library(tidyverse)

library(tsibble)

library(lubridate)

library(feasts)

library(fable)

In the [previous post](https://robjhyndman.com/hyndsight/fable/) about the new [**fable** package](https://fable.tidyverts.org/), we saw how fable can produce forecast distributions, not just point forecasts. All my examples used Gaussian (normal) distributions, so in this post I want to show how non-Gaussian forecasting can be done.

As an example, we will use eating-out expenditure in my home state of Victoria.

vic\_cafe <- tsibbledata::aus\_retail %>%

filter(

State == "Victoria",

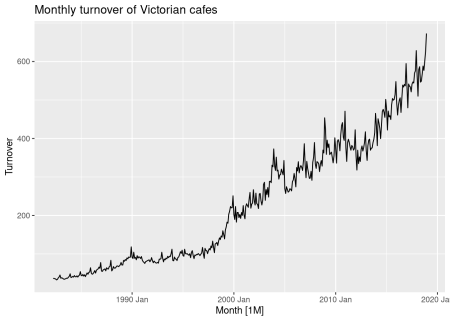
Industry == "Cafes, restaurants and catering services"

) %>%

select(Month, Turnover)

vic\_cafe %>%

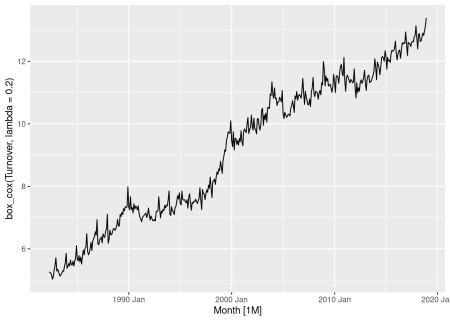
autoplot(Turnover) + ggtitle("Monthly turnover of Victorian cafes")



**Forecasting with transformations**

Clearly the variance is increasing with the level of the series, so we will consider modelling a Box-Cox transformation of the data.

vic\_cafe %>% autoplot(box\_cox(Turnover, lambda = 0.2))



The variance now looks more homogeneous across the series, allowing us to fit an additive model. I chose the value of \(\lambda=0.2\) by eye, but you can use the guerrero function for an automated approach.

vic\_cafe %>% features(Turnover, guerrero)

## # A tibble: 1 x 1

## lambda\_guerrero

##

## 1 0.124

It suggests something slightly smaller, but I will stick with 0.2.

Now to fit a model. For this post I will use ETS models

fit <- vic\_cafe %>%

model(ets = ETS(box\_cox(Turnover, 0.2)))

fit

## # A mable: 1 x 1

## ets

##

## 1

An ETS(A,A,A), or additive Holt-Winters model, has been selected for the transformed data. We can produce forecasts in the usual way.

fc <- fit %>% forecast(h = "3 years")

fc

## # A fable: 36 x 4 [1M]

## # Key: .model [1]

## .model Month Turnover .distribution

##

## 1 ets 2019 Jan 608. t(N(13, 0.020))

## 2 ets 2019 Feb 563. t(N(13, 0.028))

## 3 ets 2019 Mar 629. t(N(13, 0.036))

## 4 ets 2019 Apr 615. t(N(13, 0.044))

## 5 ets 2019 May 613. t(N(13, 0.052))

## 6 ets 2019 Jun 593. t(N(13, 0.061))

## 7 ets 2019 Jul 624. t(N(13, 0.069))

## 8 ets 2019 Aug 640. t(N(13, 0.077))

## 9 ets 2019 Sep 630. t(N(13, 0.085))

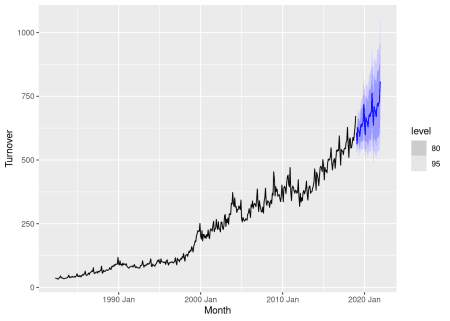
## 10 ets 2019 Oct 642. t(N(13, 0.093))

## # … with 26 more rows

Note that the distributions are given as transformed normal, denoted by t(N). The point forecast (in column Turnover) is the mean of this distribution. The back-transformation and bias adjustment is done automatically.

One particularly clever part of the package (thanks to Mitchell O’Hara-Wild) is that you can use any transformation in the model() function, and the bias adjustment is computed based on a Taylor series expansion using numerical derivatives. So you will always get the approximate mean as the point forecast, even when using some exotic transformation for which you have no analytic expression for the bias.

fc %>% autoplot(vic\_cafe)



**Bootstrapped prediction intervals**

In the preceding analysis, there was still a normality assumption for the residuals of the model applied to the transformed data. If you want to avoid that as well, you can use bootstrapped intervals. These are constructed from simulated future sample paths where the residuals are resampled as possible future errors.

