MONASH University

Rob J Hyndman

Forecasting:

Principles and Practice

12. Advanced methods

OTexts.com/fpp/9/2/

OTexts.com/fpp/9/3/

Outline

1. **Vector autoregressions**
2. Neural network models
3. Time series with complex seasonality

Vector autoregressions

Dynamic regression assumes a unidirectional relationship: forecast variable influenced by predictor variables, but not vice versa.

Vector AR allow for feedback relationships. All variables treated symmetrically.

i.e., all variables are now treated as “endogenous”.

- Personal consumption may be affected by disposable income, and vice-versa.
- e.g., Govt stimulus package in Dec 2008 increased Christmas spending which increased incomes.

Vector autoregressions

Dynamic regression assumes a unidirectional relationship: forecast variable influenced by predictor variables, but not vice versa.

Vector AR allow for feedback relationships. All variables treated symmetrically.

i.e., all variables are now treated as “endogenous”.

- Personal consumption may be affected by disposable income, and vice-versa.
- e.g., Govt stimulus package in Dec 2008 increased Christmas spending which increased incomes.

Vector autoregressions

VAR(1)

$$y_{1,t} = c_1 + \phi_{11,1}y_{1,t-1} + \phi_{12,1}y_{2,t-1} + e_{1,t}$$
$$y_{2,t} = c_2 + \phi_{21,1}y_{1,t-1} + \phi_{22,1}y_{2,t-1} + e_{2,t}$$

Forecasts:

$$\hat{y}_{1,T+1|T} = \hat{c}_1 + \hat{\phi}_{11,1}y_{1,T} + \hat{\phi}_{12,1}y_{2,T}$$
$$\hat{y}_{2,T+1|T} = \hat{c}_2 + \hat{\phi}_{21,1}y_{1,T} + \hat{\phi}_{22,1}y_{2,T}.$$

$$\hat{y}_{1,T+2|T} = \hat{c}_1 + \hat{\phi}_{11,1}\hat{y}_{1,T+1} + \hat{\phi}_{12,1}\hat{y}_{2,T+1}$$
$$\hat{y}_{2,T+2|T} = \hat{c}_2 + \hat{\phi}_{21,1}\hat{y}_{1,T+1} + \hat{\phi}_{22,1}\hat{y}_{2,T+1}.$$

Vector autoregressions

VAR(1)

$$
\begin{aligned}
y_{1,t} &= c_1 + \phi_{11,1} y_{1,t-1} + \phi_{12,1} y_{2,t-1} + e_{1,t} \\
y_{2,t} &= c_2 + \phi_{21,1} y_{1,t-1} + \phi_{22,1} y_{2,t-1} + e_{2,t}
\end{aligned}
$$

Forecasts:

$$
\begin{aligned}
\hat{y}_{1,T+1|T} &= \hat{c}_1 + \hat{\phi}_{11,1} y_{1,T} + \hat{\phi}_{12,1} y_{2,T} \\
\hat{y}_{2,T+1|T} &= \hat{c}_2 + \hat{\phi}_{21,1} y_{1,T} + \hat{\phi}_{22,1} y_{2,T}.
\end{aligned}
$$

$$
\begin{aligned}
\hat{y}_{1,T+2|T} &= \hat{c}_1 + \hat{\phi}_{11,1} \hat{y}_{1,T+1} + \hat{\phi}_{12,1} \hat{y}_{2,T+1} \\
\hat{y}_{2,T+2|T} &= \hat{c}_2 + \hat{\phi}_{21,1} \hat{y}_{1,T+1} + \hat{\phi}_{22,1} \hat{y}_{2,T+1}.
\end{aligned}
$$

VARs are useful when

- forecasting a collection of related variables where no explicit interpretation is required;
- testing whether one variable is useful in forecasting another (the basis of Granger causality tests);
- impulse response analysis, where the response of one variable to a sudden but temporary change in another variable is analysed;
- forecast error variance decomposition, where the proportion of the forecast variance of one variable is attributed to the effect of other variables.

VARs are useful when

- forecasting a collection of related variables where no explicit interpretation is required;
- testing whether one variable is useful in forecasting another (the basis of Granger causality tests);
- impulse response analysis, where the response of one variable to a sudden but temporary change in another variable is analysed;
- forecast error variance decomposition, where the proportion of the forecast variance of one variable is attributed to the effect of other variables.

VARs are useful when

- forecasting a collection of related variables where no explicit interpretation is required;
- testing whether one variable is useful in forecasting another (the basis of Granger causality tests);
- impulse response analysis, where the response of one variable to a sudden but temporary change in another variable is analysed;
- forecast error variance decomposition, where the proportion of the forecast variance of one variable is attributed to the effect of other variables.

VARs are useful when

- forecasting a collection of related variables where no explicit interpretation is required;
- testing whether one variable is useful in forecasting another (the basis of Granger causality tests);
- impulse response analysis, where the response of one variable to a sudden but temporary change in another variable is analysed;
- forecast error variance decomposition, where the proportion of the forecast variance of one variable is attributed to the effect of other variables.

