FC2018: Advanced

Samuel Orso

2018-10-23

Slides generously provided by Rob J Hyndman
Based on Chapter 9, 11 and 12 of *Forecasting: Principles and Practice* by Rob J Hyndman and George Athanasopoulos

Outline

Regression with ARIMA errors

Regression models

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t,$$

- y_t modeled as function of k explanatory variables $x_{1,t}, \ldots, x_{k,t}$.
- In regression, we assume that ε_t was WN.
- Now we want to allow ε_t to be autocorrelated.

Regression with ARIMA errors

Regression models

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t,$$

- y_t modeled as function of k explanatory variables $x_{1,t}, \ldots, x_{k,t}$.
- In regression, we assume that ε_t was WN.
- Now we want to allow ε_t to be autocorrelated.

Example: ARIMA(1,1,1) errors

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \eta_t,$$
$$(1 - \phi_1 B)(1 - B)\eta_t = (1 + \theta_1 B)\varepsilon_t,$$

where ε_t is white noise.

Stationarity

Regression with ARMA errors

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \eta_t,$$

where η_t is an ARMA process.

- All variables in the model must be stationary.
- If we estimate the model while any of these are non-stationary, the estimated coefficients can be incorrect.
- Difference variables until all stationary.
- If necessary, apply same differencing to all variables.

Stationarity

Model with ARIMA(1,1,1) errors

$$
\begin{aligned}
y_t &= \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \eta_t, \\
(1 - \phi_1 B)(1 - B)\eta_t &= (1 + \theta_1 B)\varepsilon_t,
\end{aligned}
$$

Stationarity

Model with ARIMA(1,1,1) errors

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \eta_t,$$
$$(1 - \phi_1 B)(1 - B)\eta_t = (1 + \theta_1 B)\varepsilon_t,$$

Equivalent to model with ARIMA(1,0,1) errors

$$y'_t = \beta_1 x'_{1,t} + \cdots + \beta_k x'_{k,t} + \eta'_t,$$
$$(1 - \phi_1 B)\eta'_t = (1 + \theta_1 B)\varepsilon_t,$$

where $y'_t = y_t - y_{t-1}$, $x'_{t,i} = x_{t,i} - x_{t-1,i}$ and $\eta'_t = \eta_t - \eta_{t-1}$.

Regression with ARIMA errors

Any regression with an ARIMA error can be rewritten as a regression with an ARMA error by differencing all variables with the same differencing operator as in the ARIMA model.

Regression with ARIMA errors

Any regression with an ARIMA error can be rewritten as a regression with an ARMA error by differencing all variables with the same differencing operator as in the ARIMA model.

Original data

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \eta_t$$

$$\text{where} \quad \phi(B)(1-B)^d \eta_t = \theta(B)\varepsilon_t$$

Regression with ARIMA errors

Any regression with an ARIMA error can be rewritten as a regression with an ARMA error by differencing all variables with the same differencing operator as in the ARIMA model.

Original data

$$y_t = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \eta_t$$
$$\text{where} \quad \phi(B)(1-B)^d \eta_t = \theta(B)\varepsilon_t$$

After differencing all variables

$$y'_t = \beta_1 x'_{1,t} + \cdots + \beta_k x'_{k,t} + \eta'_t.$$
$$\text{where} \quad \phi(B)\eta_t = \theta(B)\varepsilon_t$$
$$\text{and} \quad y'_t = (1-B)^d y_t$$

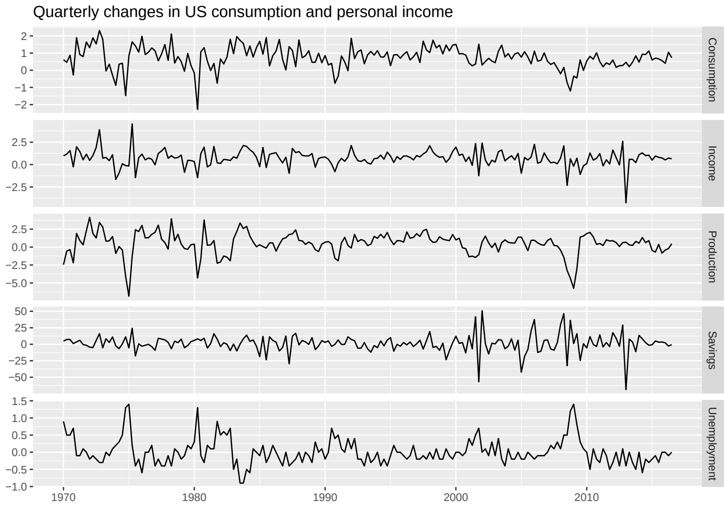
Model selection

- Check that all variables are stationary. If not, apply differencing. Where appropriate, use the same differencing for all variables to preserve interpretability.
- Fit regression model with automatically selected ARIMA errors.
- Check that ε_t series looks like white noise.

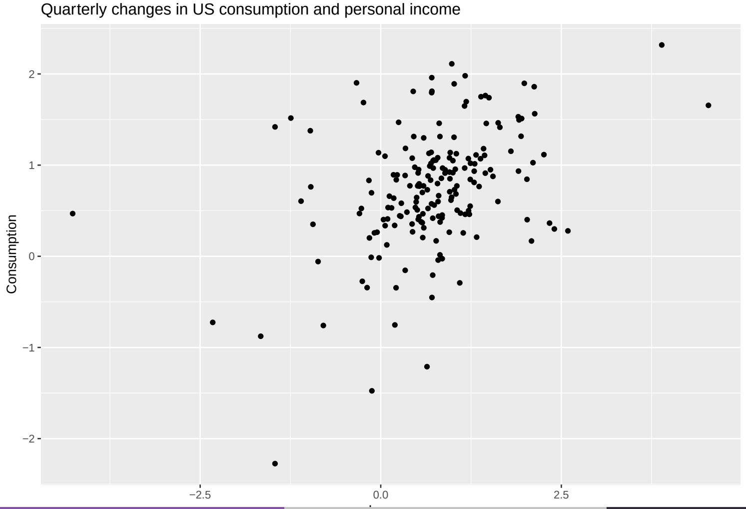
Selecting predictors

- AICc can be calculated for final model.
- Repeat procedure for all subsets of predictors to be considered, and select model with lowest AIC value.

US personal consumption and income



US personal consumption and income



US personal consumption and income

- No need for transformations or further differencing.
- Increase in income does not necessarily translate into instant increase in consumption (e.g., after the loss of a job, it may take a few months for expenses to be reduced to allow for the new circumstances). We will ignore this for now.

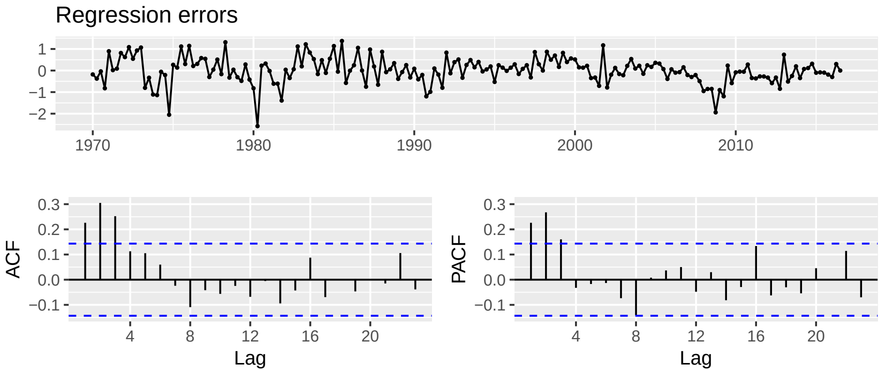
US personal consumption and income

```r
(fit <- auto.arima(uschange[,1], xreg=uschange[,2]))
```

```
## Series: uschange[, 1]
## Regression with ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1     ma1    ma2  intercept   xreg
##        0.692  -0.576  0.198      0.599  0.203
## s.e.   0.116   0.130  0.076      0.088  0.046
##
## sigma^2 estimated as 0.322:  log likelihood=-157
## AIC=326   AICc=326   BIC=345
```

