Conversations in time: interactive visualization 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 visualization 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 extends the `data.frame` and **tibble** structure to represent temporal data in tidy format [cite]. A tsibble consists of *index*, *key*, and other measured variables in one data frame. The *index* column holds time-based indices. The *key* column(s) uniquely identifies a collection of related observational units during a period of time defined by *index*. They are "sticky" columns to a `tsibble` over the course of transformation.

The **feasts** and **fable** packages, part of the **tidyverts** suite, aim to make time series analysis easier. They provide analytical and forecasting tools for the tsibble data structure, generating tsibble-centered workflow. 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.

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

Shared temporal data for coordinated views

- Symbolic formula to express structural specifications among keyed units, using `/` and `*` from Wilkinson notation (10.2307/2346786) for nesting and crossing.
- Nesting variables generate hierarchical tree, hence `plotly_key_tree()`. Overview and navigation made easier.
- One-to-many linking: marking a single point of interest highlights all other points that share particular data values (connect-type)
- Lists of key values, and json
- R6 subclass of `SharedData` from {crosstalk}

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, `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.

- Transform and send newly-transformed data via server, without updating anything else (e.g. layout and graphical elements), to avoid completely redrawing. Performance.
- Double dispatch

Case study: monthly domestic tourist trips in Australia

Conclusions and discussions

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

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