We can simulate future sample paths using the generate() function.

sim <- fit %>% generate(h = "3 years", times = 5, bootstrap = TRUE)

sim

## # A tsibble: 180 x 4 [1M]

## # Key: .model, .rep [5]

## .model .rep Month .sim

##

## 1 ets 1 2019 Jan 583.

## 2 ets 1 2019 Feb 600.

## 3 ets 1 2019 Mar 679.

## 4 ets 1 2019 Apr 669.

## 5 ets 1 2019 May 671.

## 6 ets 1 2019 Jun 624.

## 7 ets 1 2019 Jul 644.

## 8 ets 1 2019 Aug 689.

## 9 ets 1 2019 Sep 702.

## 10 ets 1 2019 Oct 690.

## # … with 170 more rows

Here we have generated five possible sample paths for future months. The .rep variable provides a new key for the tsibble. The back-transformation of the sample paths is handled automatically. If we had multiple models, each would be used to generate future sample paths provided the corresponding generate function existed. (In the current version of **fable**, we have not yet implemented this for ARIMA models.)

The plot below shows the five sample paths along with the last few years of historical data.

vic\_cafe %>%

filter(year(Month) >= 2008) %>%

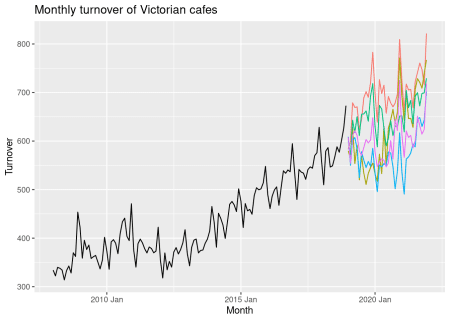
ggplot(aes(x = Month)) +

geom\_line(aes(y = Turnover)) +

geom\_line(aes(y = .sim, colour = as.factor(.rep)), data = sim) +

ggtitle("Monthly turnover of Victorian cafes") +

guides(col = FALSE)



Prediction intervals are calculated using percentiles of the future sample paths. This is all built into the forecast() function so you do not need to call generate() directly.

fc <- fit %>% forecast(h = "3 years", bootstrap = TRUE)

fc

## # A fable: 36 x 4 [1M]

## # Key: .model [1]

## .model Month Turnover .distribution

##

## 1 ets 2019 Jan 608. t(sim(=dbl[5000]))

## 2 ets 2019 Feb 563. t(sim(=dbl[5000]))

## 3 ets 2019 Mar 629. t(sim(=dbl[5000]))

## 4 ets 2019 Apr 615. t(sim(=dbl[5000]))

## 5 ets 2019 May 613. t(sim(=dbl[5000]))

## 6 ets 2019 Jun 593. t(sim(=dbl[5000]))

## 7 ets 2019 Jul 624. t(sim(=dbl[5000]))

## 8 ets 2019 Aug 640. t(sim(=dbl[5000]))

## 9 ets 2019 Sep 630. t(sim(=dbl[5000]))

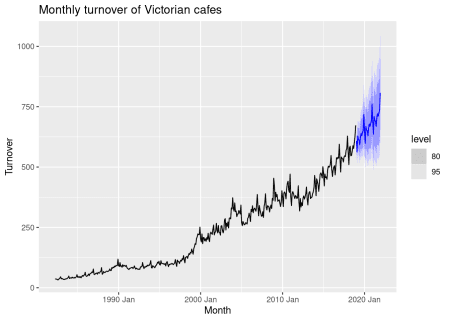
## 10 ets 2019 Oct 642. t(sim(=dbl[5000]))

## # … with 26 more rows

Notice that the forecast distribution is now represented as transformed simulation with 5000 sample paths. This default number can be modified in the times argument for forecast().

fc %>% autoplot(vic\_cafe) +

ggtitle("Monthly turnover of Victorian cafes")



In this example, the resulting forecast intervals are almost identical to those obtained when we assumed the residuals were normally distributed.

**Accuracy calculations**

We can check whether the bootstrapping helped by comparing the CRPS values from both models after doing a training/test set split.

train <- vic\_cafe %>% filter(year(Month) <= 2014)

fit <- train %>%

model(ets = ETS(box\_cox(Turnover, 0.2)))

fit %>%

forecast(h = "4 years", bootstrap = FALSE) %>%

accuracy(vic\_cafe,

measures = distribution\_accuracy\_measures

)

## # A tibble: 1 x 4

## .model .type percentile CRPS

##

## 1 ets Test 12.3 24.4

fit %>%

forecast(h = "4 years", bootstrap = TRUE) %>%

accuracy(vic\_cafe,

measures = distribution\_accuracy\_measures

)

## # A tibble: 1 x 4

## .model .type percentile CRPS

##

## 1 ets Test 12.4 24.5

In this case it makes almost no difference which of the two approaches is used, so the non-bootstrap approach is preferred because it is much faster.