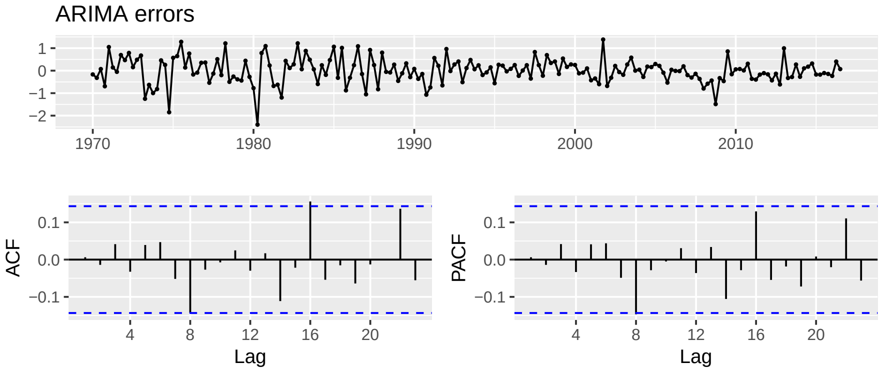
US personal consumption and income

```
ggtsdisplay(residuals(fit, type='regression'),
  main="Regression errors")
```



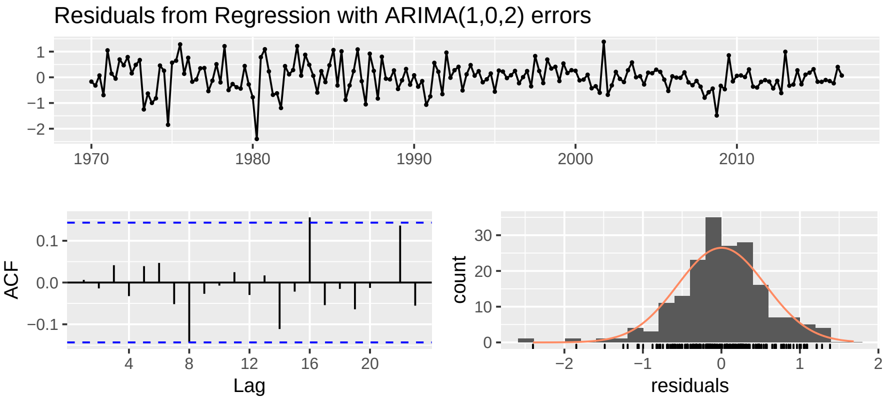
US personal consumption and income

```r
ggtsdisplay(residuals(fit, type='response'),
  main="ARIMA errors")
```



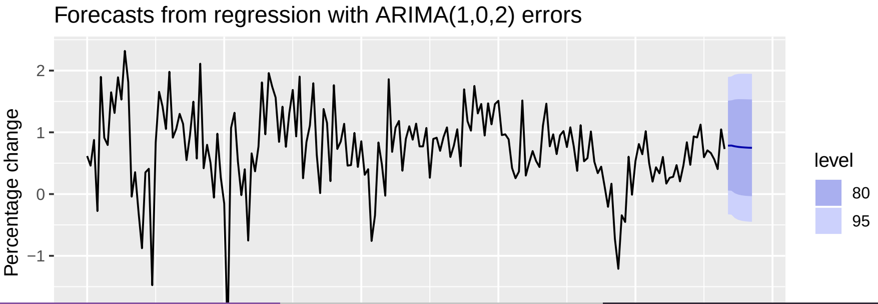
US personal consumption and income

```r
checkresiduals(fit, test=FALSE)
```



US personal consumption and income

```
fcast <- forecast(fit,
  xreg=rep(mean(uschange[,2]),8), h=8)
autoplot(fcast) + xlab("Year") +
  ylab("Percentage change") +
  ggtitle("Forecasts from regression with ARIMA(1,0,2) errors")
```



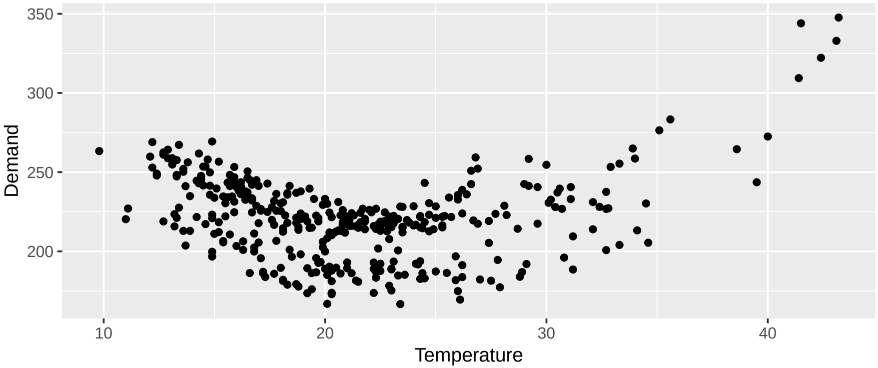
Forecasting

- To forecast a regression model with ARIMA errors, we need to forecast the regression part of the model and the ARIMA part of the model and combine the results.
- Some predictors are known into the future (e.g., time, dummies).
- Separate forecasting models may be needed for other predictors.
- Forecast intervals ignore the uncertainty in forecasting the predictors.

Daily electricity demand

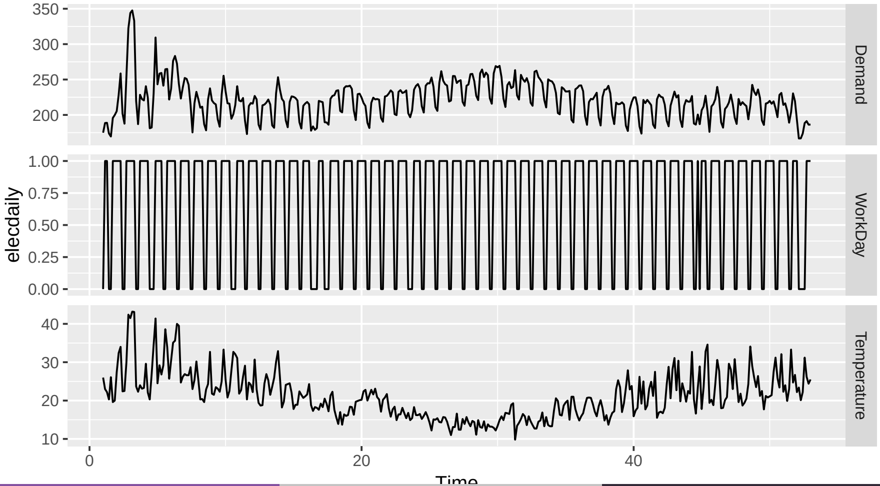
Model daily electricity demand as a function of temperature using quadratic regression with ARMA errors.

```r
qplot(elecdaily[,"Temperature"], elecdaily[,"Demand"]) +
  xlab("Temperature") + ylab("Demand")
```



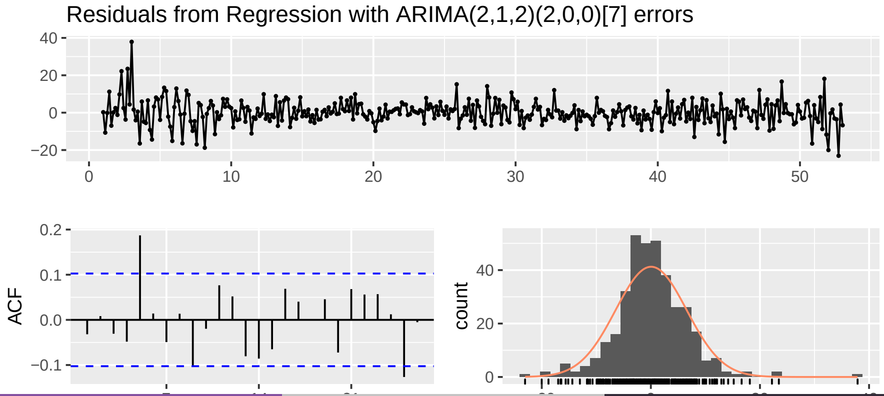
Daily electricity demand

```
autoplot(elecdaily, facets = TRUE)
```



Daily electricity demand

```
xreg <- cbind(MaxTemp = elecdaily[, "Temperature"],
              MaxTempSq = elecdaily[, "Temperature"]^2,
              Workday = elecdaily[, "WorkDay"])
fit <- auto.arima(elecdaily[, "Demand"], xreg = xreg)
checkresiduals(fit)
```



