Conversations in time: interactive visualisation to explore structured temporal data

by Earo Wang and 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.

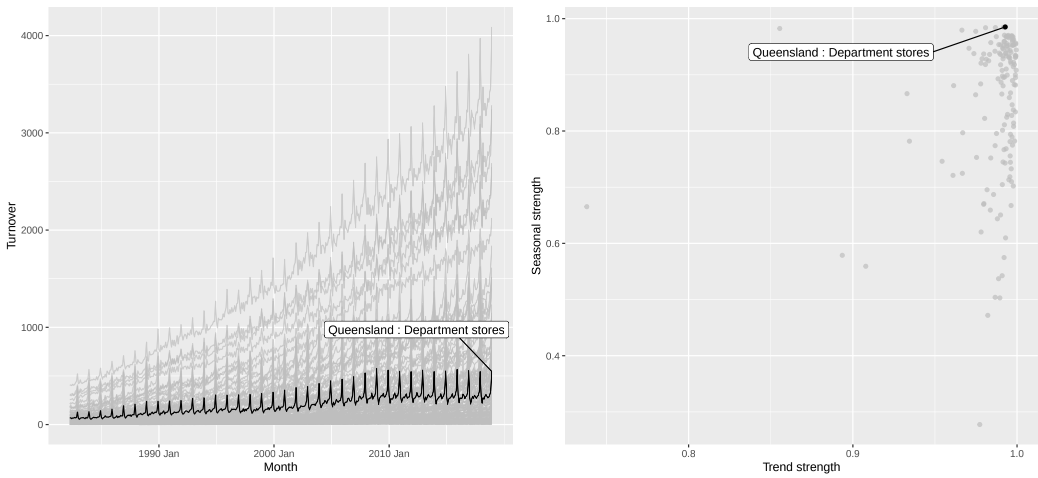
```
#> # A tsibble: 64,532 x 5 [1M]
#> # Key:       State, Industry [152]
#>   State                Industry                   `Series ID`    Month Turnover
#>   <chr>                <chr>                      <chr>          <mth>    <dbl>
#> 1 Australian Capital ~ Cafes, restaurants and cat~ A3349849A   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 1a gives an overview of 152 series for the retail data using an overlaid time series plot, while Figure 1b presents a scatterplot, where each series is represented by a dot in the feature space (trend versus seasonal strength). The plot making of Figure 1b 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 1b.

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.

 ISSN 2073-4859



(a) 1 **(b)** 2

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)
```

