Time Series Analysis in Health Research

Lecture 5: Advanced Forecasting Methods

Erfan Hoque

Dept. of Community Health & Epidemiology, University of Saskatchewan

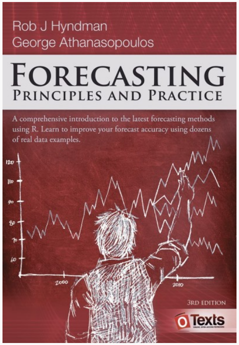
SEA 24th Annual Symposium Workshop, September 26, 2025

Schedule for this workshop

Week	Topic
Lecture 1	Introduction to forecasting, graphics
Lecture 2	Time Series toolbox and Forecasting Models
Lecture 3	Time Series Regression Models
Lecture 4	Hands-on session in R
Lecture 5	Advanced Forecasting Methods

Resources

Organization and presentation of material in these lectures will largely pull from


Rob J Hyndman
George Athanasopoulos
FORECASTING
PRINCIPLES AND PRACTICE
A comprehensive introduction to the latest forecasting methods using R. Learn to improve your forecast accuracy using dozens of real data examples.
3RD EDITION
OTexts

- Free and online (`https://otexts.com/fpp3/`)
- Data sets in associated R packages, R code for examples

Focus of this lecture

Focus of this lecture

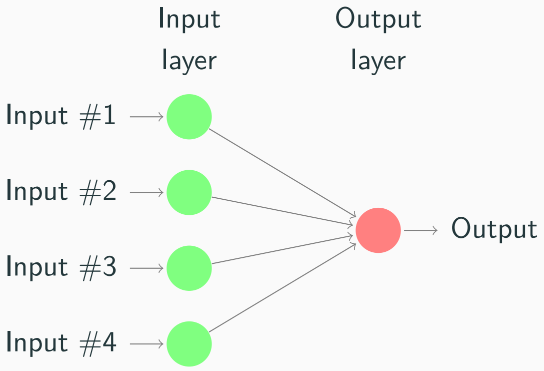
- Neural network models
  - Neural network autoregression
- Prophet model
- Forecasting

Neural network models

- Artificial neural networks are forecasting methods, inspired by the architecture of the human brain - a machine that with enough data could learn any smooth predictive relationship.
- The feed forward NN is one of the most common methods to approximate a multivariate nonlinear function.
- Their distinctive feature is the “hidden layers” in which input variables activate some nonlinear transformation functions producing a continuous signal to output nodes.
- These hidden layers represent a very efficient parallel distributed model of nonlinear regression.

Neural network models

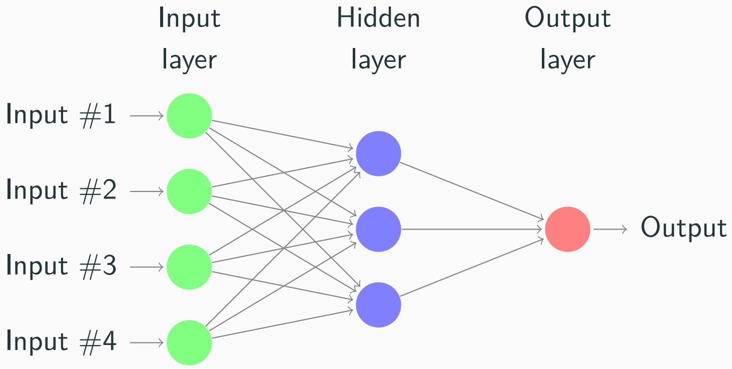
Simplest version: linear regression The simplest networks contain no hidden layers and are equivalent to linear regressions.



- A neural network can be thought of as a network of “neurons” which are organised in layers. The predictors (or inputs) form the bottom layer, and the forecasts (or outputs) form the top layer. There may also be intermediate layers containing “hidden neurons”.
- Coefficients attached to predictors are called “weights”.
- Forecasts are obtained by a linear combination of inputs.
- Weights selected using a “learning algorithm” that minimises a “cost function” (such as MSE).

Neural network models

Nonlinear model with one hidden layer



Neural network autoregression (NNAR) i

- With time series data, lagged values of the time series can be used as inputs to a neural network, just as we used lagged values in a linear autoregression model. We call this a neural network autoregression or NNAR model.
- We only consider feed-forward networks with one hidden layer, and we use the notation NNAR(p, k) to indicate there are p lagged inputs and k nodes in the hidden layer.
  - For example, a NNAR(9,5) model is a neural network with the last nine observations $y_{t-1}, \ldots, y_{t-9}$ used as inputs for forecasting the output y_t, and with five neurons in the hidden layer.