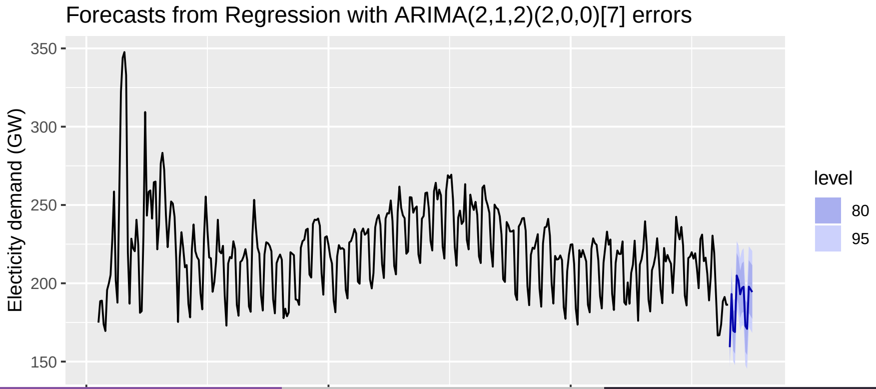
Daily electricity demand

```r
# Forecast one day ahead
forecast(fit, xreg = cbind(26, 26^2, 1))
```

```
##       Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 53.14            190   181   198   177   203
```

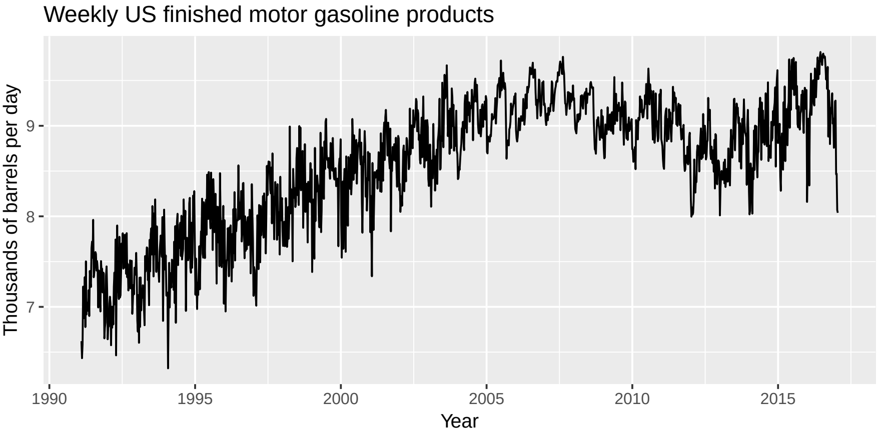
Daily electricity demand

```
fcast <- forecast(fit,
  xreg = cbind(rep(26,14), rep(26^2,14),
    c(0,1,0,0,1,1,1,1,1,0,0,1,1,1)))
autoplot(fcast) + ylab("Electicity demand (GW)")
```

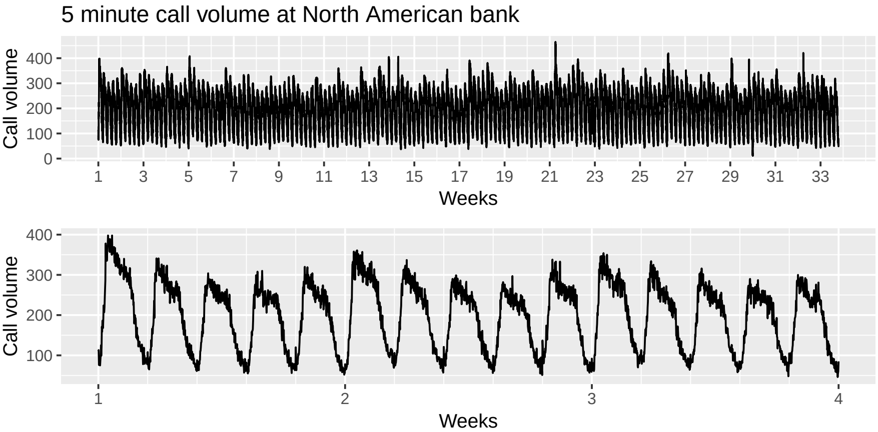


Outline

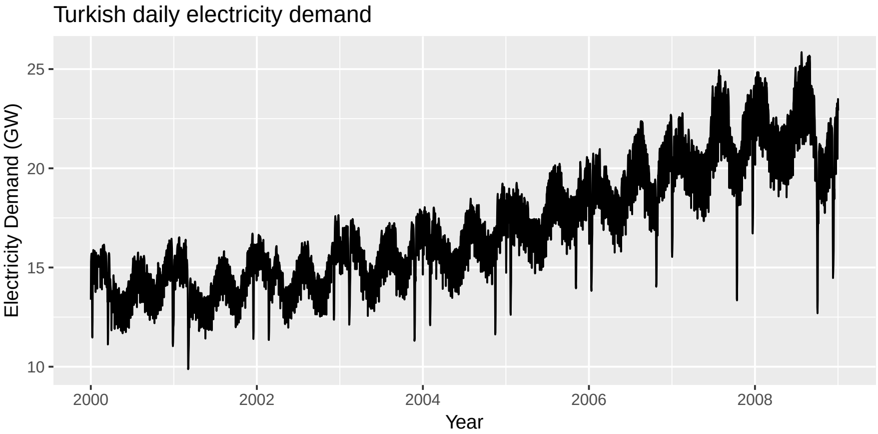
Examples


Weekly US finished motor gasoline products
Thousands of barrels per day
9
8
7
1990
1995
2000
2005
2010
2015
Year

Examples


5 minute call volume at North American bank
Call volume
400
300
200
100
0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
Weeks
Call volume
400
300
200
100
1
2
3
4
Weeks

Examples


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

Dynamic harmonic regression

Combine Fourier terms with ARIMA errors

Advantages

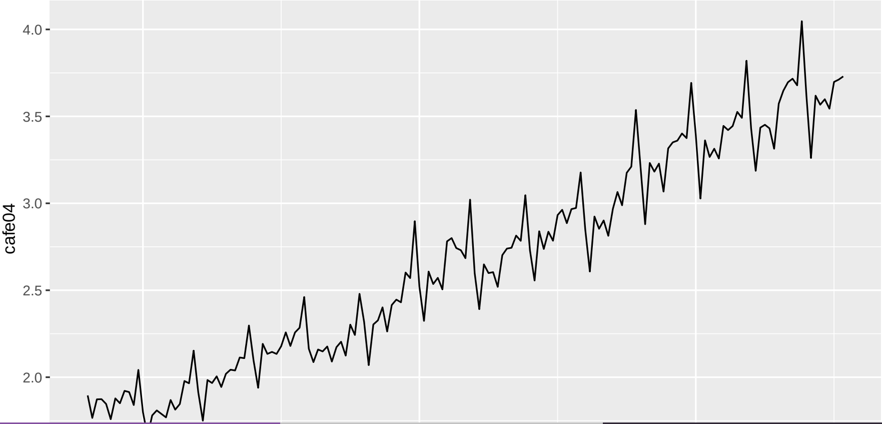
- it allows any length seasonality;
- for data with more than one seasonal period, you can include Fourier terms of different frequencies;
- the seasonal pattern is smooth for small values of K (but more wiggly seasonality can be handled by increasing K);
- the short-term dynamics are easily handled with a simple ARMA error.

Disadvantages

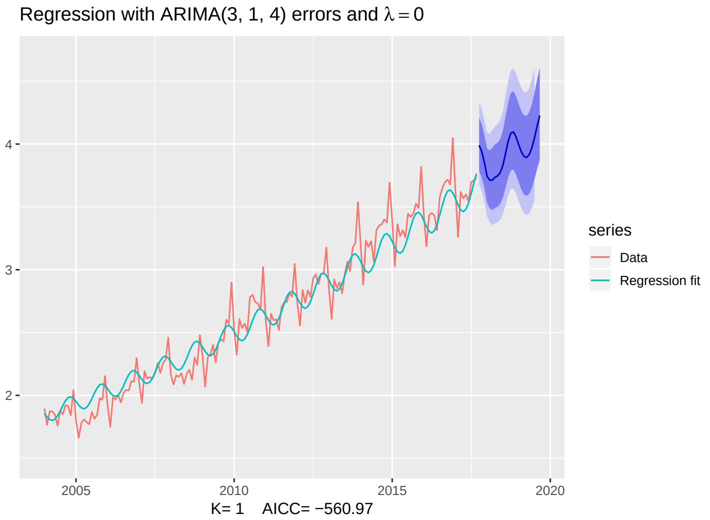
- seasonality is assumed to be fixed

Eating-out expenditure

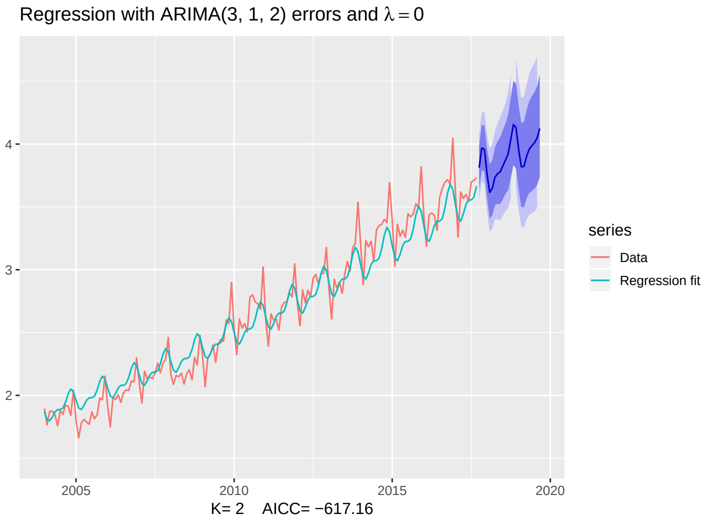
```r
cafe04 <- window(auscafe, start=2004)
autoplot(cafe04)
```



