Applied Time Series Econometrics

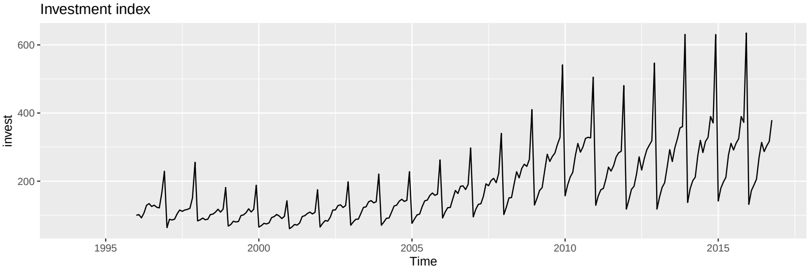
Oxana Malakhovskaya, NRU HSE

October 1, 2019

Preliminary transformation

- Sometimes a transformation of a series is needed before estimation.
- The transformation is useful when the data show different variation at different levels of the series.

```
invest_index <- read_csv("invest_index.csv")
invest <- ts(invest_index, start = c(1994, 1), freq = 12)
autoplot(invest) + ggtitle("Investment index")
```



Transformation examples

Denote original series as y_t and transformed series as w_t.

1. Square root $w_t = \sqrt{y_t}$
2. Cube root $w_t = \sqrt[3]{y_t}$
3. Logarithm $w_t = log(y_t)$

The strength of transformation increases from 1 to 3.

Logarithmic transformation is particularily useful as changes at log scale represent relative deviations at original scale.

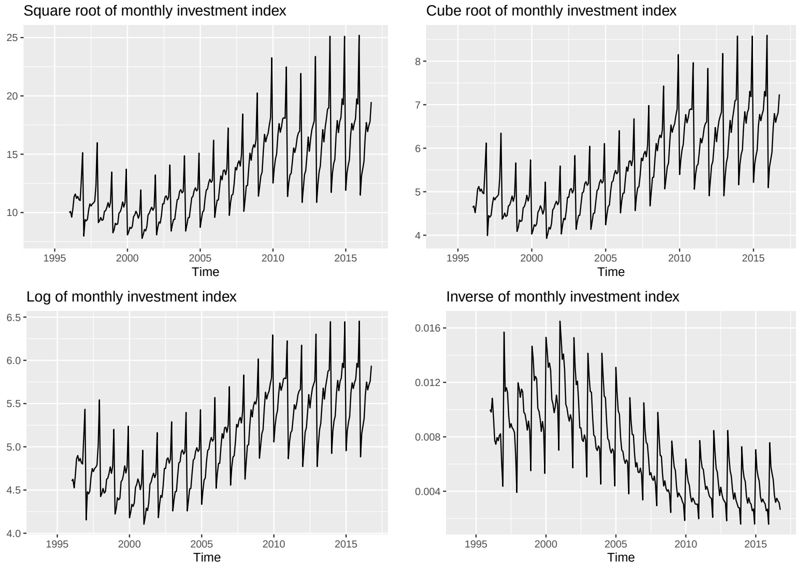
Relative deviations are percent deviations divided by 100.

Transformed series

```
plot1 <- autoplot(sqrt(invest)) + ylab("")
  ggtitle("Square root of monthly investment index")
plot2 <- autoplot(invest^(1/3)) + ylab("")+
  ggtitle("Cube root of monthly investment index")
plot3 <- autoplot(log(invest)) + ylab("")+
  ggtitle("Log of monthly investment index")
plot4 <- autoplot(invest^(-1)) + ylab("")+
  ggtitle("Inverse of monthly investment index")
```

Transformed series(2)

```r
grid.arrange(plot1, plot2, plot3, plot4, nrow = 2)
```



Box-Cox transformation: idea

Each of these transformations is a member of the family of Box-Cox transformations:

$$w_t = \begin{cases} log(y_t), & \lambda = 0 \\ \frac{y_t^\lambda - 1}{\lambda} & \lambda \neq 0 \end{cases}$$

