Model specification

Susan Thomas

September 25, 2009

Goal: How to choose one out of alternative AR/MA models?

- Recap: Specification = formulation + selection.
- Recap: Formulation: Box–Jenkins Methodology.
- Specification based on Likelihood values at the optimum parameter values.
- Specification based on residual analysis.

Recap: Specification = formulation + selection

Formulation is based on a mixture of

- theoretical knowledge about the problem and
- diagnostic, exploratory tests of the data.

The **Box–Jenkins** methodology: separate the identification of the model from the estimation of the model.

Selection is based on post-estimation analysis. This includes:

- value of the likelihood at the optimal parameters,
- hypothesis tests of coefficients,
- analysis of the residuals, and
- performance of model forecasts.

Recap: Box–Jenkins identification methodology

Step 1 Graph the data.
Aim: To identify possible trends, seasonality, breaks, etc. **Action:** Use OLS to estimate the trends, de-trend the raw data.
The analysis is carried out on the residuals from the OLS trend model.

Step 2 Plot acfs, pacfs
Aim: To identify the pattern of stochastic dependence in data. **Action:** Postulate a possible model of

- Pure AR? What order, (p)?
- Pure MA? What order, (q)?

Step 3: Estimation.

Part I

Likelihood measure for model specification

Estimation measures for model specification

- The AR order selection criteria, $C(p)$, generally takes the form:

  $$C(p) = \log L(p) + w_T c(p)$$

  where

  - $\log L(p)$ is the log likelihood of the AR(p) model. This typically decreases for a larger order.
  - w_T is a weighting function of the sample size T.
  - $c(p)$ is the function that rewards parsimony.

  That model is chosen which minimises $C(p)$.
- The ACF/PACF identify a set of possible models (say, k) for a given time series.

  $C(p)$ is used to select one/subset (say, k_1) out of k models implied by the ACF/PACF analysis.

Estimation measures for model specification

- **Standard forms** of $C(p)$:
  1. Akaike Information Criteria (**AIC**): $(2\log L)/T + (k2)/T$
  2. Schwarz–Bayes Criteria (**SBC**): $(2\log(L))/T + (k\log T)/T$ This is also sometimes referred to as the Bayes Information Criteria (**BIC**).

  where k is the number of parameters in the model, and T is the number of observations.
- **Facts:**
  - AIC asymptotically overestimates the order with positive probability.
  - SBC is strongly consistent in identifying the correct order of the process if the actual DGP is a finite-order AR process.
- Thus, if $\hat{p}(AIC), \hat{p}(SBC)$ are the orders selected by the AIC and the SBC respectively, and they are different,

$$\hat{p}(SBC) < \hat{p}(AIC)$$

even in small sample.

Identifying ARMA models

- In principle, the same approach can be used for specifying the more general ARMA model.
- However, since there are two orders to simultaneously get "right",
  - there will be a much larger number of models to be estimated, which in turn, could lead to over-specified orders.
  - need to cheque for the "uniqueness" of the parameters. There could be cancellation of parts of the AR/MA operators that is possible.
- The specification of ARMA models is therefore done using multiple steps using simpler estimation models.

Lack of uniqueness: possible cancellation of AR/MA terms in ARMA models

- Consider the following ARMA(2,1) model:

$$y_t = -0.2y_{t-1} + 0.15y_{t-2} - 0.3\epsilon_{t-1} + \epsilon_t$$

- This is a plausible model, which is stationary? invertible?
- This model can be reworked as:

$$\begin{aligned}
y_t &= (-0.2L + 0.15L^2)y_t + \epsilon_t(1 - 0.3L) \\
y_t(1 + 0.2L - 0.15L^2) &= \epsilon_t(1 - 0.3L) \\
(1 + 0.2L - 0.15L^2) &= (1 + 0.5L - 0.3L - 0.15L^2) \\
&= (1 + 0.5L) - 0.3L(1 + 0.5L) = (1 + 0.5L)(1 - 0.3L) \\
y_t(1 + 0.5L)(1 - 0.3L) &= \epsilon_t(1 - 0.3L) \text{ becomes,} \\
y_t(1 + 0.5L) &= \epsilon_t \\
\longrightarrow y_t &= -0.5y_{t-1} + \epsilon_t
\end{aligned}$$

which is actually an AR(1) model with the same ϵ series!

