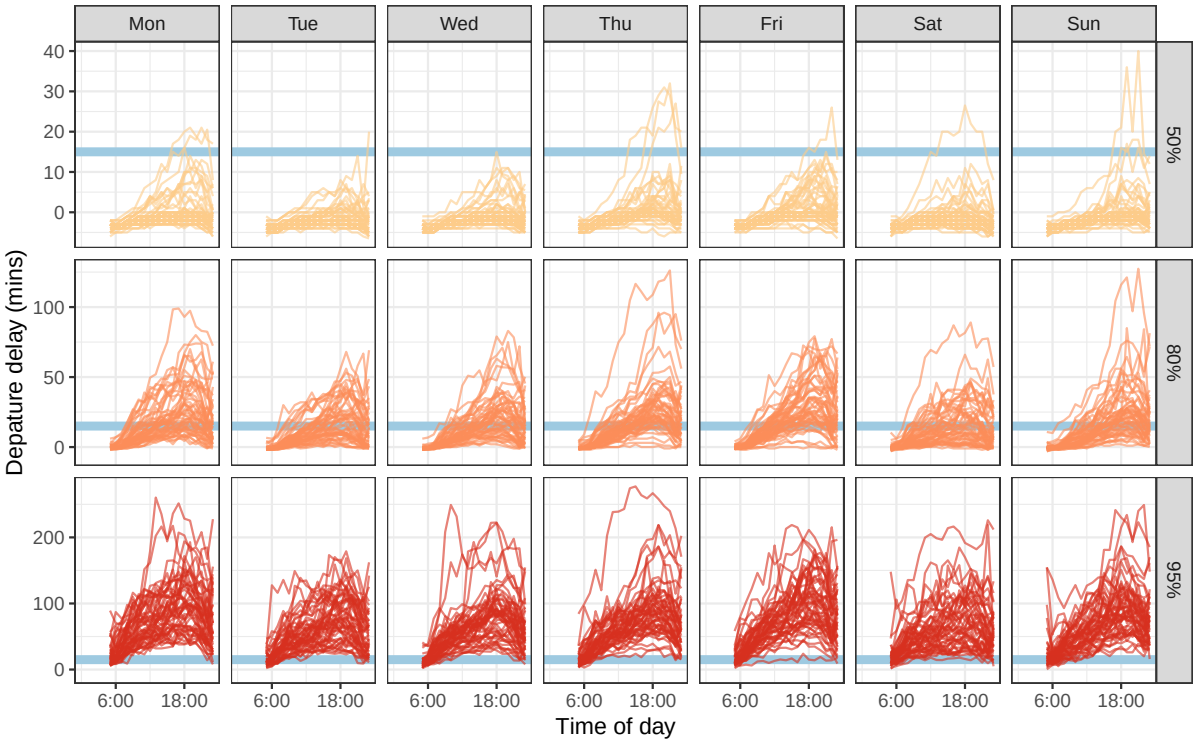

Figure 5: *Small multiples of lines about departure delay against time of day, faceting day of week and 50%, 80% and 95% quantiles. A blue horizontal line indicates the 15-minute on-time standard to help grasp the delay severity. Passengers are apt to hold up around 18 during a day, and are recommended to travel early. The variations increase substantially as the upper tails.*

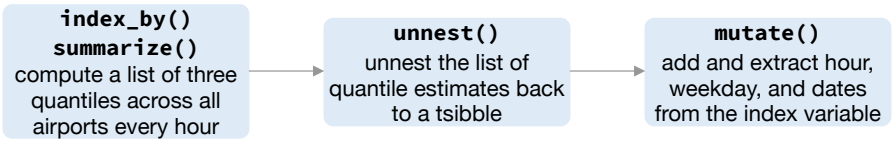

Figure 6: *Flow chart illustrates the pipeline that preprocesses the data for creating Figure 5.*

A closer examination of some big airports across the US will give an indication of 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 Seattle (SEA) airports is obtained first. The succeeding operations compute delayed percentages every day at each airport, which are framed as grey lines in Figure 7. Winter months tend to fluctuate a lot compared to the summer across all the airports. Superimposed on the plot are two-month moving averages, so the temporal trend is more visible. Since the number of days for each month is variable, moving averages over two months will require weights input. But the

weights specification can be avoid using a pair of commonly used rectangling verbs–`nest()` and `unnest()`, to wrap data frames partitioned by months into list-columns. The sliding operation with a large window size smooths out the fluctuations and gives a stable trend around 25% over the year. LAX airport has seen a gradual decline in delays over the year, whereas the SEA airport has a steady number delays over time. The IAH and JFK airports have more delays in the middle of year, while the KOA has the inverse pattern with higher delay percentage in both ends of the year. This pipeline gets the data into the daily series, and shifts the focus to five major airports.

