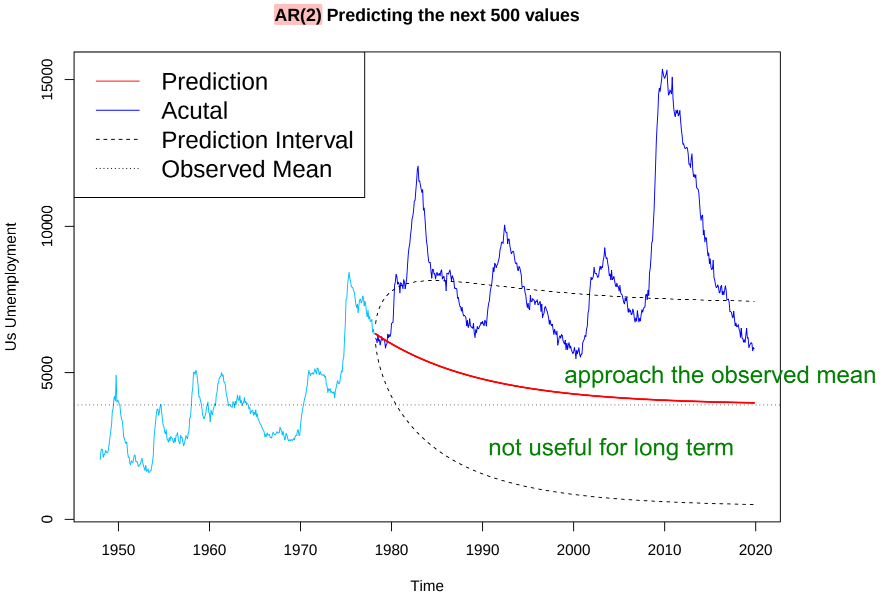
VE472 Lecture 11

Jing Liu

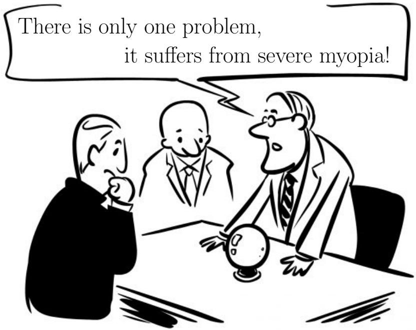
UM-SJTU Joint Institute

Summer



AR(2) Predicting the next 500 values
Prediction
Acutal
Prediction Interval
Observed Mean
approach the observed mean
not useful for long term
Us Umemployment
Time
0
5000
10000
15000
1950
1960
1970
1980
1990
2000
2010
2020

- Notice that the prediction interval round the predictions widen rapidly and the predictions themselves seem to be tending to a constant value.
- Over the long term, predictions for stationary series ultimately converge to the mean of the series and the standard errors for the forecasts tend to the standard deviation of the series.
- This means we can only expect our model to be better than simply using the mean when comes to short-term forecasts.



- We will have to live with the fact that we can only do short-term forecasts.
- However, there is one aspect, we will have to fix, namely, many time series exhibit strong seasonal characteristics. We have no such thing in ARIMA.

no seasonal part handling

- Let s denote the seasonal period, that is,
  - for monthly series, $s = 12$,
  - for quarterly series $s = 4$,
  - for daily series $s = 365$ or $s = 7$,
  - for hourly series $s = 24$, etc.
- Seasonal effects can be modelled by inducing coefficients at lags which are multiples of the seasonal period, For example, we could introduce

new coefficients

$$\Phi_1 Y_{t-s} + \Phi_2 Y_{t-2s} + \cdots + \Phi_P Y_{t-Ps} + \Theta_1 \varepsilon_{t-s} + \Theta_2 \varepsilon_{t-2s} + \cdots + \Theta_Q \varepsilon_{t-\cancel{Ps}}$$

一个season之前的、两个season之前的...

