Final Exam

Brian Detweiler

May 4, 2017, 11:59 pm

```
set.seed(48548493)
```

1. Let $\{X_t\}$ be a sequence of iid random variables with mean 0 and variance σ_X^2, and ω a finite constant. Decide whether $Y_t = X_t cos(\omega t) + X_{t-1} sin(\omega t)$ is stationary or not.

A stochastic process $\{Y_t\}$ is defined as weakly stationary if

1. The mean function is constant over time, and
2. $\gamma_{t,t-k} = \gamma_{0,k}$

$$\begin{aligned} E[Y_t] &= E[X_t cos(\omega t) + X_{t-1} sin(\omega t)] \\ &= E[X_t] cos(\omega t) + E[X_{t-1}] sin(\omega t) \\ &= 0 \cdot cos(\omega t) + 0 \cdot sin(\omega t) \\ &= 0 \end{aligned}$$

$$\begin{aligned} \gamma_{t,t-k} &= Cov(Y_t, Y_{t-k}) = Cov\big[(X_t cos(\omega t) + X_{t-1} sin(\omega t)), (X_{t-k} cos(\omega(t-k)) + X_{t-k-1} sin(\omega(t-k))\big] \\ &= Cov\big(X_t cos(\omega t), X_{t-k} cos(\omega(t-k))\big) \\ &+ Cov\big[X_t cos(\omega t), X_{t-k-1} sin(\omega(t-k)\big] \\ &+ Cov\big[X_{t-1} sin(\omega t), X_{t-k} cos(\omega(t-k)\big] \\ &+ Cov\big[X_{t-1} sin(\omega t), X_{t-k-1} sin(\omega(t-k)\big] \\ &= cos(\omega t) cos(\omega(t-k) Cov\big(X_t, X_{t-k}\big) \\ &+ cos(\omega t) sin(\omega(t-k) Cov\big[X_t, X_{t-k-1}\big] \\ &+ sin(\omega t) cos(\omega(t-k) Cov\big[X_{t-1}, X_{t-k}\big] \\ &+ sin(\omega t) sin(\omega(t-k) Cov\big[X_{t-1}, X_{t-k-1}\big] \end{aligned}$$

We note the following trig identities:

$$\begin{aligned} cos(x)cos(y) + sin(x)sin(y) &= cos(x-y) \\ cos(x)sin(y) + sin(x)cos(y) &= sin(x+y) \end{aligned}$$

Letting $x = \omega t$ and $y = \omega(t-k)$, we get

$$
\begin{aligned}
cos(\omega t)cos(\omega(t-k)) + sin(\omega t)sin(\omega(t-k)) &= cos((\omega t) - (\omega(t-k))) \\
&= cos((\omega t) - (\omega(t-k))) \\
&= cos(\omega k) \\
cos(\omega t)sin(\omega(t-k)) + sin(\omega t)cos(\omega(t-k)) &= sin((\omega t) + (\omega(t-k))) \\
&= sin((\omega t) + (\omega(t-k))) \\
&= sin(\omega(2t-k))
\end{aligned}
$$

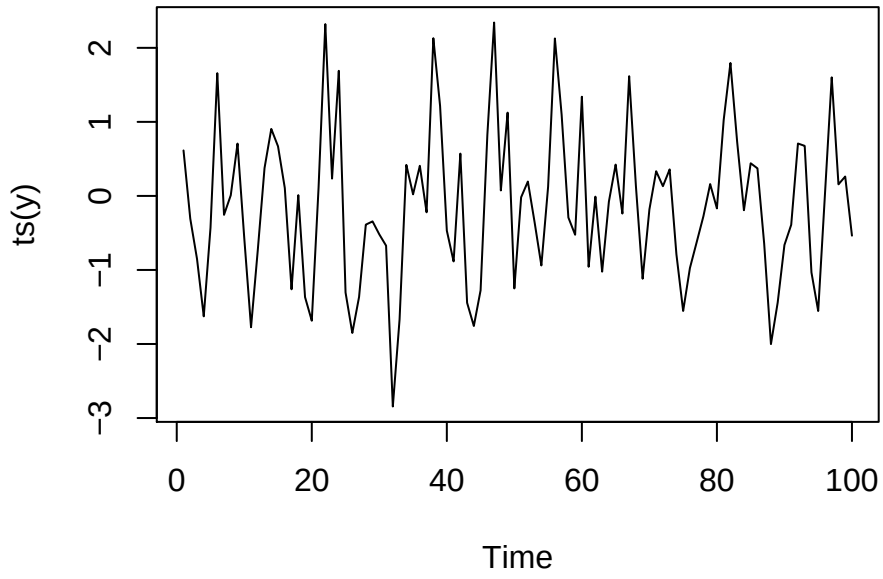
Now we have 3 possible cases:

$$
Cov(Y_t, Y_{t-k}) = \begin{cases} 2\sigma_X^2 & \text{for } k = 0 \\ sin(\omega(2t-1))2\sigma_X^2 & \text{for } k = 1 \\ 0 & \text{for } k > 1 \end{cases}
$$

So we have a constant mean and a constant variance for each time lag k, and thus it is stationary.

