3.ARMA-Properties-and-Examples

2020年5月23日 星期六 下午8:24

_8a799686
09cc4ad...

We are looking at ARMA processes. We have seen how to express simple cases of ARMA as either Moving Average or Autoregressive Processes and now we turn to an example. Remember that we form a mixed process by bringing together an *MA(q)* and an *AR(p)*

$$X_t = Noise + AutoRegressive\ Part + Moving\ Average\ Part$$

$$X_t = Z_t + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \theta_1 Z_{t-1} + \cdots + \theta_q Z_{t-q}$$

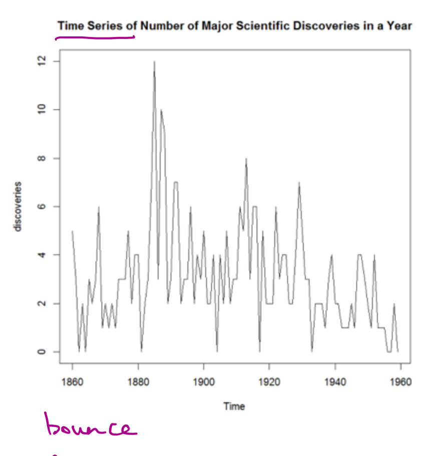
Example: ARMA Model

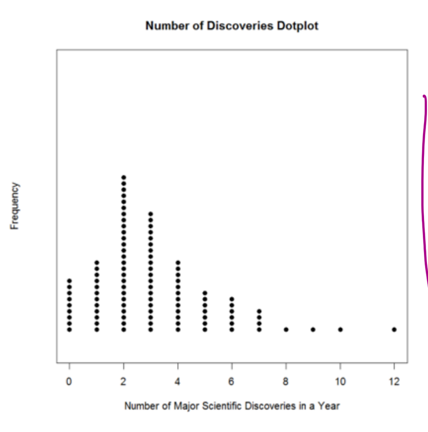
The R package has quite a few data sets installed and ready for our explorations. Let's look at some data about the number of major scientific discoveries in a given year. If you type at the command line: *discoveries* you will see the "raw" data. We make some obvious plots.

```
plot(discoveries,
     main = "Time Series of Number of Major Scientific Discoveries in a Year")
```

routine - discrete data - break data out by location better.

```
stripchart(discoveries, method = "stack", offset=.5, at=.15,pch=19,
     main="Number of Discoveries Dotplot",
     xlab="Number of Major Scientific Discoveries in a Year",
     ylab="Frequency")
```





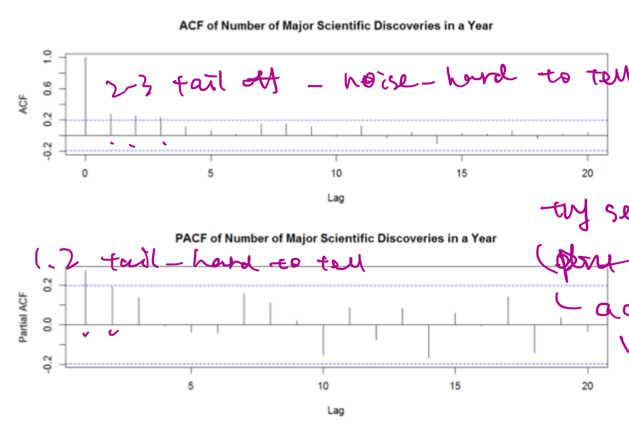
bounce
structure

poisson distribution

	AR(p)	MA(q)	ARMA(p,q)
ACF	Tails off	Cut off after lag q	Tails off
PACF	Cut off after lag p	Tails off	Tails off

The *stripchart()* command is a great way to see a "dot plot" when you are feeling casual, or have a smaller data set. We should also look at our autocorrelation and partial autocorrelation plots. We see the time plot has no obvious trends or seasonality, so we can try fitting an ARMA model. We look at our other obvious plots.

```
par(mfcol = c(2,1 ))
acf(discoveries, main="ACF of Number of Major Scientific Discoveries in a Year")
acf(discoveries, type="partial", main="PACF of Number of Major Scientific Discoveries in a Year")
```



