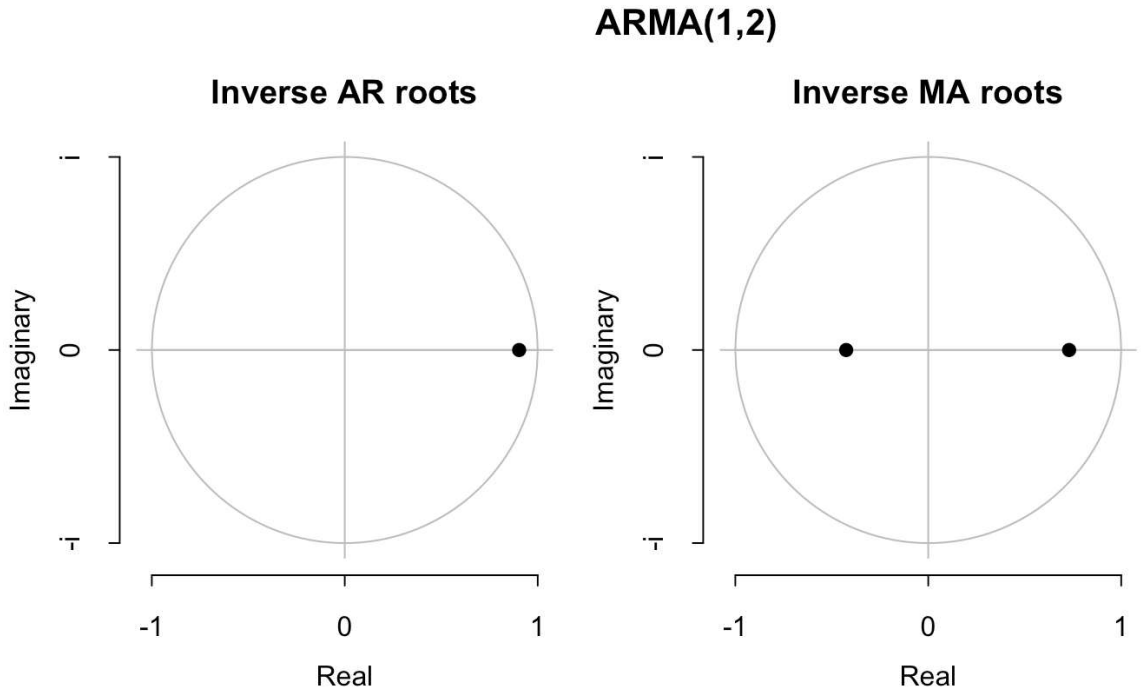
Tutorial 9 Answers

1. **ARMA(1,2)** The first plot, for the AIC-selected ARMA(1,2) model, and the corresponding parameter estimates:




```
    ar1      ma1      ma2 intercept
 0.9038  -0.3046  -0.3105    0.0040
```

For the AR(1) component, the inverted AR root is simply equal to the coefficient: 0.9038. For the MA(2) compoent the inverted roots are computed to be real (one positive and one negative) and both clearly inside the unit circle, hence the estimated model is invertible. Note the MA(2) coefficients are not directly interpretable for invertibility, it is necessary to compute the inverted roots. The plot is sufficient, but we could compute the inverted roots numerically:

```
MA2invroots <- 1/polyroot(c(1,eqlist[["ARMA12"]]$coef[c("ma1", "ma2")]))
print(round(MA2invroots,4))
```