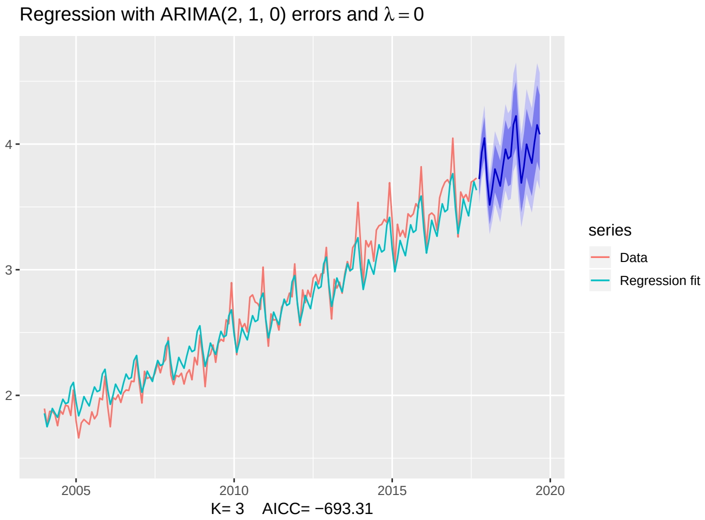
Eating-out expenditure



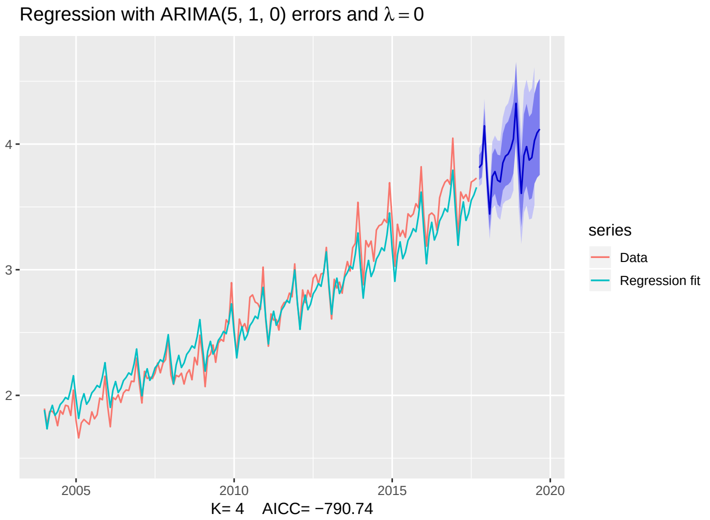
Eating-out expenditure



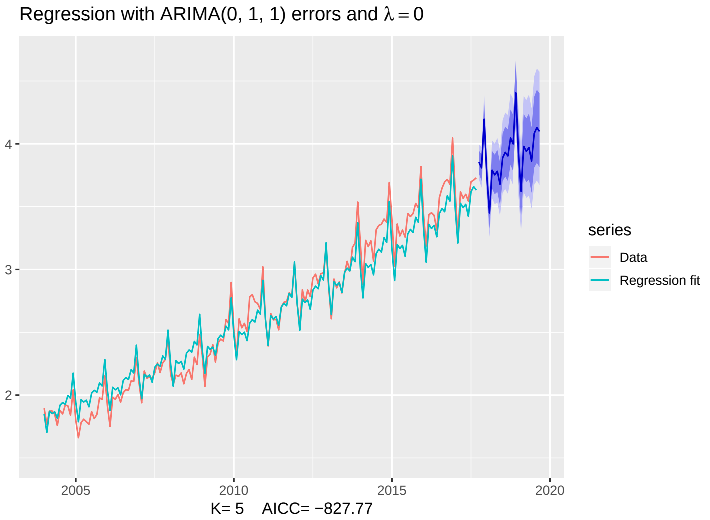
Eating-out expenditure



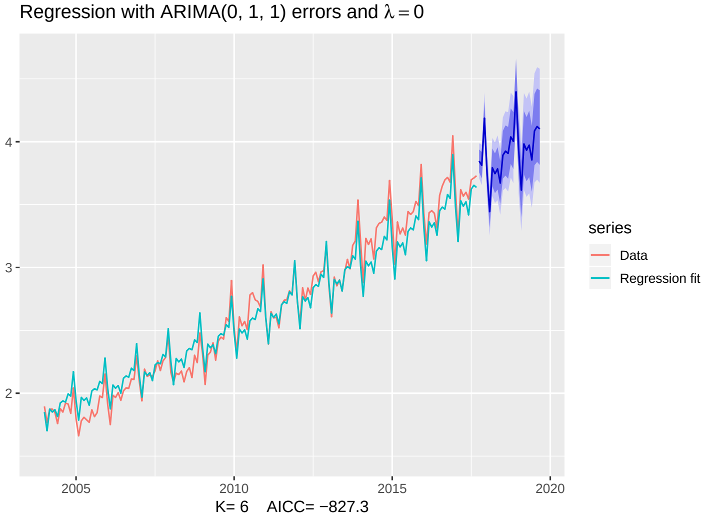
Eating-out expenditure



Eating-out expenditure



Eating-out expenditure



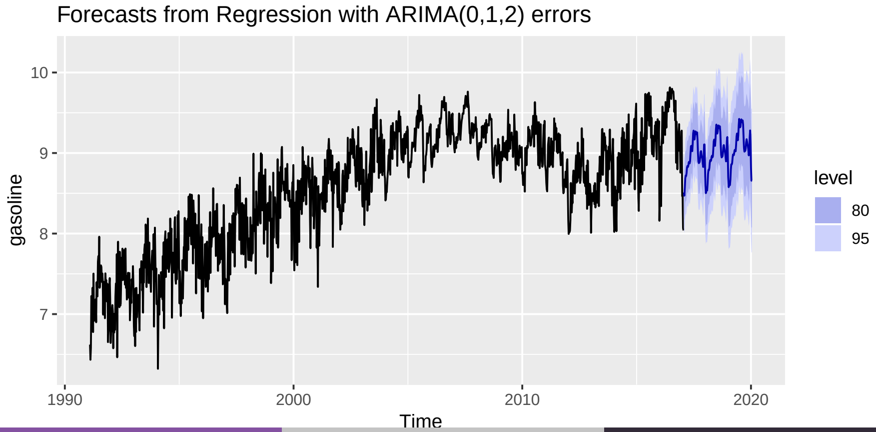
Example: weekly gasoline products

```
harmonics <- fourier(gasoline, K = 13)
(fit <- auto.arima(gasoline, xreg = harmonics, seasonal = FALSE))
```

```
## Series: gasoline
## Regression with ARIMA(0,1,2) errors
##
## Coefficients:
##          ma1    ma2  drift  S1-52   C1-52   S2-52
##       -0.961  0.094  0.001  0.031  -0.255  -0.052
## s.e.   0.027  0.029  0.001  0.012   0.012   0.009
##        C2-52  S3-52   C3-52  S4-52   C4-52
##       -0.018  0.024  -0.099  0.032  -0.026
## s.e.   0.009  0.008   0.008  0.008   0.008
##        S5-52   C5-52  S6-52   C6-52  S7-52  C7-52
##       -0.001  -0.047  0.058  -0.032  0.028  0.037
## s.e.   0.008   0.008  0.008   0.008  0.008  0.008
##       S8-52  C8-52   S9-52  C9-52  S10-52  C10-52
##       0.024  0.014  -0.017  0.012  -0.024   0.023
## s.e.  0.008  0.008   0.008  0.008   0.008   0.008
##       S11-52  C11-52  S12-52  C12-52  S13-52
##        0.000  -0.019  -0.029  -0.018   0.001
## s.e.   0.008   0.008   0.008   0.008   0.008
##       C13-52
##       -0.018
## s.e.   0.008
##
```