We can verify this with a simulation, arbitrarily letting $\sigma_X^2 = 1$ and $\omega = 0.5$. We'll look at 100 lags, and perform an augmented Dickey-Fuller test:

```
omega <- .5
x <- rnorm(n = 101, mean = 0, sd = 1)
y <- x[2:101] * cos(omega * 1:100) + x[1:100] * sin(omega * 1:100)
plot(ts(y))
```




```
adf.test(ts(y), alternative="stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts(y)
## Dickey-Fuller = -5.6636, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

■

2. For each of the following, identify it as an ARIMA model or a multiplicative seasonal ARIMA model. That is, find the values of p, d, q, P, D, Q, period s, and the values of the parameters (ϕ's, θ's, Φ's, and 'Θ's).

(a) $Y_t = 2.8Y_{t-1} - 2.6Y_{t-2} + 0.8Y_{t-3} + e_t - 0.76e_{t-1} - 0.2e_{t-2}$

The AR characteristic polynomial is

$$1 - 2.8x + 2.6x^2 - 0.8x^3 = -0.8(x - 1.25)(x^2 - 2x + 1)$$

With a real root of 1.25, it is greater than 1 so we must take the first difference. This yields

$$\begin{aligned}
\nabla Y_t &= Y_t - Y_{t-1} \\
\nabla Y_t &= \left[2.8Y_{t-1} - 2.6Y_{t-2} + 0.8Y_{t-3} + e_t - 0.76e_{t-1} - 0.2e_{t-2}\right] - Y_{t-1} \\
&= 2.8Y_{t-1} - Y_{t-1} - 2.6Y_{t-2} + 0.8Y_{t-3} + e_t - 0.76e_{t-1} - 0.2e_{t-2} \\
&= 1.8Y_{t-1} - 2.6Y_{t-2} + 0.8Y_{t-3} + e_t - 0.76e_{t-1} - 0.2e_{t-2}
\end{aligned}$$

Now the AR characteristic polynomial is

$$1 - 1.8x + 2.6x^2 - 0.8x^3 = -0.8(x - 2.56227)(x^2 - 0.687731x + 0.487849)$$

Which gives us a real root of 2.56227. This is still not less than 1, but the second difference does not produce a root less than 1 either.

It is difficult to say, so this may be an ARIMA(3, 1, 2) with parameters $\phi_1 = 1.8, \phi_2 = -2.6, \phi_3 = 0.8, \theta_1 = -0.76,$, and $\theta_2 = -0.2$.

(b) $Y_t = -0.4Y_{t-1} + Y_{t-4} + 0.4Y_{t-5} + e_t - 0.6e_{t-1} - 0.5e_{t-4} + 0.3e_{t-5}$

We can rewrite this as

$$Y_t - Y_{t-4} = 0.4(Y_{t-1} - Y_{t-5}) + e_t - 0.6e_{t-1} - 0.5e_{t-4} + 0.3e_{t-5}$$

This is a multiplicative seasonal model, ARIMA$(1, 0, 3) \times (0, 1, 0)_4$ with $\Phi_1 = 0.4$ and $\theta_1 = 0.6, \theta_2 = 0.5$, and $\theta_3 = 0.3$.

■

3. Consider an ARMA(4, 3) model:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \phi_4 Y_{t-4} + e_t - \theta_1 e_{t-1} - \theta_3 e_{t-3}$$

where $\{e_t\}$ is a sequence of iid random variables with zero mean and unit variance.

(a) Find expressions for the ψ-weights: $\psi_1, \psi_2, \psi_3, \psi_4$, and ψ_5, in terms of the ϕ's and θ's. Find a recursive equation for ψ_k for $k > 5$.

We have the following relation for a general linear process,

$$\psi_j = \theta_j + \sum_{k=1}^{p} \phi_k \psi_{j-k}$$

$$\begin{aligned}
\psi_0 &= 1 \\
\psi_1 &= \theta_1 + \phi_1 \psi_0 \\
\psi_2 &= \phi_1 \psi_0 + \phi_2 \psi_1 \\
\psi_3 &= \theta_3 + \phi_1 \psi_0 + \phi_2 \psi_1 \\
\psi_4 &= \phi_1 \psi_0 + \phi_2 \psi_1 + \phi_4 \psi_3 \\
\psi_5 &= \phi_1 \psi_0 + \phi_2 \psi_1 + \phi_4 \psi_3 + \phi_5 \psi_4 \\
\psi_j &= \phi_1 \psi_0 + \phi_2 \psi_1 + \phi_4 \psi_3 + \phi_5 \psi_4 + \ldots + \phi_j \psi_{j-k}
\end{aligned}$$

(b) Find a system of 5 equations involving $\gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4$ (and perhaps the ϕ's θ's and ψ's). You do not need to solve these equations.

Source: https://stats.stackexchange.com/questions/68644/autocovariance-of-an-arma2-1-process-derivation-of-analytical-mode

$$\gamma_k - \phi_1 \gamma_{k-1} - \ldots - \phi_p \gamma_{k-p} = 0 \text{ for } k \geq max(p, q+1)$$

with initial conditions

$$\gamma_k - \sum_{j=1}^{p} \phi_j \gamma_{k-j} = \sigma_e^2 \sum_{j=k}^{q} \theta_j \psi_{j-k} \text{ for } 0 \leq k < max(p, q+1)$$

So we have

$$\gamma_4 - \phi_1 \gamma_3 - \phi_2 \gamma_2 - \phi_3 \gamma_1 - \phi_4 \gamma_0 = 0$$

$$\gamma_4 = \sigma_e^2 \sum_{j=4}^{3} \theta_j \psi_{j-4} + \sum_{j=1}^{4} \phi_j \gamma_{4-j}$$
$$\gamma_3 = \sigma_e^2 \sum_{j=4}^{3} \theta_j \psi_{j-4} + \sum_{j=1}^{4} \phi_j \gamma_{3-j}$$
$$\gamma_2 = \sigma_e^2 \sum_{j=4}^{3} \theta_j \psi_{j-4} + \sum_{j=1}^{4} \phi_j \gamma_{2-j}$$
$$\gamma_1 = \sigma_e^2 \sum_{j=4}^{3} \theta_j \psi_{j-4} + \sum_{j=1}^{4} \phi_j \gamma_{1-j}$$
$$\gamma_0 = \sigma_e^2 \sum_{j=4}^{3} \theta_j \psi_{j-4} + \sum_{j=1}^{4} \phi_j \gamma_{0-j}$$

(c) Find a recursive equation for ρ_k for $k \geq 5$.

$$\rho_k = \phi_1 \rho_{k-1} + \phi_2 \rho_{k-2} + \ldots + \phi_p \rho_{k-p} \text{ for } k > q$$

■

4. Consider the following model:

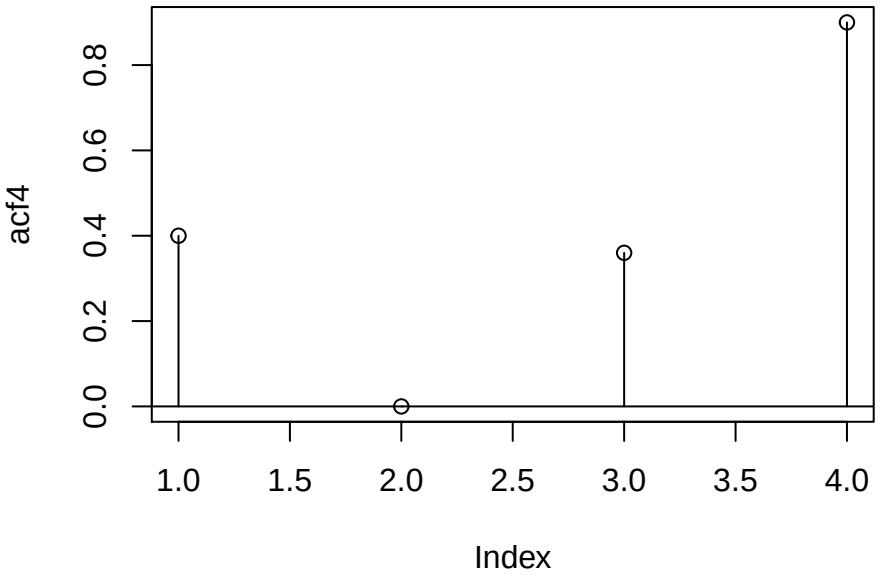
$$Y_t = 0.9Y_{t-4} + e_t - 0.5e_{t-1}$$

(a) Find the autocorrelation function, and evaluate ρ_1, ρ_2, ρ_3, and ρ_4.

```
acf4 <- ARMAacf(ar=c(0, 0, 0, 0.9), ma=c(0.5))
acf4
```

```
##    0    1    2    3    4
## 1.00 0.40 0.00 0.36 0.90
```

```
acf4 <- acf4[-1]
plot(acf4, type='h')
abline(h=0)
points(acf4, type='p')
```




```
rho1 <- acf4[1]
rho2 <- acf4[2]
rho3 <- acf4[3]
rho4 <- acf4[4]
```

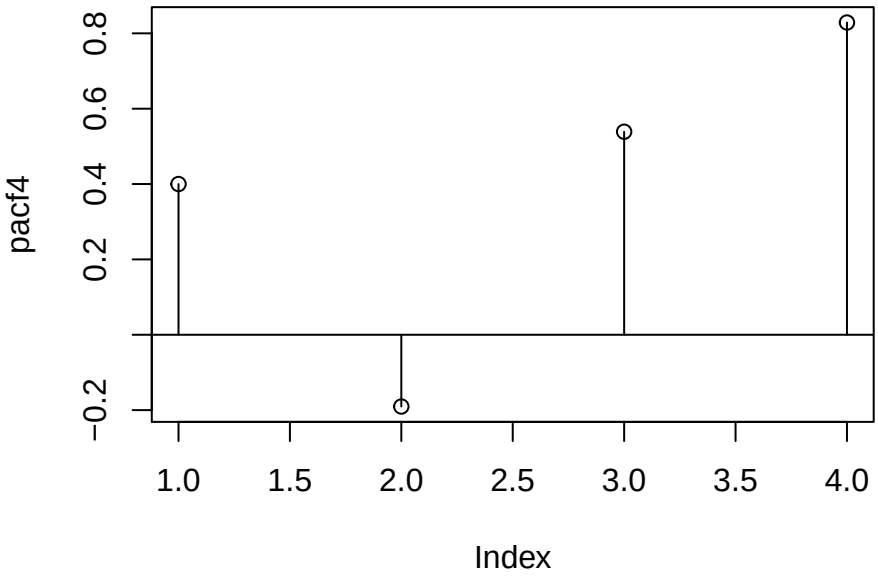
$$\rho_k = \begin{cases} 1 & , k = 0 \\ 0.4 & , k = 1 \\ 0 & , k = 2 \\ 0.36 & , k = 3 \\ 0.9 & , k = 4 \\ 0 & , k > 4 \end{cases}$$

(b) Find the partial autocorrelation function, and evaluate $\phi_{11}, \phi_{22}, \phi_{33}$, and ϕ_{44}. Show all working.

