Time Series Forecasting with R

Yazeed Alzahrani

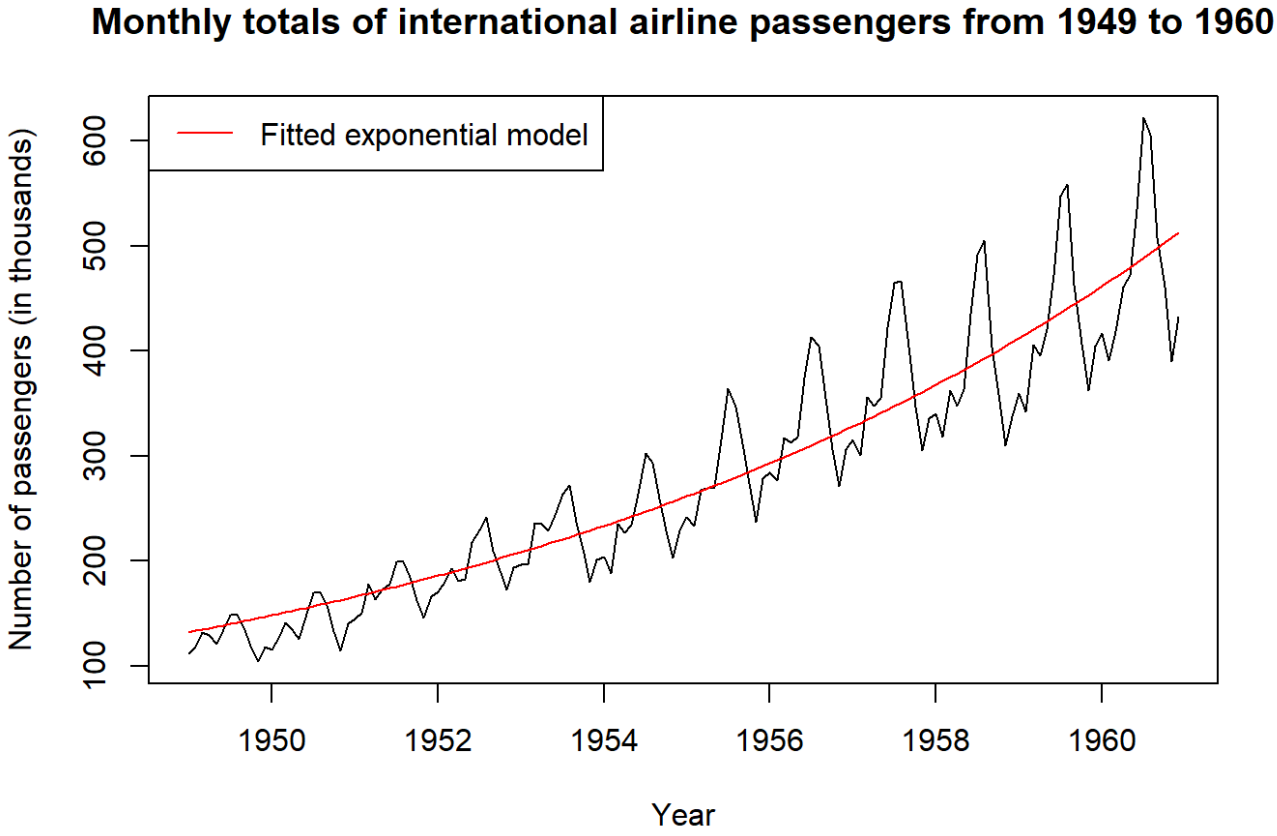
March 2024

Introduction

The purpose of this project is to apply Meta's Prophet and the Holt-Winters forecasting models to a time series. The dataset used is Box and Jenkins airline data, which is a time series of the monthly totals of international airline passengers from 1949 to 1960.

The following code plots the data and fits an exponential model using the non-linear least squares function (`nls`). The exponential model provides a reasonable fit to the data, as the number of international airline passengers grew at an accelerating rate in the 1950s due to advances in aviation technology and post-World War II economic recovery. In the final bit of code a persistent error was encountered when trying to plot the data; with the message "plot.new has not been called yet" being displayed in the console. This was only solved by separating the `plot()` , `points()` and `legend()` functions by a semicolon.