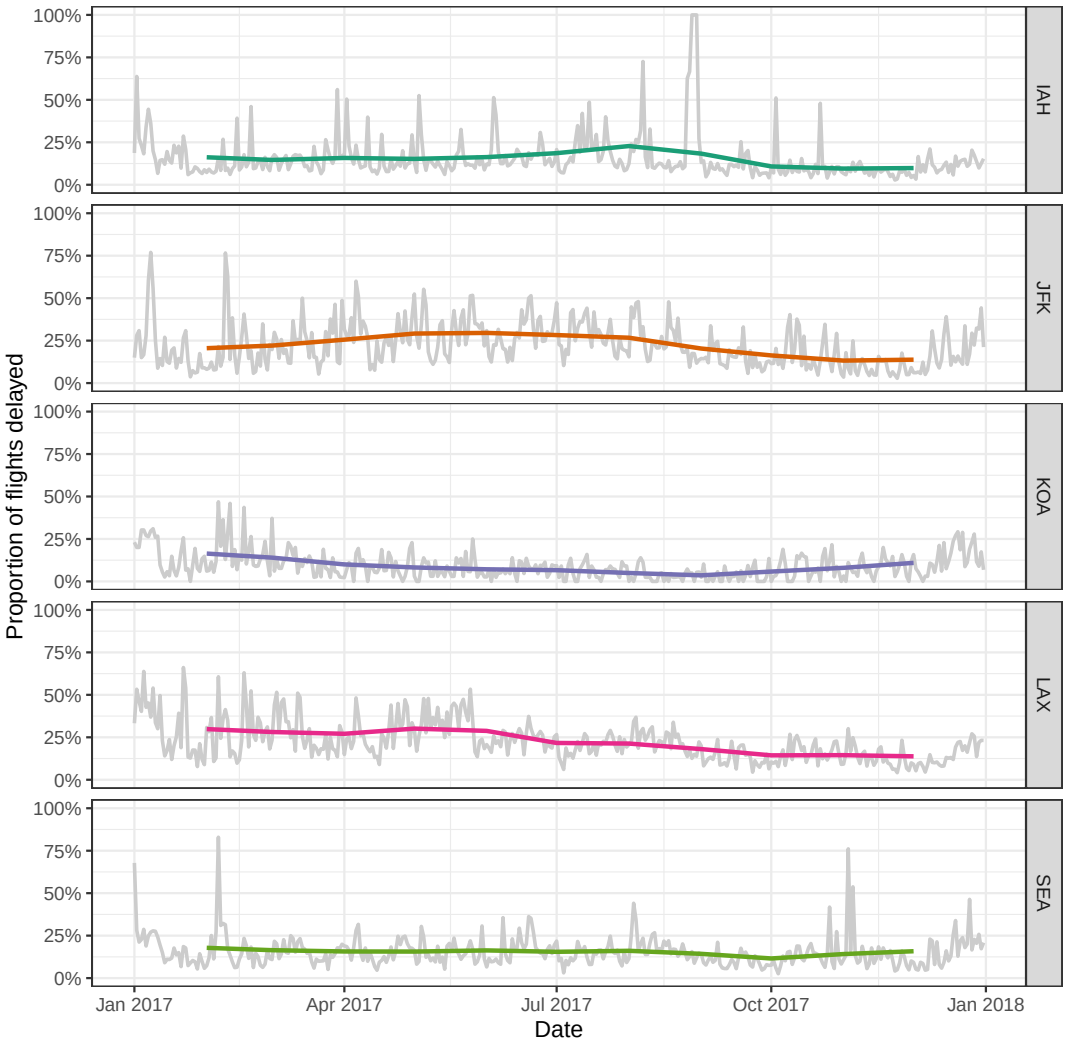

Figure 7: *Daily delayed percentages for departure with two-month moving averages overlaid at five international airports. There are least fluctuations and relatively fewer delays observed at KOA airport. The estimates of temporal trend are around 25% across the other four airports, but highlight different time periods of severe delays.*

This case study begins with duplicates fixing, that resolved the issue for constructing the tsibble. A range of temporal transformations can be handled by many free-form combinations of verbs, facilitating exploratory visualization.