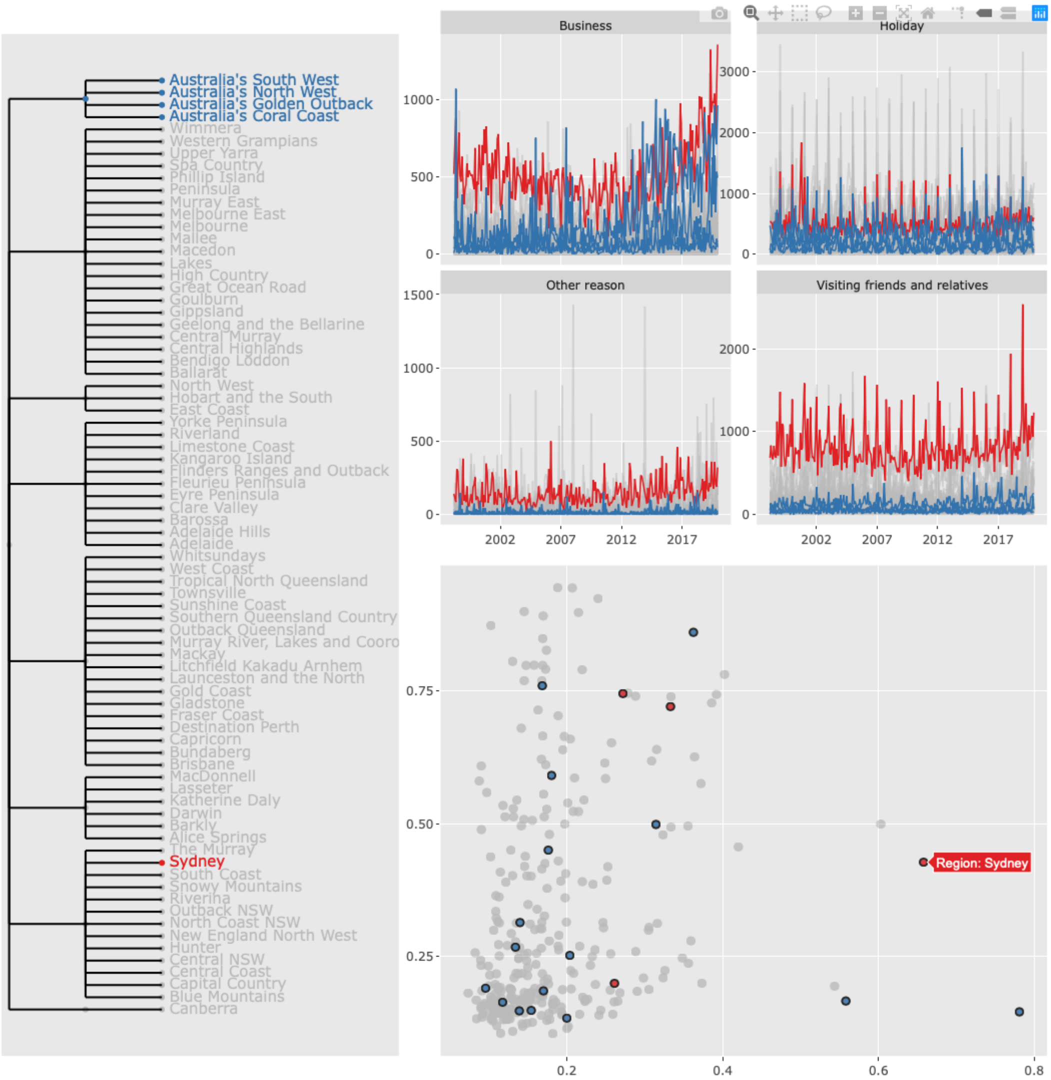


Figure 2: ToDo

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 handy function `plotly_key_tree()` has been implemented to address the need of hierarchical discovery arising from the data. It interprets hierarchies in the shared tsibble's `spec` as a tree view, built with **plotly**. The following code line produces the linked tree diagram and fills the left panel of Figure 2. The visual of tree hierarchy untangles a group of related series and snapshots the data organisation from a bird's eye view.

```
p_l <- plotly_key_tree(tourism_shared, height = 1100, width = 800)
```

The tree plot provides backbones of the data, and much flesh yet to be attached. Small multiples of time series lines are composed and placed at the top right of Figure 2 to unpack the temporal trend across regions by purposes of trips. The shared tsibble data can be directly piped into **ggplot2** code.

```
p_tr <- tourism_shared %>%
  ggplot(aes(x = Month, y = Trips)) +
  geom_line(aes(group = Region), alpha = .5, size = .4) +
  facet_wrap(~ Purpose, scales = "free_y") +
  scale_x_yearmonth(date_breaks = "5 years", date_labels = "%Y")
```

To tease apart these overlaid time series, they are funnelled through the `features()` S3 method to extract some key characteristics, including the measurements of trend and seasonality. A scatterplot is populated from these statistics for each series.

```
tourism_feat <- tourism_shared %>%
  features(Trips, feat_stl)
p_br <- tourism_feat %>%
  ggplot(aes(x = trend_strength, y = seasonal_strength_year)) +
  geom_point(aes(group = Region), alpha = .8, size = 2)
```

Lastly, three graphics are composed as an ensemble of coordinated views for multi-facetted exploration, shown as Figure 2 (the interactive realisation of Figure 1). Routine functions bring about new interaction with temporal data on the client side.

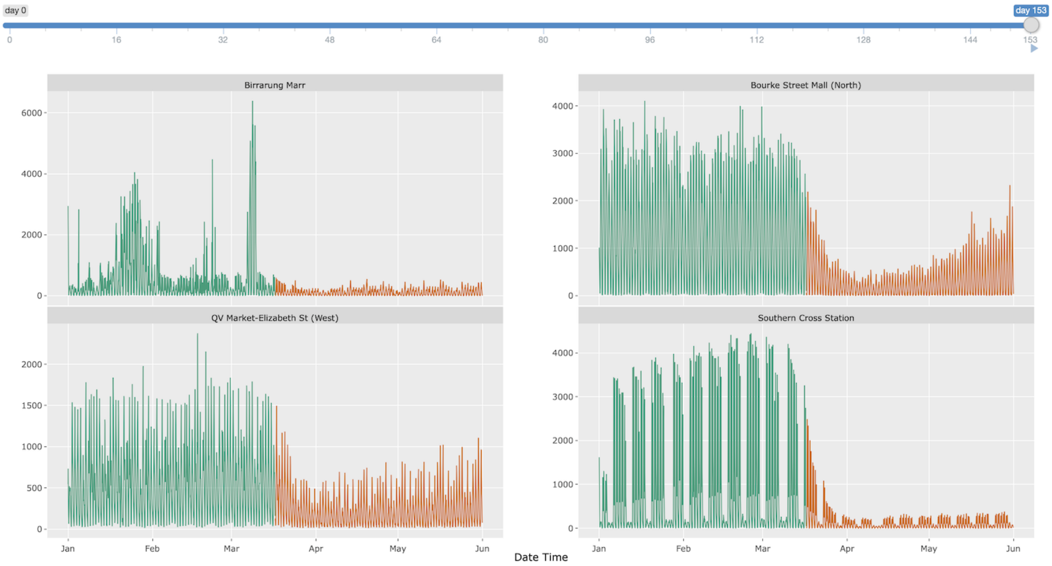
```
subplot(p_l,
  subplot(
    ggplotly(p_tr, tooltip = "Region", width = 1100),
    ggplotly(p_br, tooltip = "Region", width = 1100),
    nrows = 2),
  widths = c(.4, .6)) %>%
  highlight(dynamic = TRUE)
```