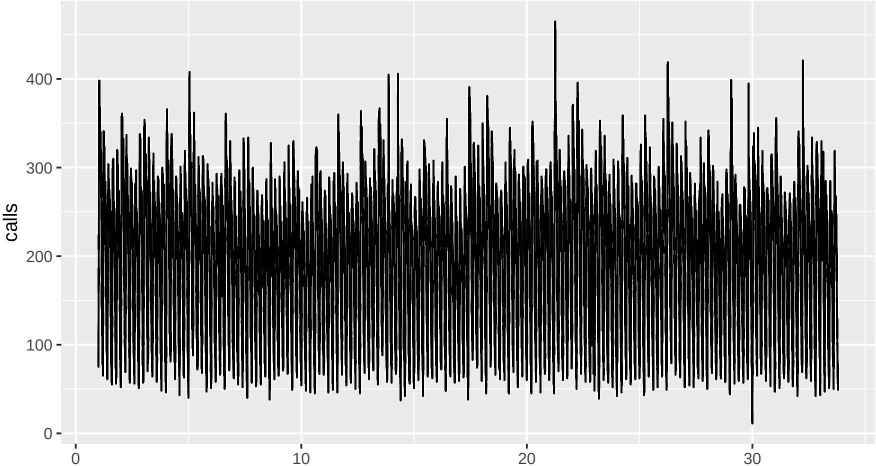
Example: weekly gasoline products

```r
newharmonics <- fourier(gasoline, K = 13, h = 156)
fc <- forecast(fit, xreg = newharmonics)
autoplot(fc)
```



5-minute call centre volume

```
autoplot(calls)
```



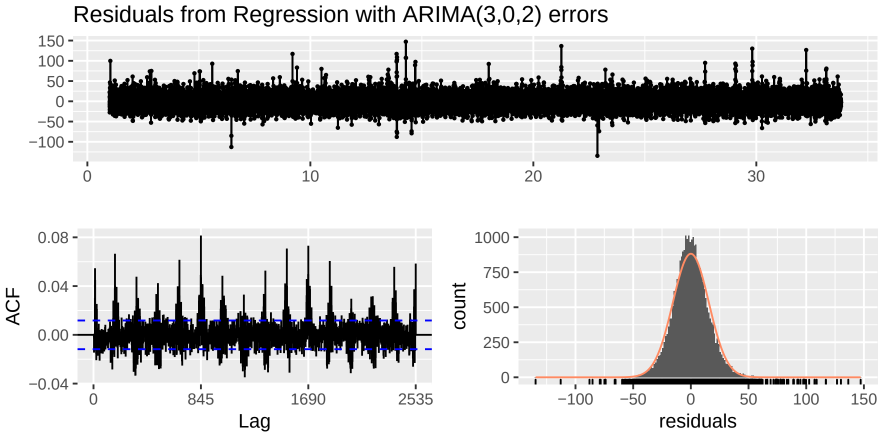
5-minute call centre volume

```
xreg <- fourier(calls, K = c(10,0))
(fit <- auto.arima(calls, xreg=xreg, seasonal=FALSE, stationary=TRUE))
```

```
## Series: calls
## Regression with ARIMA(3,0,2) errors
##
## Coefficients:
##         ar1    ar2     ar3     ma1     ma2
##       0.841  0.192  -0.044  -0.590  -0.189
## s.e.  0.169  0.178   0.013   0.169   0.137
##       intercept  S1-169   C1-169  S2-169   C2-169
##          192.07  55.245  -79.087  13.674  -32.375
## s.e.       1.76   0.701    0.701   0.379    0.379
##        S3-169  C3-169  S4-169  C4-169  S5-169
##       -13.693  -9.327  -9.532  -2.797  -2.239
## s.e.    0.273   0.273   0.223   0.223   0.196
##       C5-169  S6-169  C6-169  S7-169  C7-169
##        2.893   0.173   3.305   0.855   0.294
## s.e.   0.196   0.179   0.179   0.168   0.168
##       S8-169  C8-169  S9-169  C9-169  S10-169
##        0.857   -1.39  -0.986  -0.345    -1.20
## s.e.   0.160    0.16   0.155   0.155     0.15
##       C10-169
##         0.801
## s.e.    0.150
##
```

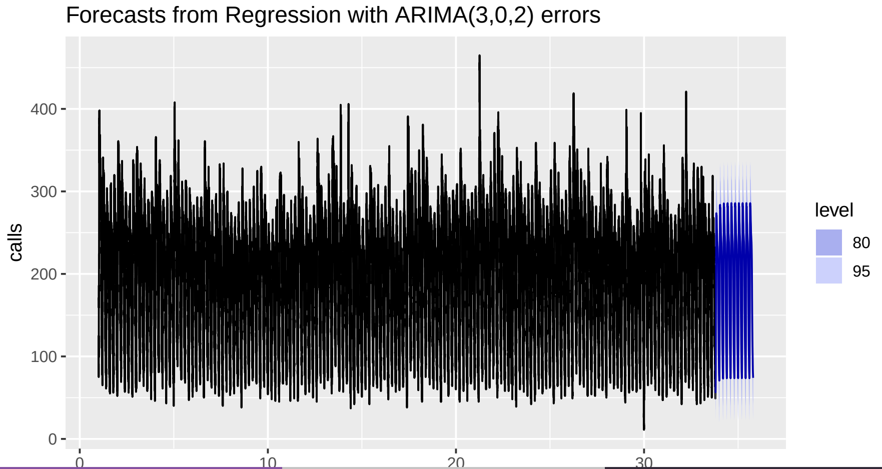
5-minute call centre volume

```r
checkresiduals(fit, test=FALSE)
```



5-minute call centre volume

```
fc <- forecast(fit, xreg = fourier(calls, c(10,0), 1690))
autoplot(fc)
```



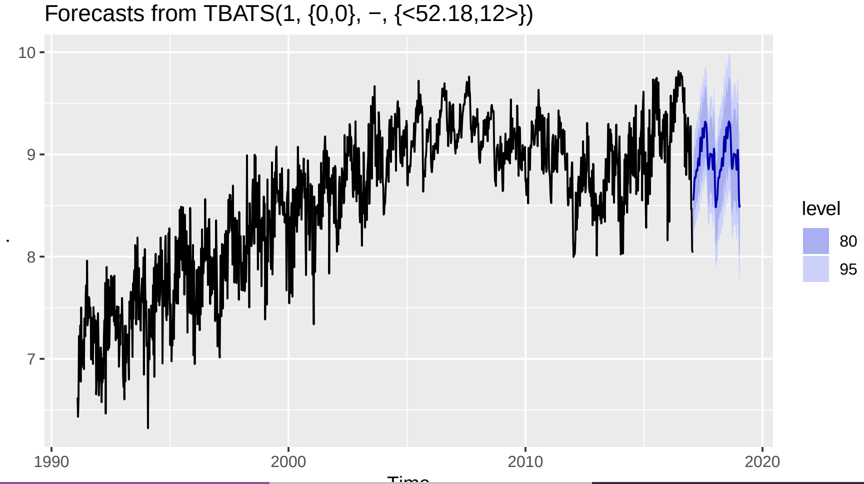
TBATS model

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)

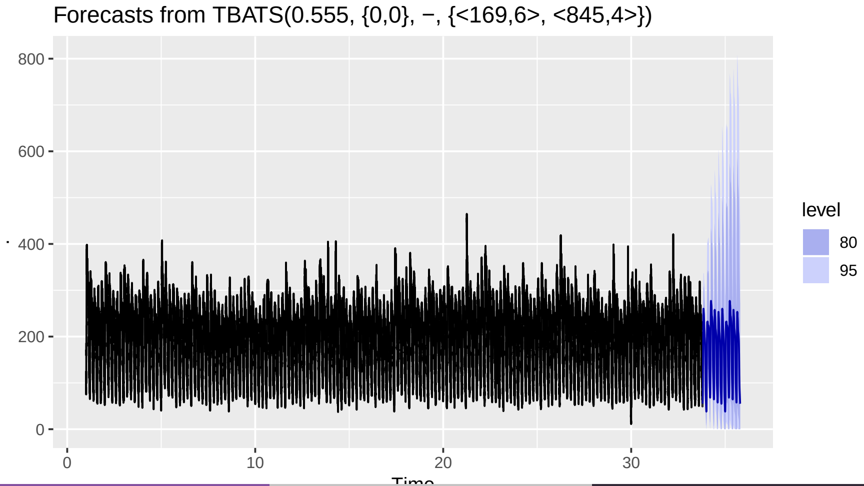
Complex seasonality

```
gasoline %>% tbats() %>% forecast() %>% autoplot()
```