2-3 tail off - noise - hard to tell

1,2 tail - hard to tell

try several competing models
(guess) identify order of p, q
↳ access model quality e.g. AIC
↳ choose model types

I'm looking at three spikes above noise in the *ACF* and one or two (feeling generous I'll say two-ish) on the *PACF*. We can explore several ARMA models and, in particular, assess the quality of the model with the AIC. I ran for all useful combinations of q = 0, 1, 2, 3 and p = 0, 1, 2, 3. Instead of looking at the total printout in each case, I'll just extract the default measure of quality, the AIC. There are automatic routines that will try to give you the order of the process as well as estimate the corresponding coefficients, and we'll explore one of these below. For now though, we are testing all reasonable candidate models by keeping our orders *p* and *q* fairly low and assessing quality of each model with the AIC. That is, *arima()* will perform our estimation after we tell it the order of the model, then the utility *AIC()* will give us the Akaike Information Criterion. There are of course other useful measures of quality, but this is one of the most popular.

We haven't talked about differencing (for stationarity) yet, but we'll explicitly tell the routine not to do any differencing (*d=0*). This means we will specify the order of the process as *(p,d,q)=(p,0,q)*.

```
AIC( arima( discoveries, order=c(0,0,1) ) )  #AIC =  [1] 445.5895
AIC( arima( discoveries, order=c(0,0,2) ) )  #AIC =  [1] 444.6742
AIC( arima( discoveries, order=c(0,0,3) ) )  #AIC =  [1] 441.323
AIC( arima( discoveries, order=c(1,0,0) ) )  #AIC =  [1] 443.3792
AIC( arima( discoveries, order=c(1,0,1) ) )  #AIC =  [1] 440.198
AIC( arima( discoveries, order=c(1,0,2) ) )  #AIC =  [1] 442.0428
AIC( arima( discoveries, order=c(1,0,3) ) )  #AIC =  [1] 442.6747
AIC( arima( discoveries, order=c(2,0,0) ) )  #AIC =  [1] 441.6155
AIC( arima( discoveries, order=c(2,0,1) ) )  #AIC =  [1] 442.0722
AIC( arima( discoveries, order=c(2,0,2) ) )  #AIC =  [1] 443.7021
AIC( arima( discoveries, order=c(2,0,3) ) )  #AIC =  [1] 441.6594
AIC( arima( discoveries, order=c(3,0,0) ) )  #AIC =  [1] 441.5658
AIC( arima( discoveries, order=c(3,0,1) ) )  #AIC =  [1] 443.5655
AIC( arima( discoveries, order=c(3,0,2) ) )  #AIC =  [1] 439.9263
AIC( arima( discoveries, order=c(3,0,3) ) )  #AIC =  [1] 441.2941
```

aic = whether extra complexity in model justified by small diff in AIC

There seem to be two strong contenders. Absent a theory, I prefer the *(p,d,q)=(1,0,1)* over the *(p,d,q)=(3,0,2)* on the basis of parsimony, but the AIC marginally likes the *(p,d,q)=(3,0,2)*. For fuller printout in the simple case:

```
arima( discoveries, order=c(1,0,1) )

Call:
arima(x = discoveries, order = c(1, 0, 1))

Coefficients:
        ar1       ma1       intercept
        0.8353    -0.6243   3.0208
s.e.    0.1379    0.1948    0.4728

sigma^2 estimated as 4.401:  log likelihood = -216.1,  aic = 440.2
```


Automatic Routines (Enjoy, but be careful)

There is a routine which will automate this process for us, appropriately named *auto.arima()*. If you haven't yet installed the *forecast* package, please do so. We haven't talked about differencing (for stationarity) yet, but we'll explicitly tell the routine not to do any differencing (*d=0*). We also haven't talked about methods for computing various quantities, but our routines will take certain liberties with longer time series and make approximations to speed up run times[1]. Our discoveries data set just sneaks in for this approximation, since it has length = 100. See if you follow the little logic puzzle that sets the flag to TRUE for this data set:

didn't see any evidence of trend

```
approximation = (length(x)>100 | frequency(x)>12)
```