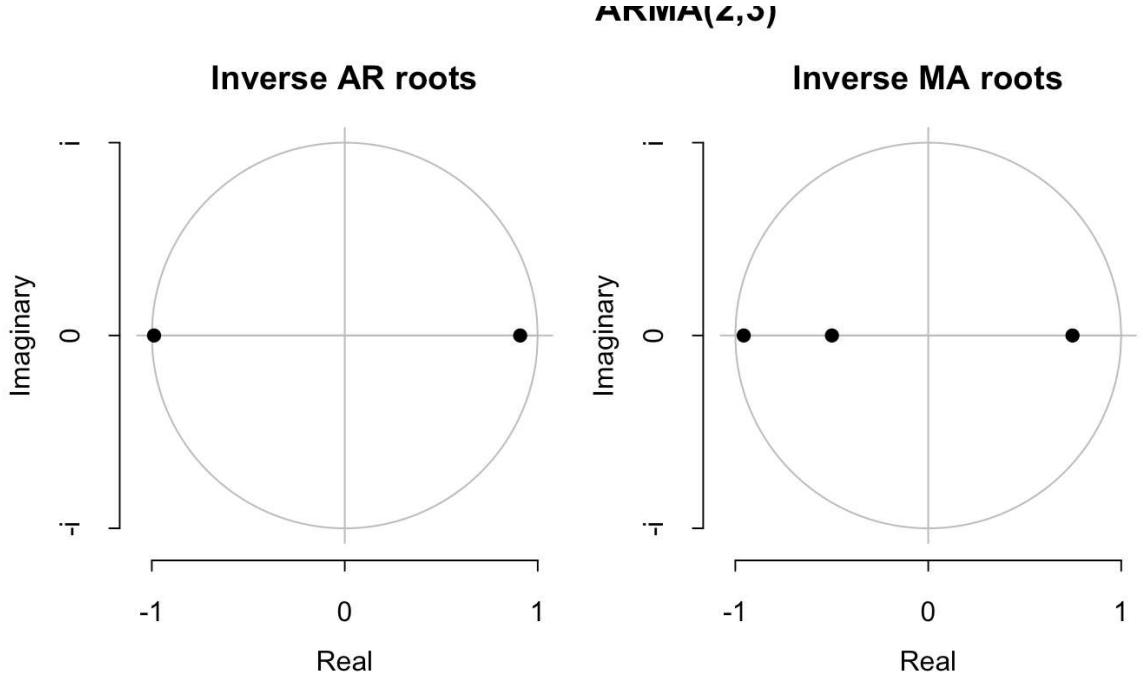
```
[1]  0.7300+0i -0.4254+0i
```

Both are real (zero imaginary part) and both less than one in magnitude.

ARMA(2,3)

The coefficients and inverted roots plot for the ARMA(2,3) are as follows.

```
     ar1      ar2      ma1      ma2      ma3 intercept
 -0.0793   0.8988   0.7080  -0.6109  -0.3570    0.0044
```




```
AR2invroots <- 1/polyroot(c(1,-eqlist[["ARMA23"]]$coef[c("ar1", "ar2")]))
print(round(AR2invroots,4))
```

```
[1]  0.9092+0i -0.9885+0i
```

```
MA3invroots <- 1/polyroot(c(1,eqlist[["ARMA23"]]$coef[c("ma1", "ma2", "ma3")]))
print(round(MA3invroots,4))
```

```
[1]  0.7477+0i -0.9564+0i -0.4993+0i
```

Incrementing the AR and MA orders by one introduces an additional inverted AR and MA root. These additional inverted roots are -0.9885+0i for the AR component and -0.9564+0i for the MA component. (Note the order in which these roots are printed out from `polyroot` is arbitrary.) Both of these additional roots are very close to one, implying this model for the same time series apparently has suddenly become both non-stationary and non-invertible. This seems puzzling in light of the results for the ARMA(1,2) model and the fact that we earlier went through the formal ADF testing process plus graphical inspection to conclude that a single difference seemed appropriate (i.e. enough differencing, but not too many differences) to render the original series stationary.