VAR example

```
> ar(usconsumption,order=3)
$ar
, , 1       consumption  income
consumption       0.222  0.0424
income            0.475 -0.2390

, , 2       consumption  income
consumption      0.2001 -0.0977
income           0.0288 -0.1097

, , 3       consumption  income
consumption       0.235 -0.0238
income            0.406 -0.0923

$var.pred
            consumption income
consumption       0.393  0.193
income            0.193  0.735
```

VAR example

```
> library(vars)
> VARselect(usconsumption, lag.max=8, type="const")$selection
AIC(n)  HQ(n)  SC(n) FPE(n)
     5      1      1      5
> var <- VAR(usconsumption, p=3, type="const")
> serial.test(var, lags.pt=10, type="PT.asymptotic")
Portmanteau Test (asymptotic)
data:  Residuals of VAR object var
Chi-squared = 33.3837, df = 28, p-value = 0.2219
```

VAR example

```
> summary(var)
VAR Estimation Results:
=========================
Endogenous variables: consumption, income
Deterministic variables: const
Sample size: 161

Estimation results for equation consumption:
============================================
               Estimate Std. Error t value Pr(>|t|)
consumption.l1  0.22280    0.08580   2.597 0.010326 *
income.l1       0.04037    0.06230   0.648 0.518003
consumption.l2  0.20142    0.09000   2.238 0.026650 *
income.l2      -0.09830    0.06411  -1.533 0.127267
consumption.l3  0.23512    0.08824   2.665 0.008530 **
income.l3      -0.02416    0.06139  -0.394 0.694427
const           0.31972    0.09119   3.506 0.000596 ***
```

VAR example

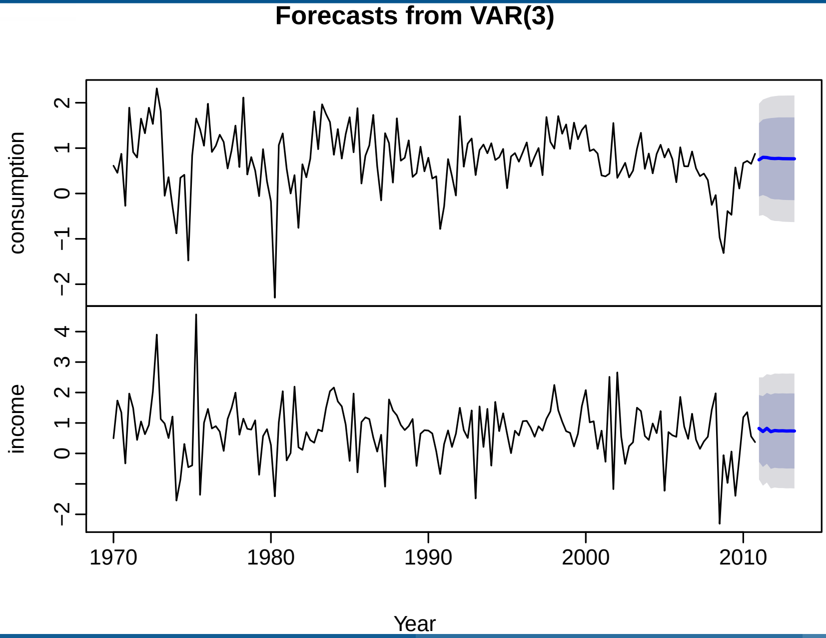
```
Estimation results for equation income:
=======================================
                Estimate Std. Error t value Pr(>|t|)
consumption.l1   0.48705    0.11637   4.186 4.77e-05 ***
income.l1       -0.24881    0.08450  -2.945 0.003736 **
consumption.l2   0.03222    0.12206   0.264 0.792135
income.l2       -0.11112    0.08695  -1.278 0.203170
consumption.l3   0.40297    0.11967   3.367 0.000959 ***
income.l3       -0.09150    0.08326  -1.099 0.273484
const            0.36280    0.12368   2.933 0.003865 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Correlation matrix of residuals:
            consumption income
consumption      1.0000 0.3639
income           0.3639 1.0000
```

VAR example

```
fcst <- forecast(var)
plot(fcst, xlab="Year")
```

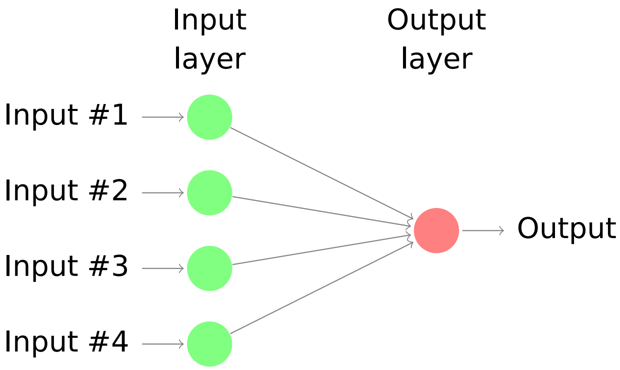
VAR example


Forecasts from VAR(3)
consumption
income
Year
−2
−1
0
1
2
3
4
1970
1980
1990
2000
2010

Outline

Neural network models

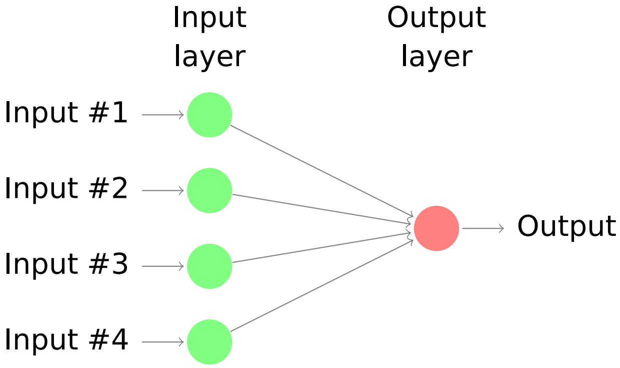
Simplest version: linear regression



- 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”.