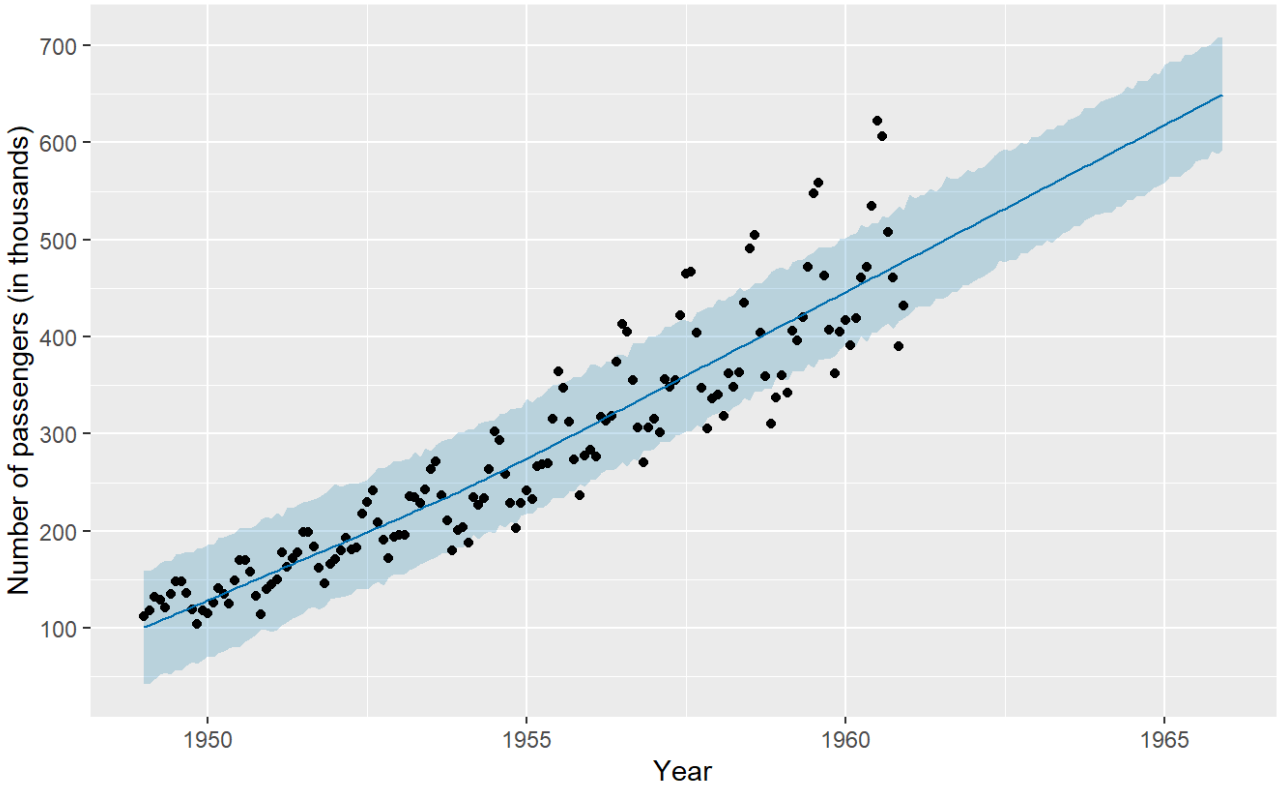
```
data(AirPassengers) # Loads the data

# Puts the data in a dataframe with two columns; ds for time and y for the number of pa
ssengers
AirPassengers.df = data.frame(ds=zoo::as.yearmon(time(AirPassengers)), y=AirPassengers)

# Initial values for the parameters a and b are specified and the function iterates thr
ough values of a and b
# until it converges on the values that provide the best fit.
exponential_model = nls(AirPassengers ~ a * exp(b * time(AirPassengers)), data = AirPas
sengers.df,
                        start=list(a = 100, b = 0))

plot(AirPassengers, main="Monthly totals of international airline passengers from 1949
to 1960", xlab = "Year", ylab ="Number of passengers (in thousands)"); points(time(AirP
assengers), fitted(exponential_model), type="l", col="red"); legend("topleft", legend
="Fitted exponential model", col="red", lty=1)
```



