# Conversations in time: interactive visualisation to explore structured temporal data

*by Earo Wang, Dianne Cook*

**Abstract** An abstract of less than 150 words.

## Introduction

- An ensemble of graphics
- Accelerate the exploratory data visualization process

## Background: tidy temporal data and workflow

The **tsibble** package (Wang et al., 2020) introduces a unified temporal data structure, referred to as a `tsibble`, to represent time series and longitudinal data in a tidy format (Wickham, 2014). That said, a `tsibble` extends the `data.frame` and **tibble** class with temporally contextual metadata: `index` and `key`. The `index` declares a data column that holds time-related indices. The `key` identifies a collection of related series or panels observed over the `index`-defined period, which can comprise multiple columns. Below displays the monthly Australian retail trade turnover data (`aus_retail`), available in the **tsibbledata** package. The `Month` column holds year-months as `index`. The `State` together with `Industry` are the identifiers for these 152 series, highlighted as `key`. Note that the column `Series ID` could be an alternative option for setting up `key`, but `State` and `Industry` are more readable and informative. The `index` and `key` are "sticky" columns to a `tsibble`, forming critical pieces for fluent temporal data analysis later.

```
#> # A tsibble: 64,532 x 5 [1M]
#> # Key:       State, Industry [152]
#>   State                Industry                    `Series ID`    Month Turnover
#>   <chr>                <chr>                       <chr>          <mth>    <dbl>
#> 1 Australian Capital ~ Cafes, restaurants and cat~ A3349849A   1982 Apr      4.4
#> 2 Australian Capital ~ Cafes, restaurants and cat~ A3349849A   1982 May      3.4
#> 3 Australian Capital ~ Cafes, restaurants and cat~ A3349849A   1982 Jun      3.6
#> 4 Australian Capital ~ Cafes, restaurants and cat~ A3349849A   1982 Jul      4
#> 5 Australian Capital ~ Cafes, restaurants and cat~ A3349849A   1982 Aug      3.6
#> # ... with 64,527 more rows
```

In the spirit of tidy data to the **tidyverse** (Wickham et al., 2019), the **tidyverts** suite features `tsibble` as the foundational data structure, in order to build a fluid and fluent pipeline for time series analysis. Besides **tsibble**, the **feasts** and **fable** packages fill the role of statistical analysis and forecasting in the **tidyverts** ecosystem. When time series analysis starts taking off, series of interest denoted by the `key` variables often remain unchanged over the course of analysis, from trend inspection to forecasting performance.

Figure 1 (a) gives an overview of 152 series for the retail data using an overlaid time series plot, while Figure 1 (b) presents a scatterplot, where each series is represented by a dot in the feature space (trend versus seasonal strength). The plot making of Figure 1 (b) is aided with the `features()` function from **feasts**, which summarises original data by each series down to various statistical features. This function along with other **tidyverts** functions is `tsibble`-aware, and outputs a table in a reduced form where each row corresponds to a series, thus graphically displayed as Figure 1 (b).

Figure 1 highlights not only a series with strongest seasonality, but also a need to querying interesting series on the fly. Without interactivity, one needs to first filter the interesting series out from the features table, and join back to the original `tsibble` in order to examine its trend in relation to others. This procedure can soon grow cumbersome if many series to be discovered. Despite that the two plots are static, they can be considered as linked views via the common key variables between two tables. This motivates enabling interactivity of `tsibble` and `tsibble`-derived objects for rapid exploratory data analysis.

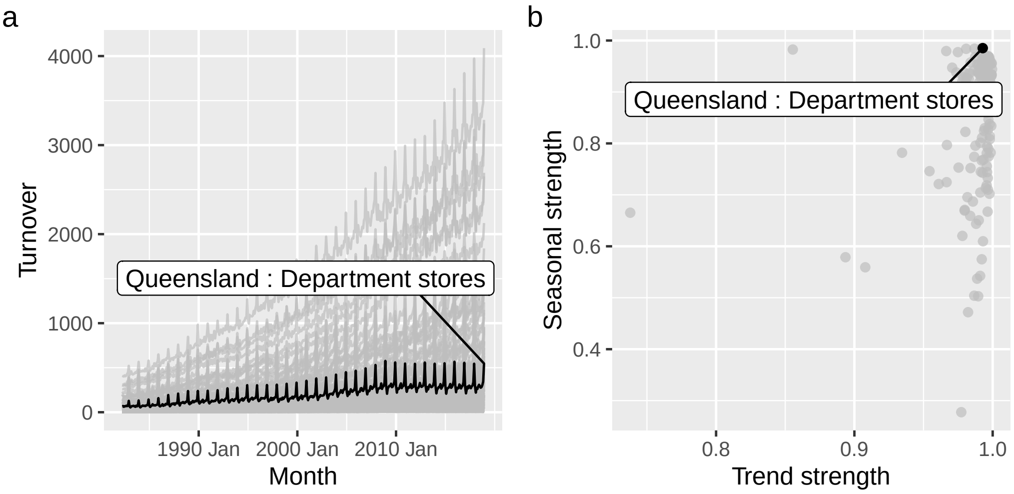



**Figure 1:** ToDo

## Overview of interactivity

- {cranvas} and {cranvastime}
- crossfilter.js & dc.js
- {crosstalk} and html widgets
- {rJava}, {rbokeh} {loon}

