ARMA Model

(Jing Li, Miami University)

Big Picture

1. ARMA model is parametric, and is widely used for forecasting
2. AR stands for autoregressive
3. MA stands for moving average
4. ARMA model is appropriate when time series is stationary (choppy, mean-reverting, no trend)
5. When series is nonstationary (smooth, trending), we apply ARMA after taking difference. The model is called ARIMA
6. ARMA model is based on idea of “history repeats itself”, and cannot predict black swan

White Noise

1. Building block of ARMA is white noise
2. White noise is a time series showing no pattern or memory
3. White noise is unpredictable
4. White noise is useful for two reasons
   (a) White noise can be used to construct predictable AR or MA process
   (b) A time series model is adequate (good enough) if residual is white noise

Simulate white noise

We use the following R codes to generate a white noise series with 100 observations

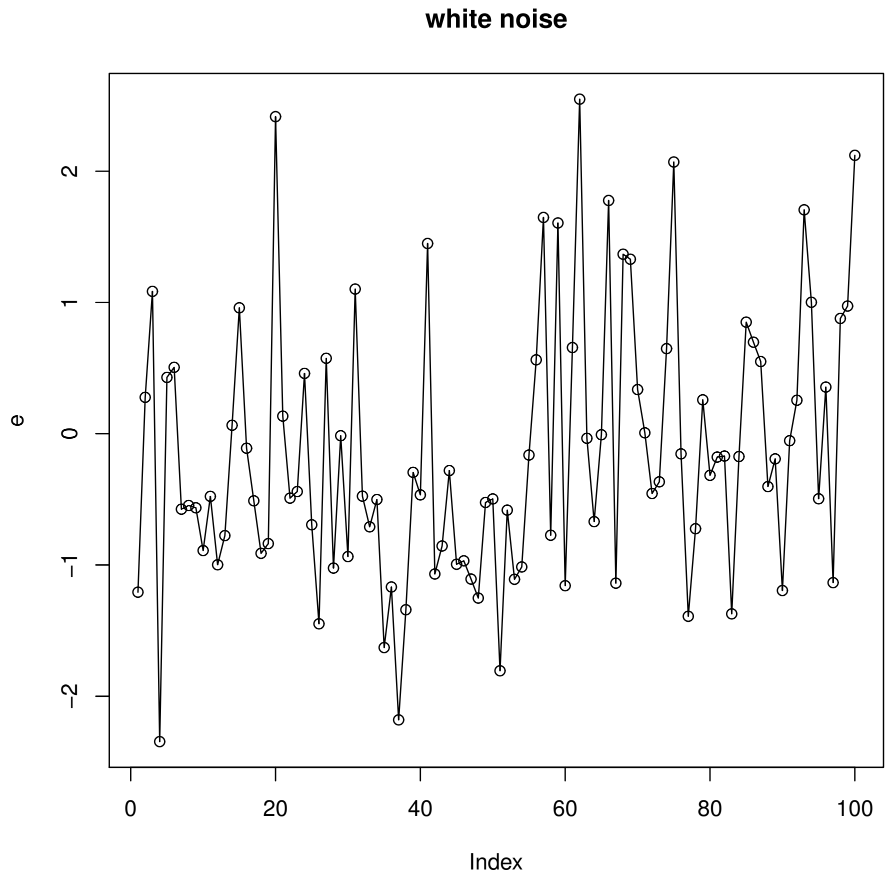
```
set.seed(1234)
n = 100
e = rnorm(n)
plot(e, type="o", main="white noise")
```

Remarks

1. 1234 is the seed number for random number generator
2. *n* is the total number of observations (sample size)
3. *e* is white noise (i.i.d random variable that follows standard normal distribution). First five observations are

   ```
   > e[1:5]
   [1] -1.2070657  0.2774292  1.0844412 -2.3456977  0.4291247
   ```

Plot of white noise



Remarks

1. The white nose is choppy (not smooth)
2. The series fluctuates around the mean value 0 (mean-reverting)
3. There is no trend

MA process

Let e denote white noise. Then we can use e to construct the first order moving average or MA(1) process as

$$y_t = e_t + \theta e_{t-1} \quad (t = 2, 3, ...n) \tag{1}$$

where the first observation is $y_1 = e_1$.

1. Since e_{t-1} appears on the right hand side for both y_t and y_{t-1}, y and its first lag are correlated, and the correlation is governed by θ.

2. By contrast y_t and y_{t-2} do not share common term on the right hand side, so y and its second lag are uncorrelated

3. In short, MA(1) process has short memory, and the memory disappears beyond two periods

4. MA(1) process is predictable due to that memory

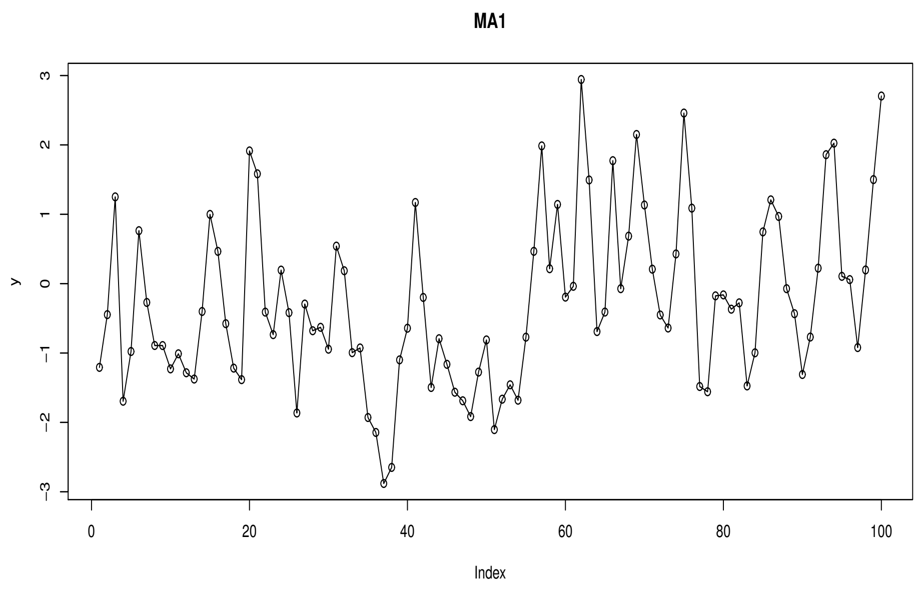
5. Intuitively, MA(1) process has memory (and can be predicted) because it contains lag value (so the past matters)

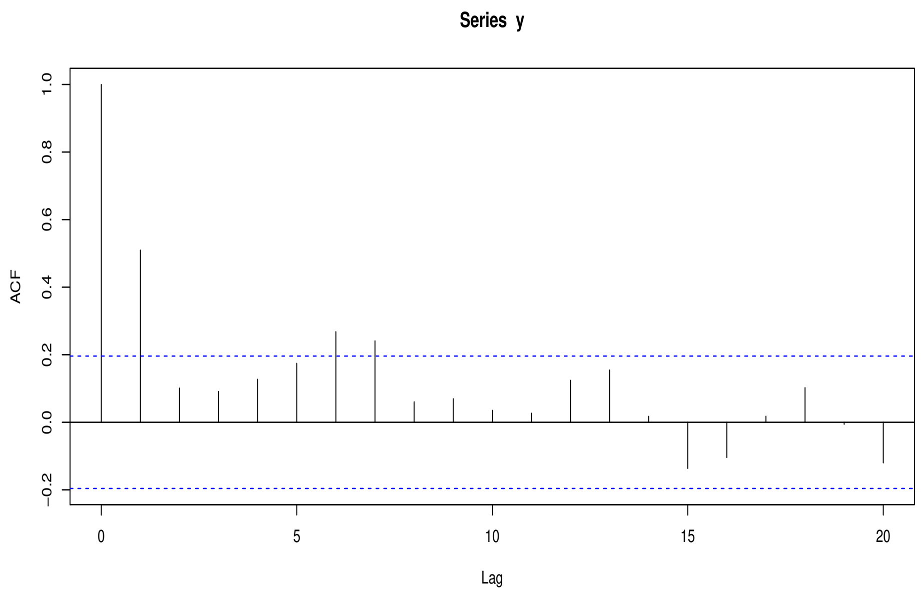
Simulate MA(1)