Neural network autoregression (NNAR) ii

- A NNAR$(p, 0)$ model is equivalent to an ARIMA$(p, 0, 0)$ model, but without the restrictions on the parameters to ensure stationarity.
- With seasonal data, it is useful to also add the last observed values from the same season as inputs.
  - For example, an NNAR $(3,1,2)_{12}$ model has inputs $y_{t-1}, \ldots, y_{t-3}$ and y_{t-12} and two neurons in the hidden layer.
- A NNAR $(p, P, 0)_s$ model is equivalent to an ARIMA $(p, 0, 0)(P, 0, 0)_s$ model but without the restrictions on the parameters that ensure stationarity.

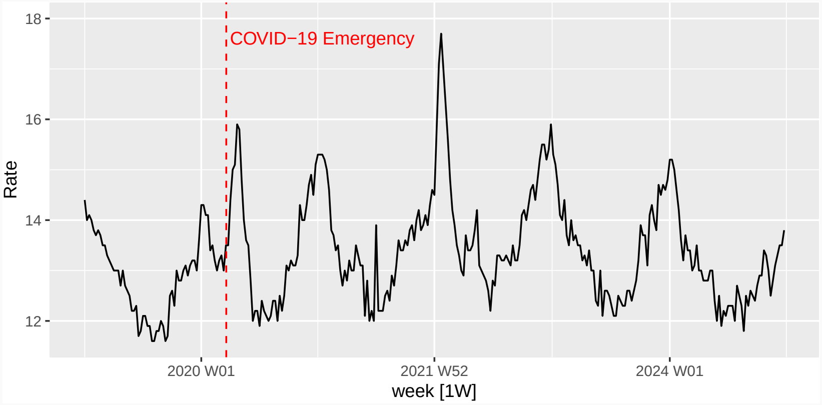
NNAR models in R

- The `NNETAR()` function fits an NNAR$(p, P, k)_m$ model.
- If p and P are not specified, they are automatically selected.
- For non-seasonal time series, default $p =$ optimal number of lags (according to the AIC) for a linear AR(p) model.
- For seasonal time series, defaults are $P = 1$ and p is chosen from the optimal linear model fitted to the seasonally adjusted data.
- Default $k = (p + P + 1)/2$ (rounded to the nearest integer).

Forecasting

- When it comes to forecasting, the network is applied iteratively.
- For forecasting one step ahead, we simply use the available historical inputs.
- For forecasting two steps ahead, we use the one-step forecast as an input, along with the historical data.
- This process proceeds until we have computed all the required forecasts.

Example: Weekly mortality data



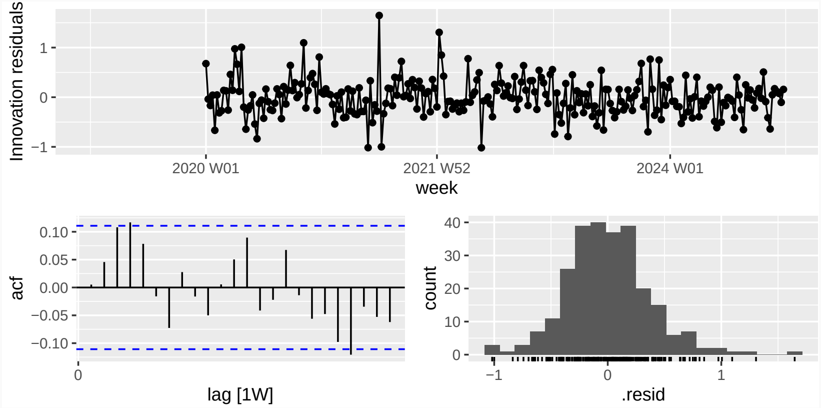
NNAR model weekly mortality data

```r
fit <- mortality_ts %>% model(NNETAR(Rate))
fit
```

```
## # A mable: 1 x 1
##       `NNETAR(Rate)`
##              <model>
## 1 <NNAR(2,1,2)[52]>
```