Qs

to the right hand side of ARMA

$$Y_t = \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

- It can be shown the polynomial lag operators in ARMA can be factored into products linear operators, i.e.

$$\phi(\mathcal{L})Y_t = \theta(\mathcal{L})\varepsilon_t$$
$$(1+a_1\mathcal{L})(1+a_2\mathcal{L})\cdots(1+a_p\mathcal{L})\,Y_t = (1+b_1\mathcal{L})(1+b_2\mathcal{L})\cdots(1+b_p\mathcal{L})\,\varepsilon_t$$

  and the order of those operators can be freely rearranged like ordinary factors
- Hence introducing the following terms

$$\Phi_1 Y_{t-s} + \Phi_2 Y_{t-2s} + \cdots + \Phi_P Y_{t-Ps} + \Theta_1 \varepsilon_{t-s} + \Theta_2 \varepsilon_{t-2s} + \cdots + \Theta_Q \varepsilon_{t-Ps}$$

  to the right hand side of ARMA

$$Y_t = \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

  merely introducing additional polynomial operators of the forms

$$(1 + A\mathcal{L}^s) \qquad \text{and} \qquad (1 + B\mathcal{L}^s)$$

  to the autoregressive operator $\phi(\mathcal{L})$ and the moving average operator $\theta(\mathcal{L})$.

- In general, we have the following operator formulation of a seasonal model

$$\Phi(\mathcal{L}^s)\phi(\mathcal{L})Y_t = \Theta(\mathcal{L}^s)\theta(\mathcal{L})\varepsilon_t$$

where

$$\Phi(\mathcal{L}^s) = (1 + A_1\mathcal{L}^s)(1 + A_1\mathcal{L}^s)\cdots(1 + A_P\mathcal{L}^s)$$
$$\Theta(\mathcal{L}^s) = (1 + B_1\mathcal{L}^s)(1 + B_1\mathcal{L}^s)\cdots(1 + B_Q\mathcal{L}^s)$$

which is denoted as ARMA $(p, q) \times (P, Q)_s$.

- The non-stationarity can be accommodated by introducing operators

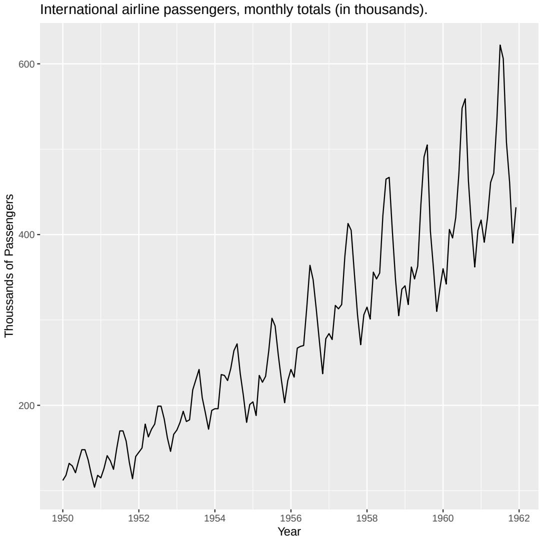
$$\nabla^d = (1 - \mathcal{L})^d \quad \text{and} \quad \nabla_s^D = (1 - \mathcal{L}^s)^d$$ notes

which leads to the following general formulation

$$\Phi(\mathcal{L}^s)\phi(\mathcal{L})\nabla^d\nabla_s^D Y_t = \Theta(\mathcal{L}^s)\theta(\mathcal{L})\varepsilon_t$$

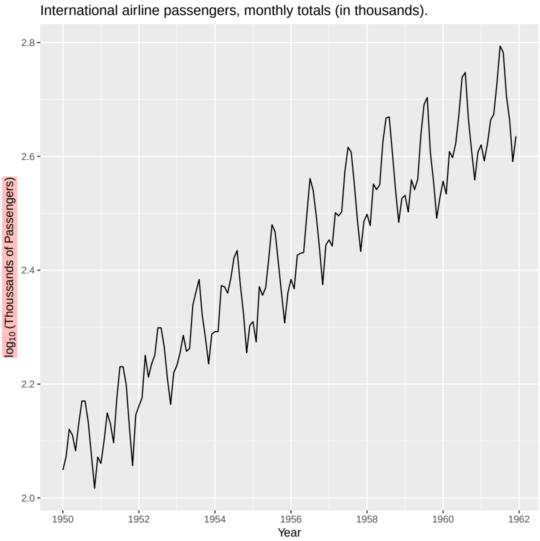
- Such a model is denoted as ARIMA $(p, d, q) \times (P, D, Q)_s$.