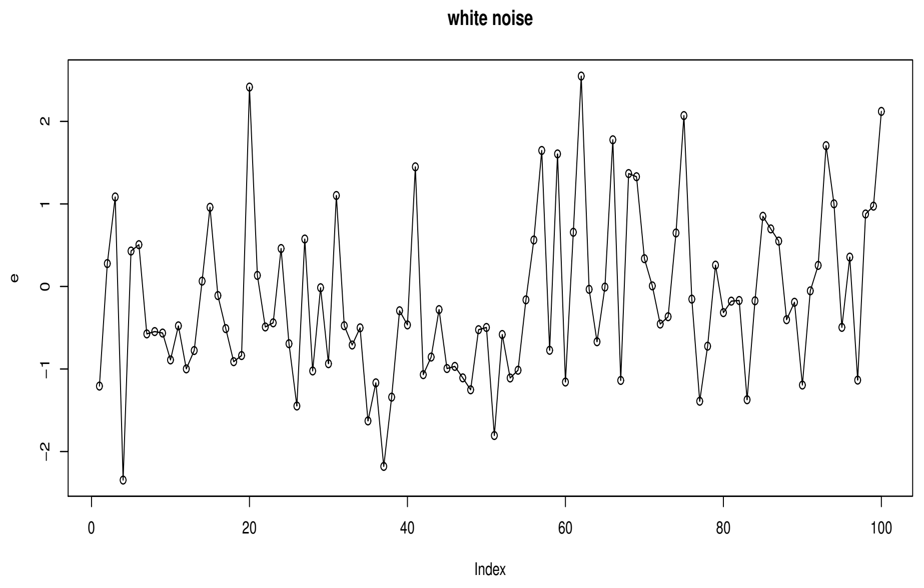
We simulate an MA(1) process with $\theta = 0.6$ and compare it to white noise

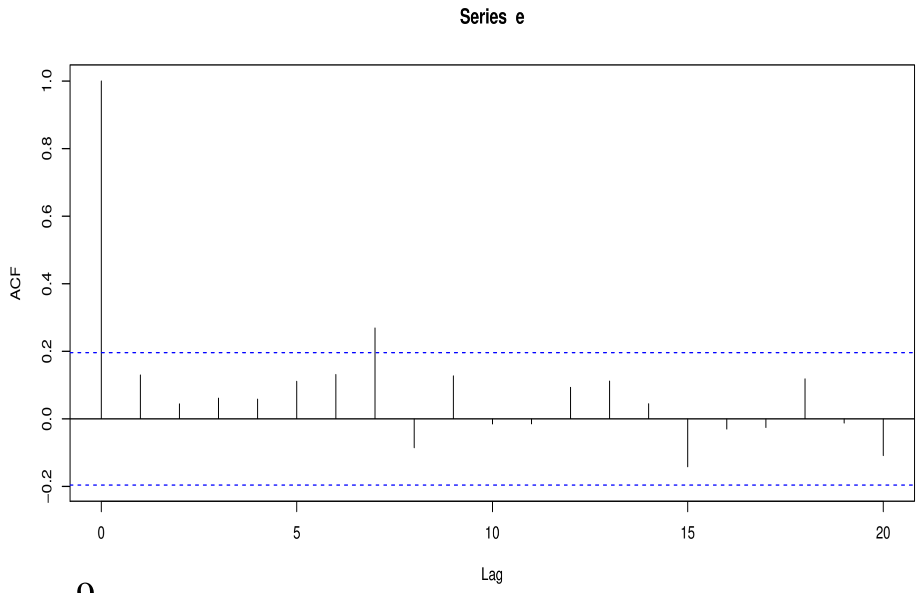
```
theta = 0.6
y = rep(0,n)
y[1] = e[1]
for (t in 2:n) {y[t]=e[t]+theta*e[t-1]}

par(mfrow=c(2,2))
plot(y, type="o", main="MA1")
acf(y)
plot(e, type="o", main="white noise")
acf(e)
```

Compare MA and White Noise









Remarks

1. The MA process is smoother than white noise. The smoothness or persistence is due to memory
2. We use auto-correlation function (ACF) or correlogram to measure the memory. We see statistically significant (i.e., exceeding the blue confidence interval) acf at lag 1 for MA(1), but insignificant acf at lag 1 for white noise
3. In theory, acf at all lags should be insignificant for white noise. Nevertheless, thanks to sampling noise, some acf may be marginally significant

Estimation

1. θ is unknown for real-life (not simulated) data
2. The R **arima** function can estimate θ

```
> arima(y, order = c(0,0,1))
Coefficients:
         ma1  intercept
      0.7295    -0.2453
s.e.  0.1070     0.1706
sigma^2 estimated as 0.9798:  log likelihood = -141.25,  aic = 288.51
```

Note that the estimated $\hat{\theta} = 0.7295$ is close to the true value 0.6, and is statistically significant. The estimate and true value differ because of sampling error

Exercise

1. Please modify the R codes to generate a second order MA or MA(2) process given by

$$y_t = e_t + 0.6e_{t-1} + 0.4e_{t-2} \quad (t = 3, 4, \ldots n) \tag{2}$$

   with $y_1 = e_1, y_2 = e_2$. Please plot the series and its ACF. Comment on the difference you see relative to the MA(1) process

2. Please estimate the MA(1) model for the white noise process, and comment on the finding

AR process

If the data is daily, then MA(1) model implies that today's value only depends on yesterday. What happened two or three days ago does not matter. This kind of short memory can be unrealistic. That is the motivation for an autoregressive or AR model, which allows for long memory. Consider the first order autoregressive or AR(1) process

$$y_t = \phi y_{t-1} + e_t \quad (t = 2, 3, ...n) \tag{3}$$

with $y_1 = e_1$. It is obvious that y_t and y_{t-1} are correlated. Since y_{t-1} and y_{t-2} are correlated, y_t and y_{t-2} are correlated as well. Actually we can show y_t and y_{t-j} are correlated no matter how large the lag j is. In that sense, the AR process has long memory.

