Air Passenger Forecasting

Seasonal patterns become especially important when analyzing consumer-related time series – consumers tend to travel, spend, and consume in seasonal patterns, and so removing these patterns is an important first step before any future analysis.

Therefore, this notebook looks at how to remove various seasonal trends from a time series of air passenger demand.

Let's start as always by importing some packages we'll need along the way:

```
require( dplyr )
require( lubridate )
require( forecast )
```

Next import the data, which comes with the forecast package:

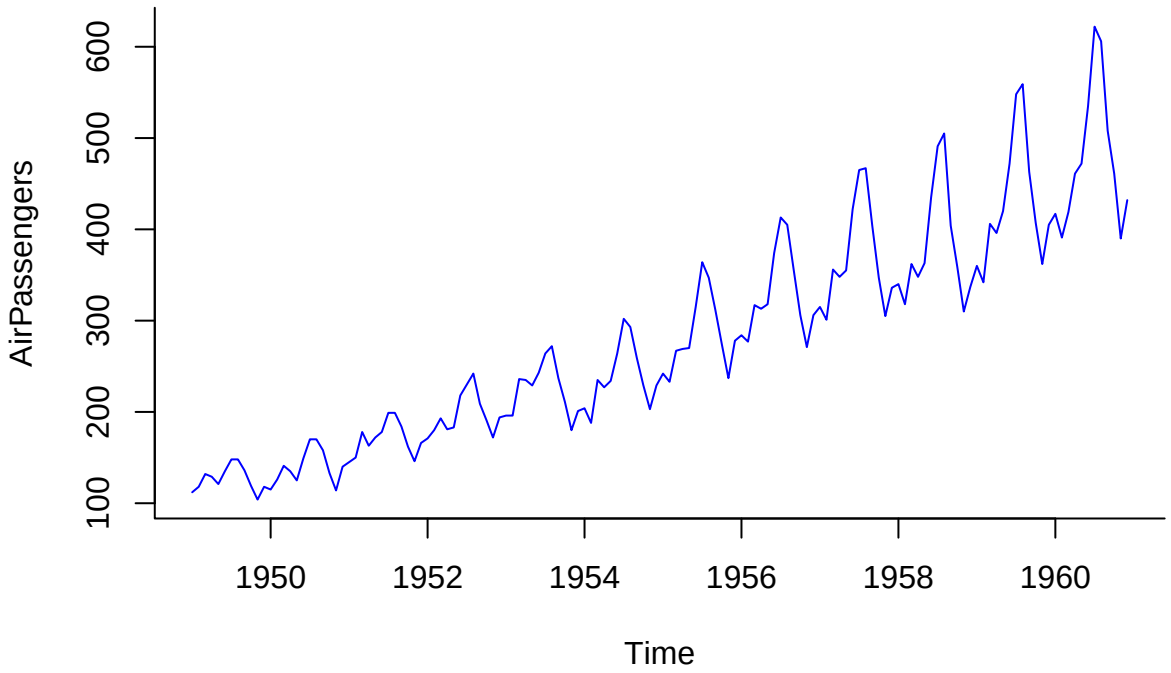
```
data("AirPassengers")
```

And also define a custom color palette for plotting:

```
custom_colors = c(
  "#0000ff",
  "#9d02d7",
  "#cd34b5",
  "#ea5f94",
  "#fa8775",
  "#ffb14e",
  "#ffd700"
)
```

We can plot the time series to inspect it visually:

```
plot( AirPassengers
      , type = 'l'
      , lwd = 1
      , col = custom_colors[1]
      , bty = 'l'   )
```