ARMA model specification using Hannan-Rissanen

- This is a multi-step process that is frequently employed to obtain the order of the ARMA(p,q) process.
- It is also used to get starting values of the p, q parameters of the model to use in the MLE.
- **Reference:** Hannan, E.J., and Rissanen, J., 1982, *Recursive estimation of fixed autoregressive-moving average order*, Biometrika 69, 81–94.

Specification process for ARMA models

- Task: identify p, q of the ARMA DGP.
- **Step 1:** Estimate a very long order AR(h) model, where $h >> p$.
  p can be guessed at from the ACF/PACF.
  Obtain the residuals $\hat{u}_t(h)$ from the AR(h) estimation.
- **Step 2:** Estimate the following model using OLS:

  $$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_n y_{t-n} + u_t + m_1 \hat{u}_{t-1}(h) + \ldots + m_l \hat{u}_{t-l}(h)$$

  This is done for all combinations of $(n, l) \leq p \leq h$.
- The order (n, l) which minimises $C(n + l)$ is chosen as the true order (p, q).

Part II

Post estimation residual analysis for model specification

Estimation residual analysis

- Once a model is specified, the residuals are analysed for model adequacy.

  An adequate model is one which captures all the dynamics and leaves only pure white noise residuals.
- **Step 1:** Graphs for visual checks of the residuals for temporal patterns.
- **Step 2:** Run statistical tests to test whether the residuals are white noise or not.

The Portmanteau test and modifications

- The **Portmanteau test** checks whether the residuals have no autocorrelation patterns.
- $H_0 : \rho_1 = \rho_2 = \ldots = \rho_h = 0$

$$H_1 : \rho_i \neq 0$$

- Test statistic: $Q_h = T \sum_{k=1}^{h} \hat{\rho}_k^2$
- Test distribution: $\sim \chi^2(h - p - q)$ under H_0.
- **Box-Ljung-Pierce test** modifies the test statistic to adjust for a better approximation to the distribution used.

$$Q_h^* = T(T+2) \sum_{k=1}^{h} \hat{\rho}_k^2/(T-k)$$

Other statistical tests on the residual autocorrelations

- **LM test**. This is also sometimes called the **Breusch-Godfrey test**. Fit an AR(h) model to the residuals:

  $$\begin{aligned} u_t &= \beta_1 u_{t-1} + \beta_2 u_{t-2} + \ldots + \beta_h u_{t-h} + \xi_t \\ H_0 : & \quad \beta_1 = \beta_2 = \ldots = \beta_h = 0 \text{ vs.} \\ H_1 : & \quad \beta_1 = 0, \text{ or } \beta_2 = 0, \text{ or } \ldots \beta_1 = h \end{aligned}$$

  This is estimated using OLS.

  The test statistic is TR^2 which is distributed as $\chi^2(h)$ under the null.

- **Higher-order moment tests**.
  The next level of tests check for the nonnormality of the residual distribution by testing the values of the higher order moments.
  - **Lomnicki-Jacque-Bera test:** for nonnormality of $3^{rd}, 4^{th}$ order moments.
  - **LM test on ARCH residuals:**
  - **RESET test**

  We do these in detail after developing models of heteroskedasticity.