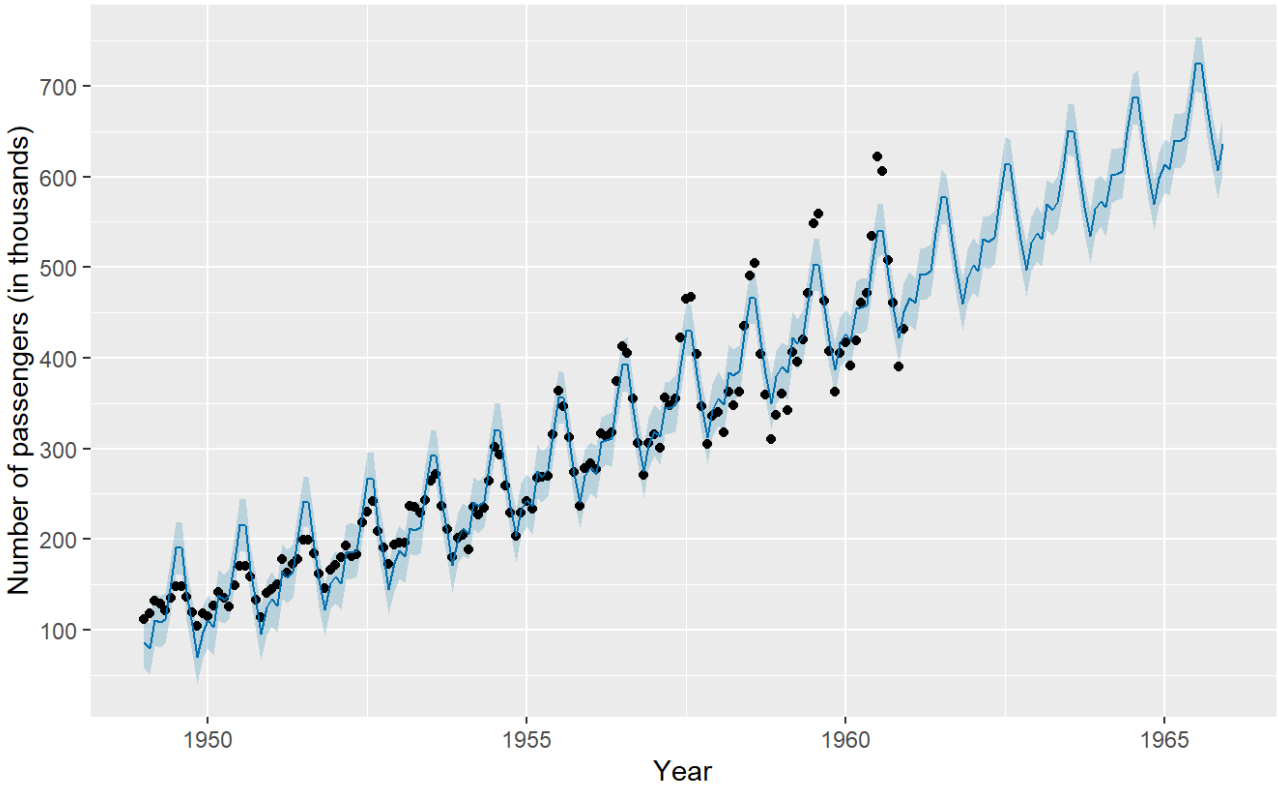
Disabling/enabling seasonality

In the code below the Prophet model is used to forecast the time series 60 months into the future. The forecast is plotted for two cases, the first case with the `yearly.seasonality` setting disabled and the second case with the setting enabled. It is evident from the two plots that the seasonality options in the `prophet()` function are of crucial importance. The `make_future_dataframe()` function sets up the period of time ahead for the which the forecast will be made and the `predict()` function creates the forecast. The `scale_y_continuous()` function from `ggplot2` library is very convenient for adjusting the y-ticks on the y-axis.

```
AirPassengers.df = data.frame(ds=zoo::as.yearmon(time(AirPassengers)), y=AirPassengers)

seasonality_OFF_model = prophet::prophet(AirPassengers.df, yearly.seasonality=FALSE)

future_dates = prophet::make_future_dataframe(seasonality_OFF_model, periods=60, freq
="month")

forecast = predict(seasonality_OFF_model, future_dates)

plot(seasonality_OFF_model, forecast, main="Prophet forecast with disabled seasonalit
y", xlabel="Year", ylabel="Number of passengers (in thousands)", xaxt="n") + ggplot2::s
cale_y_continuous(breaks = seq(0, 1000, by=100))
```



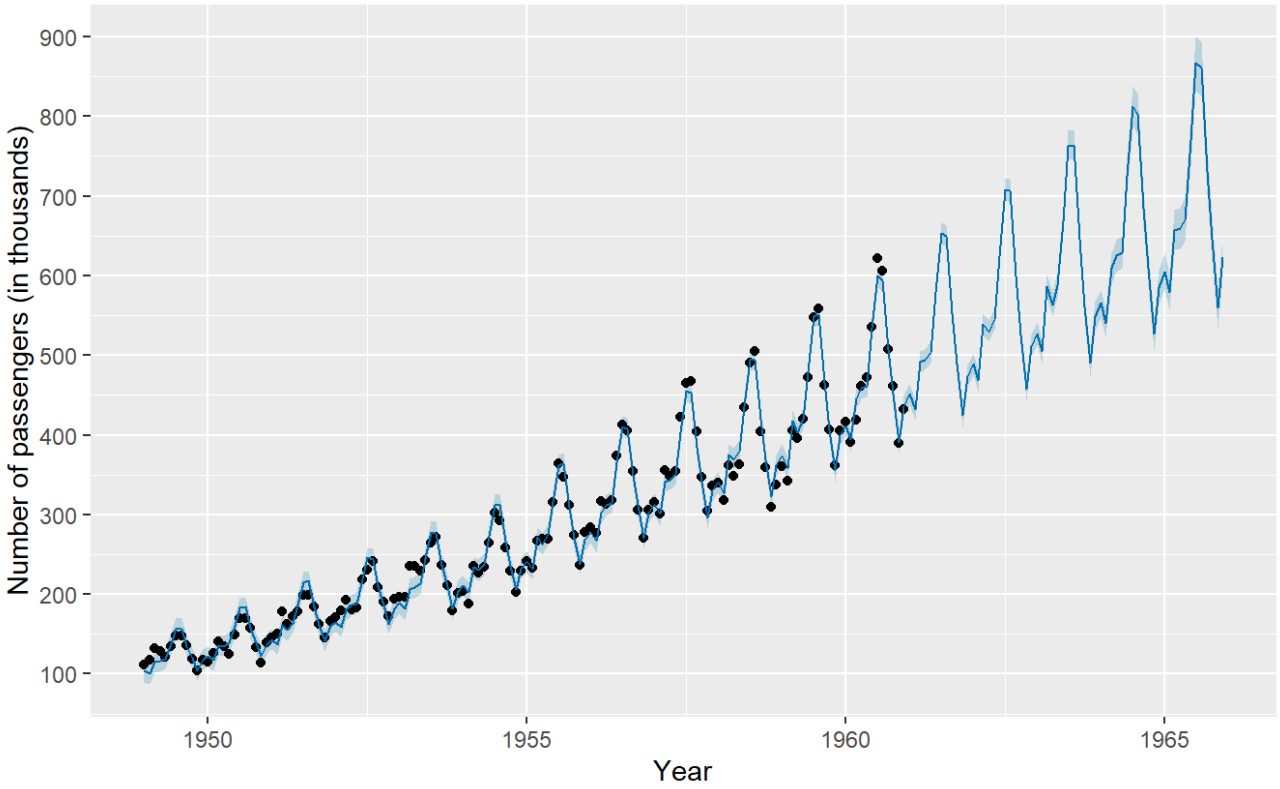

```
seasonality_ON_model = prophet::prophet(AirPassengers.df, yearly.seasonality=TRUE)

future_dates = prophet::make_future_dataframe(seasonality_ON_model, periods=60, freq="m
onth")

forecast = predict(seasonality_ON_model, future_dates)

plot(seasonality_ON_model, forecast, main="Prophet forecast with enabled seasonality",
xlabel="Year", ylabel="Number of passengers (in thousands)", xaxt="n") + ggplot2::scale
_y_continuous(breaks = seq(0, 1000, by=100))
```