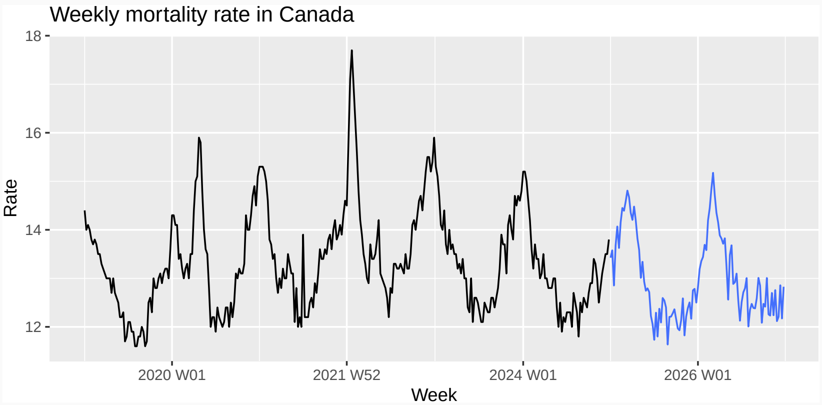
- Provides NNAR(2,1,2)[52] model for mortality rate. Here, the last 2 observations are used as predictors, and there are 2 neurons in the hidden layer.

```r
fit |> gg_tsresiduals()
```



Forecast of weekly mortality rate in Canada

```
fit %>% forecast(h="2 years", times=1) %>%
  autoplot(mortality_ts, level=NULL) +
  labs(x = "Week", y = "Rate", title = "Weekly mortality rate in Canada")
```



Prophet Models

Prophet Model

- Prophet is an open-source forecasting tool developed by Facebook for time series forecasting.
- It is designed to handle time series data with strong seasonal patterns and multiple seasonality.
- The primary purpose of Prophet is to provide a straightforward and effective way to forecast time series data, making it accessible to analysts and non-experts.
- Automatic Forecasting Procedure
- This model was introduced by Facebook (S. J. Taylor & Letham, 2018), originally for forecasting daily data with weekly and yearly seasonality, plus holiday effects.
- Prophet model is available via the `fable.prophet` package

Prophet Model

- Prophet can be considered a nonlinear regression model, of the form

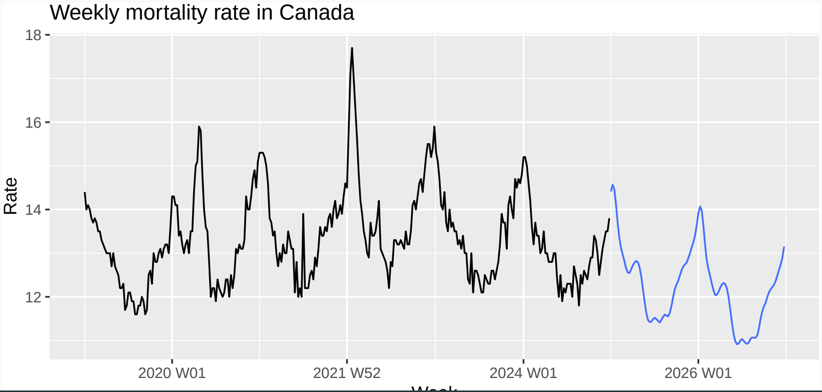
$$y_t = g(t) + s(t) + h(t) + \varepsilon_t$$

where

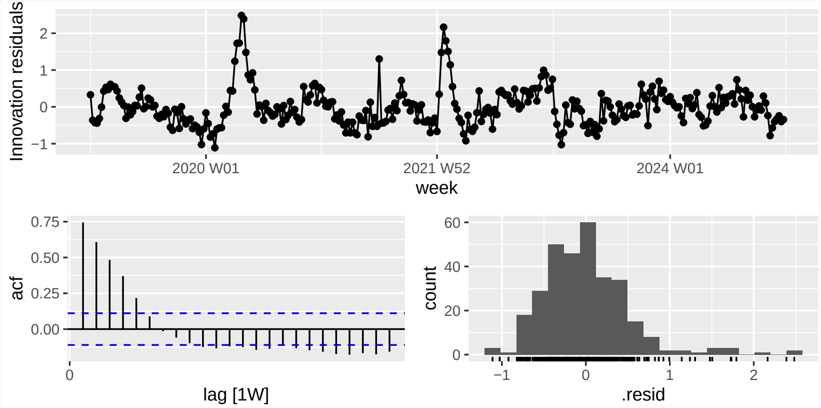
- where g(t) describes a piecewise-linear trend (or “growth term”),
- s(t) describes the various seasonal patterns,
- h(t) captures the holiday effects, and
- ε_t is a white noise error term.