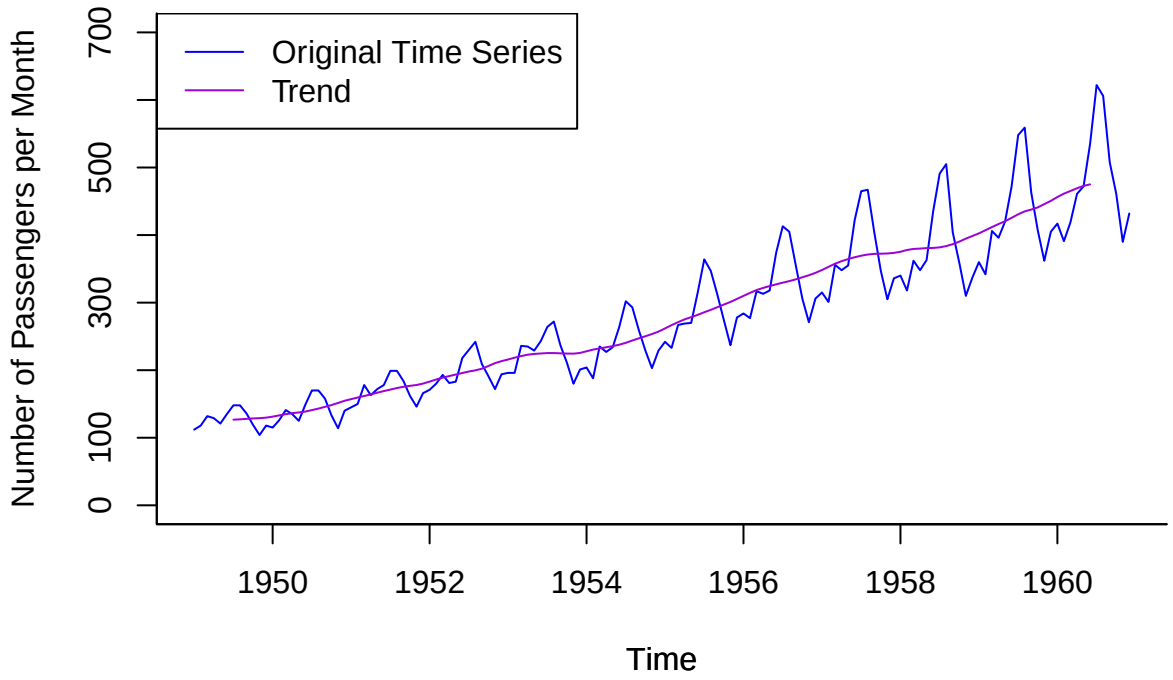
We can see a clear patter of seasonality in the data and that the data is recorded monthly.

The next step is to identify the underlying trend in the data. In other words, regardless of the ups and downs of the seasonality, is the time series increasing or decreasing over time? We do this by taking a moving average of the data.

The pattern of the data appears to repeat on an annual basis, and the data is recorded monthly, so we set the order of the moving average smoother to be 12 months.