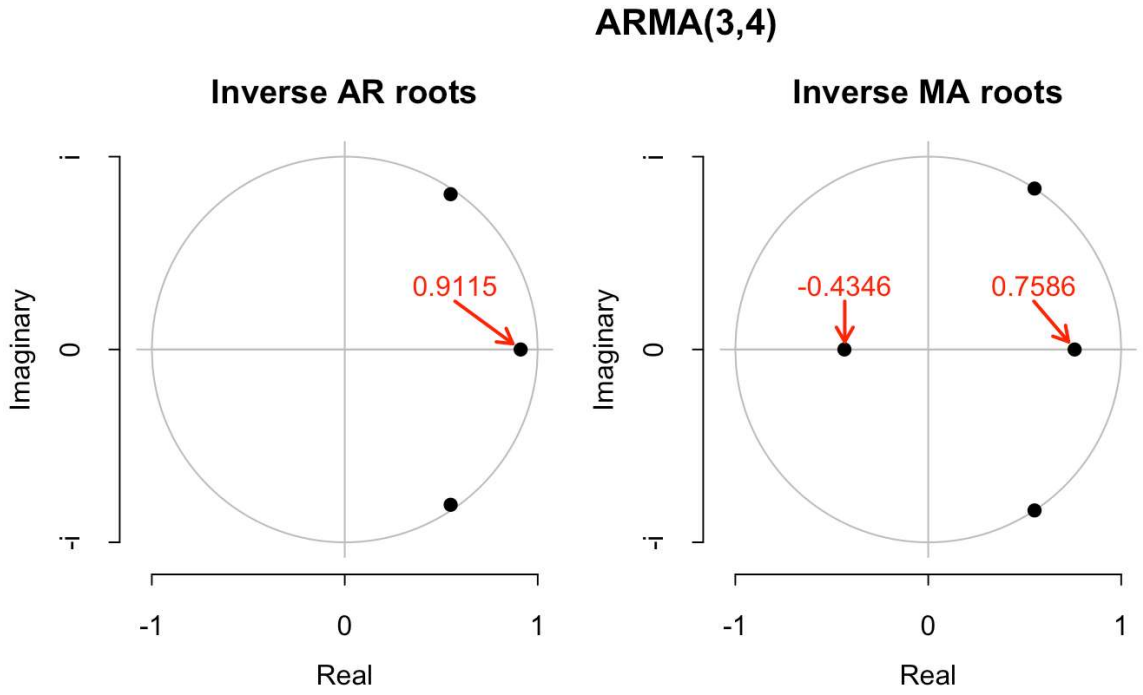
The other inverted AR and MA roots ($0.9092 + 0i$ and $0.7477 + 0i, -0.4993 + 0i$) respectively are very close to those reported for the ARMA(1,2) model. It seems that "over-specifying" the model relative to that chosen by the AIC has introduced the appearance of non-stationarity and non-invertibility while leaving the rest of the model essentially unchanged. More on this soon.

ARMA(3,4)

The coefficients and inverted roots plot for the ARMA(2,3) are as follows.

```
      ar1       ar2       ar3       ma1       ma2       ma3       ma4 intercept
```

```
2.0090   -1.9502    0.8658   -1.4255    1.0273    0.0391   -0.3297    0.0039
```