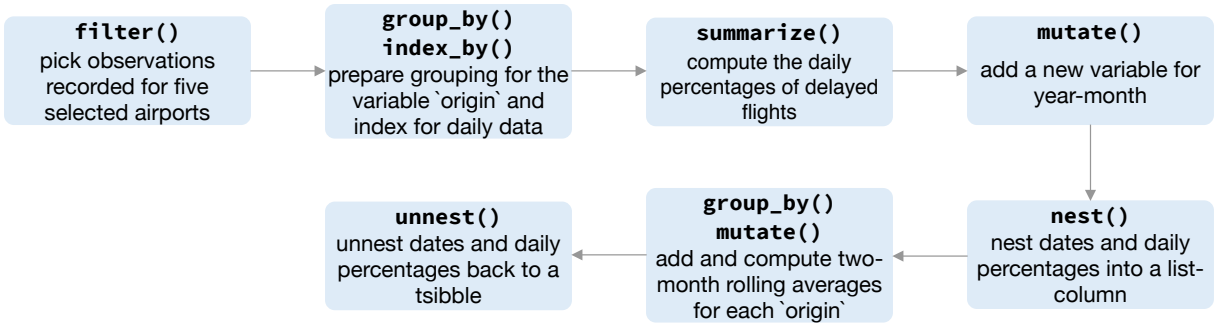

Figure 8: *Flow chart illustrating the pipeline that preprocessed the data for creating Figure 7.*

6.2 Smart-grid customer data in Australia

Sensors have been installed in households across major cities in Australia to collect data for the smart city project. 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). Data from 2013 have been sliced to examine temporal patterns of customer's energy consumption with **tsibble** for this case study. Half-hourly general supply in kwH have been recorded for 2,924 customers in the data set, resulting in 46,102,229 observations in total. Daily high and low temperatures in Newcastle in 2013 provides explanatory variables other than time in a different data table (Bureau of Meteorology 2019), obtained using the R package **bomrang** (Sparks et al. 2018). Aggregating the half-hourly energy data to the same daily time interval as the temperature data allows to join the two data tables to explore how local weather can contribute to the variations of daily electricity use and the accuracy of demand forecasting.

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 a preliminary analysis. Since the smart meters have been installed at different dates for each household, it is reasonable to assume that the records are obtainable for different time lengths for each household. Figure 9 shows the gaps for the top 49 households arranged in rows from high to low in tallies. (The remaining households values have been aggregated into a single batch and appear at the top.) Missing values can be seen to occur at any time during the entire 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 outages with more frequent occurrence in January.

Aggregation across all individuals helps to sketch a big picture of the behavioral change over time in the region, organized into a calendar display (Figure 10) using the **sugrrants** package

