This is the second post on the new [tidyverts](https://tidyverts.org/) packages for tidy time series analysis. The previous post is [here](https://robjhyndman.com/hyndsight/tsibbles).

For users migrating from the forecast package, it might be useful to see how to get similar graphics to those they are used to. The forecast package is built for ts objects, while the feasts package provides features, statistics and graphics for tsibbles. (See [first post](https://robjhyndman.com/hyndsight/tsibbles) for a description of tsibbles.)

**Exploratory graphics**

The forecast package provided facilities for plotting time series in various ways. All of these have a counterpart in the feasts package.

| **forecast** | **feasts** |
| --- | --- |
| autoplot() | autoplot() |
| ggseasonplot() | gg\_season() |
| ggsubseriesplot() | gg\_subseries() |
| gglagplot() | gg\_lag() |
| ggAcf() | ACF() %>% autoplot() |
| ggPacf() | PACF() %>% autoplot() |
| ggtsdisplay() | gg\_tsdisplay() |

The big difference is that tsibbles can contain multiple time series, while ts objects can only contain one (possibly multivariate) time series. Note also that the feasts functions will only do one thing — either compute some statistics or produce a plot — unlike the ggAcf() function which does both.

We will illustrate the above functions use the Australian quarterly holiday data by State, created in the last post.

library(tidyverse)

library(tsibble)

##

## Attaching package: 'tsibble'

## The following object is masked from 'package:dplyr':

##

## id

holidays <- tourism %>%

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

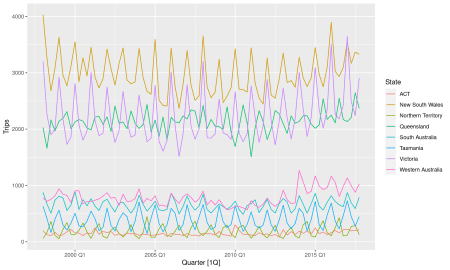
group\_by(State) %>%

summarise(Trips = sum(Trips))

First, a time plot is generated using autoplot().

library(feasts)

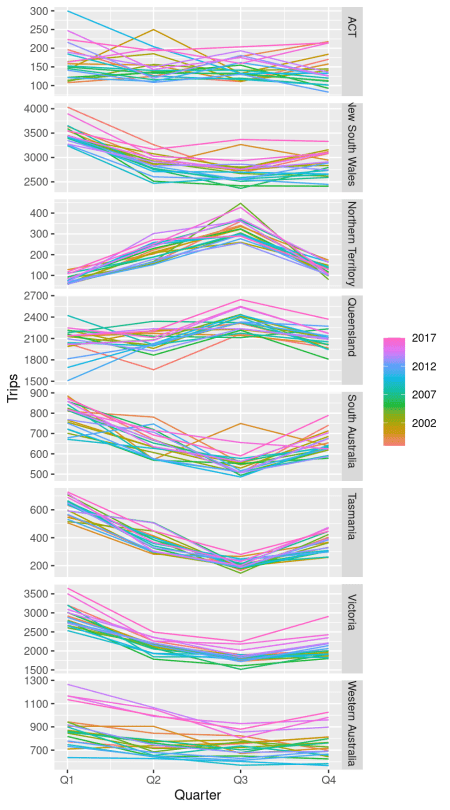
holidays %>% autoplot(Trips)



When the plotting variable (here Trips) is omitted, the first available measurement variable is used by default. When there are no keys, only one time series is shown with no legend.

