Forecasting ARIMA(1,1,1) Series
**ARIMA(1,1,1)**

1. We generate the data assuming the true process is known. Then we can compare the estimation result to the truth to ensure the coding is right.

2. In general, an ARIMA(1,1,1) process is

\[
\begin{align*}
\Delta y_t &= d + \eta_t \\
\eta_t &= \phi_1 \eta_{t-1} + e_t + \theta_1 e_{t-1}
\end{align*}
\]

In words, the first difference $\Delta y_t$ is a zero-mean ARMA(1,1) process $\eta_t$ plus the drift term $d$.

3. By substituting $\eta_t = y_t - y_{t-1} - d$, the same ARIMA(1,1,1) process can be written as

\[
(y_t - y_{t-1} - d) = \phi_1 (y_{t-1} - y_{t-2} - d) + e_t + \theta_1 e_{t-1}
\]

where $d$ is the drift term; $\phi_1$ is the AR coefficient; $\theta_1$ is the MA coefficient.

4. Here we let $d = 0.2, \phi_1 = 0.7, \theta_1 = -0.5$. Notice that the nonzero drift term causes the series to be trending.
Generating ARIMA(1,1,1)

We use R loop to generate $\eta_t$ and then $y_t$

```r
set.seed(12345)
T = 1000
tr = 1:T
e = rnorm(T)
dy = rep(0, T)
y = rep(0, T)
dy[1] = e[1]
y[1] = e[1]
for (t in 2:T) {
dy[t] = 0.7 * dy[t-1] + e[t] - 0.5 * e[t-1]
y[t] = 0.2 + y[t-1] + dy[t]
}
```

In the codes, the ARMA(1,1) $\eta_t$ is denoted as $dy[t]$
Plotting ARIMA(1,1,1)

The series has an upward trend due to the positive drift term $d = 0.2$. The trend, along with the smoothness, signifies nonstationarity.
Deterministic and Stochastic Trends

We can show that the ARIMA(1,1,1) process is trending

\[ y_t = y_0 + dt + (\eta_1 + \eta_2 + \ldots + \eta_t) \]

There is a global deterministic trend \( dt \) if \( d \neq 0 \). Even if \( d = 0 \), there can be local stochastic trend \( (\eta_1 + \eta_2 + \ldots + \eta_t) \). In graph, the deterministic trend can be dominating. We can estimate the drift term, which is the mean value of \( \Delta y_t \), without running regression

```r
mean(d.y, na.rm=T)
```

[1] 0.2769492
Estimating ARIMA(1,1,1)

1. Estimating ARIMA(1,1,1) for $y_t$ is the same as estimating ARIMA(1,0,1) for $\Delta y_t$

2. However, DO NOT use arima(y, order = c(1,1,1)) because this assumes zero drift term!!

3. Instead, use arima(d.y, order = c(1,0,1)). But be careful, the reported intercept actually is the drift term.

\[
\text{arima}(x = d.y, \text{order} = c(1, 0, 1))
\]

Coefficients:

\[
\begin{array}{ccc}
\text{ar1} & \text{ma1} & \text{intercept} \\
0.7246 & -0.5197 & 0.2767 \\
\text{s.e.} & 0.0698 & 0.0871 & 0.0550 \\
\text{log likelihood} = -1416.09, \ aic = 2840.17
\end{array}
\]

The estimated $\hat{\phi}_1 = 0.7246, \hat{\theta}_1 = -0.5197, \hat{d} = 0.2767$ are all close to the true values, and are significant.
Estimating Error Terms

The error term $e_t$ in (2) is unobservable. According to (3), we can show

$$
e_t = (y_t - y_{t-1} - d) - \phi_1(y_{t-1} - y_{t-2} - d) - \theta_1 e_{t-1}
$$

$$
= y_t - (1 - \phi_1)d - (1 + \phi_1)y_{t-1} + \phi_1 y_{t-2} - \theta_1 e_{t-1}
$$

So we use the codes below to estimate $e_1, e_2, \ldots e_t$ in a recursive way

```r
ehat = rep(0, T)
ehat[1] = y[1]
for (t in 3:T) ehat[t] = y[t] - (1-phi1) * dhat-(1+phi1) * y[t-1]+phi1 * ehat[t-1]
```
**Built-in Function for Estimating Error Terms**

Except the early observations, our estimated error terms are very close to the ones reported by R built-in function `resid`

```r
cbind(resid(arima(d.y, order = c(1,0,1)))[1:10], ehat[1:10])
```