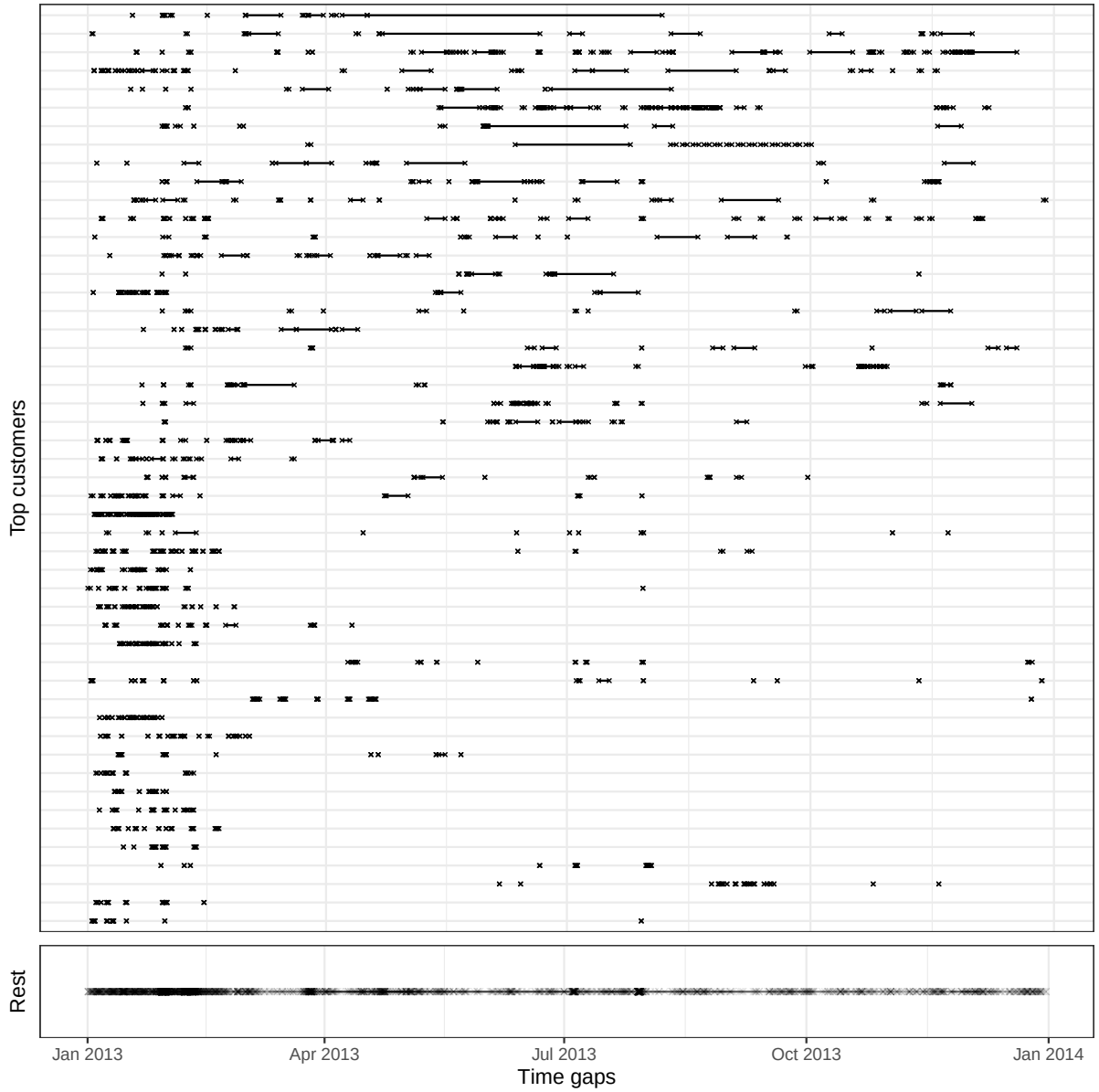

Figure 9: *Time gap plots for the 49 customers with most implicit missing values, and the remaining customers grouped into the one line at the bottom panel. Each cross represents an observation missing in time and a line between two dots shows continuous missingness over time. Each row corresponds to one customer. Missing values occur at various times, with more in January and February than other months.*

(Wang, Cook & Hyndman 2018). Each glyph represents the daily pattern of average residential electricity usage every thirty minutes. Higher consumption is indicated by higher values, and typically occurs in daylight hours. Color indicates hot days. The daily snapshots vary 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 on hot days with daily average temperature greater than 25 degrees Celsius. However, the winter

time (July and August) sees two peaks in a day, which is probably due to heating in the morning and evening.