Part III

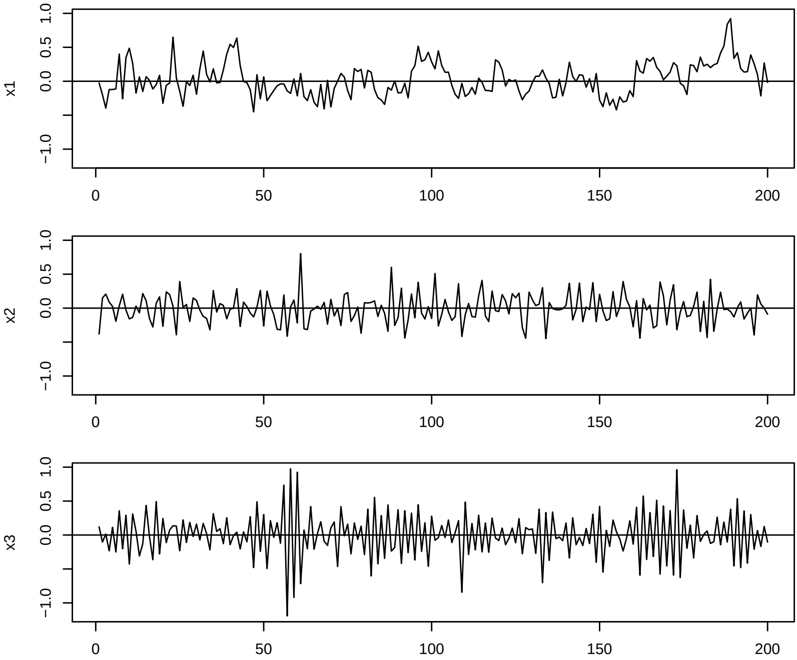
Applying our learning: Fitting models to data