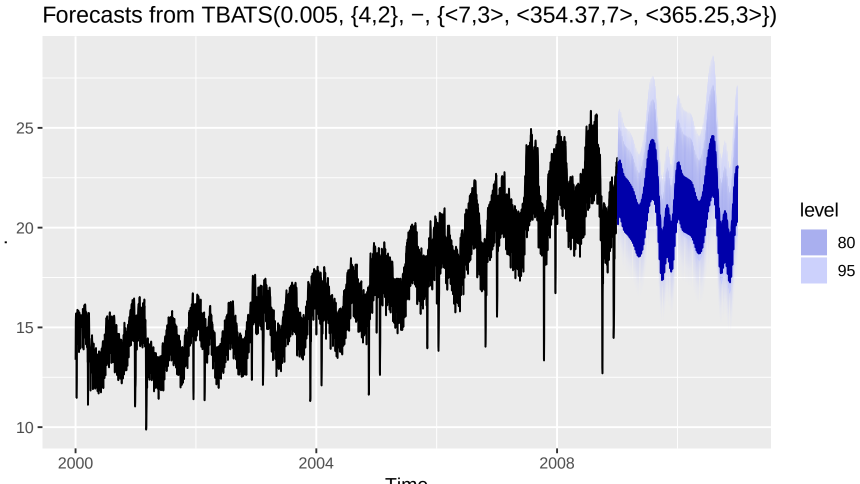
Complex seasonality

```r
calls %>% tbats() %>% forecast() %>% autoplot()
```



Complex seasonality

```
telec %>% tbats() %>% forecast() %>% autoplot()
```



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)

- Handles non-integer seasonality, multiple seasonal periods.
- Entirely automated
- Prediction intervals often too wide
- Very slow on long series

Outline

Lagged predictors

Sometimes a change in x_t does not affect y_t instantaneously

Lagged predictors

Sometimes a change in x_t does not affect y_t instantaneously

- y_t = sales, x_t = advertising.
- y_t = stream flow, x_t = rainfall.
- y_t = size of herd, x_t = breeding stock.

Lagged predictors

Sometimes a change in x_t does not affect y_t instantaneously

- y_t = sales, x_t = advertising.
- y_t = stream flow, x_t = rainfall.
- y_t = size of herd, x_t = breeding stock.

- These are dynamic systems with input (x_t) and output (y_t).
- x_t is often a leading indicator.
- There can be multiple predictors.

Lagged predictors

The model include present and past values of predictor: $x_t, x_{t-1}, x_{t-2}, \ldots$.

$$y_t = a + \nu_0 x_t + \nu_1 x_{t-1} + \cdots + \nu_k x_{t-k} + \eta_t$$

where η_t is an ARIMA process.

Lagged predictors

The model include present and past values of predictor: $x_t, x_{t-1}, x_{t-2}, \ldots.$

$$y_t = a + \nu_0 x_t + \nu_1 x_{t-1} + \cdots + \nu_k x_{t-k} + \eta_t$$

where η_t is an ARIMA process.

Rewrite model as

$$\begin{aligned} y_t &= a + (\nu_0 + \nu_1 B + \nu_2 B^2 + \cdots + \nu_k B^k) x_t + \eta_t \\ &= a + \nu(B) x_t + \eta_t. \end{aligned}$$

Lagged predictors

The model include present and past values of predictor: $x_t, x_{t-1}, x_{t-2}, \ldots$.

$$y_t = a + \nu_0 x_t + \nu_1 x_{t-1} + \cdots + \nu_k x_{t-k} + \eta_t$$

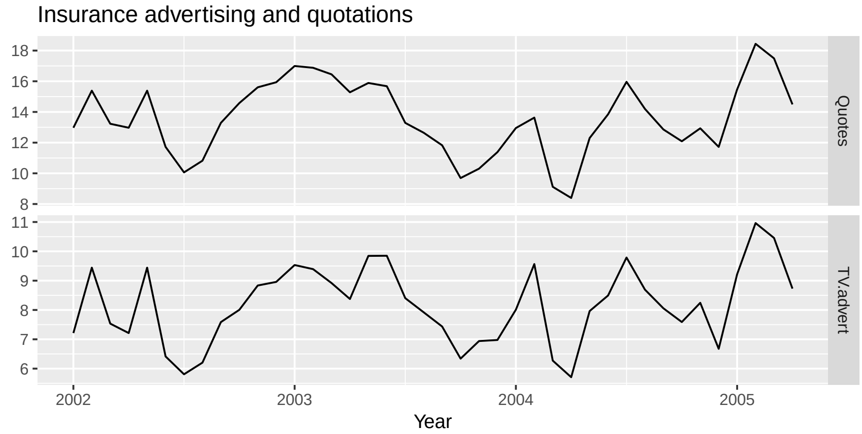
where η_t is an ARIMA process.

Rewrite model as

$$\begin{aligned} y_t &= a + (\nu_0 + \nu_1 B + \nu_2 B^2 + \cdots + \nu_k B^k) x_t + \eta_t \\ &= a + \nu(B) x_t + \eta_t. \end{aligned}$$

- $\nu(B)$ is called a *transfer function* since it describes how change in x_t is transferred to y_t.
- x can influence y, but y is not allowed to influence x.

Example: Insurance quotes and TV adverts



Example: Insurance quotes and TV adverts

```
Advert <- cbind(
    AdLag0 = insurance[,"TV.advert"],
    AdLag1 = lag(insurance[,"TV.advert"],-1),
    AdLag2 = lag(insurance[,"TV.advert"],-2),
    AdLag3 = lag(insurance[,"TV.advert"],-3)) %>%
  head(NROW(insurance))

# Restrict data so models use same fitting period
fit1 <- auto.arima(insurance[4:40,1], xreg=Advert[4:40,1],
  stationary=TRUE)
fit2 <- auto.arima(insurance[4:40,1], xreg=Advert[4:40,1:2],
  stationary=TRUE)
fit3 <- auto.arima(insurance[4:40,1], xreg=Advert[4:40,1:3],
  stationary=TRUE)
fit4 <- auto.arima(insurance[4:40,1], xreg=Advert[4:40,1:4],
  stationary=TRUE)
c(fit1$aicc,fit2$aicc,fit3$aicc,fit4$aicc)
```

Example: Insurance quotes and TV adverts

```
(fit <- auto.arima(insurance[,1], xreg=Advert[,1:2],
  stationary=TRUE))
```

```
## Series: insurance[, 1]
## Regression with ARIMA(3,0,0) errors
##
## Coefficients:
##         ar1     ar2    ar3  intercept  AdLag0
##        1.41  -0.932  0.359      2.039   1.256
## s.e.   0.17   0.255  0.159      0.993   0.067
##        AdLag1
##         0.162
## s.e.    0.059
##
## sigma^2 estimated as 0.217:  log likelihood=-23.9
## AIC=61.8   AICc=65.3   BIC=73.6
```

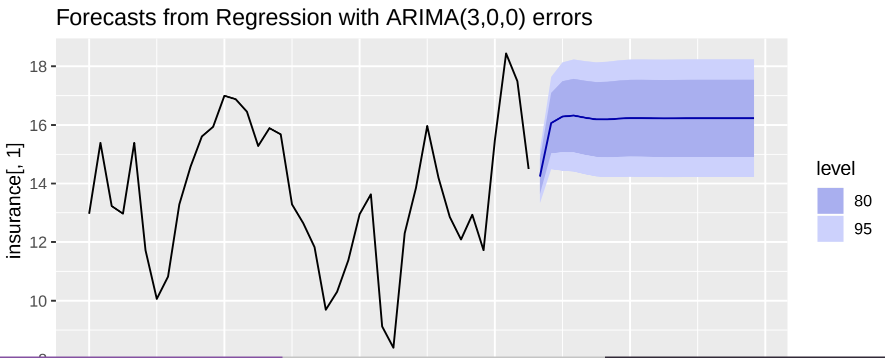
Example: Insurance quotes and TV adverts

```r
(fit <- auto.arima(insurance[,1], xreg=Advert[,1:2],
  stationary=TRUE))
```

```
## Series: insurance[, 1]
## Regression with ARIMA(3,0,0) errors
##
## Coefficients:
##         ar1     ar2    ar3  intercept  AdLag0
##        1.41  -0.932  0.359      2.039   1.256
## s.e.   0.17   0.255  0.159      0.993   0.067
##        AdLag1
##         0.162
## s.e.    0.059
##
## sigma^2 estimated as 0.217:  log likelihood=-23.9
## AIC=61.8   AICc=65.3   BIC=73.6
```