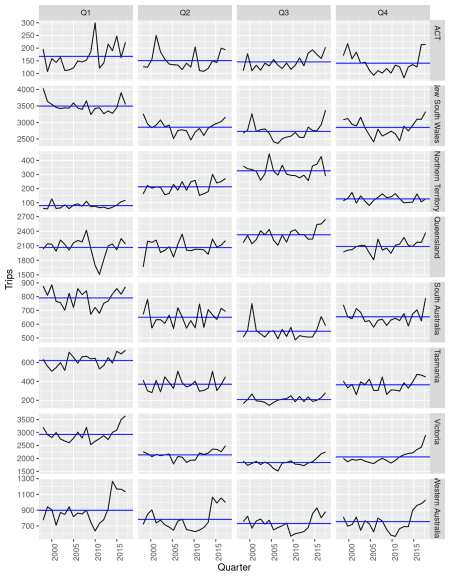
A season plot is shown below. Here it is clear that the southern states of Australia (Tasmania, Victoria and South Australia) have strongest tourism in Q1 (their summer), while the northern states (Queensland and the Northern Territory) have the strongest tourism in Q3 (their dry season).

holidays %>% gg\_season(Trips)



A subseries plot allows changes in seasonality over time to be easily visualized. The blue lines shows the mean across the years in each panel. Here it is clear that Western Australian tourism has jumped markedly in recent years, while Victorian tourism has increased in Q1 and Q4 but not in the middle of the year.

holidays %>% gg\_subseries(Trips)



The ACF is commonly used to assess the dynamic information in a time series. This is computed using the ACF() function for all series. This also produces a tsibble, but with the index being the lag.

holidays %>% ACF(Trips)

## # A tsibble: 152 x 3 [1Q]

## # Key: State [8]

## State lag acf

##

## 1 ACT 1Q 0.0877

## 2 ACT 2Q 0.252

## 3 ACT 3Q -0.0496

## 4 ACT 4Q 0.300

## 5 ACT 5Q -0.0741

## 6 ACT 6Q 0.269

## 7 ACT 7Q -0.00504

## 8 ACT 8Q 0.236

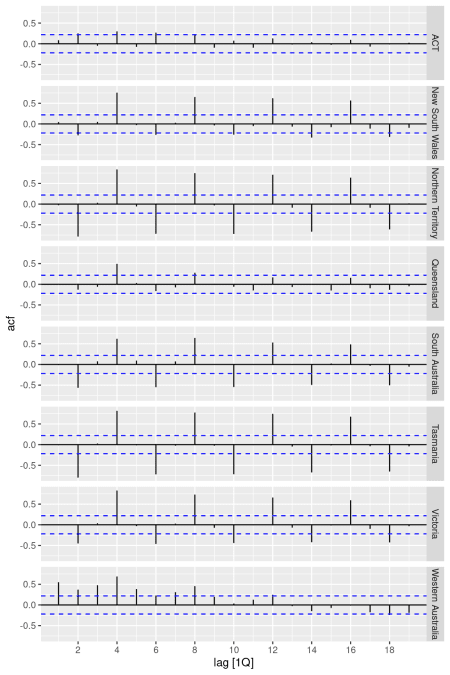
## 9 ACT 9Q -0.0953

## 10 ACT 10Q 0.0750

## # … with 142 more rows

To plot the ACFs for all series, we can pass the result to autoplot().

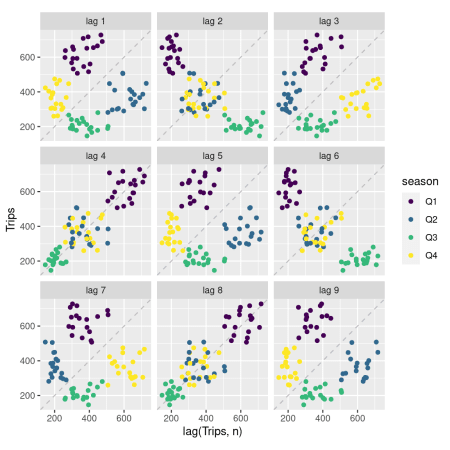
holidays %>% ACF(Trips) %>% autoplot()



Here, the low seasonality in the ACT is evident compared to the other states.

The remaining two graphical methods require only one time series. So we filter out the Tasmanian holiday data to illustrate them.