Multiplicative seasonality mode

The Prophet model assumes by default additive seasonality in which the seasonal component of the time series is added to the trend. We can see from the previous plot that this does not capture the true seasonality well enough. The exponential nature of the time series results in the seasonal component *increasing* with the trend instead of remaining constant, which is better captured by multiplicative seasonality.

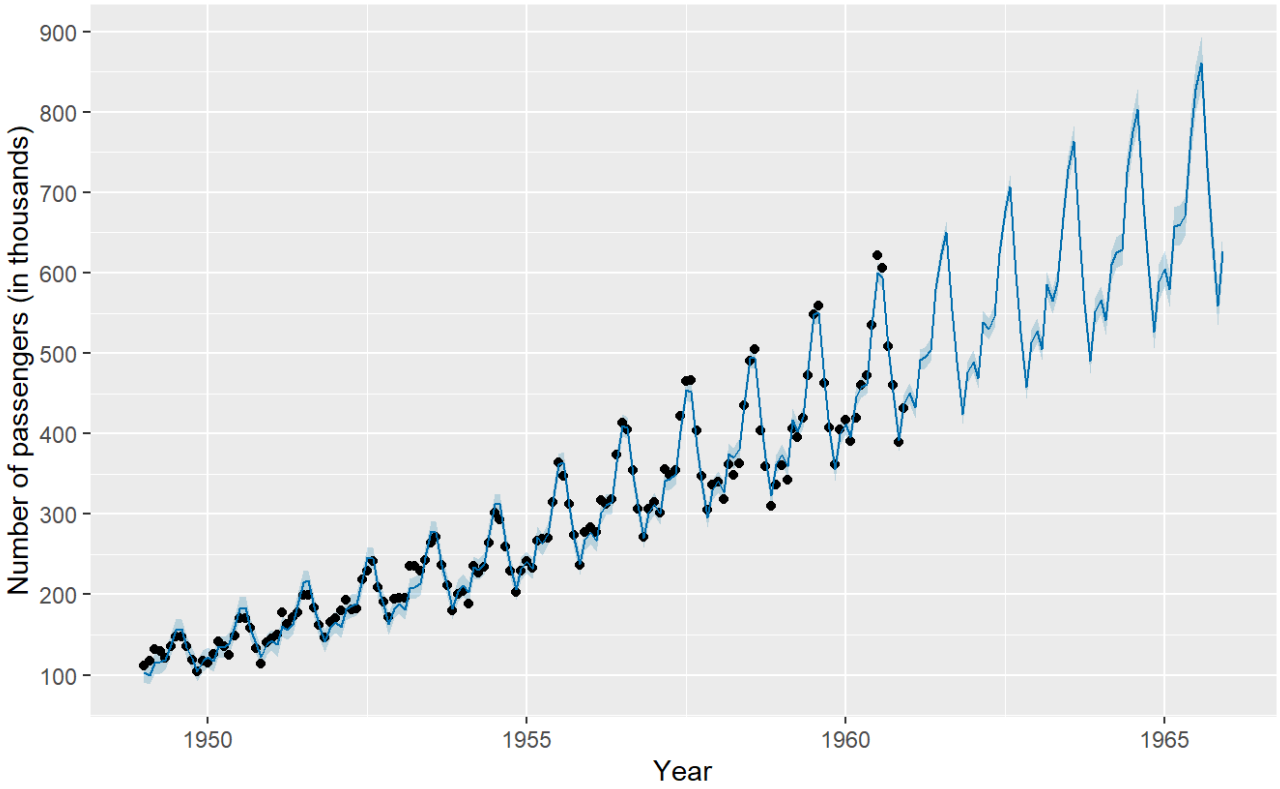
```
multiplicative_model = prophet::prophet(AirPassengers.df, yearly.seasonality=TRUE, seas
onality.mode = "multiplicative")

future_dates = prophet::make_future_dataframe(multiplicative_model, periods=60, freq="m
onth")

forecast = predict(multiplicative_model, future_dates)

plot(multiplicative_model, forecast, main="Prophet forecast with multiplicative seasona
lity", xlabel="Year", ylabel="Number of passengers (in thousands)", xaxt="n") + ggplot
2::scale_y_continuous(breaks = seq(0, 1000, by=100))
```



Adding holiday dates

The Prophet model provides the option of specifying the time periods of holidays in the dataset, which can increase the accuracy of the forecast. However the only difference the following plot shows is sharper peaks.