```
pacf4 <- ARMAacf(ar=c(0, 0, 0, 0.9), ma=c(0.5), pacf=TRUE)
pacf4
```

```
## [1]  0.4000000 -0.1904762  0.5388235  0.8287189
```

```
plot(pacf4, type='h')
abline(h=0)
points(pacf4, type='p')
```




```
phi11 <- pacf4[1]
phi22 <- pacf4[2]
phi33 <- pacf4[3]
phi44 <- pacf4[4]
```

$$\begin{aligned}\phi_{11} &= 0.4\\ \phi_{22} &= -0.1904762\\ \phi_{33} &= 0.5388235\\ \phi_{44} &= 0.8287189\end{aligned}$$

(c) Suppose that we have 100 observations generated by this process. If the last 5 observations were 24, 17, 18, 20, and 25 with corresponding residuals 0, 1, 0, -1, and 1, compute the forecasts for the next 5 observations.

From equation 9.3.28, we get

$$\begin{aligned}\hat{Y}_t(1) &= \phi_1\hat{Y}_t(0) + \phi_2\hat{Y}_t(-1) + \phi_3\hat{Y}_t(-2) + \phi_4\hat{Y}_t(-3) + \theta_0 - \theta_1 E[e_{t-1}|Y_t, Y_2, Y_3, Y_4]\\ &= 0.9(18) - 0.5\\ &= 15.7\end{aligned}$$

Once we have $\hat{Y}_t(1)$, we can use equation 9.3.30 in conjunction with this and obtain the general values for arbitrary $\hat{Y}_t(l)$.

$$
\begin{aligned}
\hat{Y}_t(1) &= \phi\hat{Y}_t + \theta_0 - \theta e_t \\
\hat{Y}_t(l) &= \phi\hat{Y}_t(l-1) + \theta_0 \\
\hat{Y}_t(2) &= 0.9(13.5) + 1 \\
&= 13.15 \\
\hat{Y}_t(3) &= 0.9(13.15) + 1 \\
&= 12.5515 \\
\hat{Y}_t(4) &= 0.9(13.15) + 1 \\
&= 12.29635 \\
\hat{Y}_t(5) &= 0.9(13.15) + 1 \\
&= 12.066715
\end{aligned}
$$

(d) If $\sigma_e^2 = 1$, calculate the variance of each of the forecasts made in (4c)

Using 9.3.38, we have $Var(e_t(l)) = \sigma_e^2 \sum_{j=0}^{l-1} \psi_j^2$ for $l \geq 1$.

We need the ψ-weights, so we can obtain them from

```
ARMAtoMA(ar=c(0, 0, 0, 0.9), ma=c(0.5), lag.max=5)
```

```
## [1] 0.50 0.00 0.00 0.90 0.45
```

So we have

$$
\begin{aligned}
\psi_0 &= 1 \\
\psi_1 &= 0.5 \\
\psi_2 &= 0 \\
\psi_3 &= 0 \\
\psi_4 &= 0.9 \\
\psi_5 &= 0.45
\end{aligned}
$$

Which gives us

Variance for 1 Day Out

$$
\begin{aligned}
\psi_0^2 &= 1^2 \\
&= 1
\end{aligned}
$$

Variance for 2 Days Out

$$
\begin{aligned}
\sum_{j=0}^{1} \psi_j^2 &= 1 + (0.5)^2 + 0 \\
&= 1.25
\end{aligned}
$$

Variance for 3 Days Out

$$\sum_{j=0}^{2} \psi_j^2 = 1 + (0.5)^2 + 0 + 0$$
$$= 1.25$$

Variance for 4 Days Out

$$\sum_{j=0}^{3} \psi_j^2 = 1 + (0.5)^2 + 0 + 0$$
$$= 1.25$$

Variance for 4 Days Out

$$\sum_{j=0}^{4} \psi_j^2 = 1 + (0.5)^2 + 0 + 0 + (0.9)^2$$
$$= 1 + 0.25 + 0.81 = 2.06$$

(e) Calculate 95% prediction limits for each of your forecasts.

Prediction Limits for 1 Day Out

$$(13.74, 17.66)$$

Prediction Limits for 2 Day Out

$$(10.7, 15.6)$$

Prediction Limits for 3 Day Out

$$(10.1015, 15.0015)$$

Prediction Limits for 4 Day Out

$$(9.84635, 14.74635)$$

Prediction Limits for 5 Day Out

$$(8.029115, 14.516715)$$

■

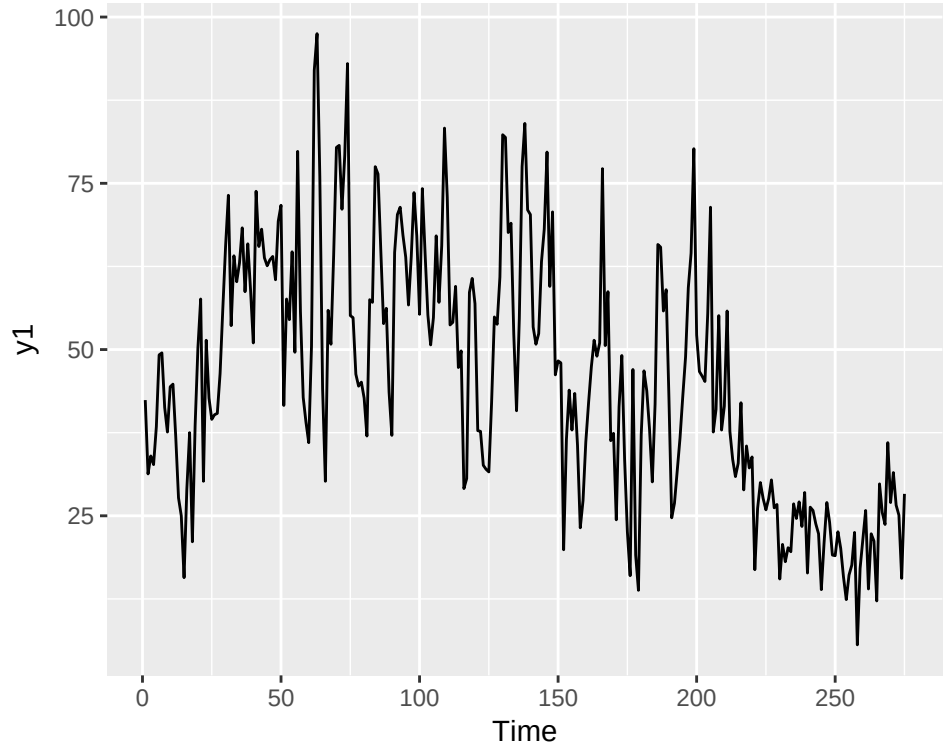
5. The file "`FinalExam.RData`" contains three datasets: `y1`, `y2`, and `y3`. You can load the data into R using the command `load("FinalExam.RData")`. Note: If you fit a model and later decide that the model is unsuitable, do not remove the model from your answer. Include it, discuss why it was unsuitable, and then move on the another model.

```
load("FinalExam.RData")
```

(a) Fit and appropriate model to `y1`. Use the third and fourth Bootstrap resampling methods to get the 95% confidence intervals for the parameters in your model.

The first thing we always do when fitting a model is to visualize the data. This time series doesn't look immediately stationary.

```
autoplot(y1)
```

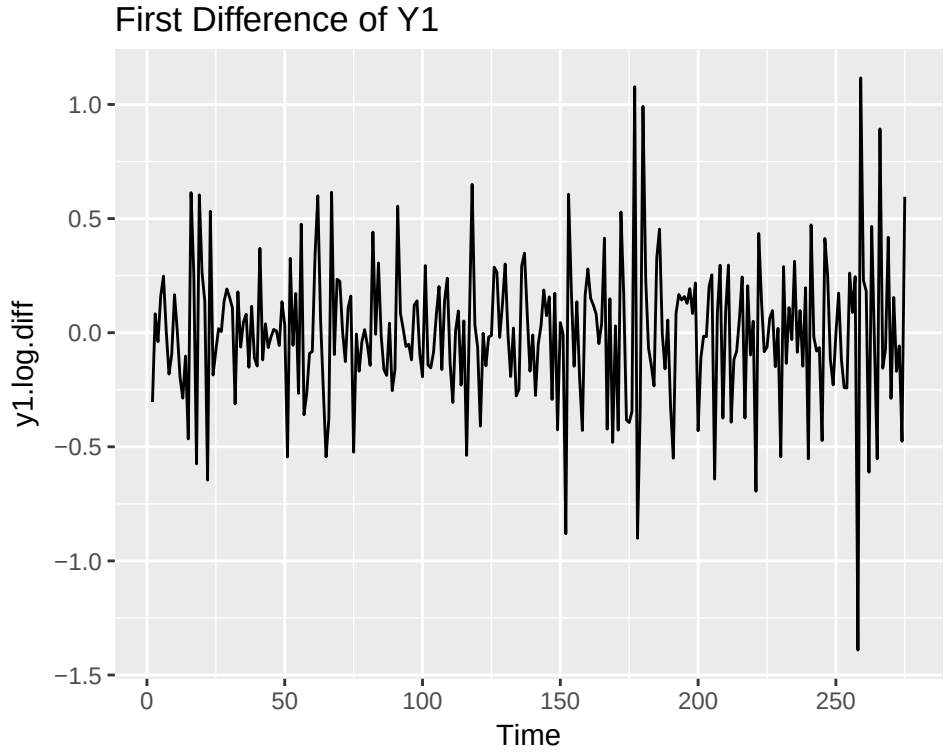



```
adf.test(y1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y1
## Dickey-Fuller = -3.4662, Lag order = 6, p-value = 0.04643
## alternative hypothesis: stationary
```

An augmented Dickey-Fuller test results in a p-value of 0.04, but we can possibly do a little better with a log difference.