Intuition

The intuition behind (3) is

$$present = past + shock$$

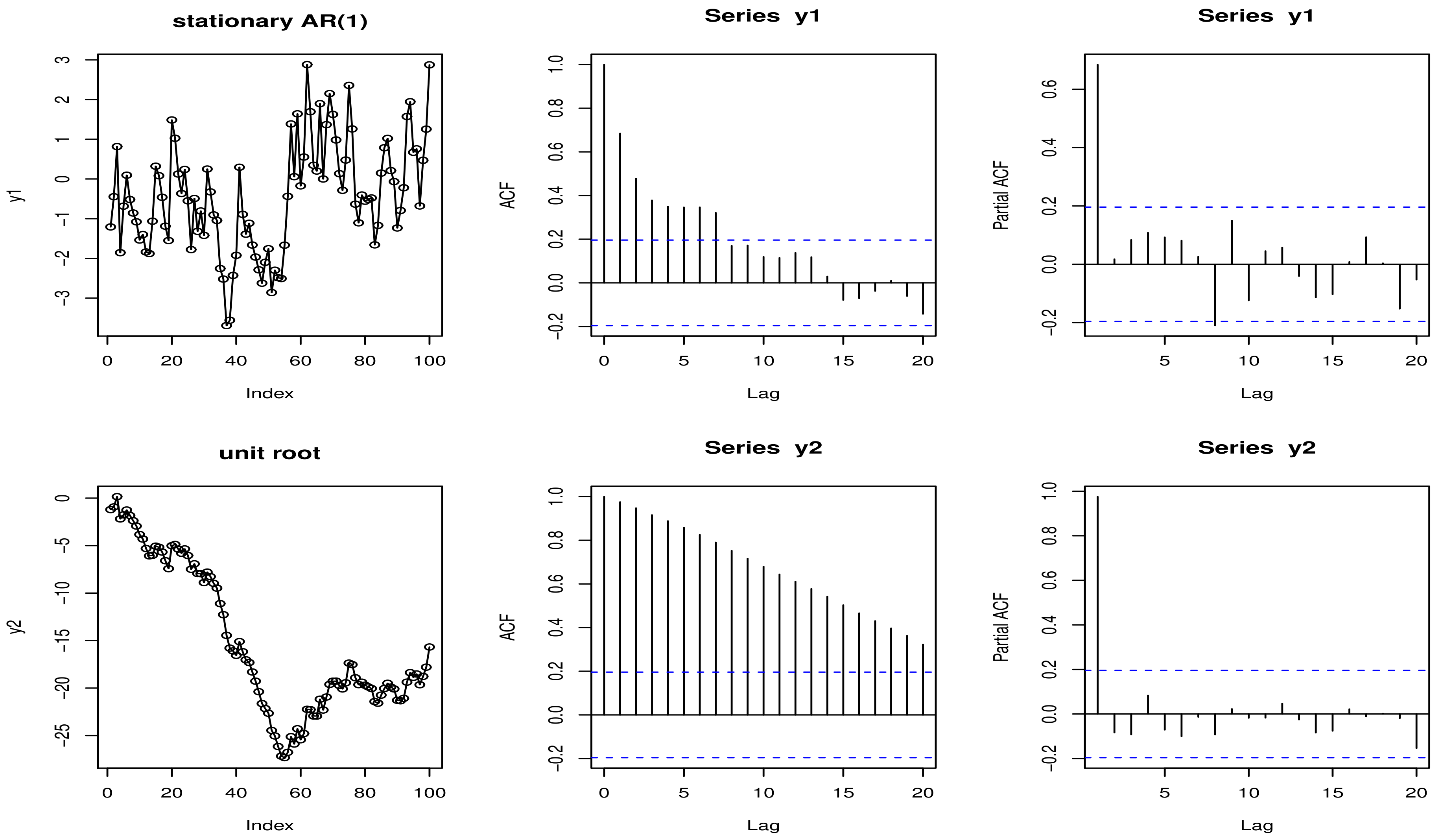
and the autoregressive coefficient ϕ determines the degree to which the past and the present are linked

Simulate AR(1)

We simulate two AR(1) processes with $\phi = 0.6$ and $\phi = 1$

```
y = rep(0,n)
y[1] = e[1]
for (t in 2:n) {y[t]=0.6*y[t-1]+e[t]}
y1 = y
for (t in 2:n) {y[t]=1*y[t-1]+e[t]}
y2 = y

par(mfrow=c(2,2))
plot(y1, type="o", main="stationary AR(1)")
acf(y1)
plot(y2, type="o", main="unit root")
acf(y2)
```

Plot of Stationary AR and Nonstationary Unit Root



Remarks

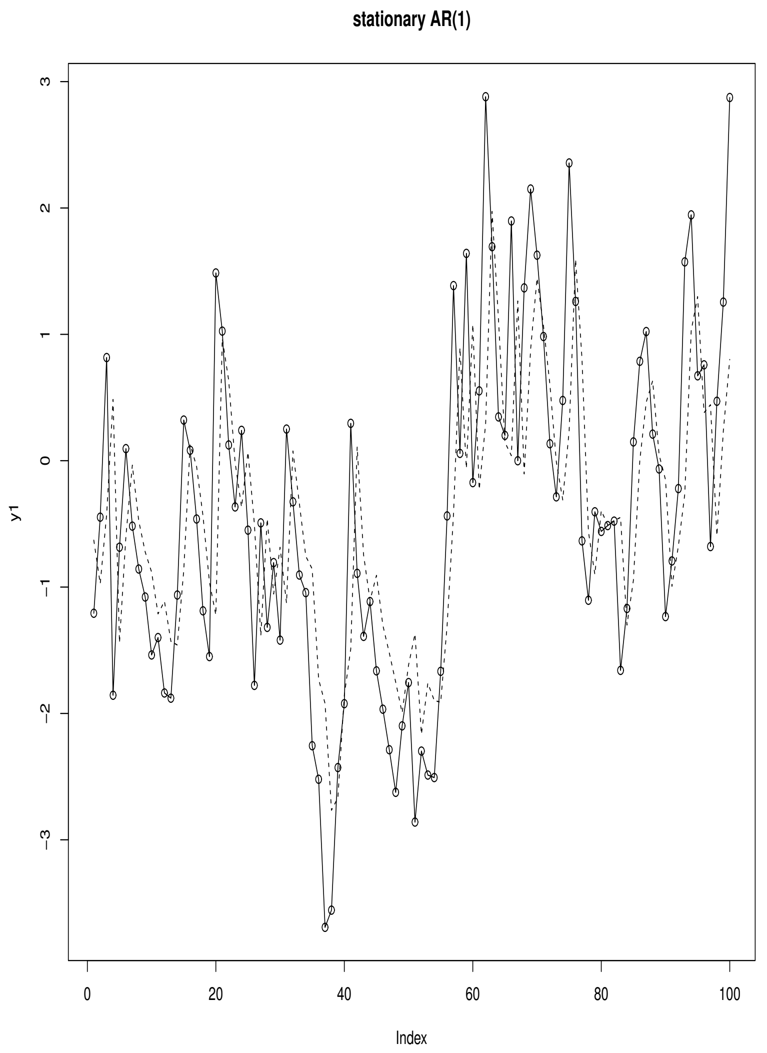
1. The AR process is even smoother than MA
2. The long memory is indicated by the ACF
3. ACF becomes slow-decaying when $\phi = 1$
4. The significant partial autocorrelation at lag 1 in PACF indicates that AR(1) model can be a candidate model