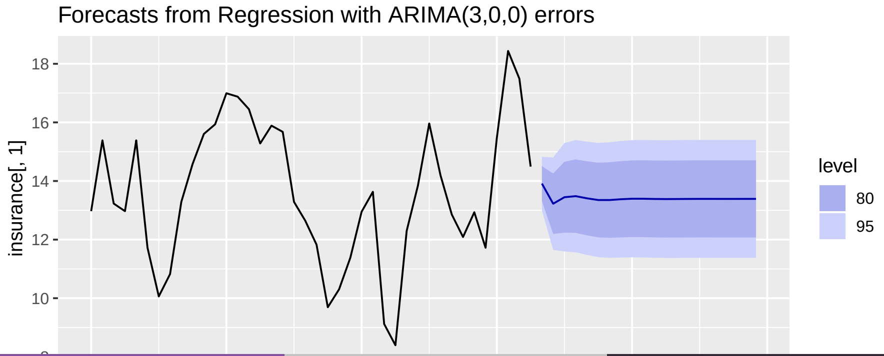
Example: Insurance quotes and TV adverts

```
fc <- forecast(fit, h=20,
  xreg=cbind(c(Advert[40,1],rep(10,19)), rep(10,20)))
autoplot(fc)
```



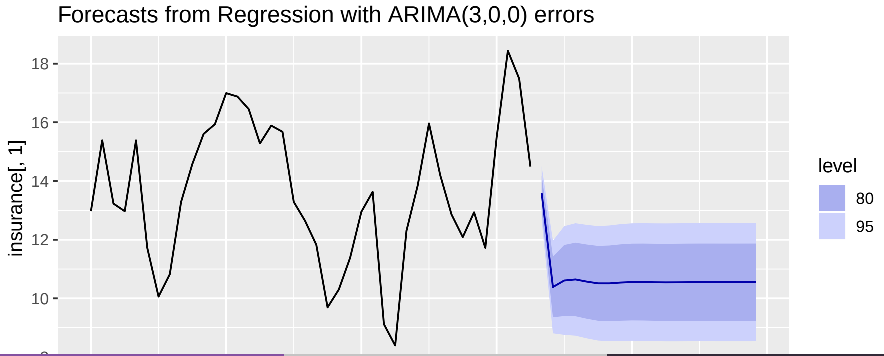
Example: Insurance quotes and TV adverts

```
fc <- forecast(fit, h=20,
  xreg=cbind(c(Advert[40,1],rep(8,19)), rep(8,20)))
autoplot(fc)
```



Example: Insurance quotes and TV adverts

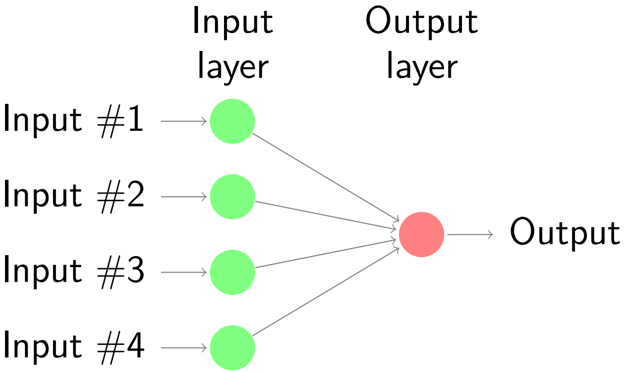
```r
fc <- forecast(fit, h=20,
  xreg=cbind(c(Advert[40,1],rep(6,19)), rep(6,20)))
autoplot(fc)
```



Outline

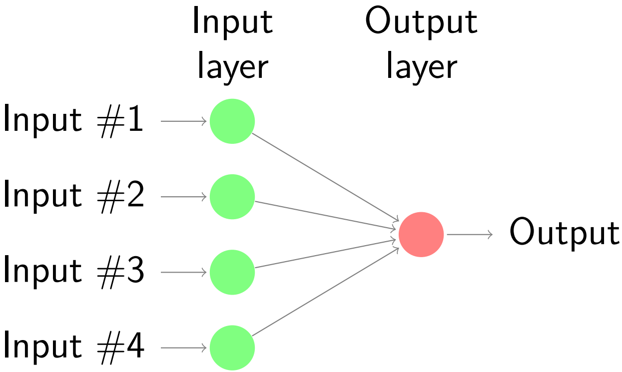
Neural network models

Simplest version: linear regression



Neural network models

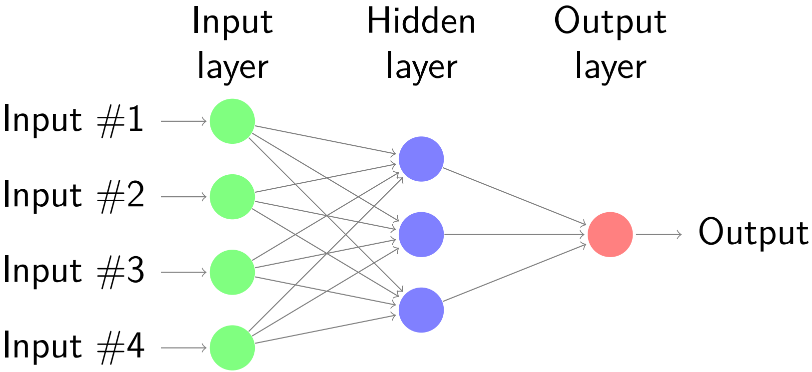
Simplest version: linear regression



- 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".

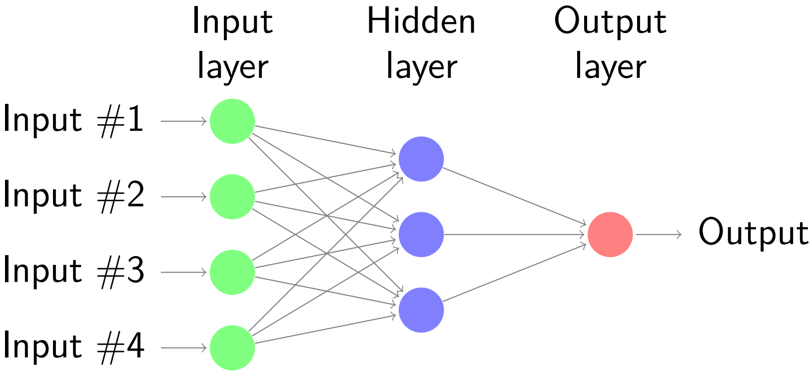
Neural network models

Nonlinear model with one hidden layer



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.

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

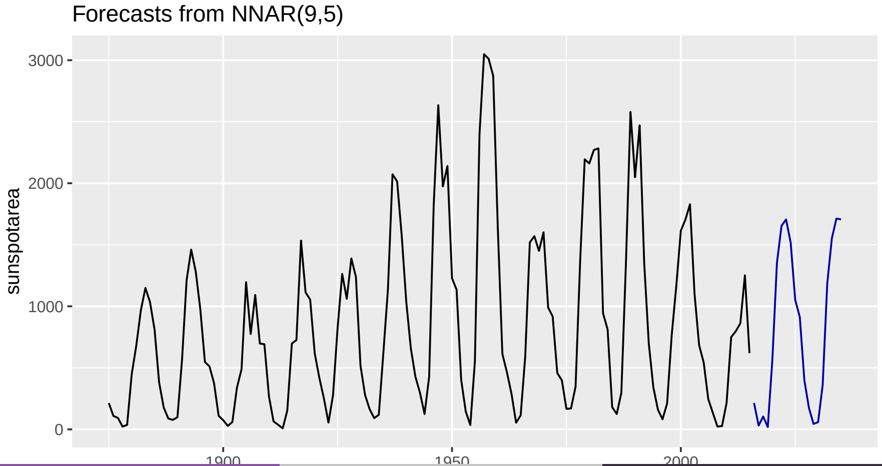
- The `nnetar()` function fits an NNAR$(p, P, k)_m$ model.
- If p and P are not specified, they are automatically selected.
- For non-seasonal time series, default $p =$ optimal number of lags (according to the AIC) for a linear AR(p) model.
- For seasonal time series, defaults are $P = 1$ and p is chosen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

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.