```
y1.log.diff <- diff(log(y1))
autoplot(y1.log.diff) + labs(title="First Difference of Y1")
```



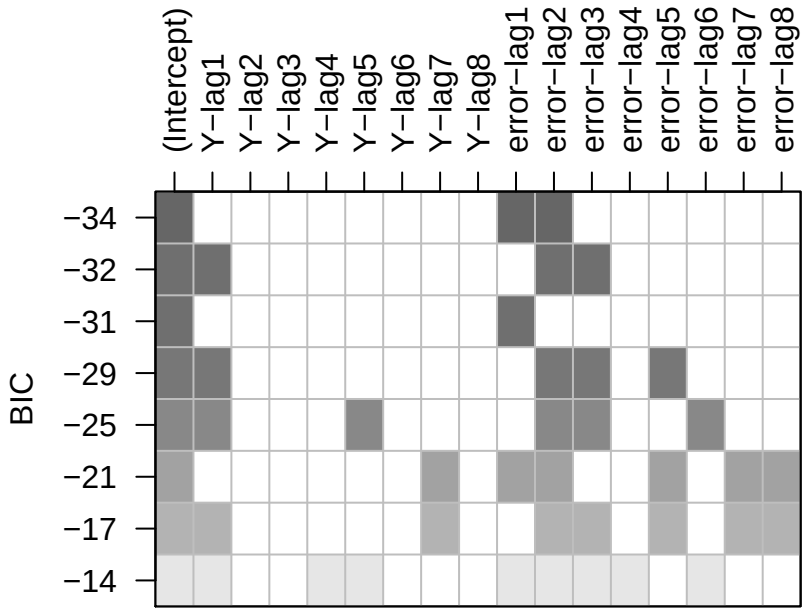

```
adf.test(y1.log.diff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y1.log.diff
## Dickey-Fuller = -9.0867, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

This series looks much more stationary, and the ADF test gives us a very small p-value.

Now we can use `armasubsets()` and `auto.arima()` to obtain the best ARMA model for the first difference.

```
subs <- armasubsets(y1.log.diff, nar = 8, nma = 8)
plot(subs)
```




```
auto.arima(y1.log.diff)
```

```
## Series: y1.log.diff
## ARIMA(1,0,1) with zero mean
##
## Coefficients:
##          ar1      ma1
##       0.4330  -0.8766
## s.e.  0.0811   0.0465
##
## sigma^2 estimated as 0.0783:  log likelihood=-39.17
## AIC=84.33   AICc=84.42   BIC=95.17
```

`armasubsets()` gives us a best fit of MA(2) on the log difference, whereas `auto.arima()` gives ARMA(1, 1).

```
f1 <- Arima(y=y1.log.diff, order = c(1, 0, 1))
f1
```

```
## Series: y1.log.diff
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ma1     mean
##       0.4375  -0.8811  -0.0019
## s.e.  0.0821   0.0475   0.0037
##
## sigma^2 estimated as 0.07851:  log likelihood=-39.04
## AIC=86.09   AICc=86.24   BIC=100.54
```

```
f2 <- Arima(y=y1.log.diff, order = c(0, 0, 2))
f2
```

```
## Series: y1.log.diff
## ARIMA(0,0,2) with non-zero mean
##
## Coefficients:
##          ma1      ma2     mean
```

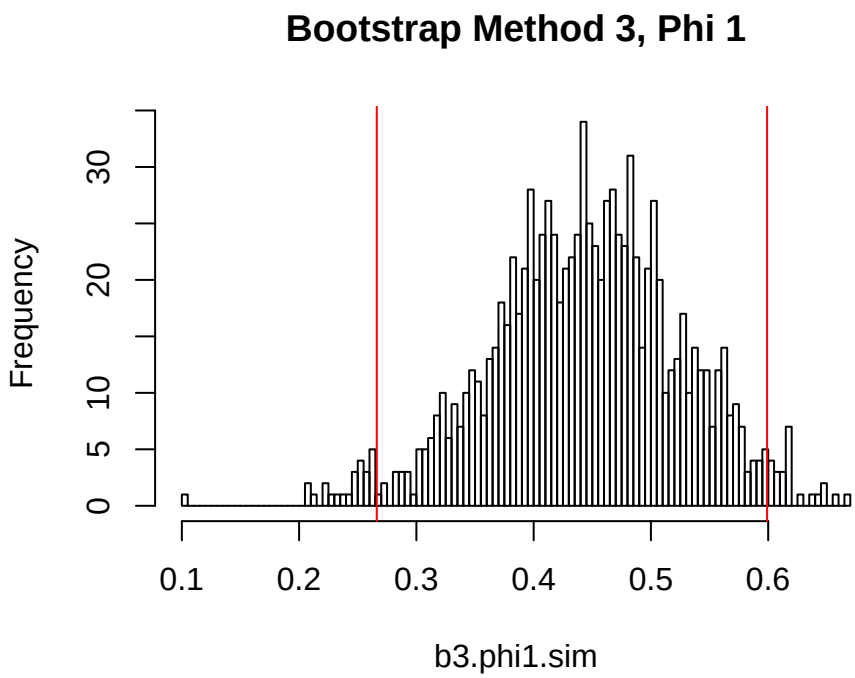
```
##       -0.4623  -0.2512  -0.0016
## s.e.   0.0572   0.0597   0.0049
##
## sigma^2 estimated as 0.0796:  log likelihood=-40.88
## AIC=89.75   AICc=89.9   BIC=104.2
```

Looking at the AICs for both of these models, we can see that the ARMA(1, 1) for the log difference gives us the lower AIC, so we will prefer this model.

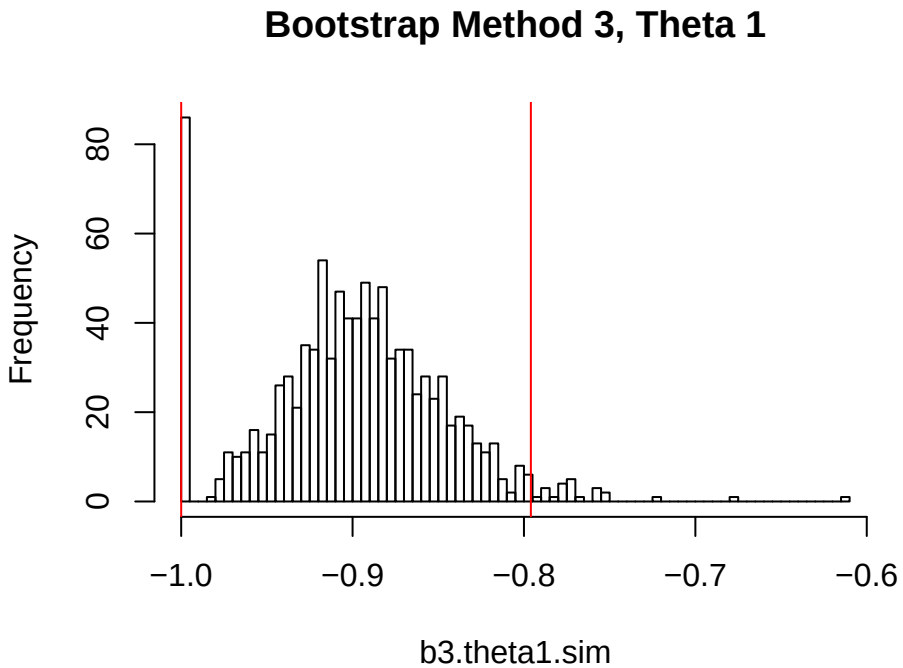
We'll bootstrap the data to get a confidence interval for our parameters.