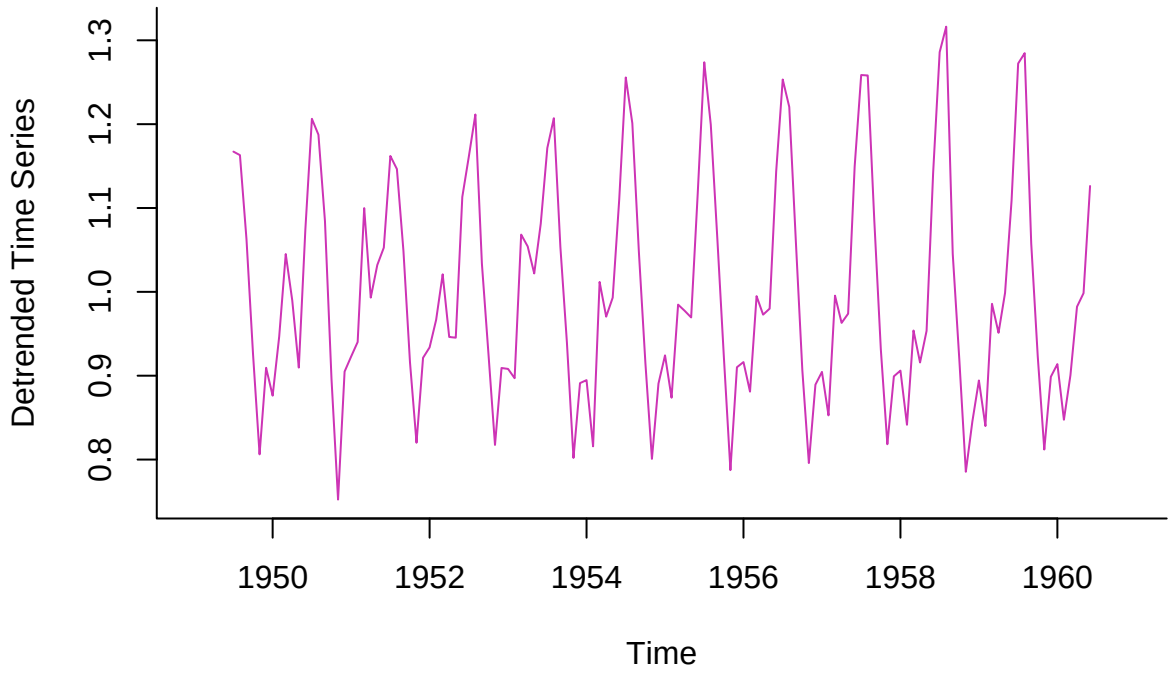
```
trend_passengers = ma( AirPassengers , order = 12 , centre = TRUE )

ylim = c( 0 , 700 )

plot( AirPassengers
      , type = 'l'
      , lwd = 1
      , col = custom_colors[1]
      , bty = 'l'
      , ylim = ylim
      , ylab = 'Number of Passengers per Month'
)

par( new = TRUE )

plot( trend_passengers
      , type = 'l'
      , lwd = 1
      , col = custom_colors[2]
      , bty = 'l'
      , ylim = ylim
      , ylab = NA
      , yaxt= 'n'
      , xaxt = 'n'
)
```

```
legend( 'topleft'
        , legend = c( 'Original Time Series' , 'Trend' )
        , col = custom_colors[1:2]
        , lwd = 1 )
```



And if we take that trend and remove it from the time series, we get the resulting de-trended series:

```
detrended_data = AirPassengers / trend_passengers

plot( detrended_data
      , type = 'l'
      , lwd = 1
      , col = custom_colors[3]
      , bty = 'l'
      , ylab = 'Detrended Time Series'
)
```

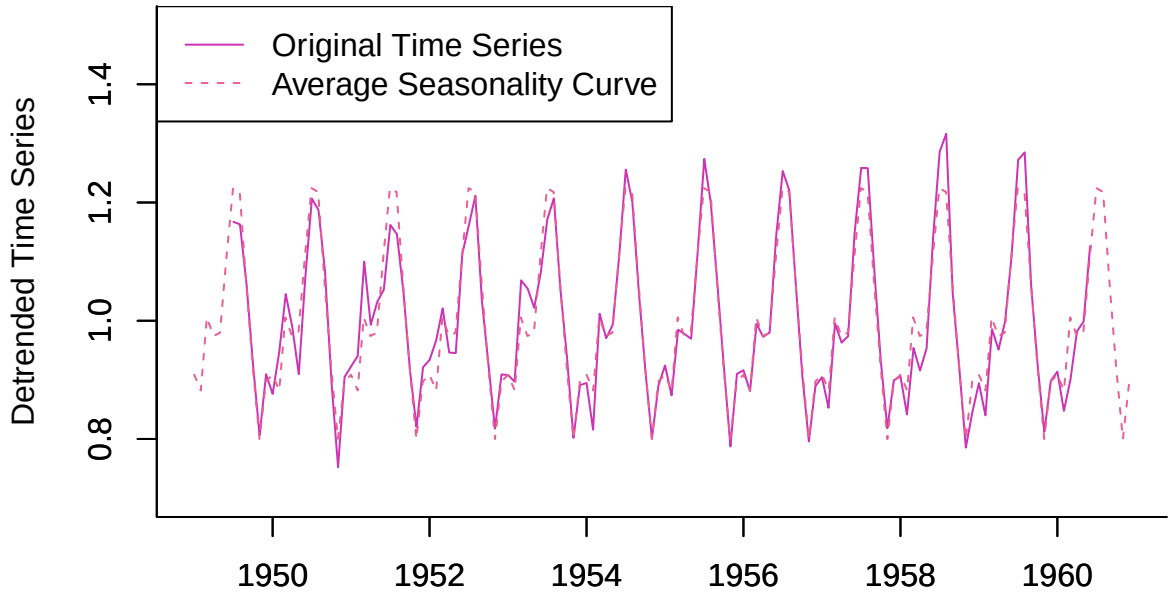


With the trend removed we can see the raw seasonality in the data even better.