Neural network models

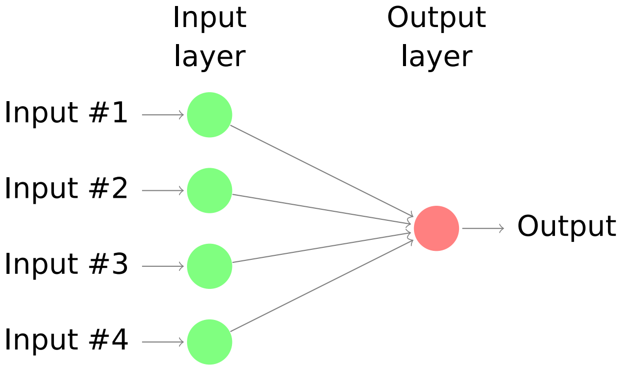
Simplest version: linear regression



- 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”.

Neural network models

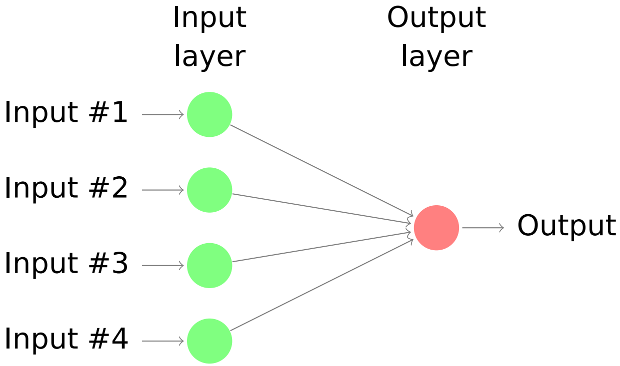
Simplest version: linear regression



- 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”.

Neural network models

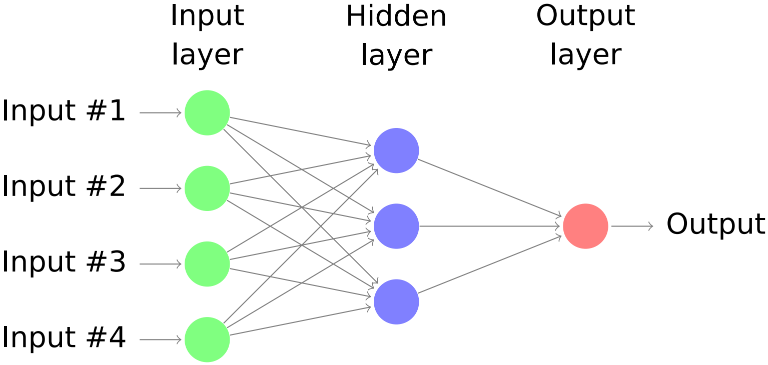
Simplest version: linear regression



- 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”.

Neural network models

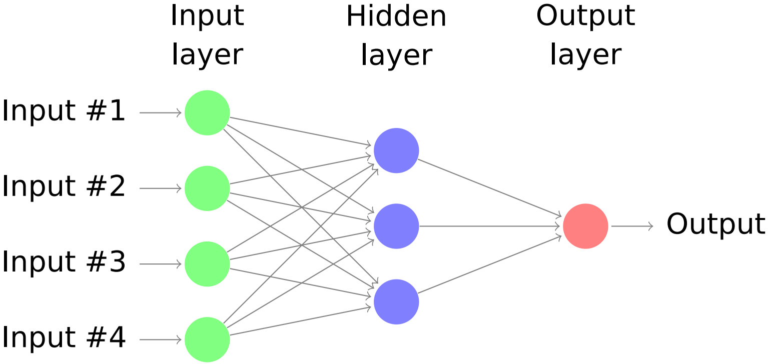
Nonlinear model with one hidden layer



- A **multilayer feed-forward network** where each layer of nodes receives inputs from the previous layers.
- Inputs to each node combined using linear combination.
- Result modified by nonlinear function before being output.

Neural network models

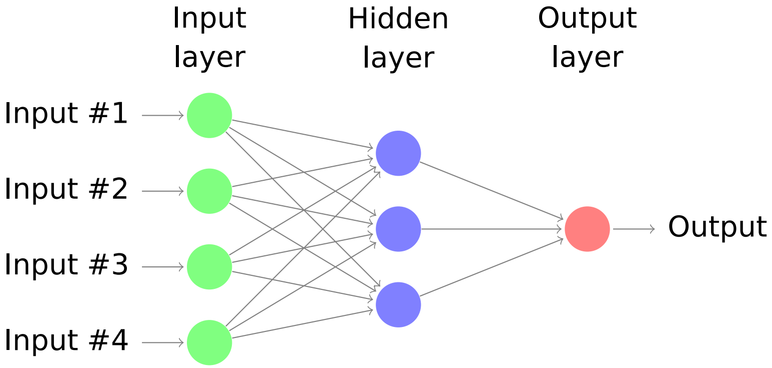
Nonlinear model with one hidden layer



- A **multilayer feed-forward network** where each layer of nodes receives inputs from the previous layers.
- Inputs to each node combined using linear combination.
- Result modified by nonlinear function before being output.