Be aware whether the additive assumption is reasonable



variebility goes up with time. multiplication series

Log transformation allows ARIMA for multiplicative series



stablize

```
> # Suppose we have used auto.arima
> # or cross-validation to find the following
> (z.arima = arima(log(airpass.ts, 10),
+                  order = c(0, 1, 1),
+                  seasonal = c(0, 1, 1)))
```

```
Coefficients:
          ma1      sma1
      -0.4018   -0.5569
s.e.   0.0896    0.0731
```

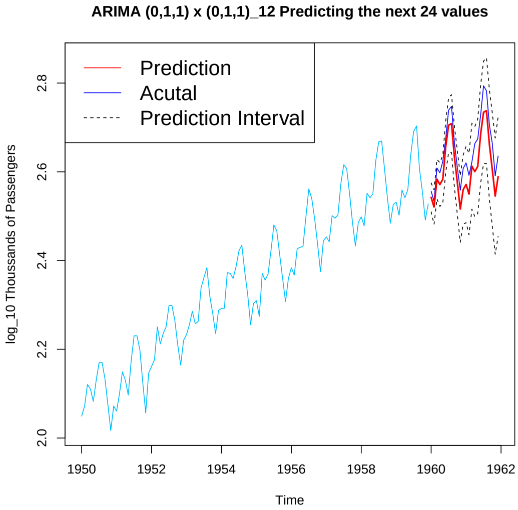
1 prediction (next y value)

```
> predict(z.arima, n.ahead = 1)
```

```
$pred
         Jan
1962 2.65362

$se
            Jan
1962 0.01594539
```

Leave out the last two years for testing prediction



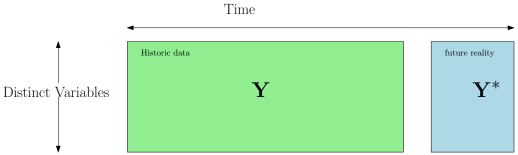
prediction involves season now

- After seeing some basic models in time series, we turn our attention back to

$$
\begin{array}{c}
Y_{11}, Y_{12} \cdots\cdots Y_{1T_1} \\
Y_{21}, Y_{22} \cdots\cdots Y_{2T_2} \\
\vdots \quad \ddots \ddots \ddots \quad \vdots \\
Y_{n1}, Y_{n2} \cdots\cdots Y_{nT_n}
\end{array}
\qquad \text{multiple time series}
$$

where n, T_1, T_2, ..., T_n could all be very large.

- Suppose $T_i = T$ for all i, so we can arrange our data into a matrix $\mathbf{Y}$



where each row stores a time series, and each column denotes a time point.

- A simple way to deal with a very large n and T is to assume distinct series are independent, and focusing on analysing one time series at a time.
- Unless we have audio or signal data, T is usually only moderately large, i.e.

```
> (n = ((1 + 4 * 365) * 24 ) * 100)
```

400years of hourly data

```
[1] 3506400
```

modern computer will have no problem to handle it using seasonal `ARIMA`

```
> ts_sim = arima.sim(list(ar = c(0.8897, -0.4858),
+                         ma = c(-0.2279, 0.2488)),
+                    n = n, sd = sqrt(0.1796))
> pryr::object_size(ts_sim)
```

```
28.1 MB
```

```
> system.time({fit=arima(ts_sim,order = c(2,0,2))})
```

```
  user  system elapsed
32.168   3.888  36.072
```

