There is a new suite of packages for tidy time series analysis, that integrates easily into the tidyverse way of working. We call these the tidyverts packages, and they are available at tidyverts.org.

The first of the packages to make it to CRAN was tsibble, providing the data infrastructure for tidy temporal data with wrangling tools. A *tsibble* (where “ts” is pronounced as in cats) is a time series object that is much easier to work with than existing classes such as ts, xts and others. Existing ts objects can be easily converted to tsibble objects using as\_tsibble(). For example

library(tidyverse)

library(tsibble)

USAccDeaths %>% as\_tsibble()

## # A tsibble: 72 x 2 [1M]

## index value

##

## 1 1973 Jan 9007

## 2 1973 Feb 8106

## 3 1973 Mar 8928

## 4 1973 Apr 9137

## 5 1973 May 10017

## 6 1973 Jun 10826

## 7 1973 Jul 11317

## 8 1973 Aug 10744

## 9 1973 Sep 9713

## 10 1973 Oct 9938

## # … with 62 more rows

This creates an Index column which species the time or date index. These are always explicit in tsibbles, and can take a rich variety of time and date classes to handle any sort of temporal data. The second column is a measurement variable. Note the [1M] in the header, indicating this is monthly data.

It is also easy to create tsibbles from csv files, by first reading them in using readr::read\_csv() and then using the as\_tsibble() function.

A more interesting tsibble object is tourism, containing quarterly overnight trips across Australia.

tourism

## # A tsibble: 24,320 x 5 [1Q]

## # Key: Region, State, Purpose [304]

## Quarter Region State Purpose Trips

##

## 1 1998 Q1 Adelaide South Australia Business 135.

## 2 1998 Q2 Adelaide South Australia Business 110.

## 3 1998 Q3 Adelaide South Australia Business 166.

## 4 1998 Q4 Adelaide South Australia Business 127.

## 5 1999 Q1 Adelaide South Australia Business 137.

## 6 1999 Q2 Adelaide South Australia Business 200.

## 7 1999 Q3 Adelaide South Australia Business 169.

## 8 1999 Q4 Adelaide South Australia Business 134.

## 9 2000 Q1 Adelaide South Australia Business 154.

## 10 2000 Q2 Adelaide South Australia Business 169.

## # … with 24,310 more rows

Here we have some additional columns called Keys. There should be one row for each unique combination of the keys and index. Columns which are not the index or key variables are measurement variables. In this example, there are three keys (Region, State and Purpose) and one Measurement (Trips)

All the usual tidyverse wrangling verbs apply. For example, to get the total visitor nights spent on Holiday by State for each quarter (ignoring Regions):

tourism %>%

filter(Purpose == "Holiday") %>%

group\_by(State) %>%

summarise(Trips = sum(Trips))

## # A tsibble: 640 x 3 [1Q]

## # Key: State [8]

## State Quarter Trips

##

## 1 ACT 1998 Q1 196.

## 2 ACT 1998 Q2 127.

## 3 ACT 1998 Q3 111.

## 4 ACT 1998 Q4 170.

## 5 ACT 1999 Q1 108.

## 6 ACT 1999 Q2 125.

## 7 ACT 1999 Q3 178.

## 8 ACT 1999 Q4 218.

## 9 ACT 2000 Q1 158.

## 10 ACT 2000 Q2 155.

## # … with 630 more rows

Note that we do not have to explicitly group by the time index as this is assumed in a tsibble.

To switch to annual data, we can re-index the tsibble:

tourism %>%

mutate(Year = lubridate::year(Quarter)) %>%

index\_by(Year) %>%

group\_by(Region, State, Purpose) %>%

summarise(Trips = sum(Trips)) %>%

ungroup()

## # A tsibble: 6,080 x 5 [1Y]

## # Key: Region, State, Purpose [304]

## Region State Purpose Year Trips

##

## 1 Adelaide South Australia Business 1998 538.

## 2 Adelaide South Australia Business 1999 641.

## 3 Adelaide South Australia Business 2000 787.

## 4 Adelaide South Australia Business 2001 608.

## 5 Adelaide South Australia Business 2002 697.

## 6 Adelaide South Australia Business 2003 690.

## 7 Adelaide South Australia Business 2004 734.

## 8 Adelaide South Australia Business 2005 490.

## 9 Adelaide South Australia Business 2006 568.

## 10 Adelaide South Australia Business 2007 584.

## # … with 6,070 more rows

The index\_by() function is the counterpart of group\_by() when dealing with the index.

I often get questions about dealing with daily and sub-daily data, for which the ts class is particularly ill-suited. The tsibble class handles such data with no problem. For example, here are some hourly pedestrian counts at four sites around Melbourne, Australia.

pedestrian

## # A tsibble: 66,037 x 5 [1h]

## # Key: Sensor [4]

## Sensor Date\_Time Date Time Count

##

## 1 Birrarung Marr 2015-01-01 00:00:00 2015-01-01 0 1630

## 2 Birrarung Marr 2015-01-01 01:00:00 2015-01-01 1 826

## 3 Birrarung Marr 2015-01-01 02:00:00 2015-01-01 2 567

## 4 Birrarung Marr 2015-01-01 03:00:00 2015-01-01 3 264

## 5 Birrarung Marr 2015-01-01 04:00:00 2015-01-01 4 139

## 6 Birrarung Marr 2015-01-01 05:00:00 2015-01-01 5 77

## 7 Birrarung Marr 2015-01-01 06:00:00 2015-01-01 6 44

## 8 Birrarung Marr 2015-01-01 07:00:00 2015-01-01 7 56

## 9 Birrarung Marr 2015-01-01 08:00:00 2015-01-01 8 113

## 10 Birrarung Marr 2015-01-01 09:00:00 2015-01-01 9 166

## # … with 66,027 more rows

The Date and Time variables split the index into two components, representing the date and the hour of day. This makes it easy to produce some interesting plots.

pedestrian %>%

mutate(

Day = lubridate::wday(Date, label = TRUE),

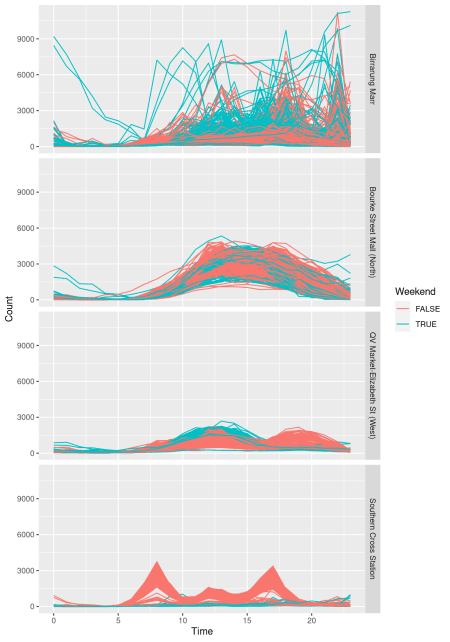
Weekend = (Day %in% c("Sun", "Sat"))

) %>%

ggplot(aes(x = Time, y = Count, group = Date)) +

geom\_line(aes(col = Weekend)) +

facet\_grid(Sensor ~ .)



The volume and pattern of pedestrian traffic at these four locations is clearly very different.

In summary, I hope tsibbles will become the standard for handling temporal data in R (including multivariate time series, panel data, ets). I’ve been using them for about a year, and I’m still amazed at how much easier it is to do things than using other structures.