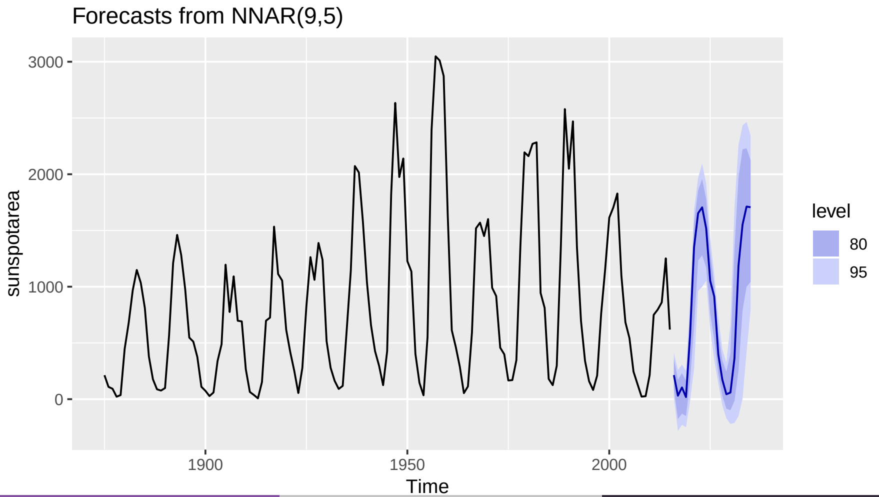
NNAR(9,5) model for sunspots

```r
fit <- nnetar(sunspotarea)
fit %>% forecast(h=20) %>% autoplot()
```



Prediction intervals by simulation

```
fit %>% forecast(h=20, PI=TRUE) %>% autoplot()
```



Outline

Forecast combinations

Clemen (1989)

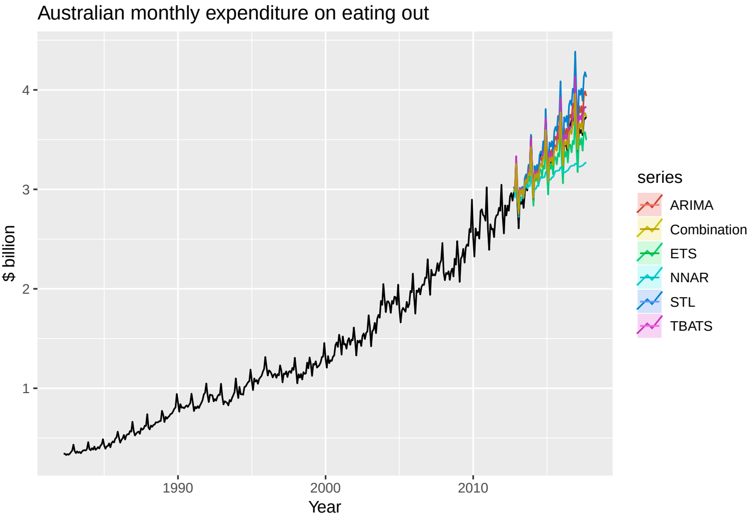
"The results have been virtually unanimous: combining multiple forecasts leads to increased forecast accuracy. ... In many cases one can make dramatic performance improvements by simply averaging the forecasts."

Forecast combinations

```
train <- window(auscafe, end=c(2012,9))
h <- length(auscafe) - length(train)
ETS <- forecast(ets(train), h=h)
ARIMA <- forecast(auto.arima(train, lambda=0, biasadj=TRUE),
  h=h)
STL <- stlf(train, lambda=0, h=h, biasadj=TRUE)
NNAR <- forecast(nnetar(train), h=h)
TBATS <- forecast(tbats(train, biasadj=TRUE), h=h)
Combination <- (ETS[["mean"]] + ARIMA[["mean"]] +
  STL[["mean"]] + NNAR[["mean"]] + TBATS[["mean"]])/5

autoplot(auscafe) +
  autolayer(ETS, series="ETS", PI=FALSE) +
  autolayer(ARIMA, series="ARIMA", PI=FALSE) +
  autolayer(STL, series="STL", PI=FALSE) +
  autolayer(NNAR, series="NNAR", PI=FALSE) +
  autolayer(TBATS, series="TBATS", PI=FALSE) +
  autolayer(Combination, series="Combination") +
  xlab("Year") + ylab("$ billion") +
  ggtitle("Australian monthly expenditure on eating out")
```

Forecast combinations



Forecast combinations

```
##          ETS       ARIMA     STL-ETS        NNAR
##       0.1370      0.1215      0.2145      0.2904
##        TBATS Combination
##       0.0941      0.0710
```

Outline

Missing values

Functions which can handle missing values

- `auto.arima()`, `Arima()`
- `tslm()`
- `nnetar()`

Models which cannot handle missing values

- `ets()`
- `stl()`
- `stlf()`
- `tbats()`

Missing values

Functions which can handle missing values

- `auto.arima()`, `Arima()`
- `tslm()`
- `nnetar()`

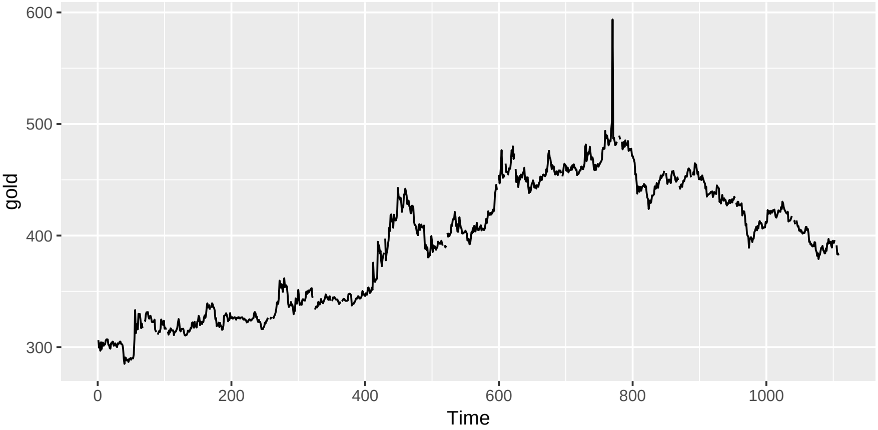
Models which cannot handle missing values

- `ets()`
- `stl()`
- `stlf()`
- `tbats()`

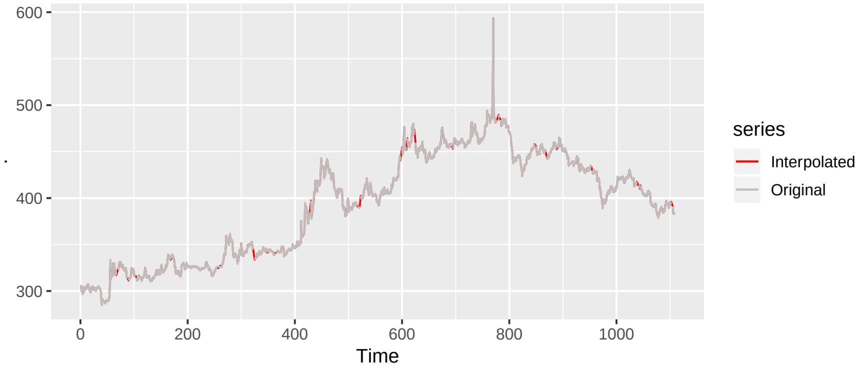
What to do?

1. Model section of data after last missing value.
2. Estimate missing values with `na.interp()`.

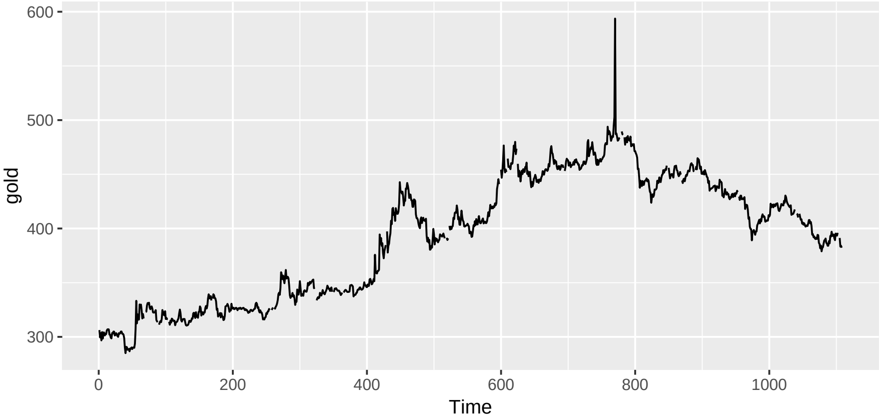
Missing values



Missing values



Outliers



Outliers

```
## $index
## [1] 770
##
## $replacements
## [1] 495
```

Outliers