Since all plots are stemmed from one shared tsibble data source, they are self-linking views. Nodes, lines, and points are hoverable and clickable. Given the `spec`, clicking either one element in any plot highlights all points that match the `Region` category, briefly "categorical linking". In Figure 2, when hovering and selecting the circle associated with "Sydney" in the scatter plot, all data records with shared values of "Sydney" listen and react to this interaction via self updating in red. In order for comparison with other regions or states, press the "Shift" key to enable persistent selection, and simultaneously select the parent node on the tree, saying "Western Australia", to include all the children by switching to the blue colour. The domestic tourism sees Sydney as one of the most popular destinations in realm of business and friends visiting over years. Despite of relatively weaker performance in Western Australia, Australia's North West region sees the strongest upward trend, bypassing Sydney in some years.

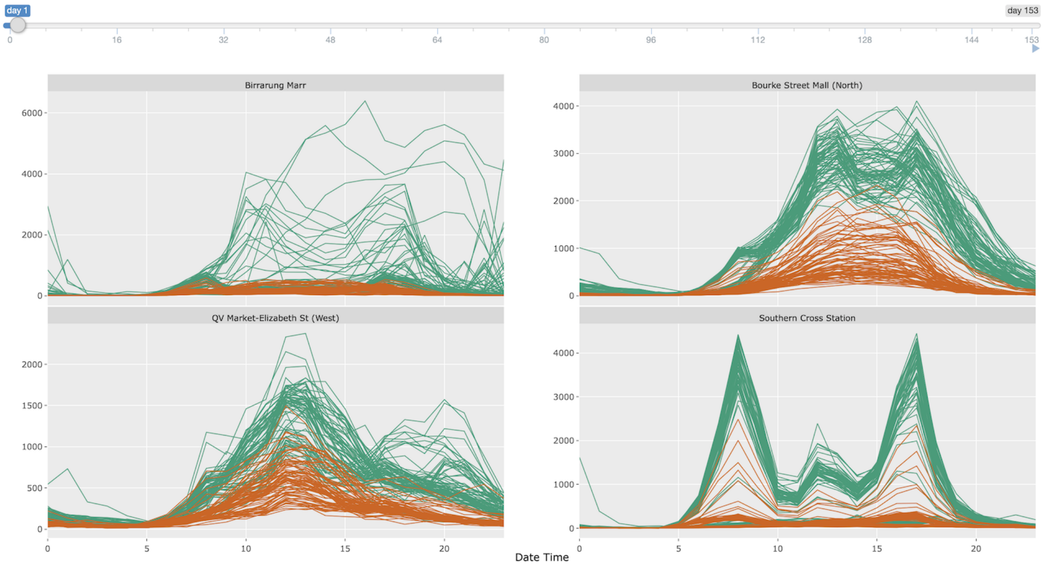
In summary, shared tsibble data nicely bridges between the **crosstalk** and **tidyverts** ecosystems. The `as_shared_tsibble()` provides a symbolic user interface for effortless construction of a hybrid of hierarchical and categorical linkings. And the `plotly_key_tree()` in turn decodes the specification to plot a tree for data overview and navigation, accompanied with more detailed plots.