We're given three time-series

	N	Mean	Variance
x1	200	0.03	0.06
x2	200	-0.01	0.05
x3	200	-0.00	0.11

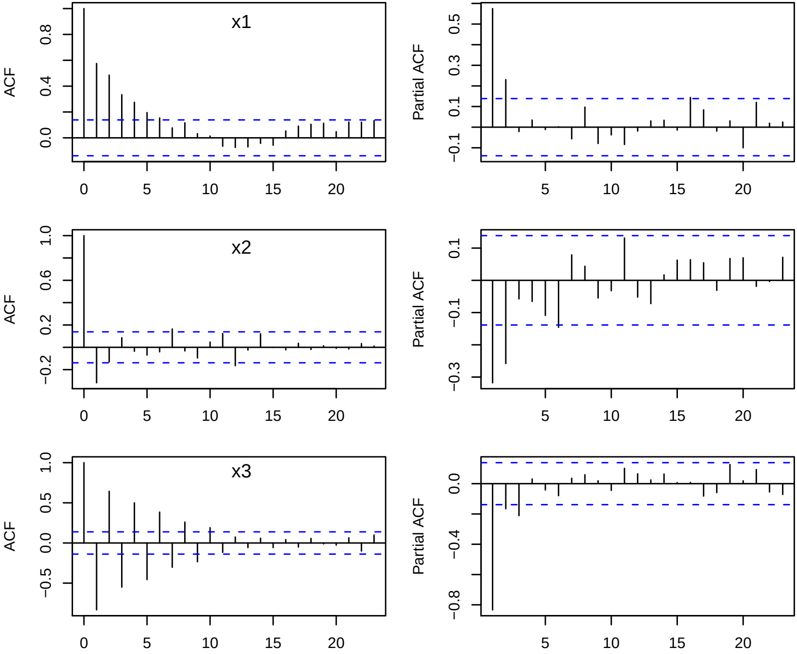
Let's look at them

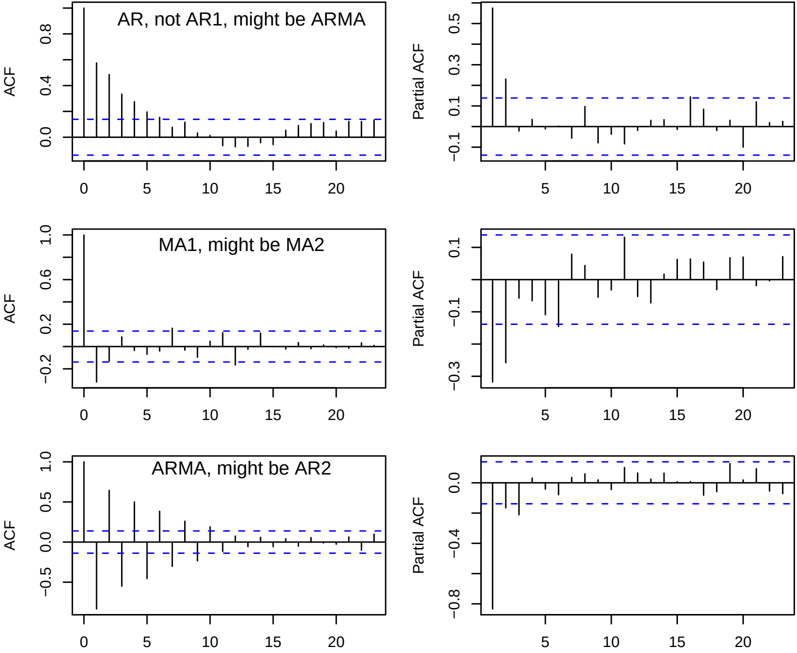
```
> par(mfrow=c(3,1),
+     mai=c(.4,.6,.1,.1))
> hilo <- range(c(x1,x2,x3))
> plot(x1, xlab="", ylim=hilo); abline(h=0)
> plot(x2, xlab="", ylim=hilo); abline(h=0)
> plot(x3, xlab="", ylim=hilo); abline(h=0)
```


x1
−1.0
0.0
0.5
1.0
0
50
100
150
200
x2
−1.0
0.0
0.5
1.0
0
50
100
150
200
x3
−1.0
0.0
0.5
1.0
0
50
100
150
200

Let's look at ACFs and PACFs

```
> par(mfrow=c(3,2), mai=c(.4,.6,.1,.1))
> mypic <- function(x, labelstring) {
+   acf(x,main="",xlab="",mar=c(1,1,1,.8))
+   text(x=12.5, y=.9, labels=labelstring, cex=1.2)
+   pacf(x,main="",xlab="",mar=c(1,1,1,.8))
+ }
> mypic(x1,"x1"); mypic(x2,"x2"); mypic(x3,"x3")
```


x1
ACF
0.0
0.4
0.8
0
5
10
15
20
Partial ACF
−0.1
0.1
0.3
0.5
5
10
15
20
x2
ACF
−0.2
0.2
0.6
1.0
0
5
10
15
20
Partial ACF
−0.3
−0.1
0.1
5
10
15
20
x3
ACF
−0.5
0.0
0.5
1.0
0
5
10
15
20
Partial ACF
−0.8
−0.4
0.0
5
10
15
20


AR, not AR1, might be ARMA
ACF
0.0
0.4
0.8
0
5
10
15
20
Partial ACF
−0.1
0.1
0.3
0.5
5
10
15
20
MA1, might be MA2
ACF
−0.2
0.2
0.6
1.0
0
5
10
15
20
Partial ACF
−0.3
−0.1
0.1
5
10
15
20
ARMA, might be AR2
ACF
−0.5
0.0
0.5
1.0
0
5
10
15
20
Partial ACF
−0.8
−0.4
0.0
5
10
15
20

Part IV

Analysing x_1

Let's evaluate AR models for x_1

Recall: We think it's AR, not AR(1), might be ARMA.

```
> results <- matrix(NA, nrow=4, ncol=3)
> rownames(results) <- paste("AR", 0:3)
> colnames(results) <- c("AIC", "SBC", "Ljung-Box p value")
> for (p in 0:3) {
+   m <- arima(x1, order=c(p, 0, 0), method="ML")
+   results[p+1,] <- c(m$aic, AIC(m, k=log(length(x1))),
+             Box.test(m$residuals, type="Ljung")$p.value)
+ }
```

Recall that the null of the Ljung-Box test is white noise

```
> Box.test(rnorm(100), type="Ljung")$p.value
[1] 0.4829763
> Box.test(cumsum(rnorm(100)), type="Ljung")$p.value
[1] 0
```

Model selection for x_1

	AIC	SBC	Ljung-Box p value
AR 0	8.5136	15.1102	0.0000
AR 1	-69.4124	-59.5175	0.0660
AR 2	-78.3769	-65.1836	0.9146
AR 3	-76.4596	-59.9680	0.9603

Suggests: AR(2) (a big gain in AIC compared with AR(1).