- In fact, we often should stop using the whole time series much earlier than the computational cost or memory consumption becomes an issue.

```
> summary(fit) # The whole time series
```

```
Coefficients:
         ar1      ar2      ma1     ma2  intercept
      0.8906  -0.4859  -0.2299  0.2486      0e+00
s.e.  0.0018   0.0010   0.0018  0.0010      4e-04
```

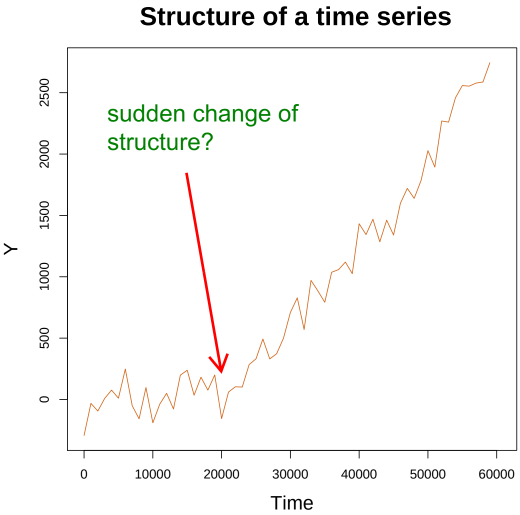
用最后5e5个

```
> summary(arima(tail(ts_sim, 5e5), order=c(2,0,2)))
```

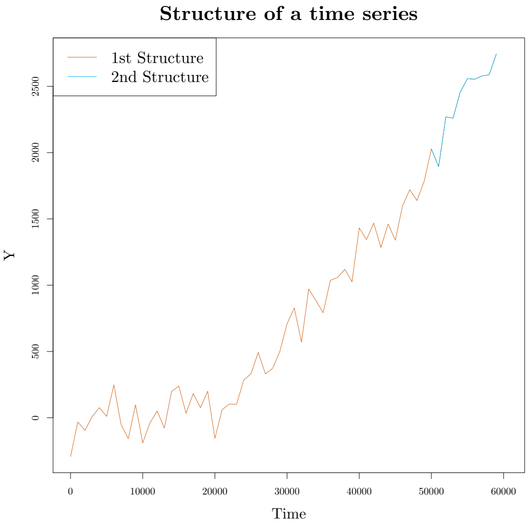
```
Coefficients:
         ar1      ar2      ma1     ma2  intercept
      0.8868  -0.4841  -0.2277  0.2484      2e-04
s.e.  0.0047   0.0025   0.0048  0.0026      1e-03
```

- Because rarely any model can stand the test of time for very long in practice!

Any special about this time series?



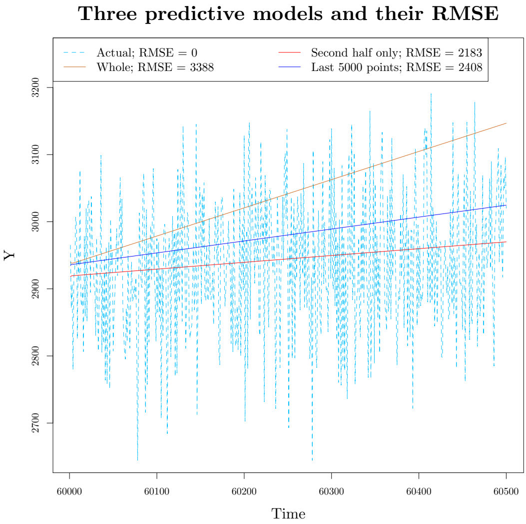
- You might think the breakdown happened around 20000, but in fact it didn't



- The data was simulated using the following

```
> z1.n = 50000
> z2.n = 10000
> pred.n = 500 # for prediction
> index = seq(from = 1, length.out =
+               z1.n+z2.n+pred.n, by = 1)
> # Quadratic deterministic trend
> z0_trend = 0.000001*(4000-index)*(8000-index)
> # First ARIMA
> z1_sim = arima.sim(
+   list(ar = c(0.69,-0.4,-0.2), ma = c(-0.2,0.3)),
+   n = z1.n , sd = sqrt(10000))
> # Second ARIMA
> z2_sim = arima.sim(
+   list(ar = c(-0.25), ma = c(0.2)),
+   n = z2.n + pred.n, sd = sqrt(10000))
> structure_break_ts = ts(
+   z0_trend + c(z1_sim, z2_sim))
```

- Notice using more data is not necessarily better in time series!