If:

- $\lambda = 1$ - no tranformation is done
- $\lambda = 1/2$ -square root plus linear transformation
- $\lambda = 0$ - natural logarithm
- $\lambda = -1$ -inverse plus one

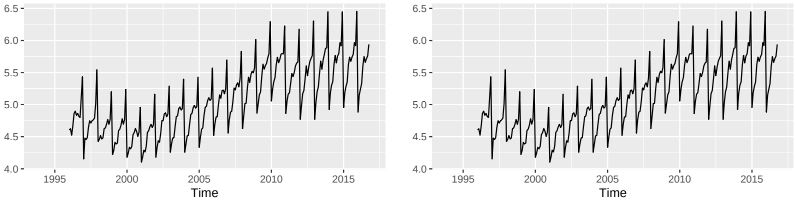
Obviousy:

- If some $y_t = 0$ then λ must be greater than zero.
- If some $y_t < 0$ then power transformation is possible only of a constant is added to the original series.

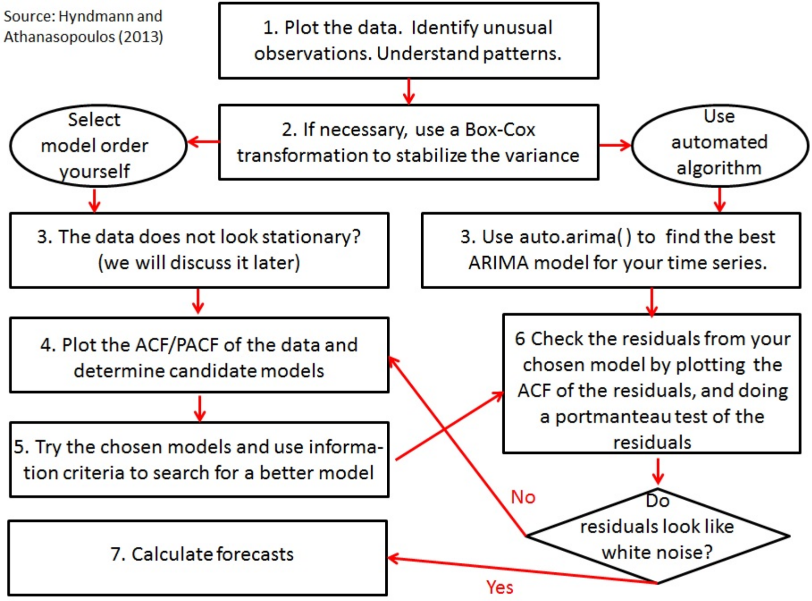
Box-Cox transformation: code

A Box-Cox type transformation is done with the `BoxCox()` function from the `forecast` package.

```
plot1 <- autoplot(log(invest))+ylab("")
plot2 <- autoplot(BoxCox(invest, lambda = 0))+ylab("")
grid.arrange(plot1, plot2, ncol = 2)
```



Algorithm of working with TS



Portmanteau test: idea

- Portmanteau test for residual autocorrelation checks the null hypothesis that there is no remaining residual autocorrelation at lags 1 to h against the alternative that at least one of the autocorelations is nonzero.

$$H_0 : \rho_{u,1} = \cdots = \rho_{u,h} = 0$$
$$H_1 : \rho_{u,i} \neq 0 \quad \text{for at least one } i = 1, \ldots, h,$$

where $\rho_{u,i} = cor(u_t, u_{t-i})$ denotes an autocorrelation coefficient of the resudual series.

$\hat{u}_t$ are residuals from an estimated ARMA(p,q) model.