Bootstrap Method 3

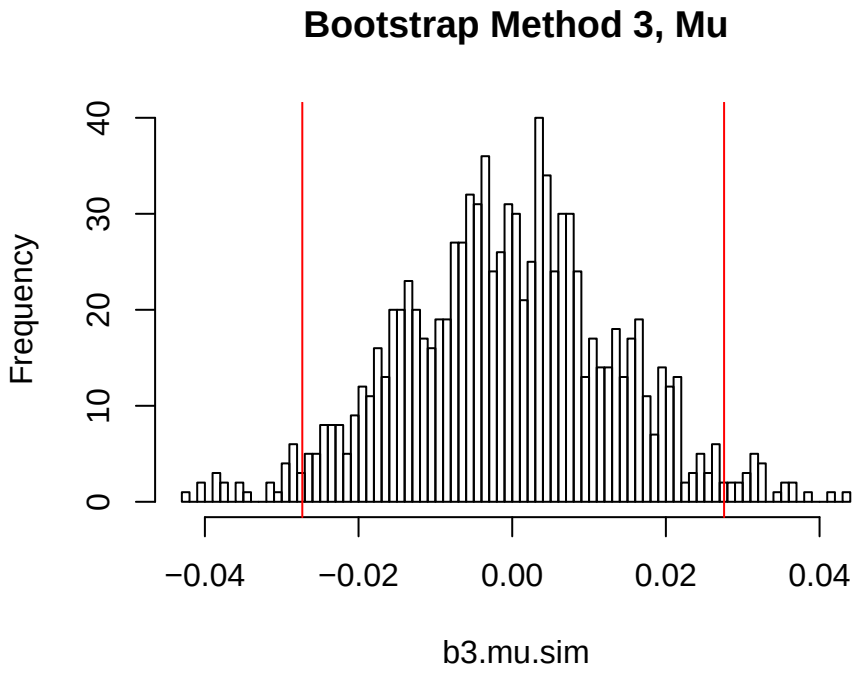
```
sims <- 1000
data.points <- 1000
b3.phi1.sim <- rep(NA, sims)
b3.theta1.sim <- rep(NA, sims)
b3.mu.sim <- rep(NA, sims)
b3.sigmae2.sim <- rep(NA, sims)

b3 <- y1.log.diff
y1.order <- c(1, 0, 1)
f <- arima(b3, order=y1.order)
coef <- f$coef

for (j in 1:sims) {

  b3 <- arima.sim(n=data.points, model=list(order=y1.order, ar=coef[1], ma=coef[2], sd=sqrt(f$sigma2)))
  b3 <- b3[-(1:(data.points-length(y1)))]

  f1 <- arima(b3, order=y1.order, method="ML")

  b3.phi1.sim[j] <- f1$coef[[1]]
  b3.theta1.sim[j] <- f1$coef[[2]]
  b3.mu.sim[j] <- mean(b3)
  b3.sigmae2.sim[j] <- var(resid(f1))

}

hist(b3.phi1.sim, main="Bootstrap Method 3, Phi 1", breaks=100)
phi1.ci.lower <- quantile(b3.phi1.sim, c(0.025,0.975))[[1]]
phi1.ci.upper <- quantile(b3.phi1.sim, c(0.025,0.975))[[2]]
abline(v=phi1.ci.lower, col="red")
abline(v=phi1.ci.upper, col="red")
```



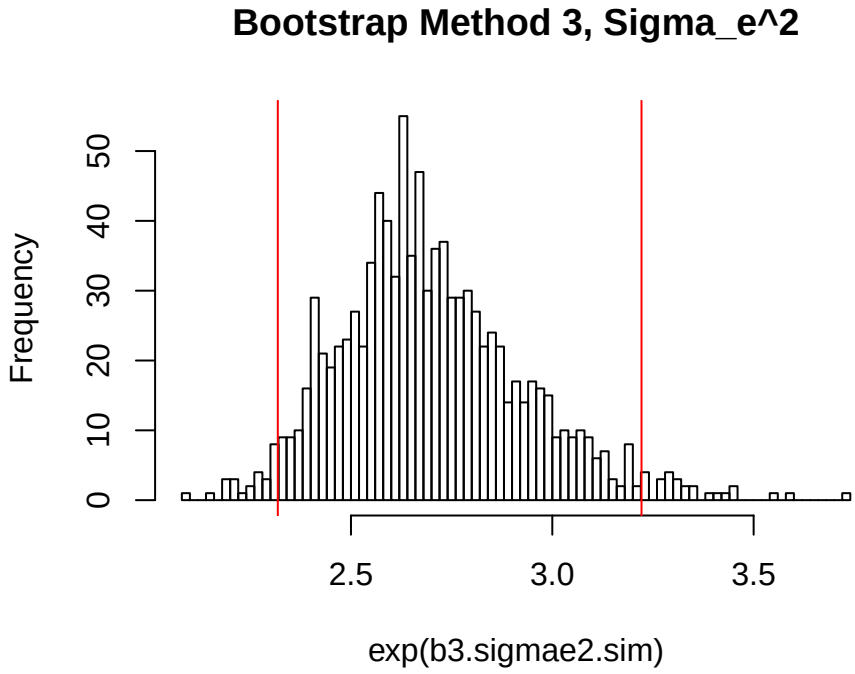

```
hist(b3.theta1.sim, main="Bootstrap Method 3, Theta 1", breaks=100)
theta1.ci.lower <- quantile(b3.theta1.sim, c(0.025,0.975))[[1]]
theta1.ci.upper <- quantile(b3.theta1.sim, c(0.025,0.975))[[2]]
abline(v=theta1.ci.lower, col="red")
abline(v=theta1.ci.upper, col="red")
```




```
hist(b3.mu.sim, main="Bootstrap Method 3, Mu", breaks=100)
mu.ci.lower <- quantile(b3.mu.sim, c(0.025,0.975))[[1]]
mu.ci.upper <- quantile(b3.mu.sim, c(0.025,0.975))[[2]]
abline(v=mu.ci.lower, col="red")
abline(v=mu.ci.upper, col="red")
```




```
hist(exp(b3.sigmae2.sim), main="Bootstrap Method 3, Sigma_e^2", breaks=100)
sigmae2.ci.lower <- quantile(exp(b3.sigmae2.sim), c(0.025,0.975))[[1]]
sigmae2.ci.upper <- quantile(exp(b3.sigmae2.sim), c(0.025,0.975))[[2]]
abline(v=sigmae2.ci.lower, col="red")
abline(v=sigmae2.ci.upper, col="red")
```

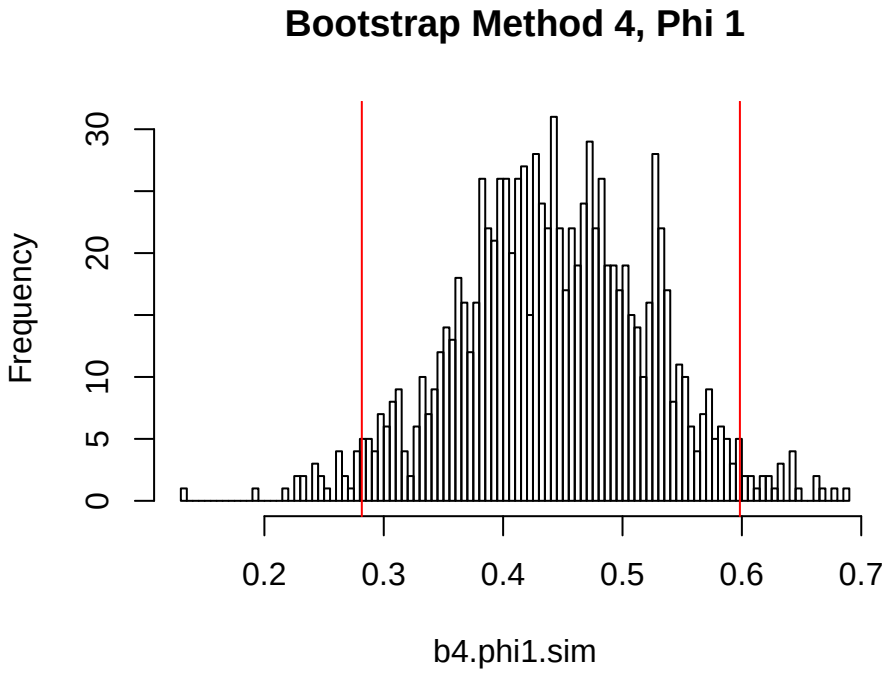


We have the following 95% confidence intervals:

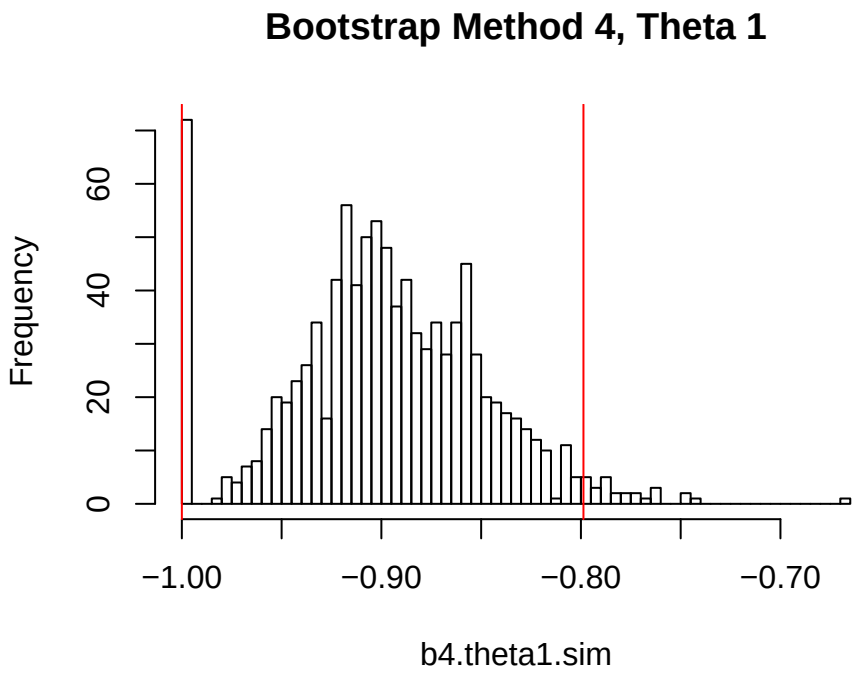
$$\begin{aligned}
\phi_1 &: (0.2662311, 0.599063) \\
\theta_1 &: (-0.9999995, -0.795976) \\
\mu &: (-0.0273214, 0.0275753) \\
\sigma_e^2 &: (2.3182636, 3.2215393)
\end{aligned}$$