## Interactivity for coordinated views via shared temporal data

The **tsibbletalk** package, inspired by the **crosstalk** package, introduces a shared tsibble data structure on top of a `tsibble` to allow for frictionless communication between different plots for temporal data. The `as_shared_tsibble()` function provides an entry point in the integrated flow, turning a `tsibble` to a shared instance (i.e. `SharedTsibbleData` subclassing of `SharedData` from **crosstalk**) that powers data transmission across multiple views. The **tsibbletalk** package aims to streamline interactive graphical analysis with the focus of temporal and structured linking.

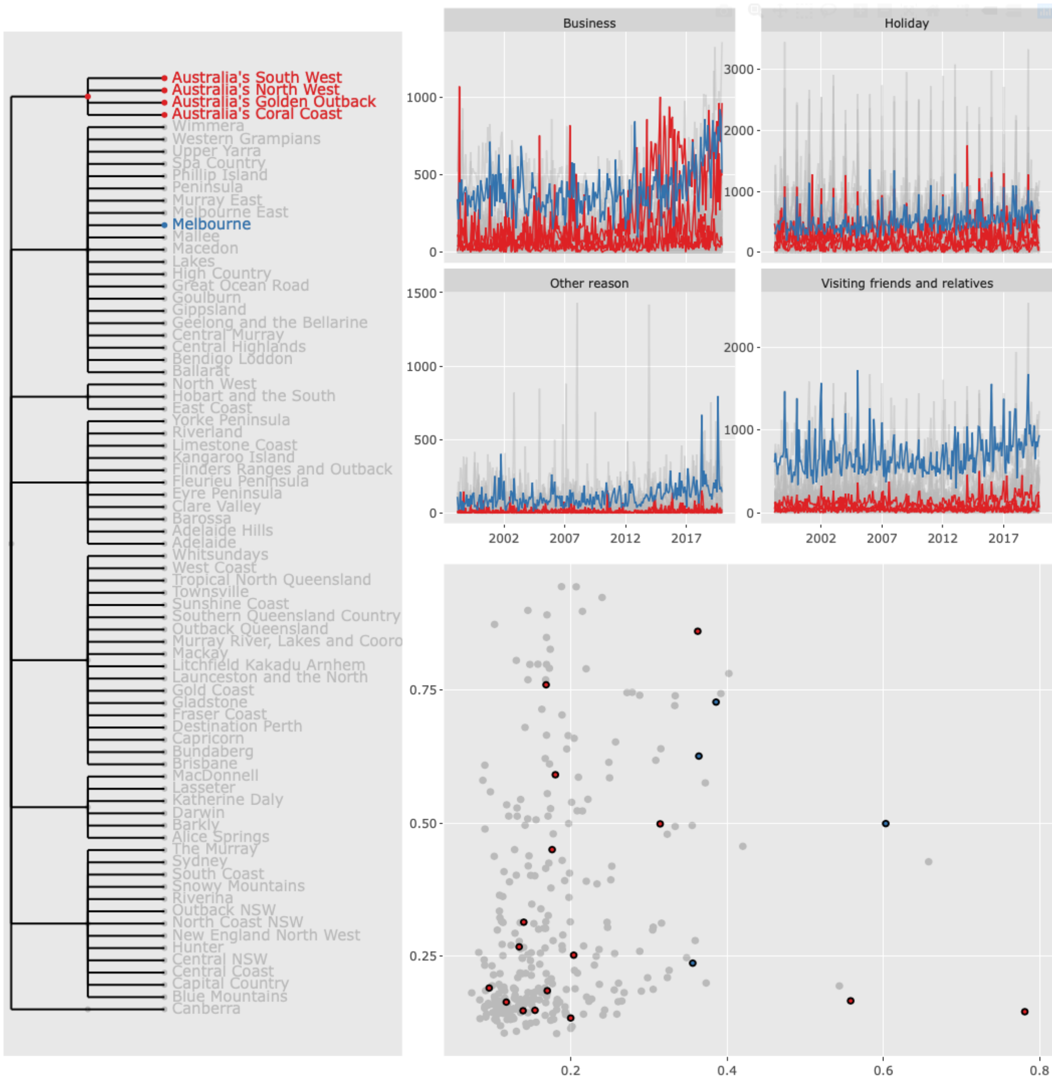
As opposed to one-to-one linking, **tsibbletalk** defaults to categorical linking where marking one or more observations in one category will broadcast to all other observations in this category. Given time series plots, click any data point on a line, highlighting the whole line as a result. The `as_shared_tsibble()` uses `tsibble`'s `key` variables to achieve these types of linking, and the `spec` argument takes one step further in constructing hybrid linking, such as hierarchical and categorical linking. For example, each series in the `aus_retail` data corresponds to all possible combinations of the `State` and `Industry` variables. They are intrinsically crossed with each other. If one variable is nested within another, this lends itself to a hierarchical structure, like geographical hierarchy. Such collection of inter-related time series are referred to as hierarchical and grouped time series in the literature (Hyndman and Athanasopoulos, 2017).