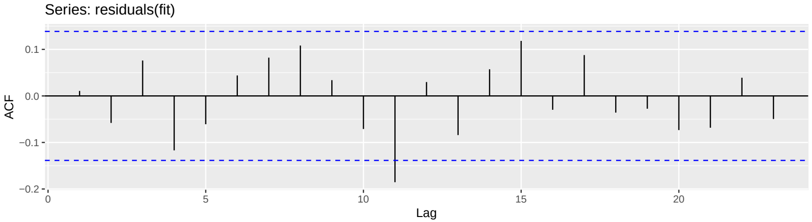
$$Q_h = T^2 \sum_{j=1}^{h} \frac{1}{T-j} \hat{\rho}_{u,j} \sim \chi^2(h-p-q)$$

Portmanteau test: code

- It is the same Box-Ljung test that we discussed in Practice session 5 but now it is applied to residuals and the option *fitdf* is not zero.

```
y <- arima.sim(model = list(ar = c(0.4,  0.5), ma = 0.4,
                            order = c(2,0,1), mean = 3),
                            n = 200)
fit <- Arima(y,order = c(2,0,1), include.constant = TRUE)
ggAcf(residuals(fit))
Box.test(residuals(fit), lag = 24, fitdf = 3,
                            type = 'Ljung')
```

Portmanteau test: results




```
##
##  Box-Ljung test
##
## data:  residuals(fit)
## X-squared = 29.454, df = 21, p-value = 0.1035
```

Forecasting: an algorithm

Point forecast

1. Rearrange ARIMA so y_t is on LHS and all other terms on RHS.
2. Rewrite the equation by replacing t by $T + h$
3. On the RHS, replace future observarions by their forecasts, future errors by zero, and past errors by corresponding residuals.

Example: ARMA(p,q)

$$y_t = \hat{\mu} + \hat{\phi}_1 y_{t-1} + \hat{\phi}_2 y_{t-2} + \cdots + \hat{\phi}_p y_{t-p} + \nu_t + \\ + \hat{\psi}_1 \nu_{t-1} + \cdots + \hat{\psi}_q \nu_{t-q}$$

Forecast for one period

$h = 1$

Replace t by $T + 1$

$$y_{T+1} = \hat{\mu} + \hat{\phi}_1 y_T + \hat{\phi}_2 y_{T-1} + \cdots + \hat{\phi}_p y_{T-p+1} + \nu_{T+1} + \\ + \hat{\psi}_1 \nu_T + \cdots + \hat{\psi}_q \nu_{T-q+1}$$

Replace ν_{T+1} by 0, and ν_{T-j} by $\hat{\nu}_{T-j} \forall j \geq 0$, and y_{T+1} by $y_{T+1|T}$

$$y_{T+1|T} = \hat{\mu} + \hat{\phi}_1 y_T + \hat{\phi}_2 y_{T-1} + \cdots + \hat{\phi}_p y_{T-p+1} + \\ + \hat{\psi}_1 \hat{\nu}_T + \cdots + \hat{\psi}_q \hat{\nu}_{T-q+1}$$

Forecast for two periods

$h = 2$

Replace t by $T + 2$

$$y_{T+2} = \hat{\mu} + \hat{\phi}_1 y_{T+1} + \hat{\phi}_2 y_T + \cdots + \hat{\phi}_p y_{T+2-p} + \nu_{T+2} + \\ + \hat{\psi}_1 \nu_{T+1} + \cdots + \hat{\psi}_q \nu_{T-q+2}$$

Replace ν_{T+1} and ν_{T+2} by 0, and ν_{T-j} by $\hat{\nu}_{T-j} \forall j \geq 0$, y_{T+1} by $\hat{y}_{T+1|T}$ and y_{T+2} by $y_{T+2|T}$

$$y_{T+2|T} = \hat{\mu} + \hat{\phi}_1 \hat{y}_{T+1|T} + \hat{\phi}_2 y_T + \cdots + \hat{\phi}_p y_{T+2-p} + \\ + \hat{\psi}_2 \hat{\nu}_T + \cdots + \hat{\psi}_q \hat{\nu}_{T-q+2}$$