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NA</td>
<td>0.5855288</td>
</tr>
<tr>
<td>2</td>
<td>0.7187304</td>
<td>0.8303763</td>
</tr>
<tr>
<td>3</td>
<td>-0.1458968</td>
<td>-0.0739732</td>
</tr>
<tr>
<td>4</td>
<td>-0.4963657</td>
<td>-0.4612424</td>
</tr>
<tr>
<td>5</td>
<td>0.5616954</td>
<td>0.5796414</td>
</tr>
<tr>
<td>6</td>
<td>-1.8649630</td>
<td>-1.8558010</td>
</tr>
<tr>
<td>7</td>
<td>0.5906055</td>
<td>0.5951970</td>
</tr>
<tr>
<td>8</td>
<td>-0.3141356</td>
<td>-0.3117367</td>
</tr>
<tr>
<td>9</td>
<td>-0.3219821</td>
<td>-0.3207393</td>
</tr>
<tr>
<td>10</td>
<td>-0.9561814</td>
<td>-0.9555372</td>
</tr>
</tbody>
</table>
Forecasting ARIMA(1,1,1)

1. Assuming our sample ends at $T$.

2. Rewrite (3) as

$$y_t = (1 - \phi_1) d + (1 + \phi_1)y_{t-1} - \phi_1 y_{t-2} + e_t + \theta_1 e_{t-1}$$

(4)

3. The one-step and two-step (out-of-sample) forecasts are

$$E(y_{T+1}|\Omega_T) = (1 - \phi_1) d + (1 + \phi_1)y_T - \phi_1 y_{T-1} + \theta_1 e_T$$

(5)

$$E(y_{T+2}|\Omega_T) = (1 - \phi_1) d + (1 + \phi_1)E(y_{T+1}|\Omega_T) - \phi_1 y_T$$

(6)

For $k$–th horizon ($k > 2$) we have

$$E(y_{T+k}|\Omega_T) = (1 - \phi_1) d + (1 + \phi_1)E(y_{T+k-1}|\Omega_T) - \phi_1 E(y_{T+k-2}|\Omega_T)$$

(7)
R Codes

Again, we use R loop to generate 12 out-of-sample forecasts, and display the first 5 forecasts

```r
f = rep(0, 12)
f[1] = (1-phi1)*dhat+(1+phi1)*y[T]-phi1*y[T-1]+theta1*ehat[T]
f[2] = (1-phi1)*dhat+(1+phi1)*f[1]-phi1*y[T]
for (t in 3:12) f[t] = (1-phi1)*dhat+(1+phi1)*f[t-1]-phi1*f[t-2]
f[1:5]
[1] 277.3508 277.4945 277.6748 277.8817 278.1078
```
yall = c(y,f)
plot(yall[901:1012], type="l", main="Last 100 obs and 12 out-of-sample forecasts")
abline(v = c(101), col = "red", lty=1)
Plotting Forecasting Values II

```
plot(c(yall[901:1000], rep(NA, 12)), type="l", lwd=2, lty=1, main="Last 100 obs and 12 out-of-sample forecasts")
lines(c(rep(NA, 100), f), lty=2, col = "red", lwd=2)
```
## R Forecast Package

There is a Forecast package that can provide the out-of-sample forecasts

```r
library(forecast)

forecast(Arima(y, order=c(1,1,1), include.drift=T), h=12)
```

<table>
<thead>
<tr>
<th>Point</th>
<th>Forecast</th>
<th>Lo 80</th>
<th>Hi 80</th>
<th>Lo 95</th>
<th>Hi 95</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>277.3508</td>
<td>276.0693</td>
<td>278.6324</td>
<td>275.3908</td>
<td>279.3108</td>
</tr>
<tr>
<td>1002</td>
<td>277.4945</td>
<td>275.4878</td>
<td>279.5012</td>
<td>274.4255</td>
<td>280.5635</td>
</tr>
<tr>
<td>1003</td>
<td>277.6748</td>
<td>275.0224</td>
<td>280.3272</td>
<td>273.6182</td>
<td>281.7313</td>
</tr>
<tr>
<td>1004</td>
<td>277.8816</td>
<td>274.6349</td>
<td>281.1283</td>
<td>272.9162</td>
<td>282.8471</td>
</tr>
<tr>
<td>1005</td>
<td>278.1077</td>
<td>274.3089</td>
<td>281.9066</td>
<td>272.2979</td>
<td>283.9175</td>
</tr>
<tr>
<td>1006</td>
<td>278.3478</td>
<td>274.0336</td>
<td>282.6619</td>
<td>271.7498</td>
<td>284.9457</td>
</tr>
<tr>
<td>1007</td>
<td>278.5979</td>
<td>273.8010</td>
<td>283.3948</td>
<td>271.2617</td>
<td>285.9341</td>
</tr>
<tr>
<td>1008</td>
<td>278.8554</td>
<td>273.6047</td>
<td>284.1060</td>
<td>270.8252</td>
<td>286.8855</td>
</tr>
<tr>
<td>1009</td>
<td>279.1181</td>
<td>273.4395</td>
<td>284.7967</td>
<td>270.4335</td>
<td>287.8028</td>
</tr>
<tr>
<td>1010</td>
<td>279.3847</td>
<td>273.3011</td>
<td>285.4684</td>
<td>270.0806</td>
<td>288.6889</td>
</tr>
</tbody>
</table>
Remarks

1. We obtain the same forecasts!

2. Arima is the function in the forecast package, which is different from arima in the stats package.

3. Notice that include.drift=T allows for a nonzero drift term.