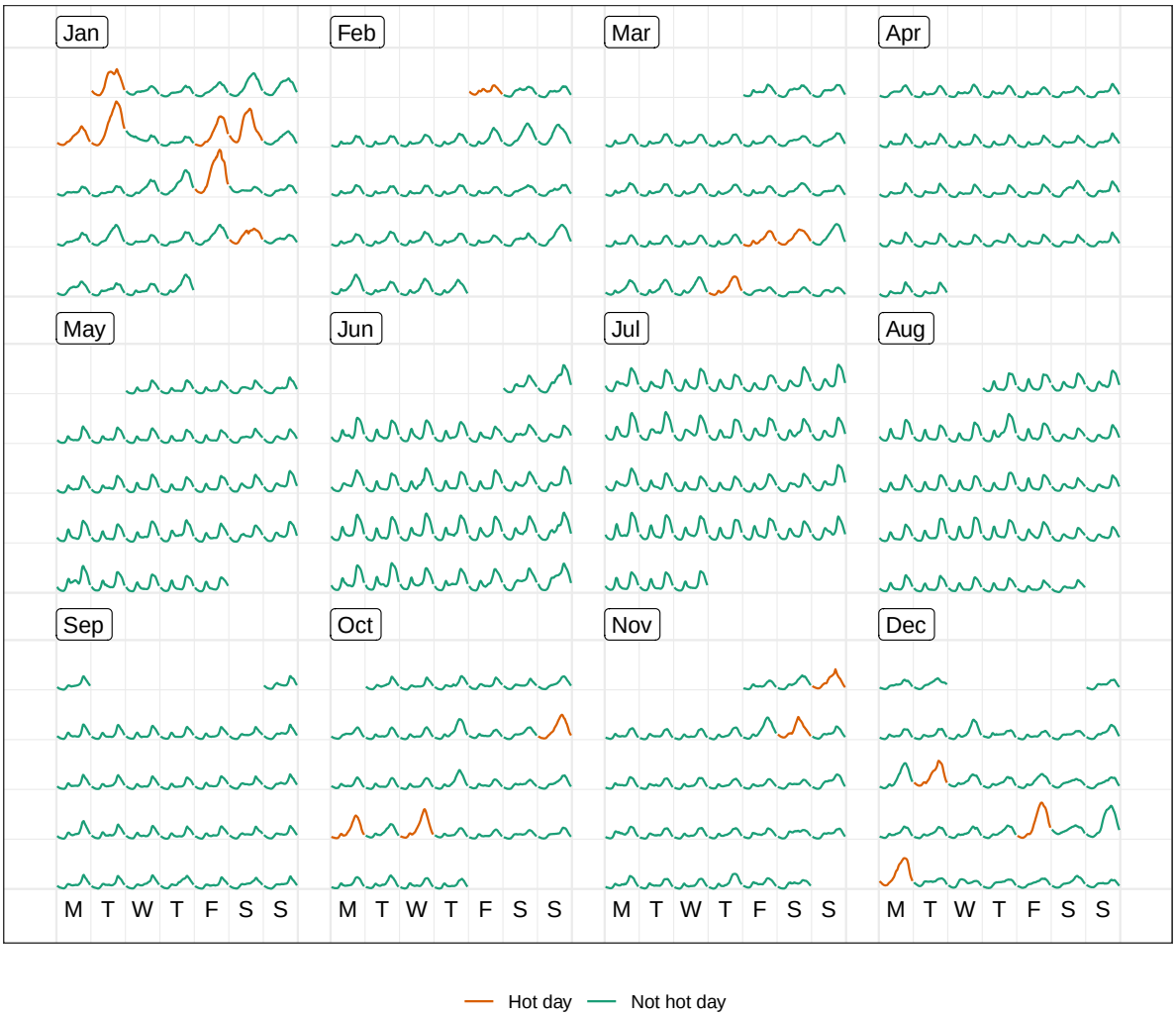

Figure 10: *Half-hourly average electricity use across all customers in the region, organized into calendar format, with color indicating hot days. Energy use of hot days tends to be higher, suggesting air conditioner use. Days in the winter months have a double peak suggesting heater use.*

Lastly, a common practice in the energy sector is load forecasting. Forecasting the demand for the last day of 2013 is undertaken, holding the original energy data as the validation set. The most recent December data is fitted by ARIMA models with and without temperature information using automatic order selection (Hyndman & Khandakar 2008). The logarithmic transformation is applied to the average demand to ensure the positive forecasts. Figure 11 plots the one-day forecasts obtained from both models against the actual demand within the last two-week window. ARIMA regressed by average temperatures gives a better fit than the one without, although both tend to underestimate the night demand. The forecasting performance is reported in Table 2, consistent with the findings in Figure 11.