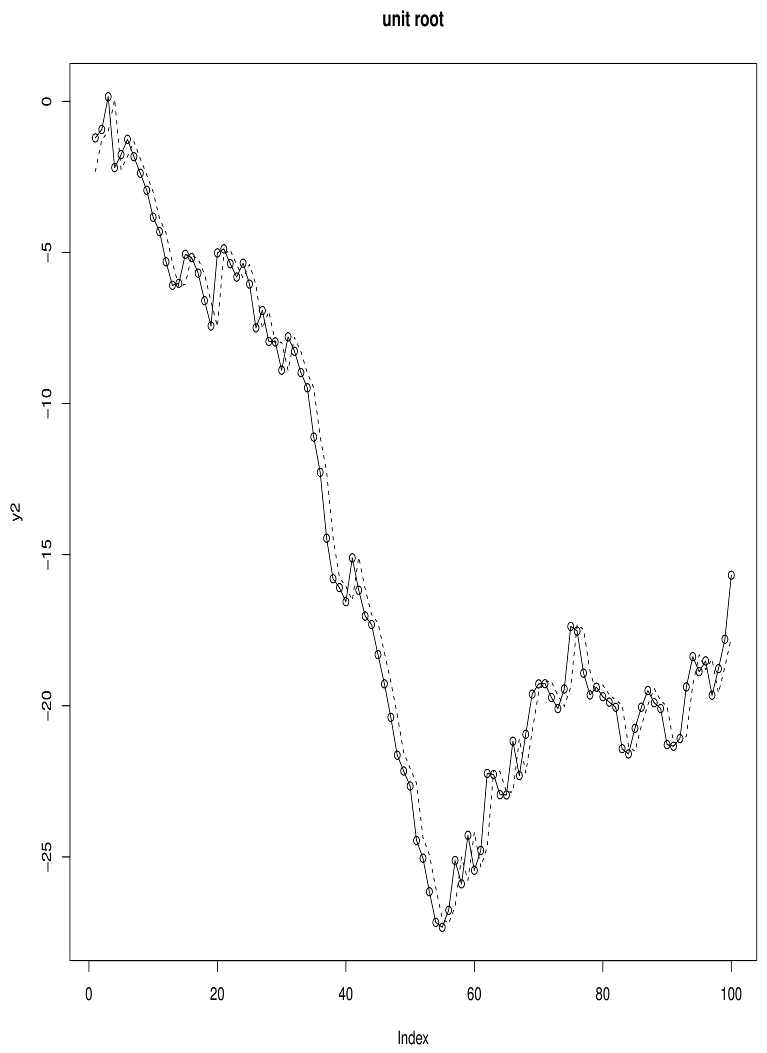
Estimating AR(1) process

```
> arima(y1, order = c(1,0,0))
Coefficients:
         ar1  intercept
      0.7208    -0.3727
s.e.  0.0717     0.3433

sigma^2 estimated as 0.9621:  log likelihood = -140.33,  aic = 286.66

> arima(y2, order = c(1,0,0))
Coefficients:
         ar1  intercept
      0.9925   -10.2736
s.e.  0.0078     7.1799

sigma^2 estimated as 1.014:  log likelihood = -144.7,  aic = 295.4
```

The estimated coefficients 0.7208 and 0.9925 are closed to true values of 0.6 and 1

Plot of y and predicted value





Checking model adequacy—a model is adequate if residual is white noise (or p value of Box-Ljung test is greater than 0.05)

```
> Box.test(arima(y1, order = c(1,0,0))$res, lag = 1, type = "Ljung")
Box-Ljung test
data:  arima(y1, order = c(1, 0, 0))$res
X-squared = 0.00079897, df = 1, p-value = 0.9774

> Box.test(arima(y2, order = c(1,0,0))$res, lag = 1, type = "Ljung")
Box-Ljung test
data:  arima(y2, order = c(1, 0, 0))$res
X-squared = 1.7537, df = 1, p-value = 0.1854
```

Here AR(1) model is good enough since p-value 0.9774 and 0.1854 are both greater than 0.05. The residual is white noise (unpredictable), so there is no need to find refinement. We may try AR(2), AR(3)...if the p-value is less than 0.05.

Exercise

1. Simulate an AR(2) process with 100 observation

$$y_t = 1.1y_{t-1} - 0.24y_{t-2} + e_t, \quad (t - 3, 4, ...100)$$

   where $y_1 = e_1, y_2 = e_2$

2. Plot the series, ACF and PACF

3. Estimate an AR(2) model and apply Box-Ljung test to residual. Is the model adequate?

4. Estimate an AR(1) model and apply Box-Ljung test to residual. Is the model adequate?

Unit root process

The AR(1) process with $\phi = 1$ is called unit root process (random walk)

$$y_t = y_{t-1} + e_t \tag{4}$$

Unit root process is an example of nonstationary time series. We can use iteration to show that

$$y_t = y_0 + e_t + e_{t-1} + \ldots + e_1 \tag{5}$$

and the variance of y_t is

$$y_t = t\sigma_e^2 \tag{6}$$

The series is nonstationary because the variance depends on t (not constant). If a cluster of shocks take values of the same sign, then a local trend can merge

Visual signal for nonstationarity

The plot of unit root process indicates three signals for nonstationarity

1. The series is smooth
2. The series has slow-decaying ACF
3. The series may be trending (sometimes locally, sometime globally)

Plotting time series before statistical analysis. Consider taking difference if there are signals for nonstationarity

Formally, we can conduct the augmented Dickey-Fuller (ADF) unit root test for the null hypothesis that there is unit root (the series is nonstationary). P-value less than 0.05 leads to rejection

Apply ADF test to stationary AR(1)

```
> library(tseries)
> adf.test(y1, alternative = c("s"), k = 1)

Augmented Dickey-Fuller Test


data:  y1
Dickey-Fuller = -3.895, Lag order = 1, p-value = 0.0171
alternative hypothesis: stationary
```

We find the series is stationary since p value 0.0171 is less than 0.05 (unit root hypothesis is rejected)

Apply ADF test to unit root process

```
> adf.test(y2, alternative = c("s"), k = 1)

Augmented Dickey-Fuller Test


data:  y2
Dickey-Fuller = -0.19324, Lag order = 1, p-value = 0.99
alternative hypothesis: stationary
```

We find the unit root is nonstationary since p value 0.99 is greater than 0.05 (unit root hypothesis is not rejected)

We can apply ARMA model only for stationary series (with no unit root). For nonstationary series (with unit roots) we need to take difference (possibly several times) before we apply the ARMA model. Unit root test can determine whether the series is stationary (having no unit root) or nonstationary (having unit roots)

We may try several ARIMA models, pick the one that returns smallest AIC, and test adequacy for the chosen model

ARIMA(p,d,q) model

1. Example 1: $p = 1, d = 0, q = 0$: the series is stationary AR(1)

$$(y_t - drift) = \phi(y_{t-1} - drift) + e_t \tag{7}$$

2. Example 2: $p = 0, d = 0, q = 1$: the series is stationary MA(1)

$$(y_t - drift) = e_t + \theta e_{t-1} \tag{8}$$

3. Example 3: $p = 1, d = 1, q = 0$: the series is integrated AR(1)

$$(y_t - y_{t-1}) = \phi(y_{t-1} - y_{t-2}) + e_t \tag{9}$$

4. Example 4: $p = 0, d = 1, q = 1$: the series is integrated MA(1)

$$(y_t - y_{t-1}) = e_t + \theta e_{t-1} \tag{10}$$

Example: forecasting AR(2) model

Consider an AR(2) model. The key is to recall that the best forecast is conditional mean

$$y_{t+1} = \phi_1 y_t + \phi_2 y_{t-1} + e_{t+1} \tag{11}$$

$$\hat{y}_{t+1} = E(y_{t+1}|\Omega_t) = E(\phi_1 y_t + \phi_2 y_{t-1} + e_{t+1}|\Omega_t) = \phi_1 y_t + \phi_2 y_{t-1} \tag{12}$$

$$y_{t+2} = \phi_1 y_{t+1} + \phi_2 y_t + e_{t+2} \tag{13}$$

$$\hat{y}_{t+2} = E(y_{t+2}|\Omega_t) = E(\phi_1 y_{t+1} + \phi_2 y_t + e_{t+2}|\Omega_t) = \phi_1 \hat{y}_{t+1} + \phi_2 y_t \tag{14}$$

$$\hat{y}_{t+j} = \phi_1 \hat{y}_{t+j-1} + \phi_2 \hat{y}_{t+j-2}, \quad (j = 3, 4, \ldots) \tag{15}$$

where $\Omega_t = (y_t, y_{t-1}, \ldots, y_1)$ is the information set. The R **forecast** package can be used to generate those forecasts.