- Over a long period of time, the structure of a time series in practice is very unlikely to remain fixed, so using a model like ARMA may not be optimal.
- If the data span many many years, then the structure of the time series may significantly change. If we assume it is changing slowing, then constructing `ARIMA` using only data on the most recent years may greatly reduces errors.

prediction error

```
> last.n = 5000
> index.last = tail((z1.n+1):(z1.n+z2.n), last.n)
>
> sim.last.arima = arima(
+   structure_break_ts[index.last],
+   order = c(2, 2, 2))
>
> pred.last = predict(sim.last.arima,
+                     n.ahead = pred.n)
```

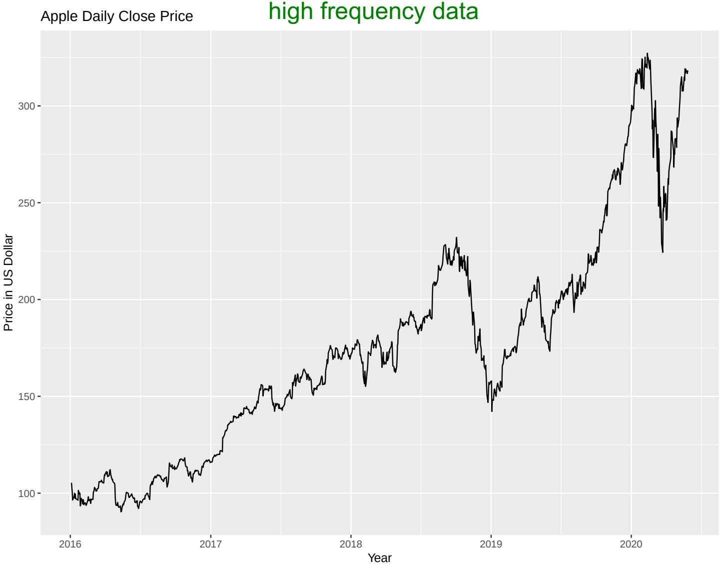
not the true model

Q: This may seem crude and ill-advised, but have you seen something similar?

trade off between bias and consistency

- Now if T is big but the data is actually over a relatively short period of time,

high frequency data



- `ARIMA` $(p,d,q)\times(P,D,Q)_s$ is inadequate for such a high frequency dataset

```
> AAPL.ts = ts(AAPL$AAPL.Close, frequency=365.25)
```

and it is often unclear what s value to use. In those cases, Fourier terms

$$Y_t = \sum_{j=1}^{k}\left(\alpha_j \sin\left(\frac{2\pi jt}{356.25}\right) + \beta_j \cos\left(\frac{2\pi jt}{356.25}\right)\right) + Z_t$$

provides an alternative way to model the seasonality, where Z_t is ARIMA.