Bootstrap Method 4

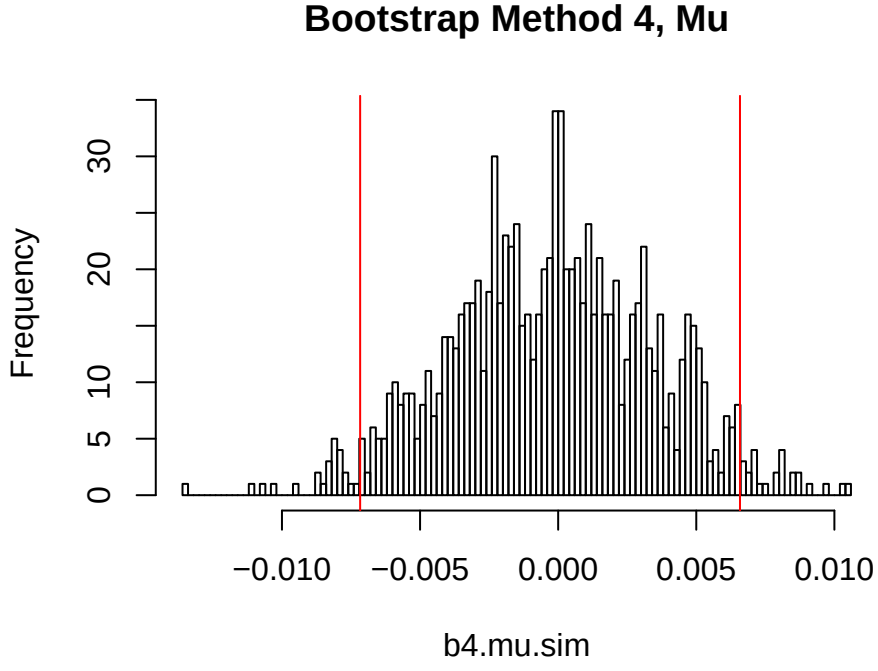
```
b4.phi1.sim <- rep(NA, sims)
b4.theta1.sim <- rep(NA, sims)
b4.mu.sim <- rep(NA, sims)
b4.sigmae2.sim <- rep(NA, sims)

b4 <- y1.log.diff
y1.order <- c(1, 0, 1)
f <- arima(b4, order=y1.order)
coef <- f$coef

for (j in 1:sims) {

  b4 <- arima.sim(n=data.points,
                  model=list(ar=coef[1], ma=coef[2], order=y1.order),
                  rand.gen=function(n, r=f$residuals){ sample(r, n, replace=TRUE) })
  b4 <- b4[-(1:(data.points-length(y1)))]

  f1 <- arima(b4, order=y1.order, method="ML")
  b4.phi1.sim[j] <- f1$coef[[1]]
  b4.theta1.sim[j] <- f1$coef[[2]]
  b4.mu.sim[j] <- mean(b4)
  b4.sigmae2.sim[j] <- var(resid(f1))

}

hist(b4.phi1.sim, main="Bootstrap Method 4, Phi 1", breaks=100)
phi1.ci.lower <- quantile(b4.phi1.sim, c(0.025,0.975))[[1]]
phi1.ci.upper <- quantile(b4.phi1.sim, c(0.025,0.975))[[2]]
abline(v=phi1.ci.lower, col="red")
abline(v=phi1.ci.upper, col="red")
```



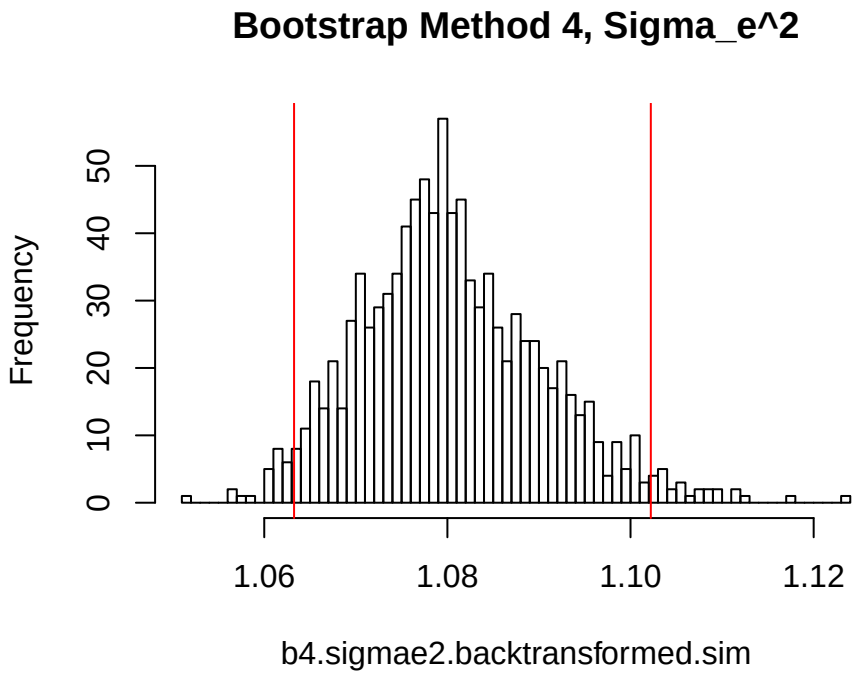
```r
hist(b4.theta1.sim, main="Bootstrap Method 4, Theta 1", breaks=100)
theta1.ci.lower <- quantile(b4.theta1.sim, c(0.025,0.975))[[1]]
theta1.ci.upper <- quantile(b4.theta1.sim, c(0.025,0.975))[[2]]
abline(v=theta1.ci.lower, col="red")
abline(v=theta1.ci.upper, col="red")
```




```r
hist(b4.mu.sim, main="Bootstrap Method 4, Mu", breaks=100)
mu.ci.lower <- quantile(b4.mu.sim, c(0.025,0.975))[[1]]
mu.ci.upper <- quantile(b4.mu.sim, c(0.025,0.975))[[2]]
abline(v=mu.ci.lower, col="red")
abline(v=mu.ci.upper, col="red")
```



```
b4.sigmae2.backtransformed.sim <- exp(b4.sigmae2.sim)
hist(b4.sigmae2.backtransformed.sim, main="Bootstrap Method 4, Sigma_e^2", breaks=100)
sigmae2.ci.lower <- quantile(b4.sigmae2.backtransformed.sim, c(0.025,0.975))[[1]]
sigmae2.ci.upper <- quantile(b4.sigmae2.backtransformed.sim, c(0.025,0.975))[[2]]
abline(v=sigmae2.ci.lower, col="red")
abline(v=sigmae2.ci.upper, col="red")
```

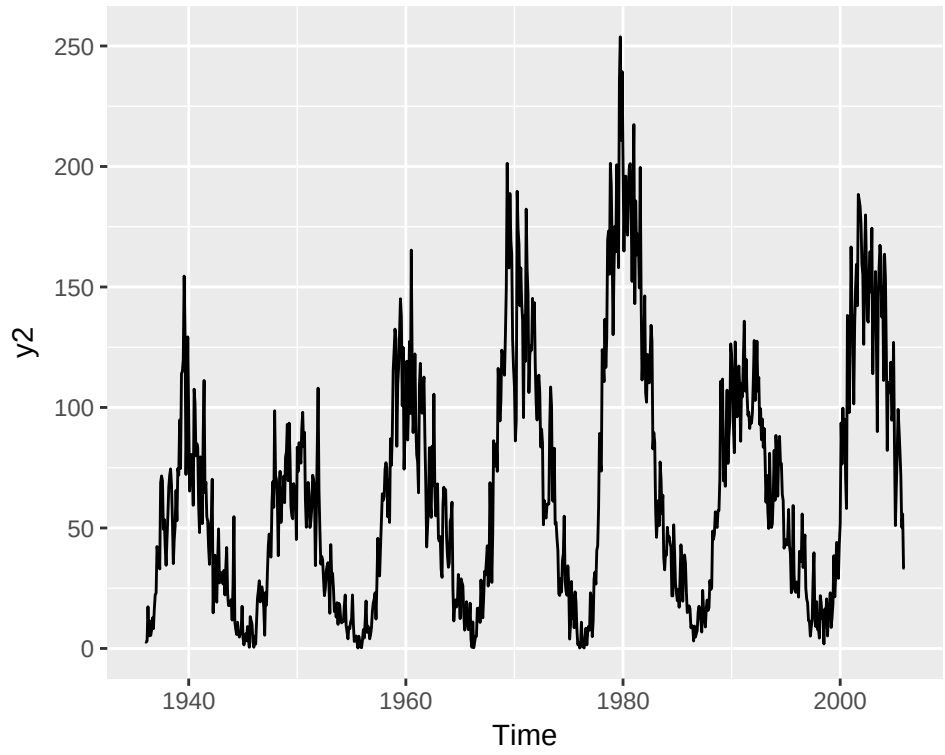


We have the following 95% confidence intervals:

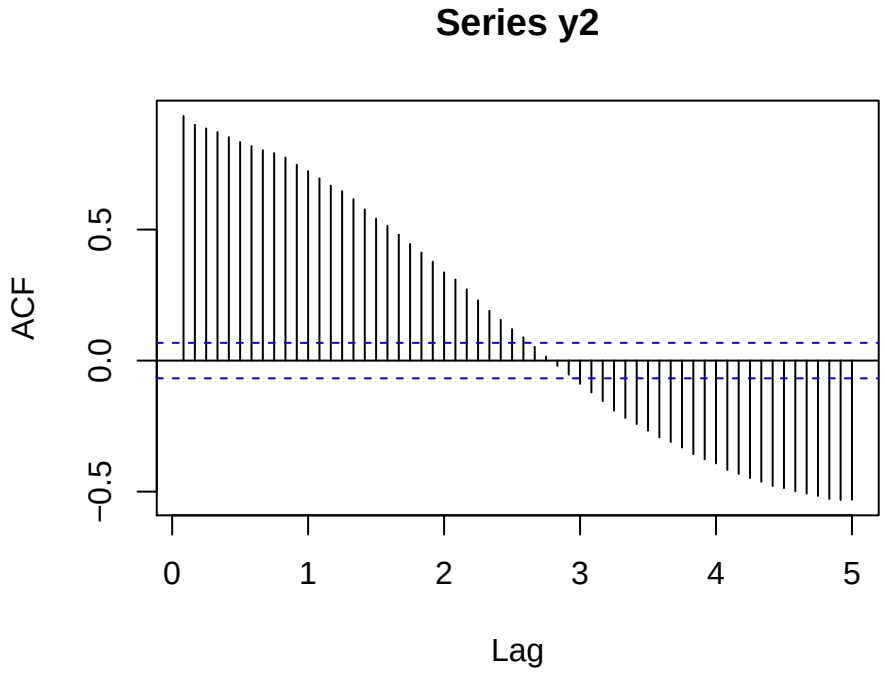
$$\begin{aligned}\phi_1 &: (0.2816003, 0.5983535)\\ \theta_1 &: (-0.9999995, -0.7987233)\\ \mu &: (-0.007169, 0.0065823)\\ \sigma_e^2 &: (1.0632569, 1.1022142)\end{aligned}$$

(b) Fit an appropriate model to y2. Predict the next 10 observations with 95% prediction limits.

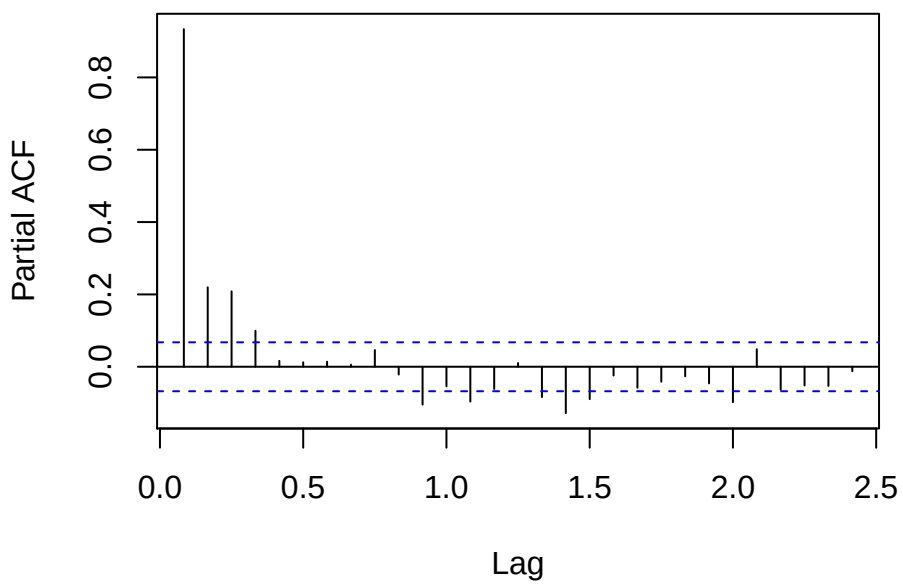
```
autoplot(y2)
```




```
acf(y2, lag.max = 60)
```




```
pacf(y2)
```

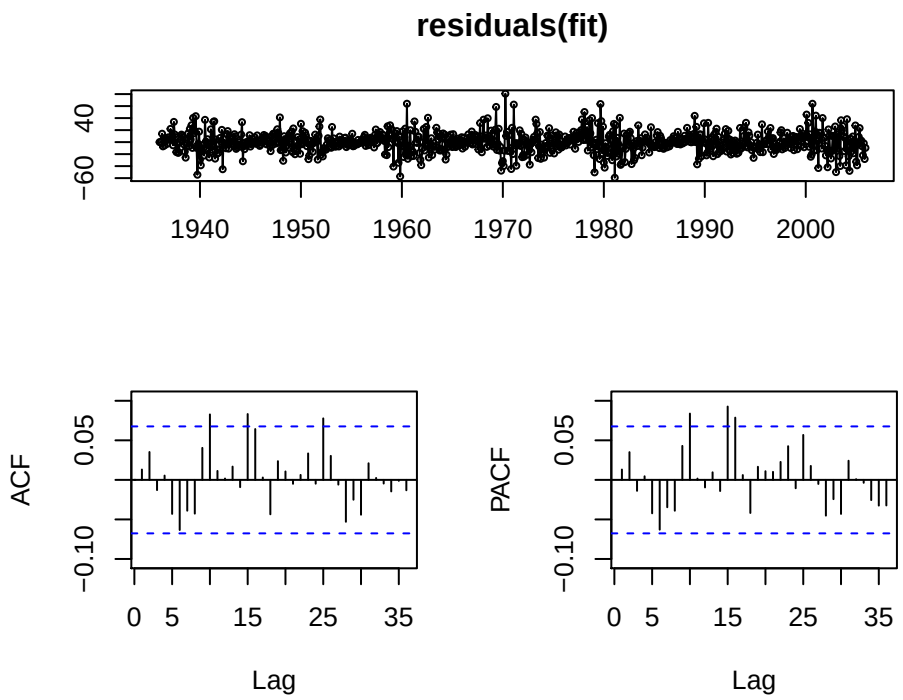


It is clear we have a seasonal model here, but there are some problematic lags in the PACF. We'll get some help from the `auto.arima()` to get a starting model.

```
fit <- auto.arima(y2)
fit
```

```
## Series: y2
## ARIMA(1,1,3)(0,0,2)[12]
##
## Coefficients:
##          ar1      ma1     ma2     ma3    sma1     sma2
##       0.9482  -1.3128  0.1380  0.2225  0.0485  -0.0640
## s.e.  0.0201   0.0394  0.0548  0.0371  0.0354   0.0337
##
## sigma^2 estimated as 296.7:  log likelihood=-3575.82
## AIC=7165.64   AICc=7165.78   BIC=7198.77
```