Smoothing Method

(Jing Li, Miami University)

Smoothing (filtering) method

1. Smoothing method is model-free, so non-parametric
2. Smoothing is essentially averaging, and weights can be equal or unequal
3. Two-sided smoothing can approximate trend, or, extract signal from noise
4. One-sided smoothing can be used as forecast

Optional math for averaging

Let σ^2 be the variance of one observation. The variance of an average of many independent observations is

$$var(\texttt{average}) = var\left(\frac{y_1 + y_2 \ldots + y_n}{n}\right) = \frac{\sigma^2}{n} < \sigma^2 \tag{16}$$

Decreasing variance implies <u>smoothness</u>. Thus, the average or filtered series is smoother than original series

Averaging can highlight trend or extract signal if we assume they are smooth

Two-sided $k-$average

Let k be an odd number, and $s = \frac{k-1}{2}$ be the one-side length. The two-sided $k-$average is computed as

$$\bar{y}_{t,k} = \frac{y_{t-s} \ldots + y_{t-1} + y_t + y_{t+1} + \ldots + y_{t+s}}{k} \tag{17}$$

For instance, let $k = 5, t = 3$. Then the $5-$average at the third observation is

$$\bar{y}_{3,5} = \frac{y_1 + y_2 + y_3 + y_4 + y_5}{5} \tag{18}$$

1. Basically we average the values before and after y_3
2. Note that $\bar{y}_{1,5}$ and $\bar{y}_{2,5}$ are not available (missing values are denoted by NA in R) since there are no data for y_0 and y_{-1}. Similarly, NA appears in the end of average data
3. Two-sided average is unsuitable for forecasting

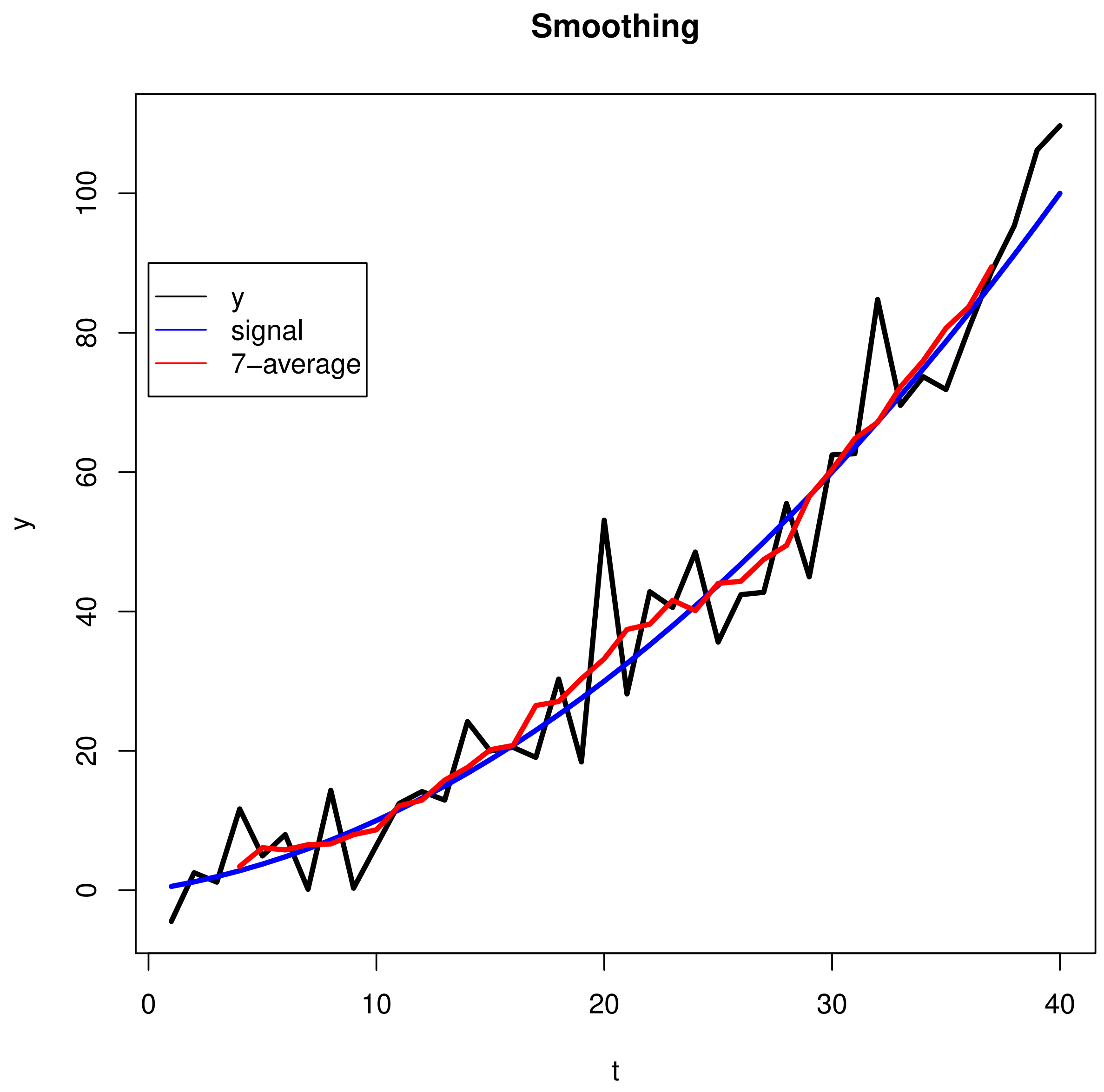
User-defined fma function

```
fma = function(data, k) {
n = length(data)
oneside = (k-1)/2
fs = rep(NA, n)
for (t in (1+oneside):(n-oneside)) {
fs[t] = mean(data[(t-oneside):(t+oneside)])
}
return(fs)
}
```

Remarks

1. The first input is name of time series that we want to average
2. The second input is k, which has to be odd
3. The output is two-sided $k-$ average
4. Missing values (NA) are in the two ends of average series

Extracting signal (trend) with 7−average



Remarks

1. To illustrate $k-$average we use simulation and let

$$y = signal + noise \tag{19}$$

2. Signal is a smooth quadratic function of time
3. Noise is choppy and unpredictable
4. The previous graph plots the y series, the signal, and $7-$ average
5. For this example, the $7-$ average does a good job of extracting signal (trend)
6. For real-life data the blue line is unknown. The red line is the estimated trend