Forecasts:example

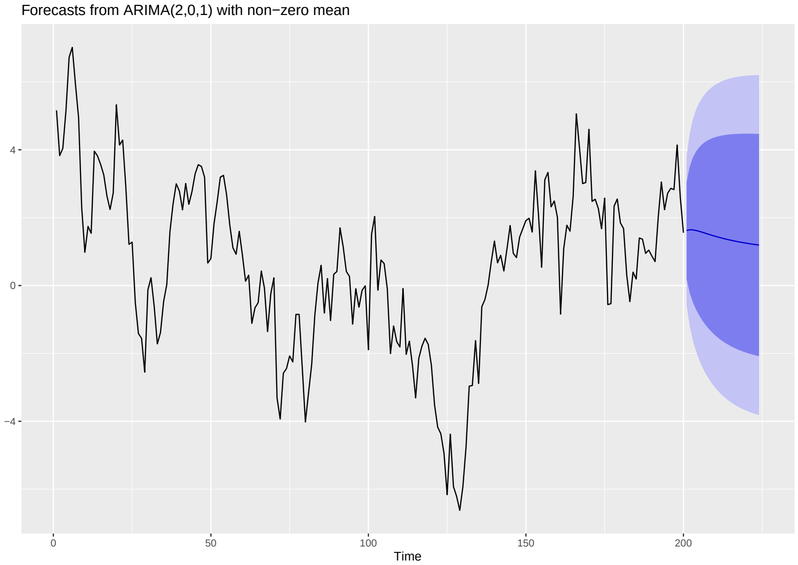
```r
y <- arima.sim(model = list(ar = c(0.4,  0.5), ma = 0.4,
                            order = c(2,0,1), mean = 3),
                            n = 200)
fit <- Arima(y,order = c(2,0,1), include.constant = TRUE)
```

Forecasts: code(1)

```
forecast(fit, h = 12, level = 95)
autoplot(forecast(fit, h=24)) + ylab("")
```

```
##      Point Forecast      Lo 95    Hi 95
## 201        1.621051 -0.5545635 3.796666
## 202        1.642560 -1.2005734 4.485693
## 203        1.639041 -1.6207569 4.898839
## 204        1.621686 -1.9361801 5.179552
## 205        1.597049 -2.1900159 5.384113
## 206        1.568948 -2.4022788 5.540176
## 207        1.539592 -2.5838462 5.663030
## 208        1.510237 -2.7413744 5.761848
## 209        1.481583 -2.8793349 5.842500
## 210        1.454001 -3.0009560 5.908958
## 211        1.427674 -3.1086965 5.964044
## 212        1.402672 -3.2045041 6.009848
```

Forecasts: code(2)



Forecast intervals

95% forecast interval:

$$\hat{y}_{T+h|T} \pm 1.96\sqrt{var_{T+h|T}},$$

where $var_{T+h|T}$ is estimated forecast variance

$var_{T+1|T} = \hat{\sigma}^2$ for all ARIMA models regardless of parameters and orders

- Forecast variance increases in size with forecast horizon
- Calculations assume residuals are uncorrelated and normally distributed
- Forecast intervals tend to be too narrow
  - the uncertainty in parameter estimates
  - the ARIMA model assumes historical patterns will not change during the forecast period.

Test for nornnormality: an idea

- Jarque and Bera(1987) proposed a test for nonnormality that checks whether the third and forth moments of the standardized residuals are consistent with a standard normal distribution.

$$\begin{aligned} H_0 &: E(u_t^s)^3 = 0 \quad \text{and} \quad E(u_t^s)^4 = 3 \\ H_1 &: E(u_t^s)^3 \neq 0 \quad \text{or} \quad E(u_t^s)^4 \neq 3, \end{aligned}$$

where $u_t^s = u_t/\sigma_u$ denotes the standardized true model residuals.