Neural network models

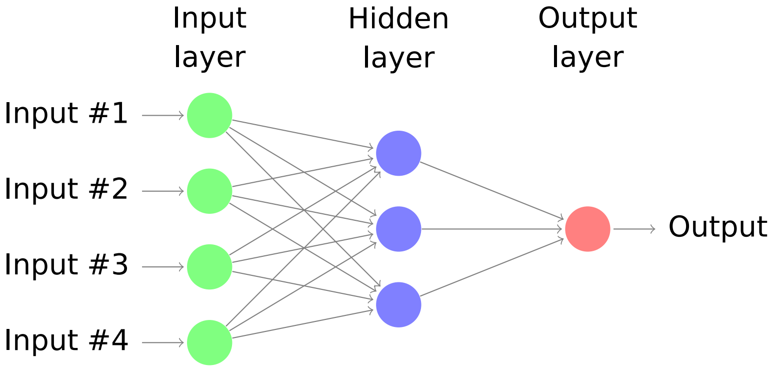
Nonlinear model with one hidden layer



- A **multilayer feed-forward network** where each layer of nodes receives inputs from the previous layers.
- Inputs to each node combined using linear combination.
- Result modified by nonlinear function before being output.

Neural network models

Nonlinear model with one hidden layer



- A **multilayer feed-forward network** where each layer of nodes receives inputs from the previous layers.
- Inputs to each node combined using linear combination.
- Result modified by nonlinear function before being output.

Neural network models

Inputs to hidden neuron *j* linearly combined:

$$z_j = b_j + \sum_{i=1}^{4} w_{i,j} x_i.$$

Modified using nonlinear function such as a sigmoid:

$$s(z) = \frac{1}{1 + e^{-z}},$$

This tends to reduce the effect of extreme input values, thus making the network somewhat robust to outliers.

Neural network models

- Weights take random values to begin with, which are then updated using the observed data.
- There is an element of randomness in the predictions. So the network is usually trained several times using different random starting points, and the results are averaged.
- Number of hidden layers, and the number of nodes in each hidden layer, must be specified in advance.

Neural network models

- Weights take random values to begin with, which are then updated using the observed data.
- There is an element of randomness in the predictions. So the network is usually trained several times using different random starting points, and the results are averaged.
- Number of hidden layers, and the number of nodes in each hidden layer, must be specified in advance.

Neural network models

- Weights take random values to begin with, which are then updated using the observed data.
- There is an element of randomness in the predictions. So the network is usually trained several times using different random starting points, and the results are averaged.
- Number of hidden layers, and the number of nodes in each hidden layer, must be specified in advance.

NNAR models

- Lagged values of the time series can be used as inputs to a neural network.
- NNAR(p, k): p lagged inputs and k nodes in the single hidden layer.
- NNAR($p, 0$) model is equivalent to an ARIMA($p, 0, 0$) model but without stationarity restrictions.
- Seasonal NNAR(p, P, k): inputs $(y_{t-1}, y_{t-2}, \ldots, y_{t-p}, y_{t-m}, y_{t-2m}, y_{t-Pm})$ and k neurons in the hidden layer.
- NNAR$(p, P, 0)_m$ model is equivalent to an ARIMA$(p, 0, 0)(P,0,0)_m$ model but without stationarity restrictions.

NNAR models

- Lagged values of the time series can be used as inputs to a neural network.
- NNAR(p, k): p lagged inputs and k nodes in the single hidden layer.
- NNAR($p, 0$) model is equivalent to an ARIMA($p, 0, 0$) model but without stationarity restrictions.
- Seasonal NNAR(p, P, k): inputs $(y_{t-1}, y_{t-2}, \dots, y_{t-p}, y_{t-m}, y_{t-2m}, y_{t-Pm})$ and k neurons in the hidden layer.
- NNAR$(p, P, 0)_m$ model is equivalent to an ARIMA$(p, 0, 0)(P,0,0)_m$ model but without stationarity restrictions.

NNAR models

- Lagged values of the time series can be used as inputs to a neural network.
- NNAR(p, k): p lagged inputs and k nodes in the single hidden layer.
- NNAR($p, 0$) model is equivalent to an ARIMA($p, 0, 0$) model but without stationarity restrictions.
- Seasonal NNAR(p, P, k): inputs $(y_{t-1}, y_{t-2}, \dots, y_{t-p}, y_{t-m}, y_{t-2m}, y_{t-Pm})$ and k neurons in the hidden layer.
- NNAR$(p, P, 0)_m$ model is equivalent to an ARIMA$(p, 0, 0)(P,0,0)_m$ model but without stationarity restrictions.

NNAR models

- Lagged values of the time series can be used as inputs to a neural network.
- NNAR(p, k): p lagged inputs and k nodes in the single hidden layer.
- NNAR($p, 0$) model is equivalent to an ARIMA($p, 0, 0$) model but without stationarity restrictions.
- Seasonal NNAR(p, P, k): inputs $(y_{t-1}, y_{t-2}, \ldots, y_{t-p}, y_{t-m}, y_{t-2m}, y_{t-Pm})$ and k neurons in the hidden layer.
- NNAR$(p, P, 0)_m$ model is equivalent to an ARIMA$(p, 0, 0)(P,0,0)_m$ model but without stationarity restrictions.