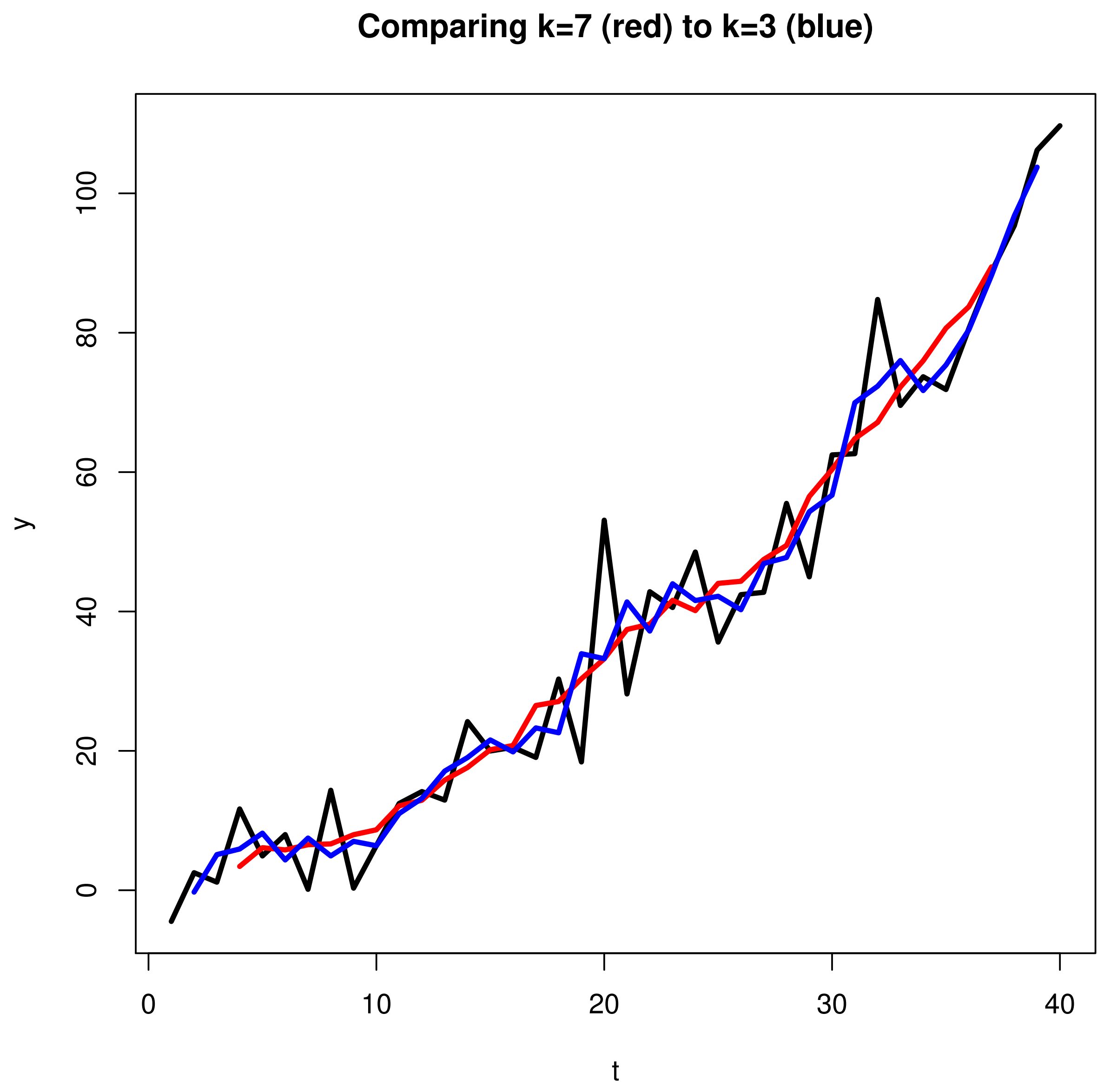
Codes

```
set.seed(12345)
t = 1:40
signal = 0.5*t+0.05*t^2
noise = 10*rnorm(40)
y = signal + noise
plot(t,y,type="l",col="black",lwd=3,main="Smoothing")
lines(signal,lty=c(1),col="blue",lwd=3)
lines(fma(y,7),lty=c(1),col="red",lwd=3)
legend(0, 90, c("y","signal","7-average"),col = c("black","blue","red"),lt
```

Two real-life examples

1. You can think of y (black line) is actual GDP
2. The signal or trend (blue line) is potential GDP
3. The gap between black and blue lines is GDP gap, which measures business cycle
4. Here we use 7-average (red line) to estimate the potential GDP (blue line)
5. The second example is applying 7-average to daily stock price in attempt to find the trend

Compare $k = 7$ to $k = 3$



Compare

```
plot(t, y, type="l",col="black",lwd=3, main="Comparing k=7 (red) to k=3 (b
lines(fma(y,7),lty=c(1),col="red",lwd=3)
lines(fma(y,3),lty=c(1),col="blue",lwd=3)
```

```
> sd(fma(y,7),na.rm=T)
[1] 25.81981
> sd(fma(y,3),na.rm=T)
[1] 29.39308
> length(na.omit(fma(y,7)))
[1] 34
> length(na.omit(fma(y,3)))
[1] 38
```

A bigger *k* leads to a smoother and shorter average

One-sided smoothing forecast

1. To forecast the trend, we use one-sided averaging

$$\hat{y}_{t+1} = \frac{y_t + y_{t-1} + \ldots + y_{t-k+1}}{k} \tag{20}$$

   where $\hat{y}_{t+1}$ is the forecast for the $t+1$th period, and is the average of past k periods

2. The forecasting error is

$$\hat{u}_{t+1} = y_{t+1} - \hat{y}_{t+1} \tag{21}$$

3. We can evaluate forecasting accuracy using total squared forecasting error (TSFE)

$$TSFE = \sum \hat{u}^2_{t+1} \tag{22}$$

   where squaring is to avoid cancellation of positive and negative forecasting errors.

4. In practice, k can be chosen so that TSFE is minimized for a testing set

Weights

1. The values in formula (20) have $\underline{\text{equal}}$ weights of $\frac{1}{k}$
2. Formula (20) works best if the trend is horizontal (or no trend)
3. If there are trends, it is better to assign bigger weights to recent values than remote values
4. That is the motivation for exponential smoothing forecast

Exponential smoothing forecast (ESF)

ESF is computed in a recursive manner

$$\hat{y}_2 = y_1 \tag{23}$$

$$\hat{y}_{t+1} = \hat{y}_t + \alpha(y_t - \hat{y}_t) \quad (t = 2, 3, \ldots) \tag{24}$$

The idea behind ESF is

$$\texttt{new forecast} = \texttt{old forecast} + \texttt{error correction} \tag{25}$$

where $0 < \alpha < 1$ controls the speed of error correction.

1. There is under-prediction if $y_t - \hat{y}_t > 0$. The error correction term is positive, and the new forecast adjusts upward $(\hat{y}_{t+1} > \hat{y}_t)$
2. There is over-prediction if $y_t - \hat{y}_t < 0$. The error correction term is negative, and the new forecast adjusts downward $(\hat{y}_{t+1} < \hat{y}_t)$
3. This <u>negative feedback</u> is the key insight of ESF

ESF uses exponential-decaying weights (optional)

Using recursive substitution we can show

$$\hat{y}_{t+1} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \dots \tag{26}$$

where weights $\alpha, \alpha(1-\alpha), \alpha(1-\alpha)^2, \dots$ decay to zero exponentially. It is evident that recent values receive bigger weights than remote values