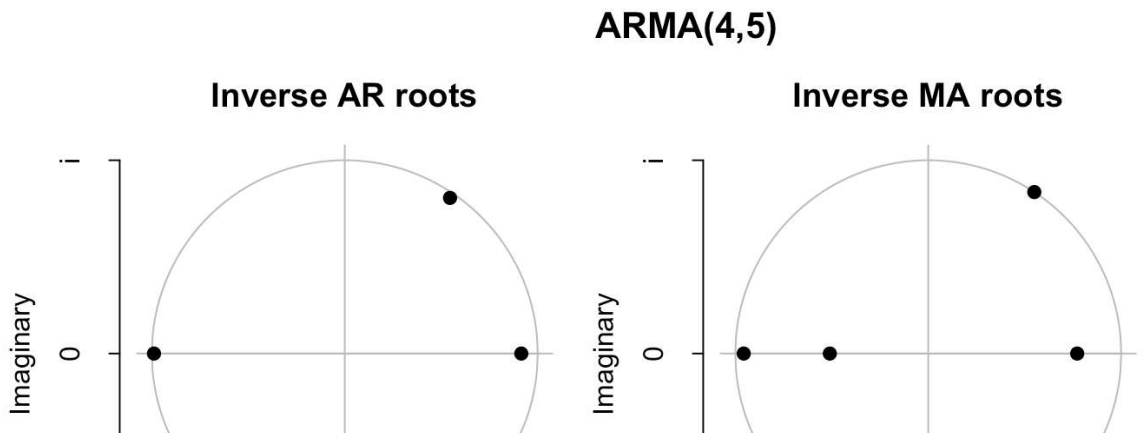


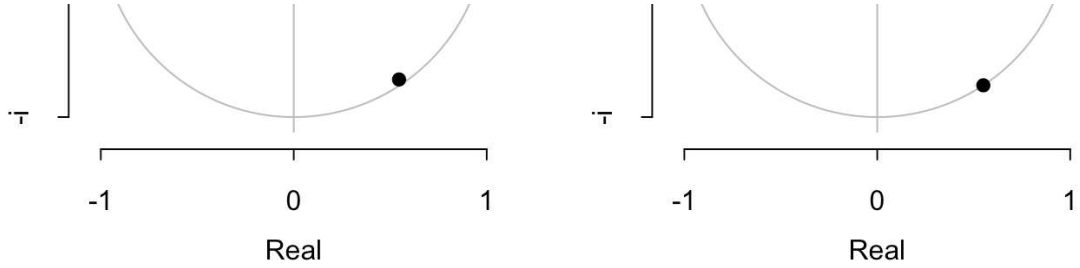
The inverted roots that clearly correspond to the original ARMA(1,2) inverse roots are indicated in red. With the addition of 2 to both AR and MA orders, we now have two additional (very close to) non-stationary AR inverse roots and two non-invertible MA inverse roots, all four of which happen to be complex. The exact values of the roots is not important (although can easily be computed with `polyroot`), what matters is that they are (very close to) the unit circle.

ARMA(4,5)

The pattern is probably already becoming clear, but here are the ARMA(4,5) results.

```
     ar1       ar2       ar3       ar4       ma1       ma2       ma3       ma4
  1.0171    0.0391   -1.0569    0.8559   -0.4046   -0.4075    1.0260   -0.2301
     ma5 intercept
 -0.3765    0.0045
```





Now there are *three* additional (very close to) non-stationary AR inverse roots and non-invertible MA inverse roots.

What is going on?

The AIC-selected ARMA(1,2) model can be written symbolically as

$$\Delta Y_t = \beta_0 + Z_t$$
$$(1 - \phi_1 L) Z_t = (1 + \theta_1 L + \theta_2 L^2) U_t$$

As we saw in lectures, lag polynomials can be factored into linear factors expressed in terms of their inverse roots. For the current analysis, let us use $\xi_1, \ldots, \xi_p$ to denote the roots of an AR(p) polynomial and $\zeta_1, \ldots, \zeta_q$ to denotes the roots of an MA(q) polynomial. For the ARMA(1,2) model we therefore have

$$(1 - \xi_1^{-1} L) Z_t = (1 - \zeta_1^{-1} L)(1 - \zeta_2^{-1} L) U_t$$

where $\xi_1^{-1} = \phi_1$ and $\zeta_1^{-1}, \zeta_2^{-1}$ can be computed from θ_1, θ_2 (eg numerically by `polyroot`). Above we found $\xi_1^{-1} = 0.9038$, $\zeta_1^{-1} = 0.7300$, $\zeta_2^{-1} = -0.4254$.