We'll tell it not to make any approximations since the computational time here is really quite modest. Then we can make the calls:

```
library(forecast)
auto.arima(discoveries, d=0, approximation=FALSE)
```

Evidently, *auto.arima()* likes a *(p,d,q)=(2,0,0)* model when approximation is FALSE:

```
Series: discoveries
ARIMA(2,0,0) with non-zero mean

Coefficients:
        ar1       ar2       mean
        0.2251    0.1929    3.0877
s.e.    0.0985    0.0984    0.3594

sigma^2 estimated as 4.605:  log likelihood=-216.81
AIC=441.62          AICc=442.04          BIC=452.04
```

[1] *According to the help page: auto.arima() has a flag called "approximation", If it is set to TRUE, estimation is via conditional sums of squares and the information criteria used for model selection are approximated. The final model is still computed using maximum likelihood estimation. Approximation should be used for long time series or a high seasonal period to avoid excessive computation times.*

Take a moment and run the routine when approximation is set to TRUE:

```
Series: discoveries
ARIMA(1,0,1) with non-zero mean

Coefficients:
        ar1       ma1       mean
        0.8353    -0.6243   3.0208
s.e.    0.1379    0.1948    0.4728

sigma^2 estimated as 4.538:  log likelihood=-216.1
AIC=440.2           AICc=440.62          BIC=450.62
```

The routine changed its mind! We have a choice (especially for a fairly short time series) about whether or not to use the approximation flag and this choice may influence the model selection process.

Another choice is that there are a few different ways to specify the criterion used for model (order) selection in the routine *auto.arima()*. Just as the *SSE* might specify a different model as "best" when compared to the best under the *AIC* value, the three quality criteria used by *auto.arima()* can return different models as "best". The user is allowed to specify one of the following:

- *Bayesian Information Criterion (BIC),*
- *Akaike Information Criterion (AIC), or*
- *"corrected AIC" (AICC).*

We use the information criterion flag *ic=c("aicc", "aic", "bic")* to specify which one we want, though the corrected AIC (aicc) is the default. (That's why it's listed first in the concatenation.) This is what was used in the automatic selection above.

If we look for the best model under Baysian Information Criterion using *approximation* as FALSE we get the following:

```
auto.arima(discoveries, d=0, ic="bic", approximation=FALSE)
```



```
Series: discoveries
ARIMA(1,0,1) with non-zero mean

Coefficients:
        ar1       ma1       mean
        0.8353    -0.6243   3.0208
s.e.    0.1379    0.1948    0.4728

sigma^2 estimated as 4.538:  log likelihood=-216.1
AIC=440.2    AICc=440.62        BIC=450.62
```

And, if we look for the best under AIC we get the following, again using *approximation* as FALSE:

```
auto.arima(discoveries, d=0, ic="aic", approximation=FALSE)
Series: discoveries
ARIMA(3,0,0) with non-zero mean

Coefficients:
        ar1       ar2       ar3       mean
        0.1967    0.1613    0.1451    3.0637
s.e.    0.0995    0.0998    0.1007    0.4136

sigma^2 estimated as 4.556:  log likelihood=-215.78
AIC=441.57   AICc=442.2   BIC=454.59
```

Look for the best under the AIC criterion when *approximation* is TRUE. Consistent with the *arima()* calculation we performed above, we get the following:

```
auto.arima(discoveries, d=0, ic="aic", approximation=TRUE)
Series: discoveries
ARIMA(1,0,1) with non-zero mean

Coefficients:
        ar1       ma1       mean
        0.8353    -0.6243   3.0208
s.e.    0.1379    0.1948    0.4728

sigma^2 estimated as 4.538:  log likelihood=-216.1
AIC=440.2   AICc=440.62   BIC=450.62
```