AR(2) model for x_1

```
> arima(x1, order=c(2, 0, 0), method="ML")

Call:
arima(x = x1, order = c(2, 0, 0), method = "ML")

Coefficients:
         ar1     ar2  intercept
      0.4386  0.2314     0.0259
s.e.  0.0686  0.0689     0.0412

sigma^2 estimated as 0.03792:  log likelihood = 43.19,  aic
```

Part V

Analysing x_2

Let's evaluate MA models for x_2

Recall: We think it's MA(1) (likely), might be MA(2).

```
> results <- matrix(NA, nrow=4, ncol=3)
> rownames(results) <- paste("MA", 0:3)
> colnames(results) <- c("AIC", "SBC", "Ljung-Box p value")
> for (q in 0:3) {
+   m <- arima(x2, order=c(0, 0, q), method="ML")
+   results[q+1,] <- c(m$aic, AIC(m, k=log(length(x2))),
+                 Box.test(m$residuals, type="Ljung")$p.value)
+ }
```

Model selection for x_2

	AIC	SBC	Ljung-Box p value
MA 0	-39.9416	-33.3450	0.0000
MA 1	-72.9462	-63.0512	0.4325
MA 2	-73.6604	-60.4671	0.9507
MA 3	-71.9553	-55.4637	0.9808

Suggests: MA(1). MA(2) is only a small gain in AIC compared with MA(1).

MA(1) model for x_2

```
> arima(x2, order=c(0, 0, 1), method="ML")

Call:
arima(x = x2, order = c(0, 0, 1), method = "ML")

Coefficients:
          ma1  intercept
      -0.4947    -0.0078
s.e.   0.0697     0.0071

sigma^2 estimated as 0.0394:  log likelihood = 39.47,  aic
```

Part VI

Analysing x_3

Let's evaluate AR and ARMA models for x_3

Recall: We think it's ARMA(1,1) or AR(2)

```
> results <- matrix(NA, nrow=5, ncol=3)
> rownames(results) <- c("AR0", "AR(1)","AR(2)","AR(3)",
+                         "ARMA(1,1)")
> colnames(results) <- c("AIC", "SBC", "Ljung-Box p value")
> for (p in 0:3) {
+   m <- arima(x3, order=c(p, 0, 0), method="ML")
+   results[p+1,] <- c(m$aic, AIC(m, k=log(length(x3))),
+             Box.test(m$residuals, type="Ljung")$p.value)
+ }
> m <- arima(x3, order=c(1,0,1), method="ML")
> results[5,] <- c(m$aic, AIC(m, k=log(length(x3))),
+             Box.test(m$residuals, type="Ljung")$p.value)
```

Model selection for x_3

	AIC	SBC	Ljung-Box p value
AR0	130.3049	136.9015	0.0000
AR(1)	-104.1459	-94.2509	0.0450
AR(2)	-107.6283	-94.4350	0.5794
AR(3)	-114.6651	-98.1735	0.9713
ARMA(1,1)	-111.3014	-98.1081	0.7441

Suggests: ARMA(1,1) or AR(3). The AR(3) model has a better AIC but requires one more free parameter.

AR(3) model for x_3

```
> arima(x3, order=c(3, 0, 0), method="ML")

Call:
arima(x = x3, order = c(3, 0, 0), method = "ML")

Coefficients:
          ar1      ar2      ar3  intercept
      -1.0021  -0.3675  -0.2089    -0.0009
s.e.   0.0689   0.0954   0.0687     0.0049

sigma^2 estimated as 0.03118:  log likelihood = 62.33,  aic
```

Strange AR(1) coefficient!