Test for nornnormality: statistics

Under H_0

$$JB = \frac{T}{6}\left[T^{-1}\sum_{t=1}^{T}(\hat{u}_t^s)^3\right]^2 + \frac{T}{24}\left[T^{-1}\sum_{t=1}^{T}(\hat{u}_t^s)^4 - 3\right]^2 \sim \chi^2(2)$$

- If H_0 is rejected, the normal distribution is also rejected. If the null hypothesis holds, this does not necessarily mean that underlying distribution is actually normal but only that it has the same first four moments as the normal distribution.

Test for nonnormality: code

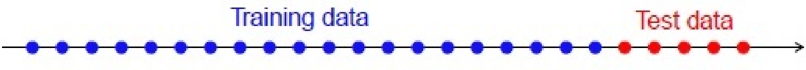
To apply the Jarque-Bera test we use the `jarque.bera.test()` function from the `tseries` package.

```
library(tseries)
jarque.bera.test(residuals(fit))
```

```
##
##  Jarque Bera Test
##
## data:  residuals(fit)
## X-squared = 1.7284, df = 2, p-value = 0.4214
```

Forecast evaluation

- We have just discussed how to make a forecast for just one model in hand.
- But how to compare different models in terms of their forecasting accuracy if we have a set of candidate models?
- First, we calculate pseudo real-time forecasts for the periods for which we have the data available.



Measures of accuracy

Scale-dependent errors

Forecast error: $e_i = y_i - \hat{y}_i$

Mean absolute error: $MAE = mean(|e_i|)$

Root mean squared error: $RMSE = \sqrt{mean(e_i^2)}$

Percentage errors

Percentage error: $p_i = \frac{100e_i}{y_i}$

Mean absolute percentage error: $MAPE = mean(|p_i|)$

Scaled errors

Scaled error: $q_j = \frac{e_j}{\frac{1}{T-1}\sum_{t=2}^{T}|y_t - y_{t-1}|}$

Mean absolute scaled error: $MASE = mean(|q_j|)$

Accuracy measures

So, some popular accuracy measures are calculated as:

$$MAE_h = \frac{1}{T}\sum_{t=1}^{T}|y_t - \hat{y}_{t|t-h}|$$

$$MSE_h = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_{t|t-h})^2$$

$$RMSE_h = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_{t|t-h})^2}$$

$$MAPE = \frac{100}{T}\sum_{t=1}^{T}\frac{|y_t - \hat{y}_{t|t-h}|}{y_t}$$

Accuracy measures:code

- We can compare the accuracy of forecasts based on several models.
- In practice session 5 we had two alternative fits for the same series

```
US_inv <- read_csv("US_investment.dat")
US_inv_ts <- ts(US_inv, start = c(1947, 2), freq = 4)
train = window(US_inv_ts, end = c(1969,4))
fit1 <- Arima(train, order = c(4,0,4))
fore1 <- forecast(fit1, h = 12)
fit2 <- auto.arima(train, d = 0, seasonal = FALSE)
fore2 <- forecast(fit2, h = 12)
options(digits = 3)
accuracy(fore1, US_inv_ts)
accuracy(fore2, US_inv_ts)
```

```
##                   ME RMSE  MAE  MPE MAPE  MASE     ACF1
## Training set 0.0111  2.3 1.75  NaN  Inf 0.472 0.000678
## Test set     1.8527  3.8 2.98 80.6  106 0.803 0.243160
```

```
##                   ME RMSE  MAE  MPE MAPE  MASE    ACF1
## Training set 0.00618 2.36 1.83 -Inf  Inf 0.492 -0.0273
## Test set     1.81046 3.68 2.94   64  108 0.794  0.2564
```

Literature

1. Hyndman, R. J. and Athanasopoulos, G. (2013) Forecasting: principles and practice, Otexts
2. Lütkepohl, H., and Krätzig, M. (2004) Applied Time Series Econometrics, New York, Cambridge University Press