NNAR models

- Lagged values of the time series can be used as inputs to a neural network.
- NNAR(p, k): p lagged inputs and k nodes in the single hidden layer.
- NNAR($p, 0$) model is equivalent to an ARIMA($p, 0, 0$) model but without stationarity restrictions.
- Seasonal NNAR(p, P, k): inputs $(y_{t-1}, y_{t-2}, \dots, y_{t-p}, y_{t-m}, y_{t-2m}, y_{t-Pm})$ and k neurons in the hidden layer.
- NNAR$(p, P, 0)_m$ model is equivalent to an ARIMA$(p, 0, 0)(P,0,0)_m$ model but without stationarity restrictions.

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).

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).

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).

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).

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).

Sunspots

- Surface of the sun contains magnetic regions that appear as dark spots.
- These affect the propagation of radio waves and so telecommunication companies like to predict sunspot activity in order to plan for any future difficulties.
- Sunspots follow a cycle of length between 9 and 14 years.

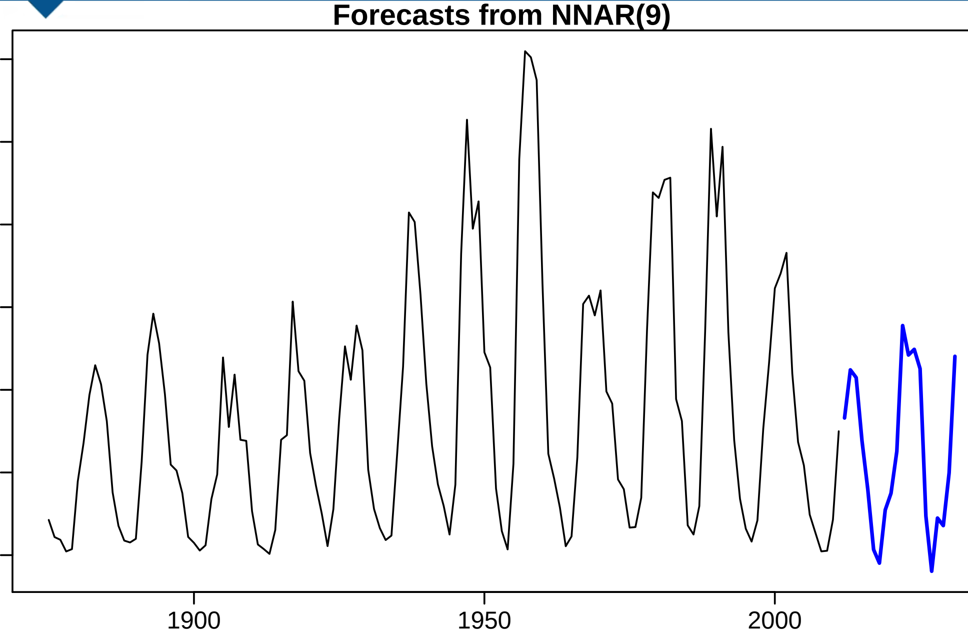
Sunspots

- Surface of the sun contains magnetic regions that appear as dark spots.
- These affect the propagation of radio waves and so telecommunication companies like to predict sunspot activity in order to plan for any future difficulties.
- Sunspots follow a cycle of length between 9 and 14 years.

Sunspots

- Surface of the sun contains magnetic regions that appear as dark spots.
- These affect the propagation of radio waves and so telecommunication companies like to predict sunspot activity in order to plan for any future difficulties.
- Sunspots follow a cycle of length between 9 and 14 years.

NNAR(9,5) model for sunspots



NNAR model for sunspots

```
fit <- nnetar(sunspotarea)
plot(forecast(fit,h=20))
```

To restrict to positive values:

```
fit <- nnetar(sunspotarea,lambda=0)
plot(forecast(fit,h=20))
```

NNAR model for sunspots

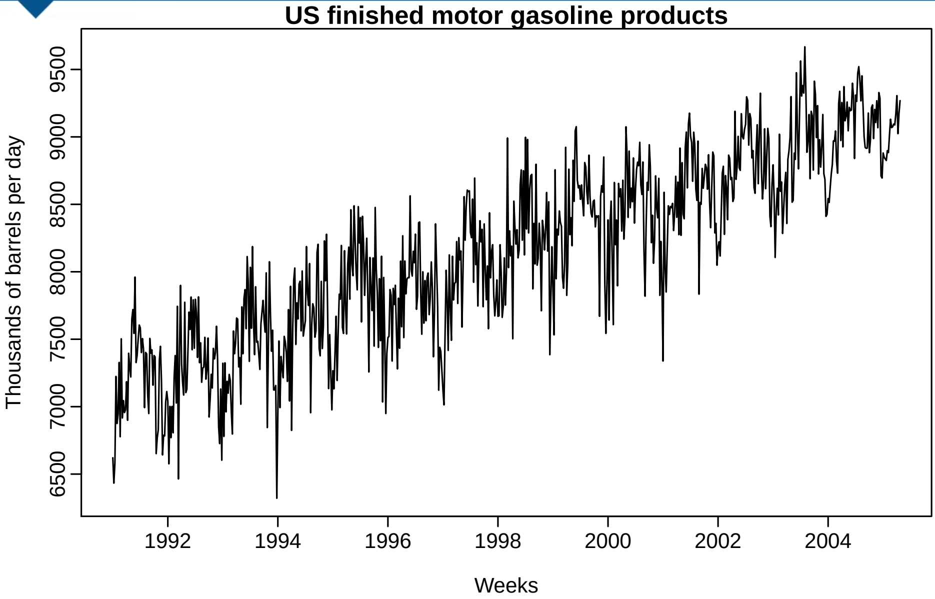
```
fit <- nnetar(sunspotarea)
plot(forecast(fit,h=20))
```