Suppose the ARMA(1,2) is indeed "correct". Then we increment the specification to ARMA(2,3):

$$(1 - \xi_1^{-1} L)(1 - \xi_2^{-1} L) Z_t = (1 - \zeta_1^{-1} L)(1 - \zeta_2^{-1} L)(1 - \zeta_3^{-1} L) U_t.$$

It would be reasonable to take the "correct" values of ξ_2^{-1} and ζ_3^{-1} to be zero, i.e. they are excess terms for the model. Unfortunately things get more complicated in this specification, and we can't rely on estimates of ξ_2^{-1} and ζ_3^{-1} to be near zero. By introducing the extra terms on *both* sides of the model we have introduced what is referred to as a "common factor". i.e. *for any* value z such that $\xi_2^{-1} = \zeta_3^{-1} = z$ (not only $z = 0$) we have

$$(1 - \xi_1^{-1} L)(1 - zL) Z_t = (1 - \zeta_1^{-1} L)(1 - \zeta_2^{-1} L)(1 - zL) U_t,$$

and the common factor $(1 - zL)$ can be "cancelled" from both sides. The problem is that in the ARMA(2,3) specification there is not one "correct" value $\xi_2^{-1} = \zeta_3^{-1} = 0$, but an entire continuum of "correct" values $\xi_2^{-1} = \zeta_3^{-1} = z$ for any z.

When a model specification is such that there are multiple "correct" values, we say that model is *not identified*. This is the fundamental problem — there is no way of knowing what "correct" value z is being estimated since all z are equivalent. At that stage we are at the mercy of the numerical algorithm that is estimating the model. In this application the algorithm is preferring the push the estimates of the excess parameters ξ_2^{-1}, ζ_3^{-1} towards

the unit circle, even though intuitively we may think those estimates should be pushed towards zero.

Some practical observations:

- If you find an estimated ARMA(p,q) model with inverse roots that appear to nearly "cancel out", especially if they are on / near the unit circle, it would be prudent to reconsider the orders p,q and query whether both might be reduced.
- It would usually be the case that the AIC model selection step will work to avoid this issue (as it did in this application by preferring the ARMA(1,2) model). Nevetheless the AIC has a non-zero probability of selecting a model that is too large, hence it is worth being aware of this possibility.
- When doing model selection searches, it is generally advised to avoid increasing the order of *both* AR and MA components simultaneously. It is better to increase AR or MA components separately, in which case this potential identification problem should be avoided.
- Building on the previous point, notice our approach to ARMA selection with AIC has been quite "brute force", i.e. estimating ARMA(p,q) models for every combination of p and q. If $p_{\max}$ and $q_{\max}$ aren't excessively large this will usually work ok… until it doesn't! If the "correct" model (better we say something like "best compromise of model fit and parismony") within the class has small p and q then we may find some strange or unstable estimates, or even errors attempting to estimate, when trying larger p and q. There are cleverer ways to doing the model selection search, implemented for example by the command `auto.arima` (which we are not using), that attempt to avoid this problem and provide a more computationally efficient search.
- More pragmatically, at least in economics and finance it has become more common to rely simply on AR models rather than ARMA. This necessarily avoids the common factor problem, reduced the model selection search to a single dimension, simplifies estimation to OLS (quicker and more stable than ARMA estimation), and simplifies extensions to multivariate models. This may sometimes come at the cost of requiring more parameters (a clear example is using AR(p) to approximate MA(1)) but practically the forecasts are often very similar anyway (compare ARMA(1,2) and AR(3) for the current interest rate time series).

---

2. **a.** We can algebraically manipulate lag polynomials like any other polynomial:

$$(1-\phi_1 L)(1-\Phi_1 L^4) = 1-\phi_1 L-\Phi_1 L^4+\phi_1\Phi_1 L^5$$

noting that $L\cdot L^4 = L^5$, i.e. five lags. This is an AR(5) model

$$Z_t = \phi_1^* Z_{t-1}+\phi_2^* Z_{t-2}+\phi_3^* Z_{t-3}+\phi_4^* Z_{t-4}+\phi_5^* Z_{t-5}+U_t$$

with restrictions $\phi_1^* = \phi_1$, $\phi_2^* = \phi_3^* = 0$, $\phi_4^* = \Phi_1$ and $\phi_5^* = -\phi_1\Phi_1$.