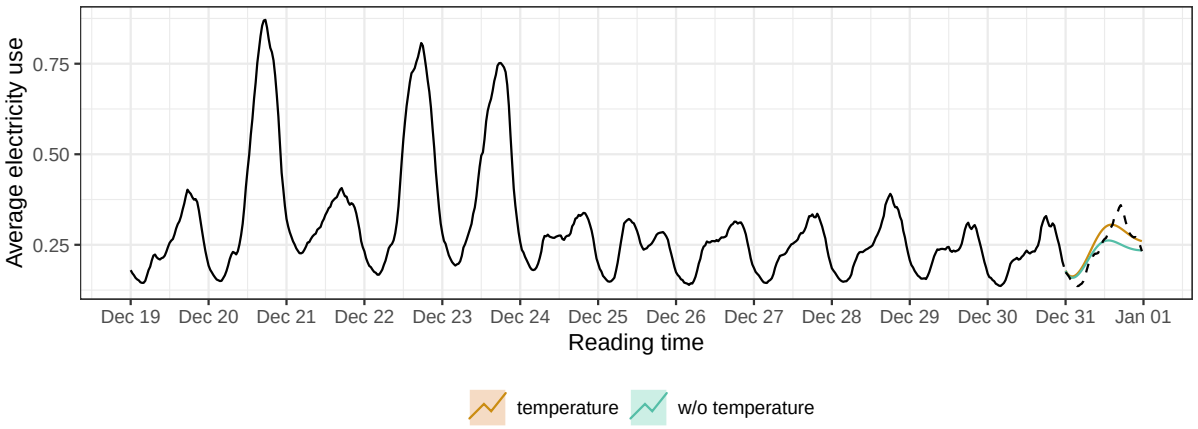

Figure 11: *One-day forecasts generated by ARIMA models with and without temperatures plotted against the actual demand. Both nicely capture the temporal dynamics, but ARIMA with temperatures performs better than the one without.*

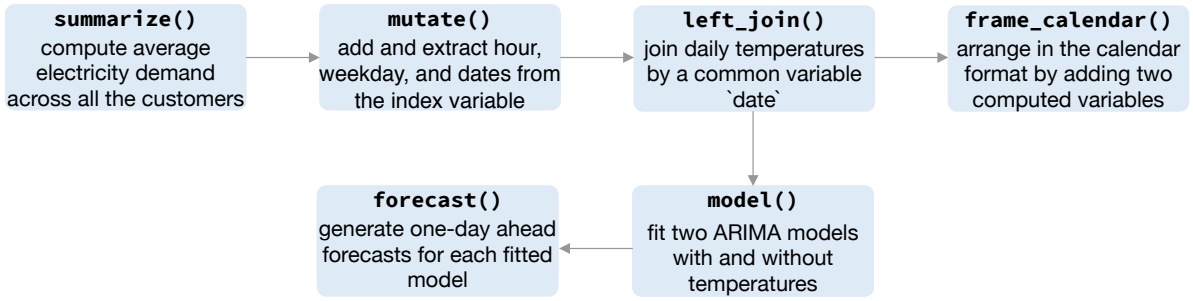

Figure 12: *Flow chart illustrating the pipeline involved for creating Figure 10 and Figure 11.*

Table 2: *Accuracy measures to evaluate the forecasting performance between ARIMA models with and without temperatures, using the validation set.*

model	ME	RMSE	MAE	MPE	MAPE
temperature	-0.009	0.030	0.025	-6.782	11.446
w/o temperature	0.016	0.043	0.032	2.634	12.599

This case study demonstrates the significance of tsibble in lubricating the plumbing of handling time gaps, visualizing, and forecasting in general.

7 Conclusion and future work

A data abstraction, tsibble, is proposed to represent temporal data, allowing the "tidy data" principles to be brought to the time domain. Tidy data begins to take shape in the state of time with new contextual semantics: index and key. The index variable provides direct support to an exhaustive set of ordered objects. The key identifies observational units over time, which can consist of single or multiple variables to be useful in more scenarios. These semantics further determine unique data entries required for a valid tsibble. It shepherds raw temporal data through the "tidy" stage to the next exploration stage for gaining more insights.

The supporting toolkits articulate the temporal data pipeline, with the shared goal of reducing the time between the framings of data questions and the code realization. The rapid iteration for better understanding data is achieved through frictionlessly shifting among transformation, visualization, and modelling, using the standardized tsibble data infrastructure.

A nice feature that the tsibble structure should take into account is to allow user-defined calendars and to respect structurally missing observations. For example, a call center may operate only between 9:00 am and 5:00 pm on week days, and stock trading resumes on Monday straight after Friday. No data available outside trading hours would be labeled as structural missingness, which tsibble currently disregards. Customer calendars can be embedded into the tsibble framework in theory. A few R packages provide functionality to create and manage many specific calendars, such as the **bizdays** package (Freitas 2018) for business days calendars. However, a generic flexible calendar system is lacking, that delays the implementation. This is left for future work.

Acknowledgments

The authors would like to thank Mitchell O'Hara-Wild for many discussions on the software development and Davis Vaughan for contributing ideas on rolling window functions. We also thank Stuart Lee for the feedback on this manuscript. We are grateful for anonymous reviewers for helpful feedback that has led to many improvements in the paper. This article was created with knitr (Xie 2015) and R Markdown (Xie, Allaire & Grolemund 2018). The project's Github repository https://github.com/earowang/paper-tsibble houses all materials required to reproduce this article and a history of the changes.