Next we can compute the average seasonality curve.

```
monthly_matirx = matrix( data = detrended_data , nrow = 12 , byrow = TRUE )
seasonal_averages = colMeans( monthly_matirx , na.rm = TRUE )

# Get the first year of the data
start.year =
  detrended_data %>%
  time %>%
  as.numeric %>%
  date_decimal %>%
  min %>%
  year

seasonal_average_dates =
  seq( from = as.Date(  paste0( start.year,'-01-01' ) ) , length = 12*12 , by = '1 month' )

ylim = c( 0.7 , 1.5 )

plot( detrended_data
      , type = 'l'
      , lwd = 1
      , col = custom_colors[3]
      , bty = 'l'
      , ylab = 'Detrended Time Series'
      , ylim = ylim
      , xlab = NA
)

par( new = TRUE )
```

```
plot( y = rep( seasonal_averages , 12 )
      , x = seasonal_average_dates
      , type = 'l'
      , lwd = 1
      , lty = 2
      , col = custom_colors[4]
      , bty = 'l'
      , ylab = NA
      , ylim = ylim
      , xlab = NA
)

legend( 'topleft'
        , legend = c( 'Original Time Series'
                      , 'Average Seasonality Curve' )
        , col = custom_colors[3:4]
        , lwd = 1
        , lty = c( 1, 2 ) )
```

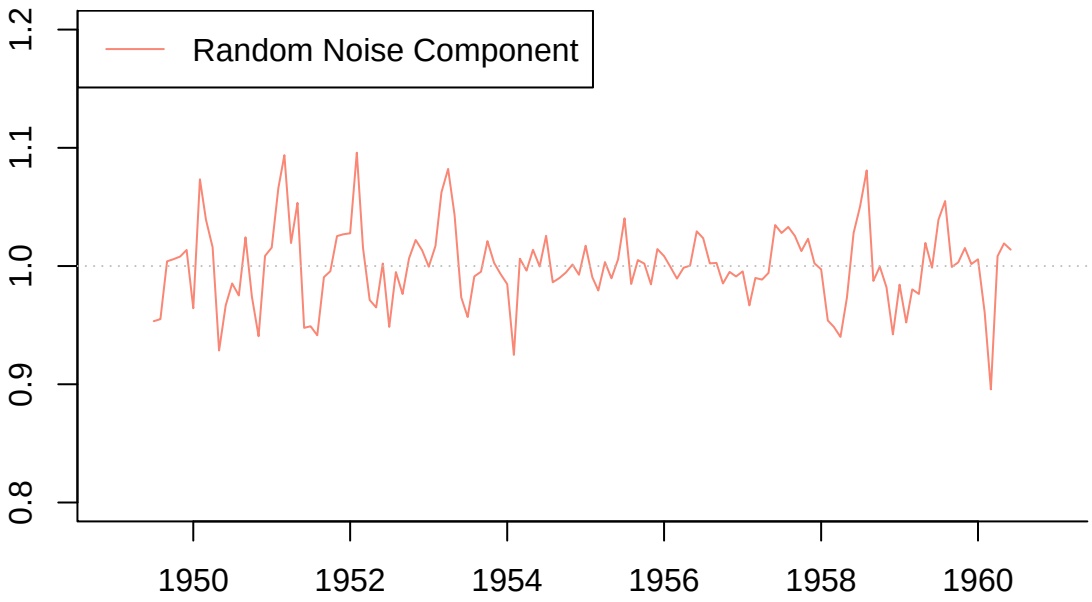


We can see that the average seasonality curve does a good job at explaining the detrended time series.