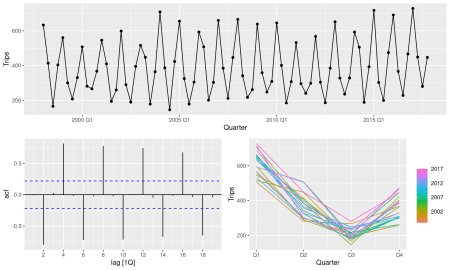
holidays %>% filter(State=="Tasmania") %>% gg\_lag(Trips, geom="point")



This lag plot shows a scatterplot of the lagged observation (vertical axis) against the current observation, with points coloured by the current quarter. The correlations of these lag plots are what make up the ACF. In this example, it is clear that Q1 is a strong quarter for Tasmania, and that the seasonality induces positive correlations at lags 4 and 8, but negative correlations at lags 2 and 6.

Finally, we show a composite plot created using gg\_tsdisplay(). This is a little different from the corresponding ggtsdisplay() function in the forecast package which showed the PACF in the bottom right panel by default. I think the season plot is a little more informative for exploratory data analysis, so that is what is shown by default in this new function. The other panels are the same.

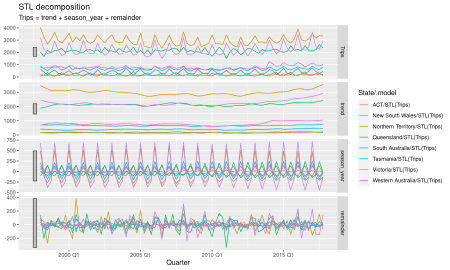
holidays %>% filter(State=="Tasmania") %>% gg\_tsdisplay(Trips)



**Decompositions**

The stats package provides the stl() function for STL decomposition of single time series with one seasonal period. The forecast package extended this with mstl() to allow for multiple seasonal periods. The feasts package allows for more flexible seasonality and for multiple series to be handled simultaneously.

holidays %>% STL(Trips) %>% autoplot()



All components from all series are shown here. Note that the annual seasonality has been estimated by default. With time series containing other seasonal periods, more than one seasonal component will be produced. These can be controlled using the STL arguments.

To demonstrate on a more difficult series, here is an STL decomposition for half hourly electricity data.

library(lubridate)

##

## Attaching package: 'lubridate'

## The following objects are masked from 'package:tsibble':

##

## interval, new\_interval

## The following object is masked from 'package:base':

##

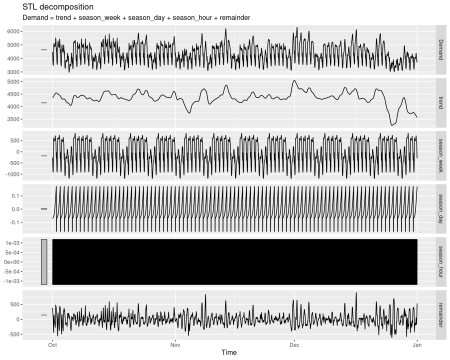
## date

tsibbledata::vic\_elec %>%

filter(yearmonth(Date) >= yearmonth("2014 Oct")) %>%

STL(Demand ~ trend(window=77) + season(window="periodic")) %>%

autoplot()



The hourly seasonality is largely meaningless – we do not expect electricity demand to have a periodic effect within the hour – and the daily seasonality has been largely captured in the weekly seasonality above it. The confounding of these two components makes it hard to interpret the daily seasonality. So we can drop the hourly and daily components and just model the weekly seasonality instead.

tsibbledata::vic\_elec %>%

filter(yearmonth(Date) >= yearmonth("2014 Oct")) %>%

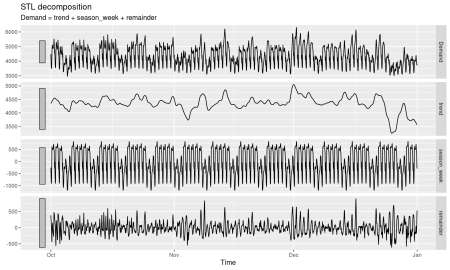
STL(

Demand ~ trend(window=77) +

season("week", window="periodic")

) %>%

autoplot()



The remainder term captures the difference from what you would expect if the demand was simply a function of the time of week. The variations from the weekly pattern, due to holidays or unusual weather, will show up in the remainder series.

**Features and statistics**

The feasts package does much more than graphics,