ARMA(1,1) model for x_3

```
> arima(x3, order=c(1, 0, 1), method="ML")

Call:
arima(x = x3, order = c(1, 0, 1), method = "ML")

Coefficients:
          ar1      ma1  intercept
      -0.7382  -0.3046    -0.0009
s.e.   0.0576   0.0904     0.0051

sigma^2 estimated as 0.03204:  log likelihood = 59.65,  aic
```

How different are these, really?

```
> tmp <- function(x, order) {
+   arima(x, order=order, method="ML")$residuals
+ }
> e <- cbind(tmp(x3, order=c(2,0,0)),
+            tmp(x3, order=c(3,0,0)),
+            tmp(x3, order=c(1,0,1)))
> colnames(e) <- c("AR(2)","AR(3)","ARMA(1,1)")
> cor(e)
```

```
              AR(2)     AR(3) ARMA(1,1)
AR(2)     1.0000000 0.9779688 0.9942808
AR(3)     0.9779688 1.0000000 0.9872445
ARMA(1,1) 0.9942808 0.9872445 1.0000000
```

Part VII

Bottom line

Systematic steps in ARMA identification

- Analyse the ACF and PACF
- Estimate some candidates
- Use the AIC, SBC and tests of serial correlation of the residual

How did we fare?

Problem	We think	The truth
x_1	AR(2), (0.44, 0.23)	AR(2), (0.4, 0.3)
x_2	MA(1), (-0.49)	MA(1) (-0.4)
x_3	ARMA(1,1), (-0.73, -0.3)	ARMA(1,1), (-0.7, -0.4)

Part VIII

Expert systems for ARMA identification

The idea

- An expert system : encode in computer software all the steps that a sophisticated human expert would make.
- Writing an expert system forces us to go up from vague intuition to writing down hard rules.
- The field of: *automatic identification of ARMA models*

Automatic choice of AR order

```
> ar(x1)$order
[1] 2
> ar(x2)$order
[1] 6
> ar(x3)$order
[1] 3
```

- He gets x_1 right
- He thinks x_2 is a long lag AR process
- He gets x_3 wrong.

forecast::auto.arima()

```
> library(forecast)
This is forecast 1.24
> auto.arima(x1, ic="aic", trace=TRUE)
 ARIMA(2,0,2) with non-zero mean : -74.03315
 ARIMA(0,0,0) with non-zero mean : 8.91052
 ARIMA(1,0,0) with non-zero mean : -68.45697
 ARIMA(0,0,1) with non-zero mean : -36.42676
 ARIMA(1,0,2) with non-zero mean : -75.10331
 ARIMA(1,0,1) with non-zero mean : -76.05008
 ARIMA(1,0,1) with zero mean     : -77.60985
 ARIMA(0,0,1) with zero mean     : -36.89050
 ARIMA(2,0,1) with zero mean     : -76.9843
 ARIMA(1,0,0) with zero mean     : -69.80587
 ARIMA(1,0,2) with zero mean     : -76.63824
 ARIMA(0,0,0) with zero mean     : 9.297301
 ARIMA(2,0,2) with zero mean     : -75.35354
```

What does it find?

```
> names(auto.arima(x1, ic="aic")$coef)
[1] "ar1" "ma1"
> names(auto.arima(x2, ic="aic")$coef)
[1] "ma1" "ma2"
> names(auto.arima(x3, ic="aic")$coef)
[1] "ar1" "ma1" "ma2"
```

SBC fares better?

```
> names(auto.arima(x1, ic="bic")$coef)

[1] "ar1" "ar2"

> names(auto.arima(x2, ic="bic")$coef)

[1] "ma1"

> names(auto.arima(x3, ic="bic")$coef)

[1] "ar1" "ma1"
```

Conclusion

- Your first best approach is to study the data, draw graphs, think.
- If you're in a hurry, you can go far with forecast::auto.arima(x, ic="bic")