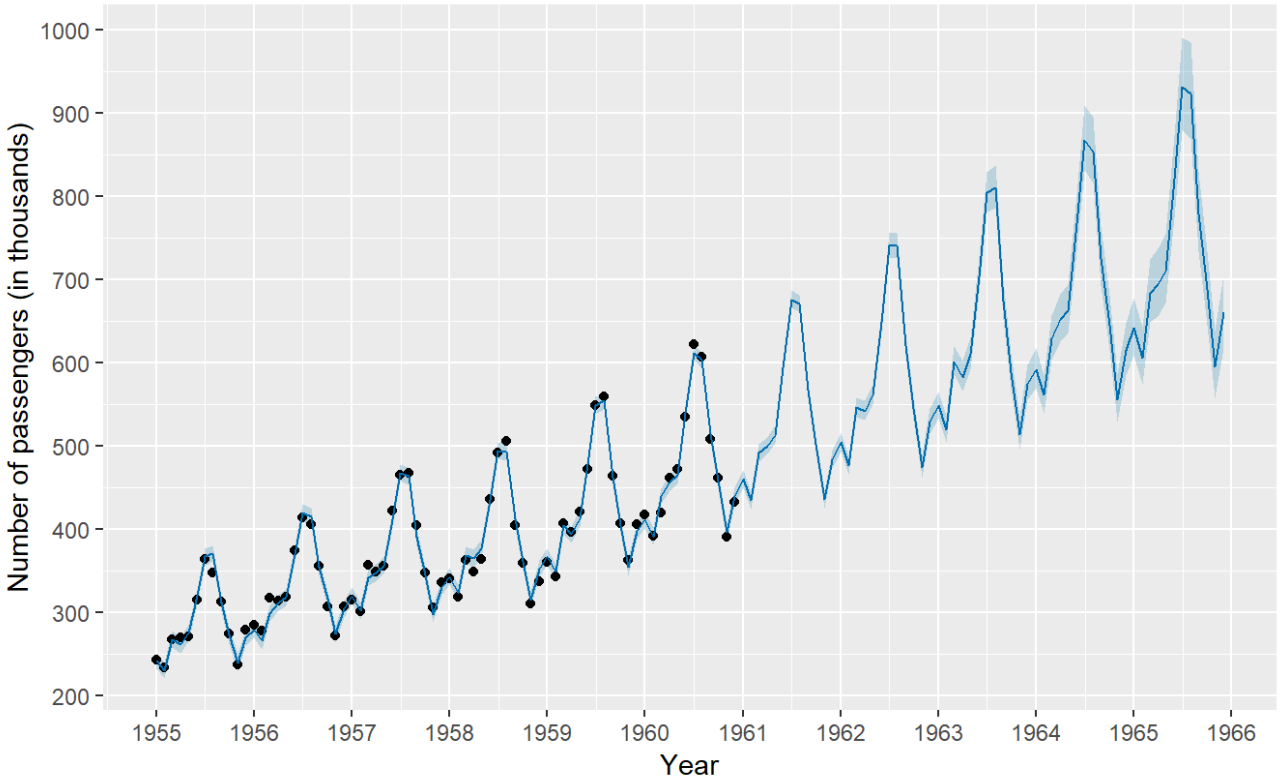
```
# Creates a sequence of dates at 1 year intervals within the specified time periods.
summer_holiday_dates = seq(as.Date("1949-07-01"), as.Date("1960-08-31"), by="year")
winter_holiday_dates = seq(as.Date("1949-12-01"), as.Date("1961-01-31"), by="year")

# A dataframe of the sequences of dates is created because the Prophet model can only w
ork with dataframes.
holidays_df = data.frame(holiday=c(rep("Summer Holiday", length(summer_holiday_dates)),
                                   rep("Winter Holiday", length(winter_holiday_date
s))),
                         ds=c(summer_holiday_dates, winter_holiday_dates))

model = prophet::prophet(AirPassengers.df, yearly.seasonality=TRUE, seasonality.mode =
"multiplicative", holidays = holidays_df)

future_dates = prophet::make_future_dataframe(model, periods=60, freq="month")

forecast = predict(model, future_dates)

plot(model, forecast, main="Prophet forecast with holiday dates included in the model",
xlabel="Year",
     ylabel="Number of passengers (in thousands)", xaxt="n") + ggplot2::scale_y_continu
ous(breaks = seq(0, 1000, by=100))
```



Using a subset of the dataset

Here only data from the last 5 years in the original dataset was used to create the forecast. This resulted in a general increase in the number of passengers in the forecast.