At this point we've extracted both the trend component (the long-term changes over time) and the seasonality component. Whatever remains after accounting for these two underlying patterns, we can consider that to be "random noise".

```
random_component = AirPassengers / ( trend_passengers * seasonal_averages )

plot( y = random_component
      , x = seasonal_average_dates
      , type = 'l'
      , lwd = 1
      , lty = 1
      , col = custom_colors[5]
      , bty = 'l'
      , ylab = NA
      , ylim = c( 0.8 , 1.2 )
```

```
      , xlab = NA
)

abline( h = 1 , col = 'gray' , lwd = 1 , lty = 3 )

legend( 'topleft'
        , legend = c( 'Random Noise Component'  )
        , col = custom_colors[5]
        , lwd = 1
        , lty = c( 1 ) )
```



A quick note on interpretation:

A value closer to one here means that our model (the combined trend and seasonal components) does a better job at explaining air passenger demand at that point in time. As the random noise value gets away from a value of one, whether that be greater or less than one, then that represents a larger difference between our model and the original data.

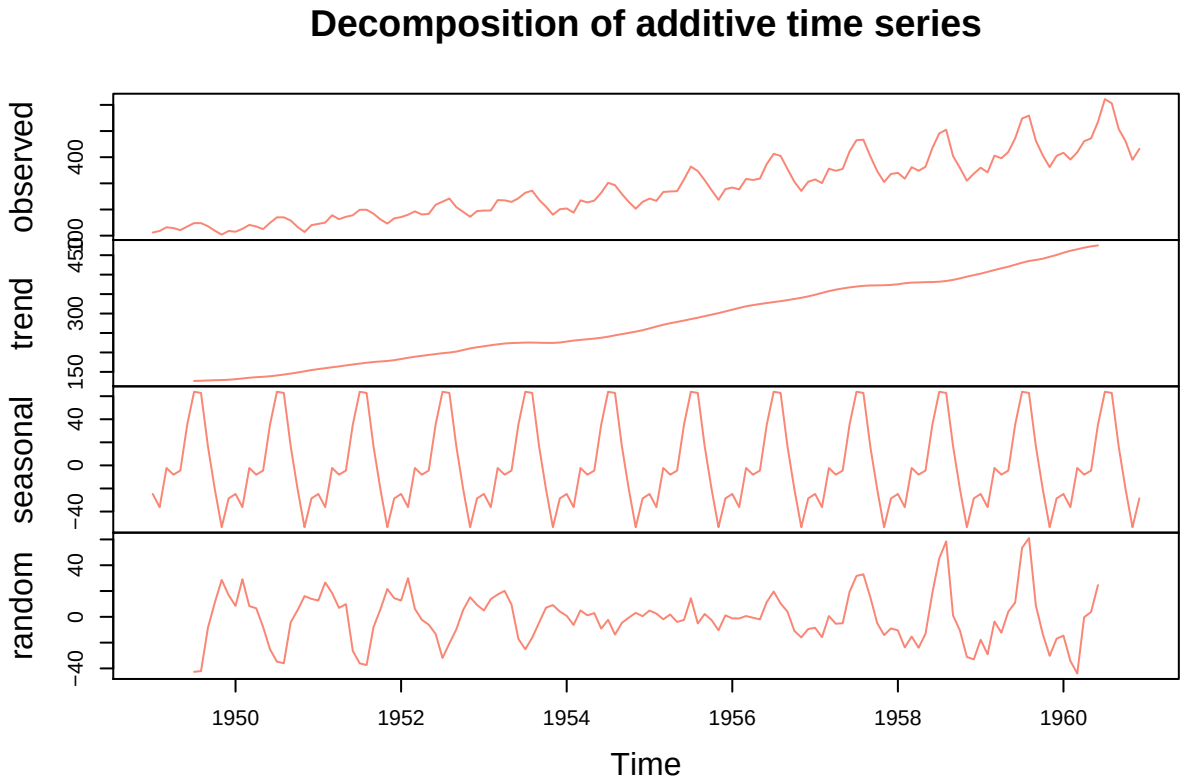
As the value of the random noise component grows larger than one, that means that our model UNDERpredicts the original data. As the value of the noise compoenent gets less than one, that means that our mode OVERpredicts the original data at that point in time.