Slicing and dicing time

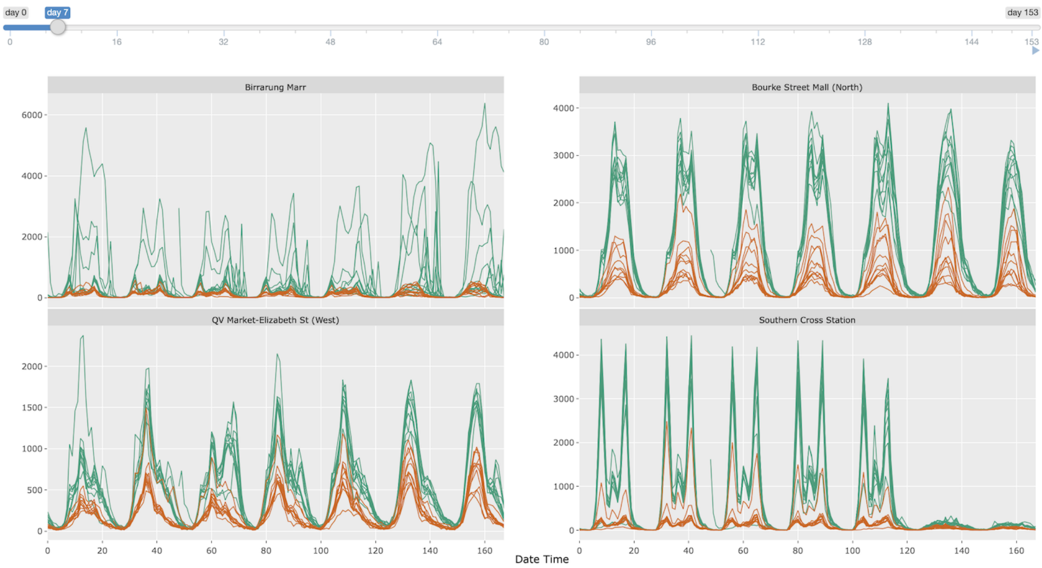
The shared tsibble data leverages the `key` attribute to converse with many coordinated views, with or without **shiny**. On the other hand, a second critical attribute–`index`–lays the foundational temporal context that augments the conversation. When temporal data are plotted and stretched against the entire span like Figure 1a, it puts emphasis on the trend perception. Yet to digest periodic/aperiodic patterns, data should be wrapped over relative time units that are origin-less, such as one quarter or one day.



(a) (1)



(b) (2)



(c) (3)

Figure 3: ToDo

The city of Melbourne has sensors installed to count hourly tallies of pedestrians in order to capture downtown daily rhythms (cite). Figure 3 shows the first five months of 2020 foot traffic at four locations, with the depiction of three pronounced slices in time. Figure 3a unfolds all counts from January to May on their absolute time scales, facetted by four sensors. On March 16, Melbourne went to the stage three lockdown due to COVID-19, seeing a significant decline in traffic volume across the city. These lines are then folded into daily and weekly sections, shown as Figure 3b and 3c respectively. Seasonal variations have been popped out to viewers, complementing the not-just-magnitude-drop story. The pre-lockdown period is coloured with dark green and lockdown with orange.

The wrapping procedure involves slicing time indices into seasonal periods of interest and their corresponding time dices. For example, hourly pedestrian data can be decomposed into 24-hour blocks grouped by all respective days, like Figure 3b. Figure 3 suggests that there could be more than one eye-catching slices out of many possible combinations, and thus repeated wrappings can be unwieldy. To visually locate an interesting slice, the **tsibbletalk** package implements a shiny module, a pair of UI and server functions, to automate this wrapping procedure.

This shiny module, decoupled to `tsibbleDiceUI()` and `tsibbleDiceServer()`, presents a clean interface and forms a resusable piece in a shiny application. Like all shiny modules, the first argument in both functions requires a user-supplied session id that must be unique. The UI function `tsibbleDiceUI()` simply shows a slider that animates or controls the number of periods to be diced. The workhorse is certainly the server function `tsibbleDiceServer()`, encapsulating the core algorithm that transforms data and sends messages to update the plot accordingly. It expects a `ggplot` or `plotly` object, where one can plot data using either lines or other graphical elements (such as boxplots). As the function name suggests, a (shared) tsibble is needed to start the engine, and thereby the time `index` can be retrieved for dissection. The `period` argument is to specify the minimum number of seasonal blocks as desired, for example data shifted by "1 day", "2 days", or "1 week", etc. The following code chunk generates Figure 3.

```
p_line <- pedestrian20 %>%
  ggplot(aes(x = Date_Time, y = Count, colour = Lockdown)) +
  geom_line(size = .3) +
  facet_wrap(~ Sensor, scales = "free_y") +
  labs(x = "Date Time") +
  scale_colour_brewer(palette = "Dark2") +
  theme(legend.position = "none")

ui <- fluidPage(
  tsibbleDiceUI("dice")
)
server <- function(input, output, session) {
  tsibbleDiceServer("dice", ggplotly(p_line, height = 700), period = "1 day")
}
shinyApp(ui, server)
```

Upon running the shiny application, Figure 3a corresponds to the initial state, with the slider incremented by 1-day unit. The "play" button near the end of slider begins animating the slicing and dicing process by walking through all 24 hours by 153 days. Alternatively, users can drag the handler to poke around certain slices themselves.

In response to the slider input, the plot will be updated and loaded with newly transformed data. Keeping the application as performant as possible is the top priority. Without completely redrawing the plot, the `plotly.js` react method is invoked internally. The underlying tsibble data is being called back and processed in R. Only transformed data gets fed back to the shiny server, along with reseting the x-axis ranges and breaks. The rest plot configurations, such as marks, y-axes, and layouts, are properly cached.

The new shiny module exploits the temporal aspect for a tsibble object. It allows users to slide through relative periods to explore seasonal insights, with slick user experience.

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`