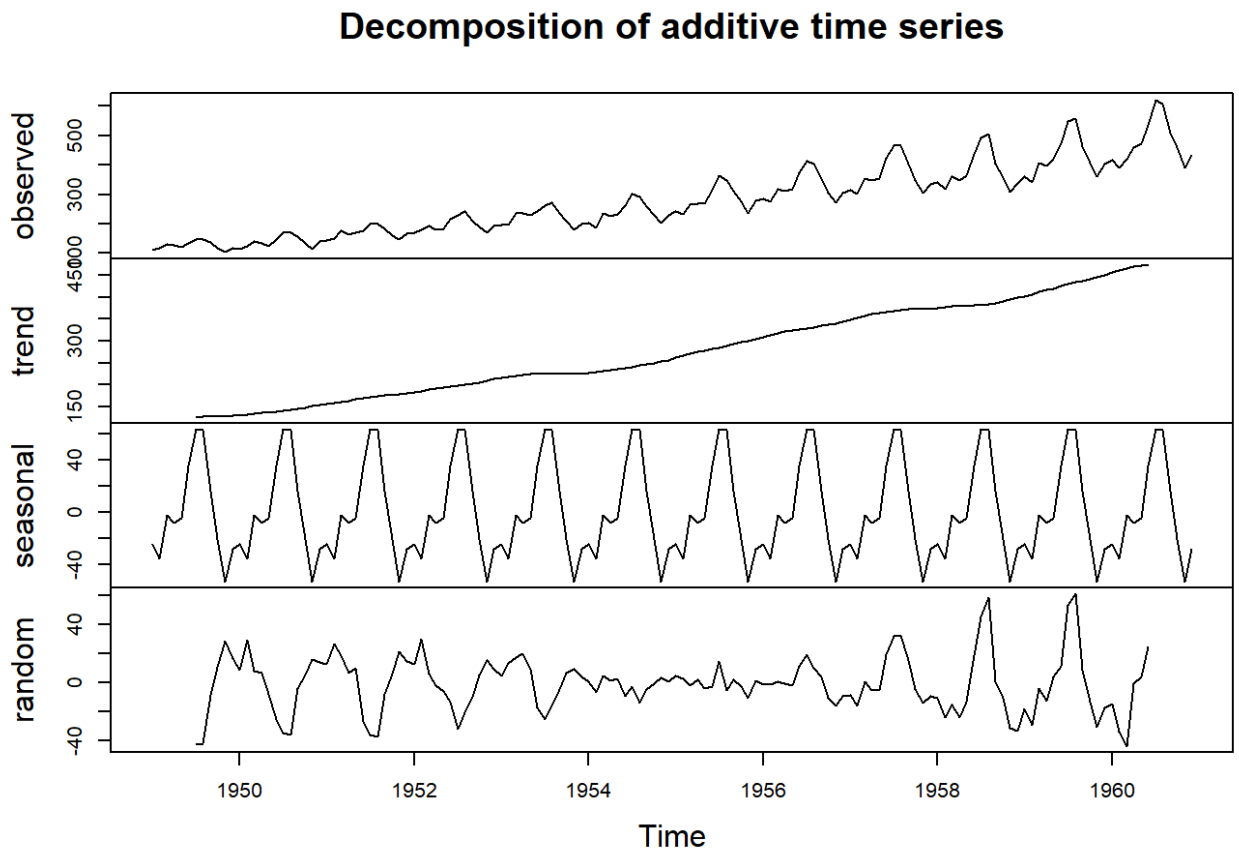
```
AirPassengers.df = data.frame(ds=zoo::as.Date(time(AirPassengers)), y=AirPassengers)

# Creates a new dataframe which contains data just from the last 5 years only.
AirPassengers_df_filtered = AirPassengers.df[AirPassengers.df$ds >= as.Date("1955-01-0
1"),]

model = prophet::prophet(AirPassengers_df_filtered, yearly.seasonality=TRUE, seasonalit
y.mode = "multiplicative")

future_dates = prophet::make_future_dataframe(model, periods=60, freq="month")

forecast = predict(model, future_dates)

plot(model, forecast, main="Prophet forecast using data from the last 5 years", xlabel
="Year", ylabel="Number of passengers (in thousands)", xaxt="n") + ggplot2::scale_y_con
tinuous(breaks=seq(0, 1000, by=100)) + ggplot2::scale_x_datetime(date_breaks="1 year",
date_labels="%Y")
```



Decomposing the time series

The function `decompose()` breaks the time series into its trend, seasonal, and random (residual error) components. We can observe in the charts below the upward trend in the data, the yearly seasonality, and the residual error which grows in magnitude at both ends of the time series.

```
data("AirPassengers")
plot(decompose(AirPassengers))
```



Holt-Winters Method

The simple exponential smoothing is not suitable for forecasting when there is an upward or a downward trend in the data. The Holt Method expands on the idea of exponential smoothing by introducing a trend component, and the Holt-Winters Method extends this further by adding a seasonal component to the model; which is convenient for our case as there is a clear upward trend and seasonality in the data. The Holt-Winters model is described by the following equations:

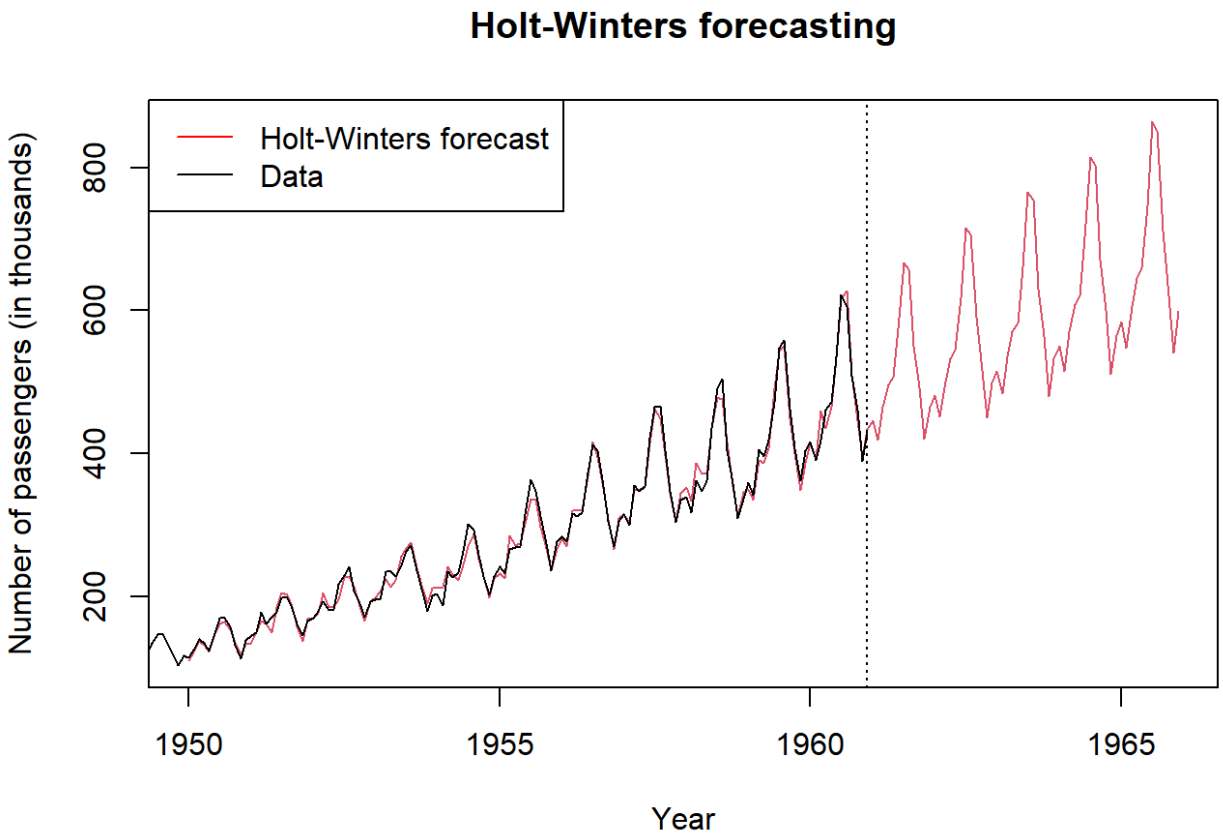
$$m_t = \alpha(x_t - s_{t-S}) + (1-\alpha)(m_{t-1} + b_{t-1})$$