Another thing to note here is that it might not be fair to call this noise "random". What this signal actually represents is: after accounting for both the trend and seasonality components, what signal is left over? This might not be random in that there might be other factors not measured by this model affecting air passenger demand, and these might be exactly the kind of factors we want to explain.

So to me the term "excess" seems more applicable – as in, "excess passengers". The number of passengers in excess of those predicted by the seasonal model.

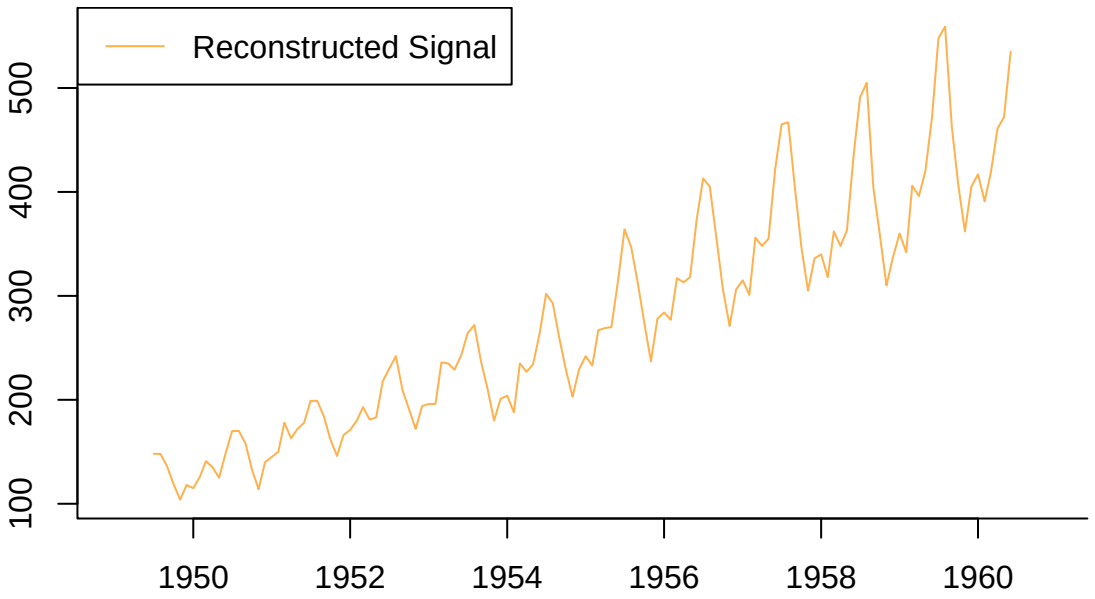
Finally, we have done all of this manually, but that was really just for investigation – R provides the forecast package, which allows you to do all of this with a simple decompose command:

```
plot(
  decompose( AirPassengers )
  , col = c( custom_colors[5] )
)
```



To illustrate all of this with one final step, we can use our three components that we extracted (trend, seasonal, and excess) to reconstruct the exaxt original signal:

```
reconstructed_signal = trend_passengers * seasonal_averages * random_component

plot( y = reconstructed_signal
      , x = seasonal_average_dates
      , type = 'l'
      , lwd = 1
      , lty = 1
      , col = custom_colors[6]
      , bty = 'l'
      , ylab = NA
      , xlab = NA
)

legend( 'topleft'
        , legend = c( 'Reconstructed Signal'  )
        , col = custom_colors[6]
        , lwd = 1
        , lty = c( 1 ) )
```



And this matches our original.

As I mentioned at the start, this kind of decomposition tends to be especially important when it comes to consumer trends – consumers tend to travel and consume various goods in seasonal patterns, and so removing these patterns is an important first step into any future analysis.