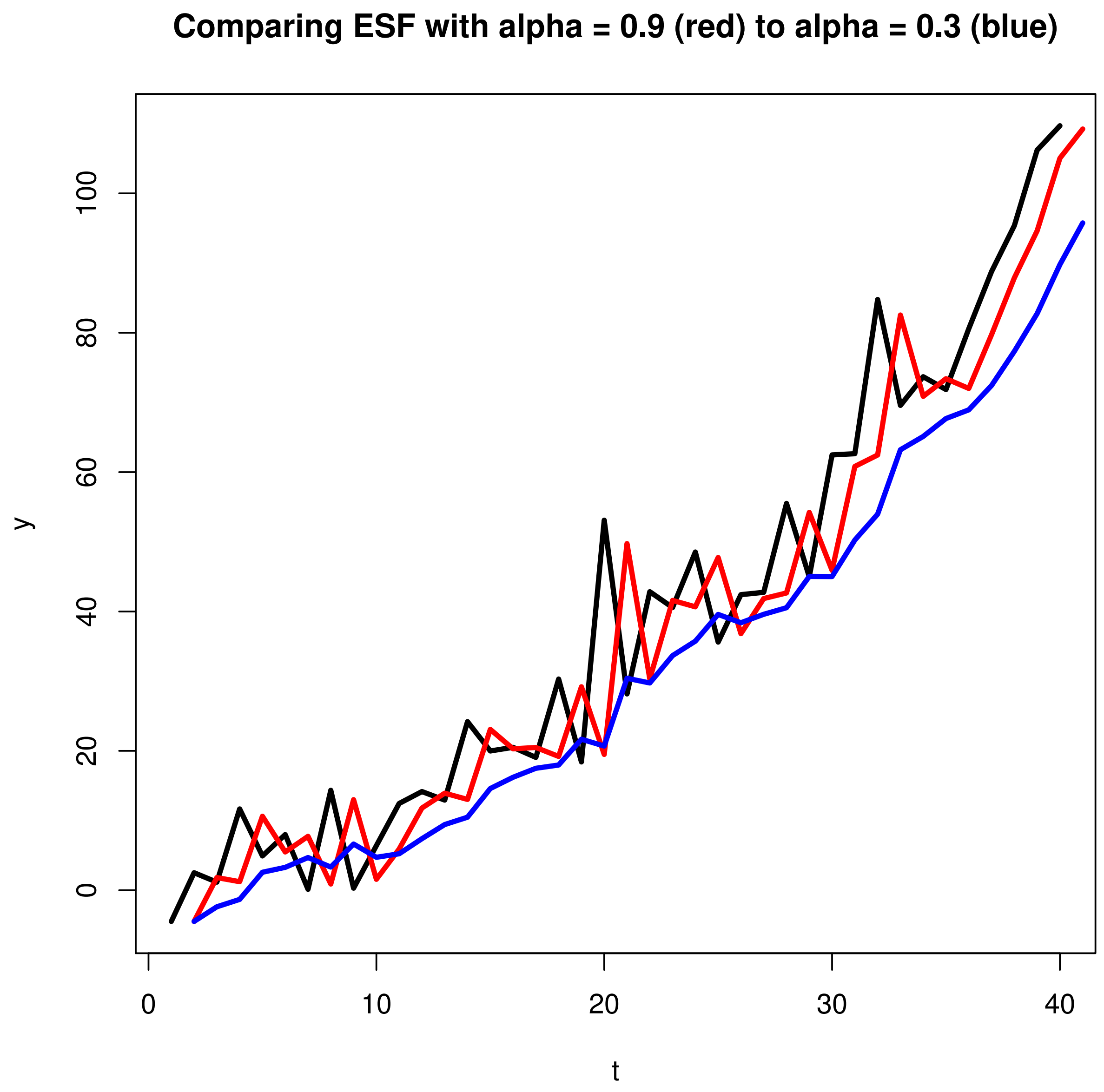
Bigger α in ESP puts heavier weight on recent values, and therefore can capture change more quickly. In practice, α can be chosen to minimize total squared forecasting error of training set

Drawbacks of ESF are (1) typically it cannot capture trend quickly enough (there is lag); (2) α is fixed. Ideally α should be data-driven and time-varying. Better forecasts can be obtained using double exponential smoothing or ARIMA forecasting

User-defined fes function

```
# user-defined ESF
fes = function(data, alpha) {
n = length(data)
yes = rep(NA, n)
yes[2] = y[1]
for (t in 2:n) {
yes[t+1]=yes[t] + alpha*(y[t]-yes[t])
}
return(yes)
}
```

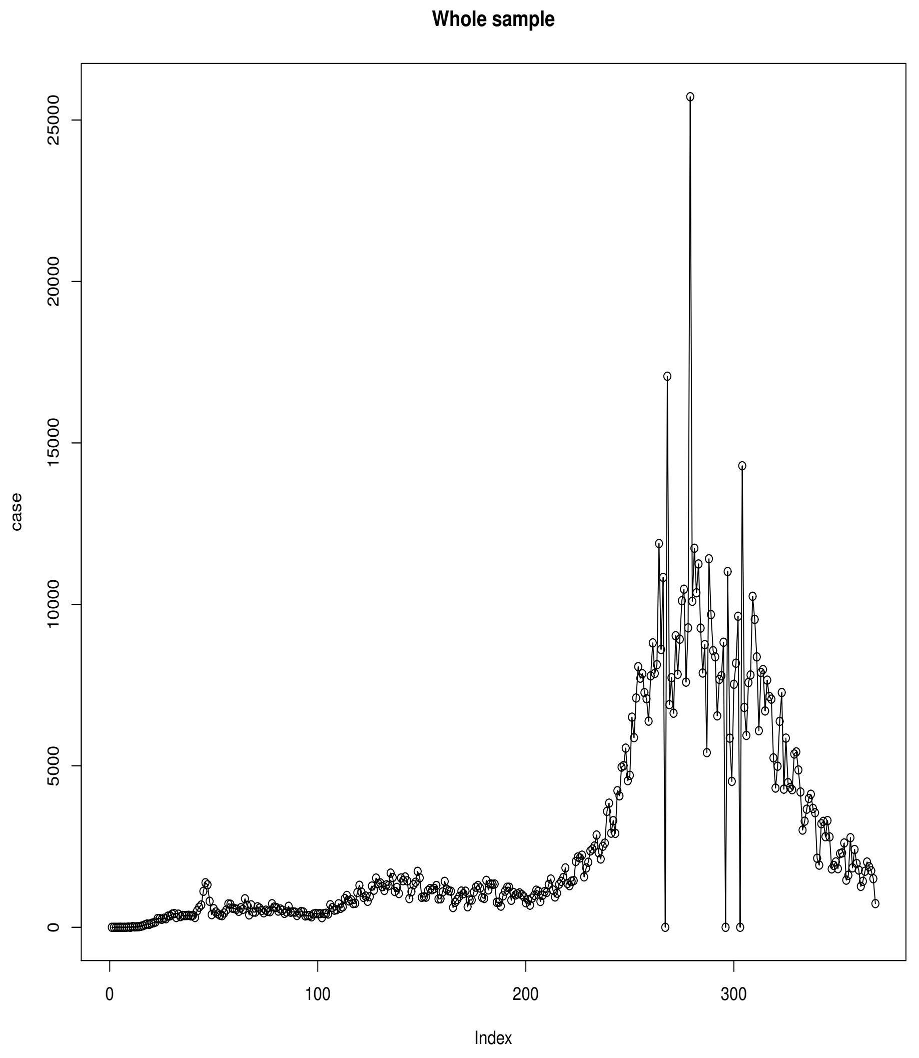
Compare $\alpha = 0.9$ to $\alpha = 0.3$ for ESF