$$b_t = \beta(m_t - m_{t-1}) + (1-\beta)b_{t-1}$$

$$s_t = \gamma(x_t - m_{t-1} - b_{t-1}) + (1-\gamma)s_{t-S}$$

where x_t is the time series, m_t the level component, b_t the trend component, s_t the seasonality component, and S is the seasonality period. A Holt-Winters model can be created in R by using the function `HoltWinters()`. It has three arguments α, β and γ which control the relative weights of the level, trend and seasonal components respectively. If these parameters are not assigned values then the Holt-Winters function will automatically compute their optimal values by minimising the squared one-step ahead prediction error.

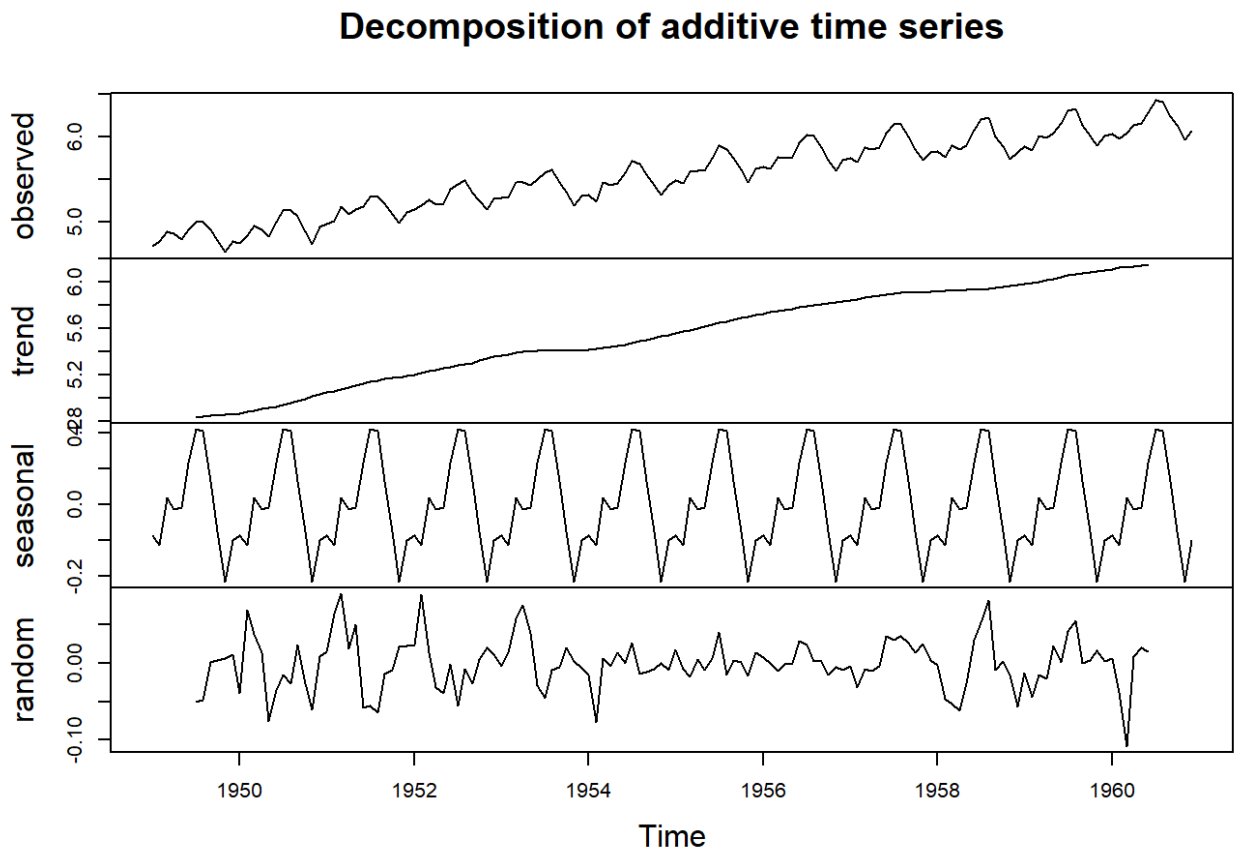
```
Holt_Winters_model = HoltWinters(AirPassengers, seasonal="multiplicative")

Holt_Winters_forecast = predict(Holt_Winters_model, 60)

plot(Holt_Winters_model, Holt_Winters_forecast, main="Holt-Winters forecasting", xlab
="Year", ylab="Number of passengers (in thousands)"); legend("topleft", legend=c("Holt-
Winters forecast", "Data"), col=c("red", "black"), lty=1)
```