To incorporate structured specifications in the `key`, a symbolic formula can be passed to the `spec` argument. Adopting Wilkinson notations for factorial models (Wilkinson and Rogers, 1973), the `spec` follows the `/` and `*` operators tradition to declare nesting and crossing variables respectively. The `spec` for the `aus_retail` data is therefore specified as `State * Industry` or `Industry * State`, which is the default for the presence of multiple `key` variables. If there is a hierarchy in the data, using `/` is required to indicate the parent-child relation, as strictly one direction `parent/child`.

The `tourism_monthly` dataset packaged in **tsibbletalk**, contains monthly domestic overnight trips across Australia, to give an illustrator of nesting and crossing. The `key` is comprised of three identifying variables: `State`, `Region`, and `Purpose` (of trip), in particular `State` nesting of `Region`, together crossed with `Purpose`. This specification can be translated as follows:

```
library(tsibbletalk)
tourism_shared <- tourism_monthly %>%
  as_shared_tsibble(spec = (State / Region) * Purpose)
```

This dataset contains a three-level hierarchy: the root node is implicitly Australia, and geographically disaggregated to states and lower-level tourism regions. A new function `plotly_key_tree()` has been implemented to address the need of hierarchical discovery arising from data. It visually interprets hierarchies in the shared tsibble's `spec` as a tree view, made with **plotly**. From a static point of view, a tree visualisation of hierarchical data facilitates to grasp the overview structure. From the interactive perspective, selecting a parent node along the tree subsequently triggers its children and their corresponding series, or vice versa. It makes navigation simpler than conditional panels, and also enables persistent brushing for comparisons across groups.




## Slicing and dicing time

The other critical aspect of a tsibble is "index", that provides foundational temporal context. A common tool in time series analytical toolkit is seasonal plots that lay time series not on the whole time scale, but on an origin-less relative time unit, for example `gg_season()` in the {feasts} package. It helps to examine and emphasise periodic/aperiodic patterns, comparing to time series plots that primarily focus on trends. Standard seasonal plots break the overall time into two components: seasonal periods on the x-axis, and grouped by their corresponding lower-resolution time. For example, monthly data can be decomposed into months separated by years, and hourly data into hours grouped by days. Data collected at lower-level resolutions often exhibits more than one seasonal patterns. To discover typical seasonal or non-typical profiles, it is helpful to quickly browse through many possible periods. Interactivity ought to be enabled.

The {tsibbletalk} package provides a pair of UI and server functions, as a shiny module, to help with finding interesting time slices in a shiny application. The pair, decoupled to `tsibbleDiceUI()` and `tsibbleDiceServer()`, presents a clean interface and forms a resusable piece. Like all shiny modules, users should supply a unique session id. The UI function `tsibbleDiceUI()` shows a slider that controls the number of periods, and a plot specified by users. The server function `tsibbleDiceServer()` is the workhorse, transforming data and updating the plot. It expects a `ggplot` (converted to `plotly` via `ggplotly()`) or `plotly` object. This plot can be line charts, or other graphical elements (such as boxplots). But it assumes that tsibble's time index is plotted on the x-axis. The other mandatory argument is to specify the number of seasonal periods that requires shifting.

(Data flows) Transformed data generally requires redrawing the plot, and worsen the performance of shiny. The underlying tsibble data is called back and transformed in R. Using the `plotly.js` react method, only transformed data is sent to the server side, while keeping the rest configuration unchanged (e.g. layout and graphical elements). It is performant, and users will not experience notable delay in response to the change in the slider input. Dissect time index, and propagate transformed data to shiny server.

## Conclusions and discussions

## Bibliography

R. J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Melbourne, Australia, 2017. URL `OTexts.org/fpp2`. [p2]

E. Wang, D. Cook, and R. J. Hyndman. A new tidy data structure to support exploration and modeling of temporal data. *Journal of Computational and Graphical Statistics*, 0(0):1–13, 2020. doi: 10.1080/10618600.2019.1695624. [p1]

H. Wickham. Tidy data. *Journal of Statistical Software*, 59(10):1–23, 2014. [p1]

H. Wickham, M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. doi: 10.21105/joss.01686. URL `https://doi.org/10.21105/joss.01686`. [p1]

G. N. Wilkinson and C. E. Rogers. Symbolic description of factorial models for analysis of variance. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 22(3):392–399, 1973. ISSN 00359254, 14679876. URL `http://www.jstor.org/stable/2346786`. [p2]

*Earo Wang*
*The University of Auckland*
*Department of Statistics*
*The University of Auckland*
*New Zealand*
`earo.wang@auckland.ac.nz`

*Dianne Cook*
*Monash University*
*Department of Econometrics and Business Statistics*
*Monash University*
*Australia*
`dicook@monash.edu`