To restrict to positive values:

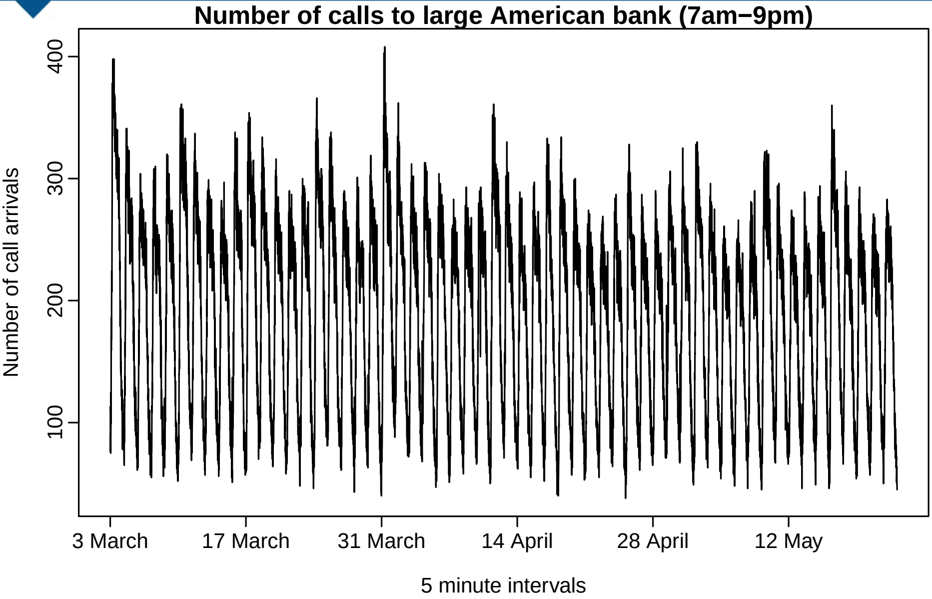
```
fit <- nnetar(sunspotarea,lambda=0)
plot(forecast(fit,h=20))
```

Outline

Examples

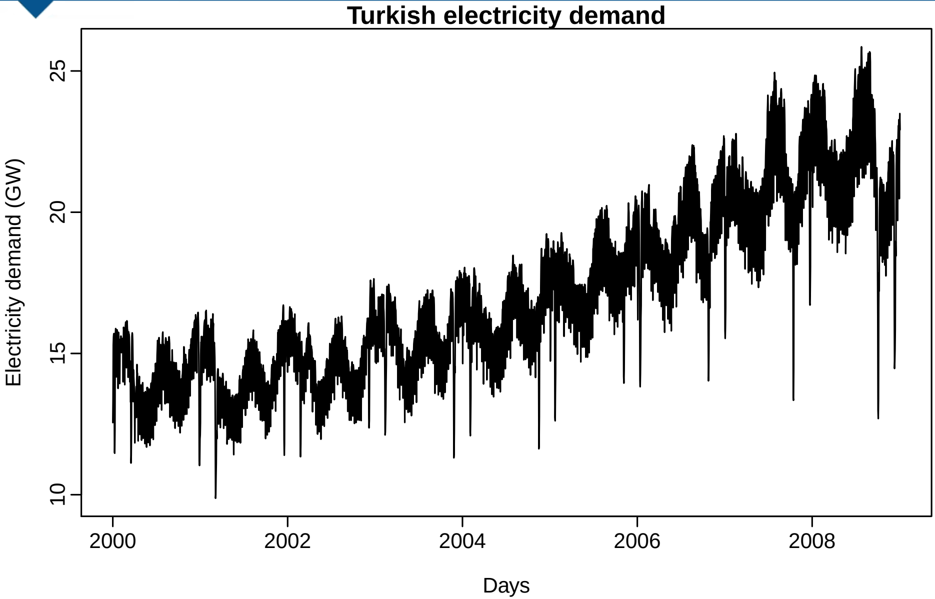

US finished motor gasoline products
Thousands of barrels per day
6500
7000
7500
8000
8500
9000
9500
1992
1994
1996
1998
2000
2002
2004
Weeks

Examples



Number of calls to large American bank (7am−9pm)
Number of call arrivals
100
200
300
400
3 March
17 March
31 March
14 April
28 April
12 May
5 minute intervals

Examples


Turkish electricity demand
Electricity demand (GW)
10
15
20
25
2000
2002
2004
2006
2008
Days

TBATS model

TBATS

Trigonometric terms for seasonality
Box-Cox transformations for heterogeneity
ARMA errors for short-term dynamics
Trend (possibly damped)
Seasonal (including multiple and non-integer periods)