Taking the logarithm of the time series

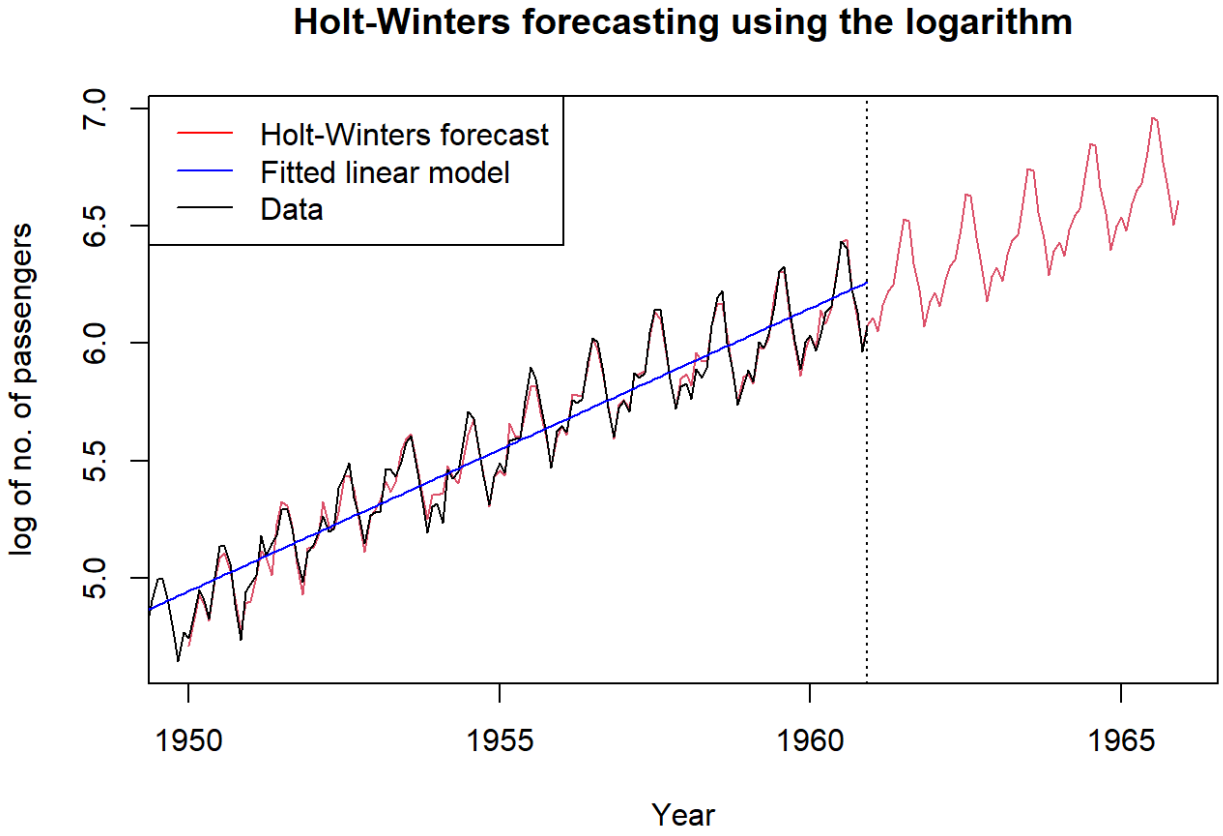
Taking the logarithm of the time series has the effect of linearising the trend and reducing the residual error as can be seen in the charts below. The residual error has also become more *homoskedastic* (more equally scattered). Constant variance of error across time is an indication of a good time series model.

```
data("AirPassengers")
plot(decompose(log(AirPassengers)))
```



The plot below shows that a linear model now makes a nice fit, which is good because the Holt-Winters model works better with linear time series.

```
Holt_Winters_log_model = HoltWinters(log(AirPassengers))

Holt_Winters_log_forecast = predict(Holt_Winters_log_model, 60)

linear_model = lm(log(AirPassengers) ~ time(AirPassengers))

plot(Holt_Winters_log_model, Holt_Winters_log_forecast, main="Holt-Winters forecasting
using the logarithm", xlab="Year", ylab="log of no. of passengers"); points(time(AirPas
sengers), fitted(linear_model), type="l", col="blue"); legend("topleft", legend=c("Holt
-Winters forecast", "Fitted linear model", "Data"), col=c("red", "blue", "black"), lty=
1)
```


Holt-Winters forecasting using the logarithm
Holt-Winters forecast
Fitted linear model
Data
log of no. of passengers
5.0
5.5
6.0
6.5
7.0
1950
1955
1960
1965
Year