# 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 and motivation

### Interactive visualisation systems with linking views

- {ggobi} and {xggobi}
- {cranvas} and {cranvastime}
- crossfilter.js & dc.js
- {crosstalk} and html widgets

### 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 contain multiple columns. Below displays the monthly Australian retail trade turnover data, 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, 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 interests defined by `key` are likely to remain unchanged over the course of analysis. Functions, such as `features()` and `model()`, summarise a sequence of indexed values down to a single statistic or model by every observational unit. The output is a normal table, where each row corresponds to an observational unit denoted by "key". In the context of relational databases, the "key" acts like a foreign key in a reduced form of tsibble, while the index and key together operates like a primary key. (tsibble-centered workflow)

At the early stage of exploratory temporal data analysis, time series plots and scatterplots goes hand by hand. (insert figures below)

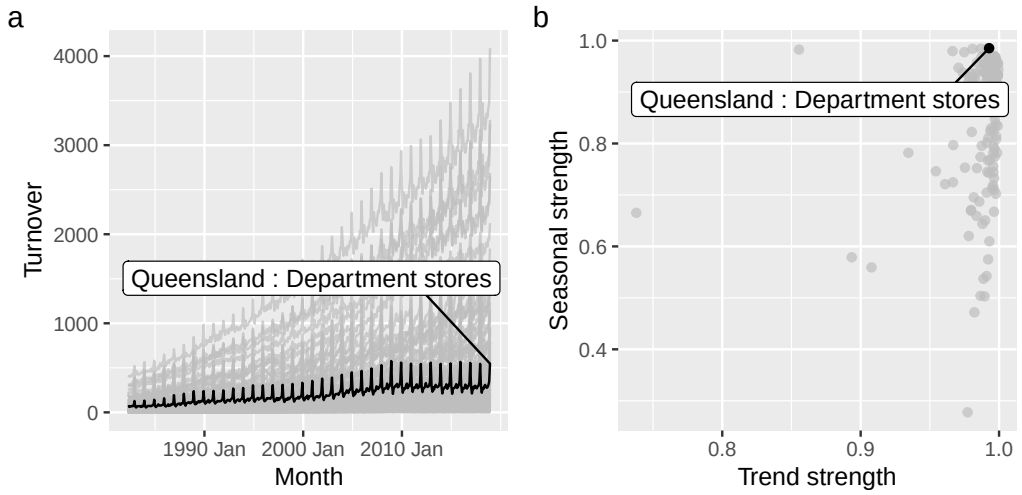


**Figure 1:** ToDo

## Shared temporal data for coordinated views

The **tsibbletalk** package, inspired by the **crosstalk** package, introduces a shared tsibble data structure to enable reactivity for temporal data (i.e. a `tsibble`). The `as_shared_tsibble()` function turns a tsibble to a mutable tsibble `SharedTsibbleData`, as an R6 subclass of `SharedData` from {crosstalk}. A *tsibble* data object holds a set of interrelated series, identified by the "key", in a data frame. When the "key" includes more than two variables, there can exist intrinsic structures in all combinations of variables: either nesting or crossing. (Add an example here)

The `spec` argument in the `as_shared_tsibble()` function takes a symbolic formula to allow users to specify structured combinations, using `/` and `*` adopted from Wilkinson notation (10.2307/2346786) for nesting and crossing expressions respectively. By default, the specification assumes crossing structure for all units.

Specifying the structure is particularly useful, when there's a hierarchical structure in the data. For such case, a new function `plotly_key_tree()` can help to visualise the "key" structure as a dendrogram, using plotly. This type of plot gives an overview of structured information in the data. Owing to its interactivity, each node in the tree plot is clickable and linked to other plots, and in turn the navigation at various levels is made easier.

When the key structure involves crossing only, one-to-one linking is the default.

- One-to-many linking: marking a single point of interest highlights all other points that share particular data values (categorical linking)
- Constructing recursive lists. Lists of key values, and json
- Self-linking: a single data source

## 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.

## Case study: monthly domestic tourist trips in Australia

## Conclusions and discussions

## Bibliography

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]

*Earo Wang*
*The University of Auckland*
*Department of Statistics*
*The Univerisity 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