TBATS model

$$y_t = \text{observation at time } t$$

$$y_t^{(\omega)} = \begin{cases} (y_t^\omega - 1)/\omega & \text{if } \omega \neq 0; \\ \log y_t & \text{if } \omega = 0. \end{cases}$$

$$y_t^{(\omega)} = \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^{M} s_{t-m_i}^{(i)} + d_t$$

$$\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha d_t$$

$$b_t = (1-\phi)b + \phi b_{t-1} + \beta d_t$$

$$d_t = \sum_{i=1}^{p} \phi_i d_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}$$

$$\begin{aligned} s_{j,t}^{(i)} &= s_{j,t-1}^{(i)} \cos \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \sin \lambda_j^{(i)} + \gamma_1^{(i)} d_t \\ s_{j,t}^{(i)} &= -s_{j,t-1}^{(i)} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t \end{aligned}$$

TBATS model

$$y_t = \text{observation at time } t$$

$$y_t^{(\omega)} = \begin{cases} (y_t^{\omega} - 1)/\omega & \text{if } \omega \neq 0; \\ \log y_t & \text{if } \omega = 0. \end{cases}$$

Box-Cox transformation

$$y_t^{(\omega)} = \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^{M} s_{t-m_i}^{(i)} + d_t$$

$$\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha d_t$$

$$b_t = (1 - \phi)b + \phi b_{t-1} + \beta d_t$$

$$d_t = \sum_{i=1}^{p} \phi_i d_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}$$

$$\begin{aligned} s_{j,t}^{(i)} &= s_{j,t-1}^{(i)} \cos \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \sin \lambda_j^{(i)} + \gamma_1^{(i)} d_t \\ s_{j,t}^{(i)} &= -s_{j,t-1}^{(i)} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t \end{aligned}$$

TBATS model

$$y_t = \text{observation at time } t$$

$$y_t^{(\omega)} = \begin{cases} (y_t^\omega - 1)/\omega & \text{if } \omega \neq 0; \\ \log y_t & \text{if } \omega = 0. \end{cases}$$

Box-Cox transformation

$$y_t^{(\omega)} = \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^{M} s_{t-m_i}^{(i)} + d_t$$

M seasonal periods

$$\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha d_t$$

$$b_t = (1-\phi)b + \phi b_{t-1} + \beta d_t$$

$$d_t = \sum_{i=1}^{p} \phi_i d_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}$$

$$\begin{aligned} s_{j,t}^{(i)} &= s_{j,t-1}^{(i)} \cos \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \sin \lambda_j^{(i)} + \gamma_1^{(i)} d_t \\ s_{j,t}^{(i)} &= -s_{j,t-1}^{(i)} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t \end{aligned}$$

TBATS model

$$y_t = \text{observation at time } t$$

$$y_t^{(\omega)} = \begin{cases} (y_t^\omega - 1)/\omega & \text{if } \omega \neq 0; \\ \log y_t & \text{if } \omega = 0. \end{cases}$$

Box-Cox transformation

$$y_t^{(\omega)} = \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^{M} s_{t-m_i}^{(i)} + d_t$$

M seasonal periods

$$\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha d_t$$

$$b_t = (1-\phi)b + \phi b_{t-1} + \beta d_t$$

global and local trend

$$d_t = \sum_{i=1}^{p} \phi_i d_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}$$

$$\begin{aligned} s_{j,t}^{(i)} &= s_{j,t-1}^{(i)} \cos \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \sin \lambda_j^{(i)} + \gamma_1^{(i)} d_t \\ s_{j,t}^{(i)} &= -s_{j,t-1}^{(i)} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t \end{aligned}$$

TBATS model

$$y_t = \text{observation at time } t$$

$$y_t^{(\omega)} = \begin{cases} (y_t^\omega - 1)/\omega & \text{if } \omega \neq 0; \\ \log y_t & \text{if } \omega = 0. \end{cases}$$

Box-Cox transformation

$$y_t^{(\omega)} = \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^{M} s_{t-m_i}^{(i)} + d_t$$

M seasonal periods

$$\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha d_t$$

$$b_t = (1-\phi)b + \phi b_{t-1} + \beta d_t$$

global and local trend

$$d_t = \sum_{i=1}^{p} \phi_i d_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

ARMA error

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}$$

$$\begin{aligned} s_{j,t}^{(i)} &= s_{j,t-1}^{(i)} \cos \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \sin \lambda_j^{(i)} + \gamma_1^{(i)} d_t \\ s_{j,t}^{(i)} &= -s_{j,t-1}^{(i)} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t \end{aligned}$$

TBATS model

$$y_t = \text{observation at time } t$$

$$y_t^{(\omega)} = \begin{cases} (y_t^{\omega} - 1)/\omega & \text{if } \omega \neq 0; \\ \log y_t & \text{if } \omega = 0. \end{cases}$$

Box-Cox transformation

$$y_t^{(\omega)} = \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^{M} s_{t-m_i}^{(i)} + d_t$$

M seasonal periods

$$\ell_t = \ell_{t-1} + \phi b_{t-1} + \alpha d_t$$

$$b_t = (1 - \phi) b + \phi b_{t-1} + \beta d_t$$

global and local trend

$$d_t = \sum_{i=1}^{p} \phi_i d_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

ARMA error

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}$$

$$s_{j,t}^{(i)} = \quad s_{j,t-1}^{(i)} \, \mathrm{c}$$

$$s_{j,t}^{(i)} = -s_{j,t-1}^{(i)} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t$$