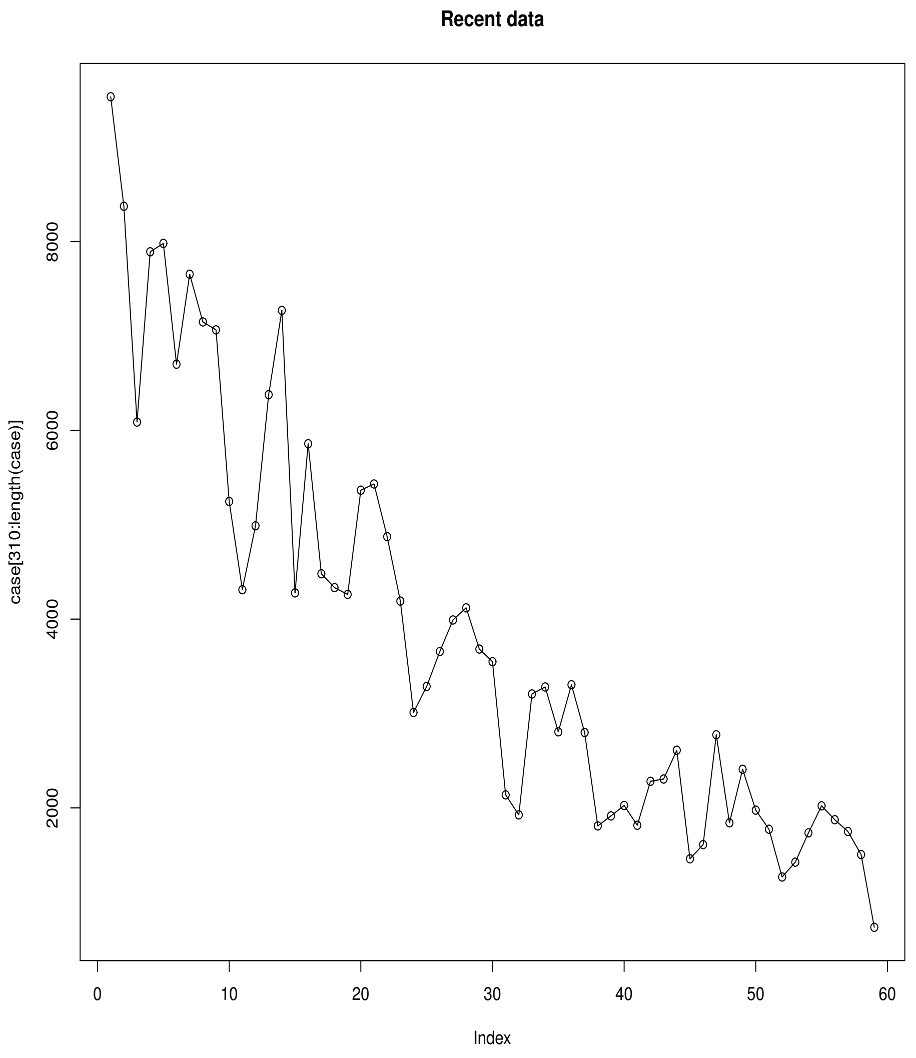


Remarks

1. For this example, because of the upward trend, ESF with $\alpha = 0.9$ works better than $\alpha = 0.3$
2. It seems that the ESF <u>lags</u> behind the y series, and this finding is common for smoothing method
3. That is the motivation for a better ARIMA forecasting

Application: smoothing recent Ohio Covid cases

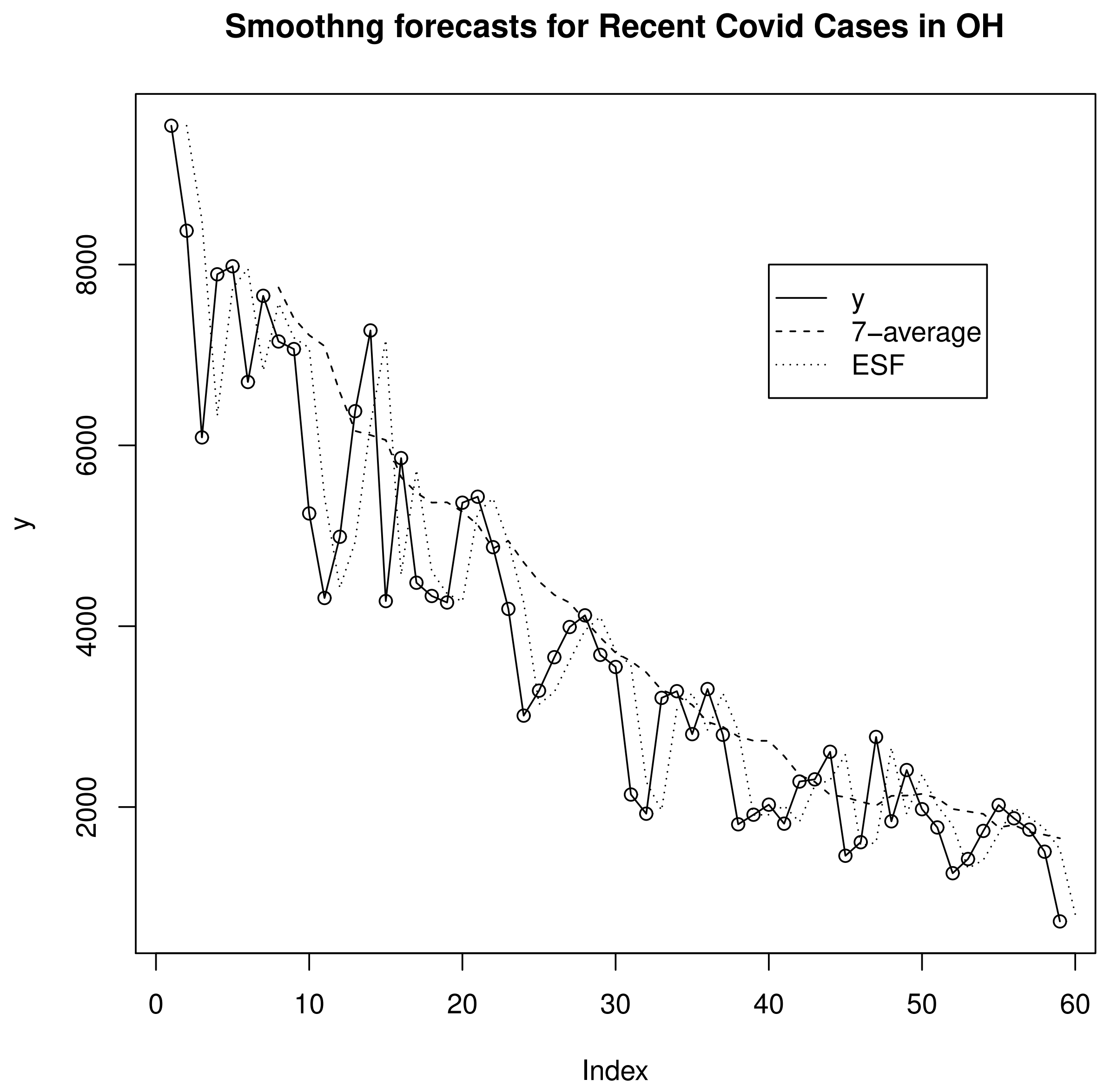




Codes

```
data = read.table("672_covidohio_data.txt", header=T)
case = data[,2]
par(mfrow=c(1,2))
plot(case, type="o", main="Whole sample")
plot(case[310:length(case)], type="o", main="Recent data")
y = case[310:length(case)]

# seven-day one-sided simple average (forecasting) and ESF
n = length(y)
yav = rep(NA, n)
for (t in 8:n) {yav[t]=sum(y[(t-7):(t-1)])/7}

plot(y, type="o", main="Smoothng forecasts for Recent Covid Cases in OH")
lines((1:n),yav,lty=2)
lines(fes(y,0.9), lty=3)
legend(40, 8000, c("y", "7-average", "ESF"), lty = c(1, 2, 3), merge = TRU
```

Plot of smoothing forecasting



Remarks

1. The one-sided 7-averaging is smoother than ESF with $\alpha = 0.9$
2. ESF has smaller forecasting error than one-sided 7-averaging

Forecast for $n+1$ period

```
> cat("7-average forecast for next period is ", yav[length(yav)], "\n")
7-average forecast for next period is  1654.571
> cat("ESF for next period is ", fes(y,0.9)[length(fes(y,0.9))], "\n")
ESF for next period is  814.6765
```

HW7 (Check syllabus for due date)

1. (1 point) Please use R to download SP500 data from FRED, and plot the SP500 time series
2. (2 points) Please apply the ADF unit root test to SP500 and its first difference. Do you find unit roots?
3. (1 points) Please report the result of AR(1) regression applied to the first difference of SP500. Is the model adequate?
4. (1 points) Please find the one-step-ahead forecast for the first difference of SP500 based on the AR(1) results