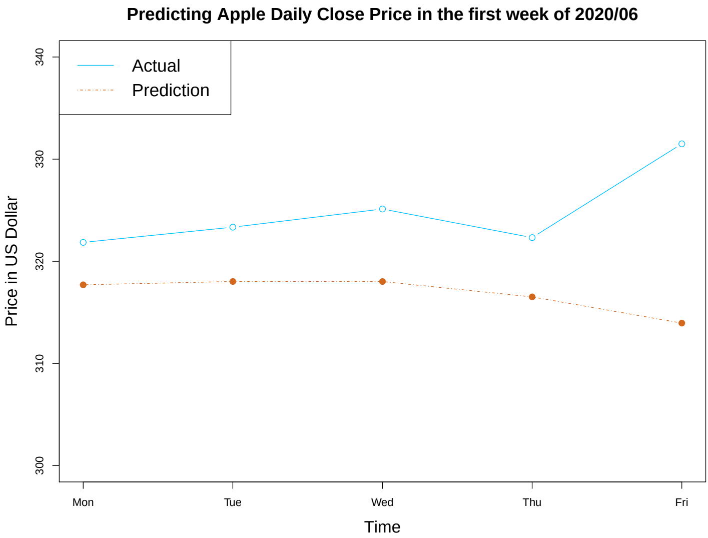
```
> f = fourier(AAPL.ts, K=100) # using Fourier freq

> AAPL.arima = auto.arima(
+   AAPL.ts, xreg = f, seasonal = FALSE)

> h = 5 # predicting the next 5 trading days
> fh = fourier(AAPL.ts, K=100, h = h)
> pred = forecast(AAPL.arima,
+                 xreg = fh, h = h)
```

stock price not seasonal

No surprise here!



- Now in terms of extremely long and extremly high frequency dataset,

```
> signal = tuneR::readWave(
+   filename = "~/Desktop/comet_rock.wav")
> signal
```

very large sampling rate

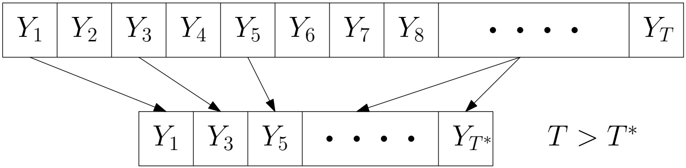
```
Wave Object
        Number of Samples:        16342384
        Duration (seconds):       113.49
        Samplingrate (Hertz):     144000
        Channels (Mono/Stereo):   Mono
        PCM (integer format):     TRUE
        Bit (8/16/24/32/64):      16
```

```
> pryr::object_size(signal)
```

```
65.4 MB
```

- If we duration is anything longer than a few hours, then we have no option but to pre-processing our data rather than using the most recent data points!

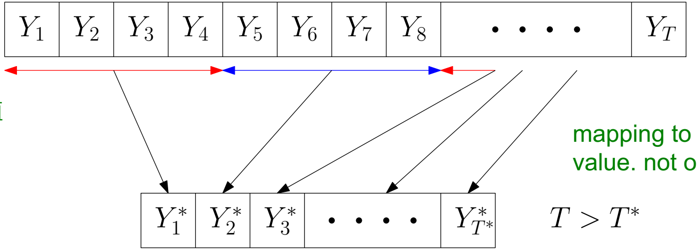
- A simple way is to thinning, taking 1 out of every k consecutive ones



$T > T^*$

or collapsing by using a function that takes k values and output 1 value

eg: 用平均值

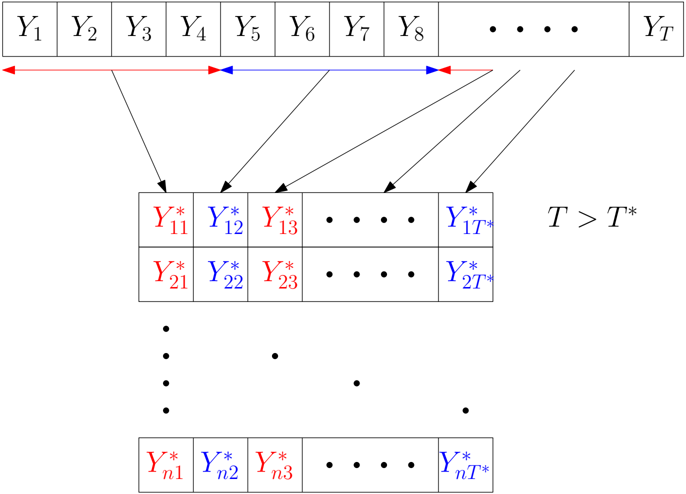


mapping to new value. not original.

$T > T^*$

- Of course, if k is too big, then we lose too much information.

- One way is to collapse k values into n values, i.e.



which bring us to back to how to handle a large number of time series!