References

Bache, SM & H Wickham (2014). *magrittr: A Forward-Pipe Operator for R*. R package version 1.5. https://CRAN.R-project.org/package=magrittr.

Bengtsson, H (2019). *future: Unified Parallel and Distributed Processing in R for Everyone*. R package version 1.11.1.1. https://CRAN.R-project.org/package=future.

Buja, A, D Asimov, C Hurley & 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 Meteorology (2019). *Australia's National Weather Data*. Australian Government, Bureau of Meteorology. https://data.gov.au/dataset/4e21dea3-9b87-4610-94c7-15a8a77907ef (visited on 01/12/2019).

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).

City of Melbourne (2017). *Pedestrian Volume in Melbourne*. http://www.pedestrian.melbourne.vic.gov.au.

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. https://www.jstatsoft.org/v027/i02.

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).

Eddelbuettel, D & L Silvestri (2018). *nanotime: Nanosecond-Resolution Time for R*. R package version 0.2.3. https://CRAN.R-project.org/package=nanotime.

Freitas, W (2018). *bizdays: Business Days Calculations and Utilities*. R package version 1.0.6. https://CRAN.R-project.org/package=bizdays.

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

Henry, L & H Wickham (2019a). *purrr: Functional Programming Tools*. R package version 0.3.0. https://CRAN.R-project.org/package=purrr.

Henry, L & H Wickham (2019b). *rlang: Functions for Base Types and Core R and 'Tidyverse' Features*. R package version 0.3.1. https://CRAN.R-project.org/package=rlang.

Henry, L & H Wickham (2019c). *Tidy Tidyverse Design Principles*. https://principles.tidyverse.org.

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

Hyndman, R & Y Khandakar (2008). Automatic Time Series Forecasting: The forecast Package for R. *Journal of Statistical Software, Articles* **27**(3), 1–22. https://www.jstatsoft.org/v027/i03.

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

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

Long, JA (2019). *panelr: Regression Models and Utilities for Repeated Measures and Panel Data*. R package version 0.7.1. https://CRAN.R-project.org/package=panelr.

Mcilroy, D, E Pinson & B Tague (1978). Unix Time-Sharing System Forward. *The Bell System Technical Journal*, 1902–1903. https://archive.org/details/bstj57-6-1899.

O'Hara-Wild, M, R Hyndman & E Wang (2019). *fable: Forecasting Models for Tidy Time Series*. R package version 0.1.0. https://fable.tidyverts.org.

Pebesma, E (2018). Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal* **10**(1), 439–446. https://journal.r-project.org/archive/2018/RJ-2018-009/index.html.

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

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

SAS Institute Inc. (2018). *SAS/STAT Software, Version 9.4*. Cary, NC. http://www.sas.com/.

Sparks, A, J Carroll, D Marchiori, M Padgham, H Parsonage & K Pembleton (2018). *bomrang: Australian Government Bureau of Meteorology (BOM) Data from R*. R package version 0.4.0. https://CRAN.R-project.org/package=bomrang.

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

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.

Tidyverse Team (2019). *Tidy Evaluation*. https://tidyeval.tidyverse.org.

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

Vaughan, D & M Dancho (2018a). *furrr: Apply Mapping Functions in Parallel using Futures*. R package version 0.1.0. https://CRAN.R-project.org/package=furrr.

Vaughan, D & M Dancho (2018b). *tibbletime: Time Aware Tibbles*. R package version 0.1.1. https://CRAN.R-project.org/package=tibbletime.

Wang, E, D Cook & R Hyndman (2019). *tsibble: Tidy Temporal Data Frames and Tools*. R package version 0.8.3. https://tsibble.tidyverts.org.

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 (2019). *dplyr: A Grammar of Data Manipulation*. http://dplyr.tidyverse.org, https://github.com/tidyverse/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 (Apr. 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 (2015). *Dynamic Documents with R and knitr*. 2nd. Boca Raton, Florida: Chapman and Hall/CRC. https://yihui.name/knitr/.

Xie, Y, J Allaire & G Grolemund (2018). *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman and Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Y, H Hofmann & X Cheng (May 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* **14**(6), 1–27. https://www.jstatsoft.org/v014/i06.