Fourier-like seasonal terms

TBATS model

$y_t =$ observation at time t

$$y_t^{(\omega)} = \begin{cases} (y_t^\omega - 1)/\omega & \text{if } \omega \neq 0; \\ \text{lo} \end{cases}$$

$$y_t^{(\omega)} = \ell_{t-1}$$

$$\ell_t = \ell_{t-1}$$

$$b_t = (1 -$$

$$d_t = \sum_{i=1}^{p}$$

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)}$$

$$s_{j,t}^{(i)} = \quad s_{j,t-1}^{(i)} \text{ c}$$

$$s_{j,t}^{(i)} = -s_{j,t-1}^{(i)} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t$$

TBATS

Trigonometric

Box-Cox

ARMA

Trend

Seasonal

Box-Cox transformation

M seasonal periods

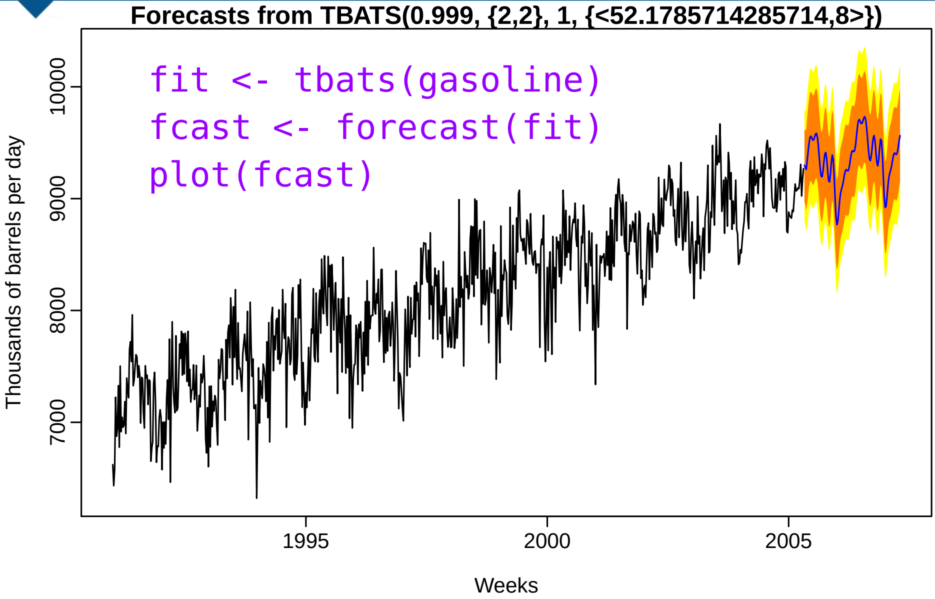
global and local trend

ARMA error

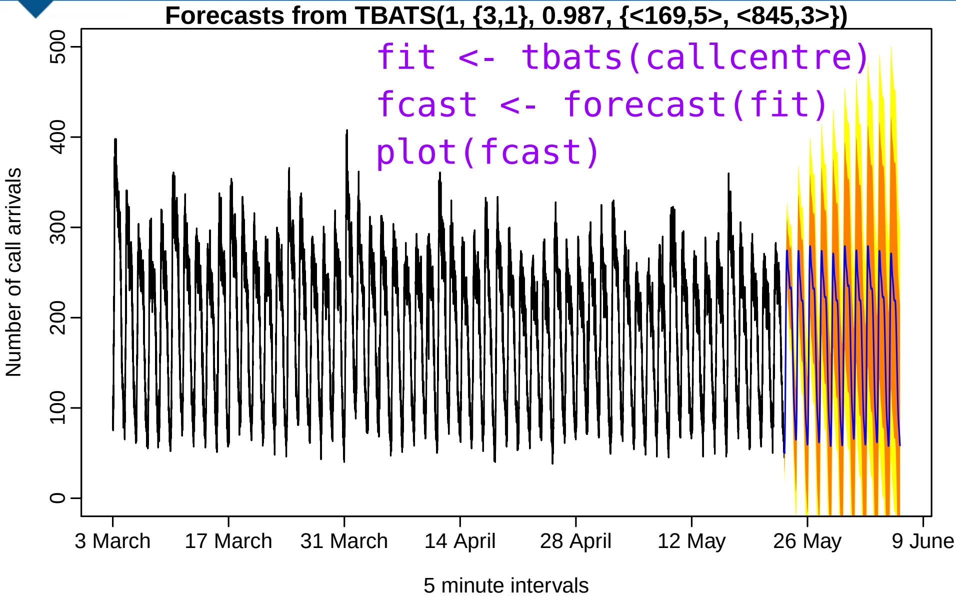
Fourier-like seasonal terms

Examples

```r
fit <- tbats(gasoline)
fcast <- forecast(fit)
plot(fcast)
```



Examples


Forecasts from TBATS(1, {3,1}, 0.987, {<169,5>, <845,3>})
fit <- tbats(callcentre)
fcast <- forecast(fit)
plot(fcast)
Number of call arrivals
0
100
200
300
400
500
3 March
17 March
31 March
14 April
28 April
12 May
26 May
9 June
5 minute intervals

Examples

```r
fit <- tbats(turk)
fcast <- forecast(fit)
plot(fcast)
```