b. The results suggest $p = 1$ and $P = 3$:

```
print(round(LBp,3))
```

```
    SAR0  SAR1  SAR2  SAR3  SAR4
AR0    0 0.000 0.000 0.000 0.000
AR1    0 0.079 0.512 0.739 0.693
AR2    0 0.086 0.433 0.711 0.634
AR3    0 0.346 0.472 0.485 0.425
AR4    0 0.000 0.402 0.000 0.426
```

```
print(AICc==min(AICc))
```

```
     SAR0  SAR1  SAR2  SAR3  SAR4
AR0 FALSE FALSE FALSE FALSE FALSE
AR1 FALSE FALSE FALSE  TRUE FALSE
AR2 FALSE FALSE FALSE FALSE FALSE
AR3 FALSE FALSE FALSE FALSE FALSE
AR4 FALSE FALSE FALSE FALSE FALSE
```

The selected model has $p = 1$ and $P = 3$. With quarterly data this implies a longest lag in the model of 13, found by expanding:

$$(1 - \phi_1 L)(1 - \Phi_1 L^4 - \Phi_2 L^8 - \Phi_3 L^{12})$$

Forecasting with the selected model:

```
p <- 1
P <- 3
eq_sar <- Arima(Y, order=c(p,0,0), xreg=X[t_est,],
      seasonal=list(order=c(P,0,0), period=4))

# Forecast through 2019
Yf_sar <- forecast(eq_sar, xreg=X[t_for,])$mean

# Compute RMSE
RMSE_sar <- sqrt(mean((Yf-Yf_sar)^2))
print(RMSE_sar)
```

```
[1] 0.00788378
```

c. Forecasting with quarterly dummies and AR(3) model:

```
# Estimation of AR(3) with XQD as deterministic matrix
eq_qd <- Arima(Y, order=c(3,0,0), xreg=XQD[t_est,])

# Forecast through 2019
Yf_qd <- forecast(eq_qd, xreg=XQD[t_for,])$mean

# Compute RMSE
RMSE_qd <- sqrt(mean((Yf-Yf_qd)^2))
print(RMSE_qd)
```

```
[1] 0.00804294
```

The seasonal autoregressive model has slightly lower RMSE than the quarterly dummy model.

d. Firstly the model with quarterly dummies is clearly preferred by the AICc to the seasonal autoregression:

```
print(eq_qd$aicc)
```

```
[1] -472.9557
```

```
print(eq_sar$aicc)
```

```
[1] -440.0512
```

Including the quarterly dummies in the seasonal autoregressive model search provides a way to check if seasonal autoregressive components *and* quarterly dummies may improve the model further:

```
pmax <- 4
Pmax <- 4
AICc <- matrix(nrow=1+pmax, ncol=1+Pmax)
rownames(AICc) <- paste0("AR",0:pmax)
colnames(AICc) <- paste0("SAR",0:Pmax)
LBp <- AICc
for (p in 0:pmax){
  for (P in 0:Pmax){
    eq_sar <- Arima(Y, order=c(p,0,0), xreg=XQD[t_est,],
          seasonal=list(order=c(P,0,0), period=4))
    AICc[1+p,1+P] <- eq_sar$aicc
  }
}
print(AICc==min(AICc))
```

```
     SAR0  SAR1  SAR2  SAR3  SAR4
AR0 FALSE FALSE FALSE FALSE FALSE
AR1 FALSE FALSE FALSE FALSE FALSE
AR2 FALSE FALSE FALSE FALSE FALSE
AR3  TRUE FALSE FALSE FALSE FALSE
AR4 FALSE FALSE FALSE FALSE FALSE
```

In this case it turns out : no, once quarterly dummies are used there is no further improvement in model fit by also using seasonal autoregressive components. The AR(3) remains the preferred specification.