Prophet model weekly mortality data

```r
fit.ph <- mortality_ts %>%
  model(prophet(Rate ~ season(period = 52, order = 10)))
# forecast
fit.ph %>% forecast(h = "2 years") %>%
  autoplot(mortality_ts, level=NULL) +
  labs(x = "Week", y = "Rate", title = "Weekly mortality rate in Canada")
```



```r
fit.ph |> gg_tsresiduals()
```



Few more advanced forecating models

- Long short term memory (LSTM) model
  - a type of recurrent neural network (RNN) architecture that is well-suited for time series forecasting.
- Forecast combinations
  - An easy way to improve forecast accuracy is to use several different methods on the same time series, and to average the resulting forecasts.
  - Look at Chapter 13 of Forecasting: Principles and Practice (3rd ed, `https://otexts.com/fpp3/`) book to know more.

Some practical forecasting issues

Models for different frequencies

Models for annual data

- ETS (exponential smoothing), ARIMA, Dynamic regression

Models for quarterly data

- ETS, ARIMA/SARIMA, Dynamic regression

Models for monthly data

- ETS, ARIMA/SARIMA, Dynamic regression

Models for different frequencies

- Weekly, daily and sub-daily data can be challenging for forecasting, although for different reasons.

Models for weekly data

- ARIMA/SARIMA, Dynamic regression

Models for daily, hourly and other sub-daily data

- ARIMA/SARIMA, Dynamic regression, STL+ETS, STL+ARIMA

Missing values

Functions which can handle missing values

- `ARIMA()`
- `TSLM()` (dynamic regression models)
- `NNETAR()`

Models which cannot handle missing values

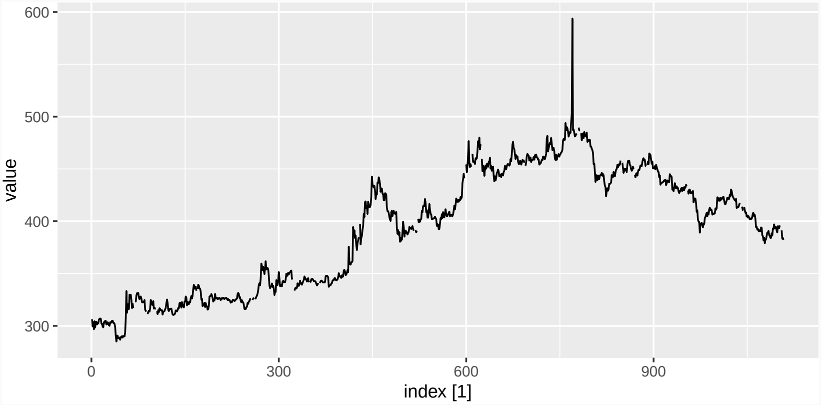
- `ETS()`
- `STL()`

What to do?

1. Just take section of data after last missing value (assuming there is a long enough series of observations to produce meaningful forecasts)
2. Estimate missing values with `interpolate()`.

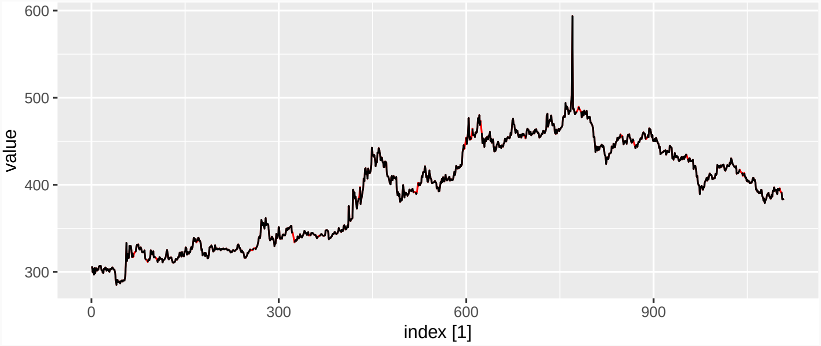
Missing values: Example - Daily morning gold prices

```r
gold <- as_tsibble(forecast::gold)
gold %>% autoplot(value)
```



Missing values

```r
gold_complete <- gold %>%
  model(ARIMA(value)) %>% interpolate(gold)
gold_complete %>%
  autoplot(value, colour = "red") + autolayer(gold, value)
```



Outliers

- All of the methods we have learned will not work well if there are extreme outliers in the data.
- In this case, we may wish to replace them with missing values, or with an estimate that is more consistent with the majority of the data.
- The `tsoutliers()` function is designed to identify outliers, and to suggest potential replacement values.

Look at Chapter 13 of Forecasting: Principles and Practice (3rd ed, `https://otexts.com/fpp3/`) book to know more.