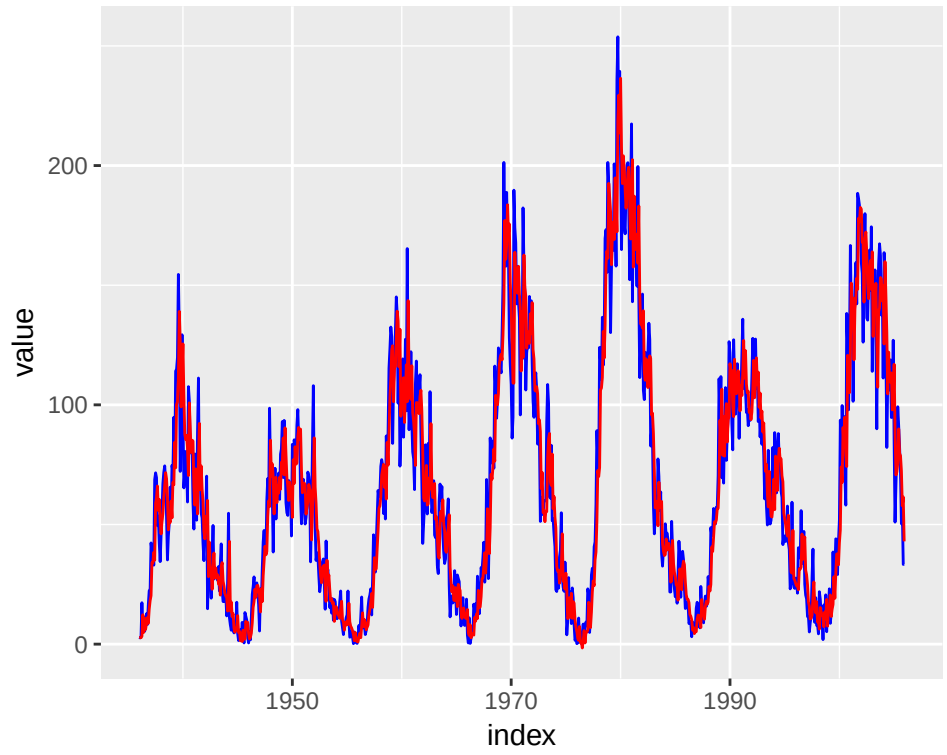
```
# fit <- Arima(y2, order = c(1, 1, 3), seasonal=list(order=c(0, 0, 2), period=12) )

tsdisplay(residuals(fit))
```




```
y2.df <- data.frame(index=index(y2),
                    value=coredata(y2),
                    fitted=fitted(fit),
                    resid.stand=rstandard(fit))
```

`auto.arima()` suggests an $\text{ARIMA}(1,1,3) \times (0,0,2)_{12}$.

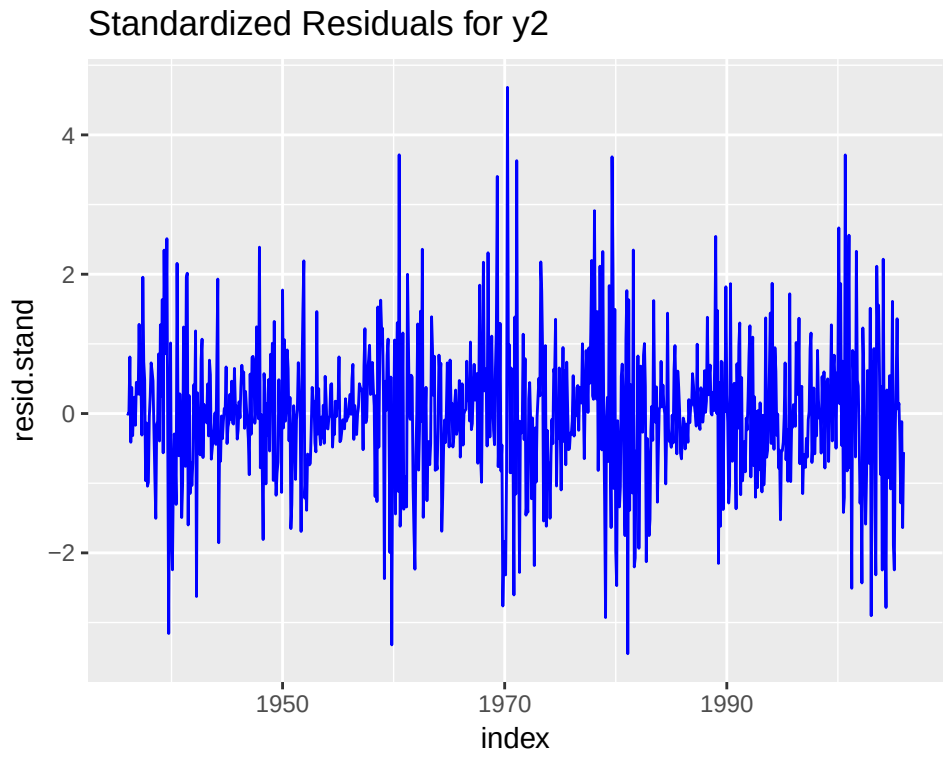
```
ggplot(y2.df, aes(x=index, y=value)) +
  geom_line(col="blue") +
  geom_line(aes(x=index, y=fitted), col="red")
```



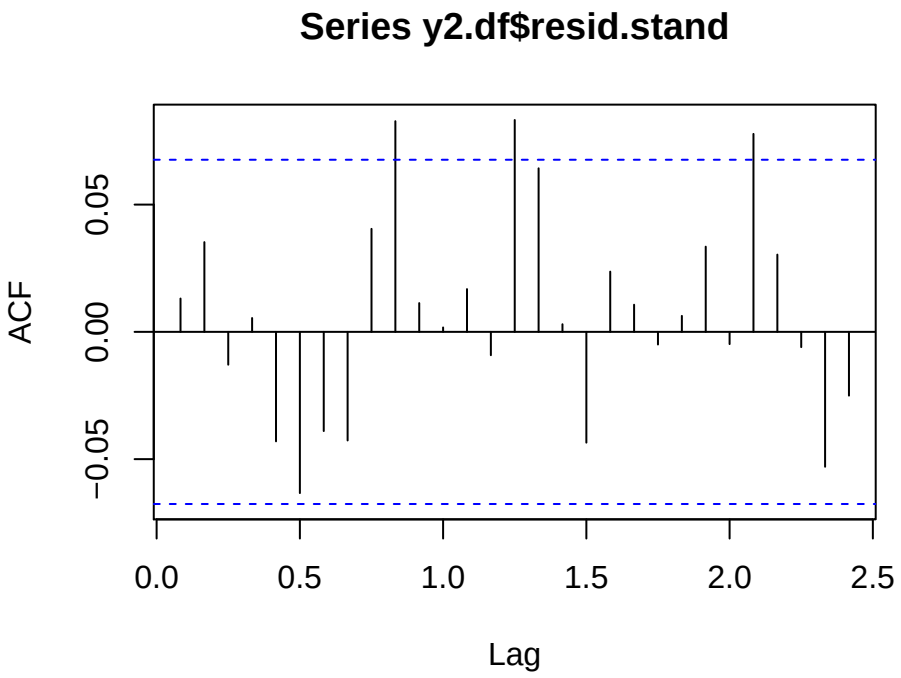
We can do an analysis on the residuals to verify our model:

```
ggplot(y2.df, aes(x=index, y=resid.stand)) +
  geom_line(col="blue") +
  labs(title="Standardized Residuals for y2")
```

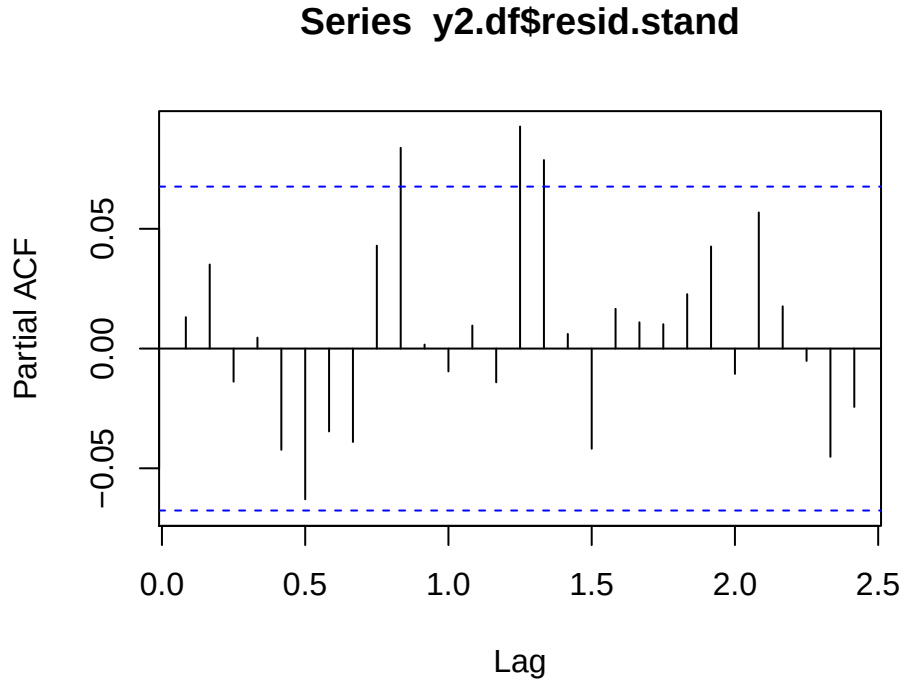
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



```
acf(y2.df$resid.stand)
```

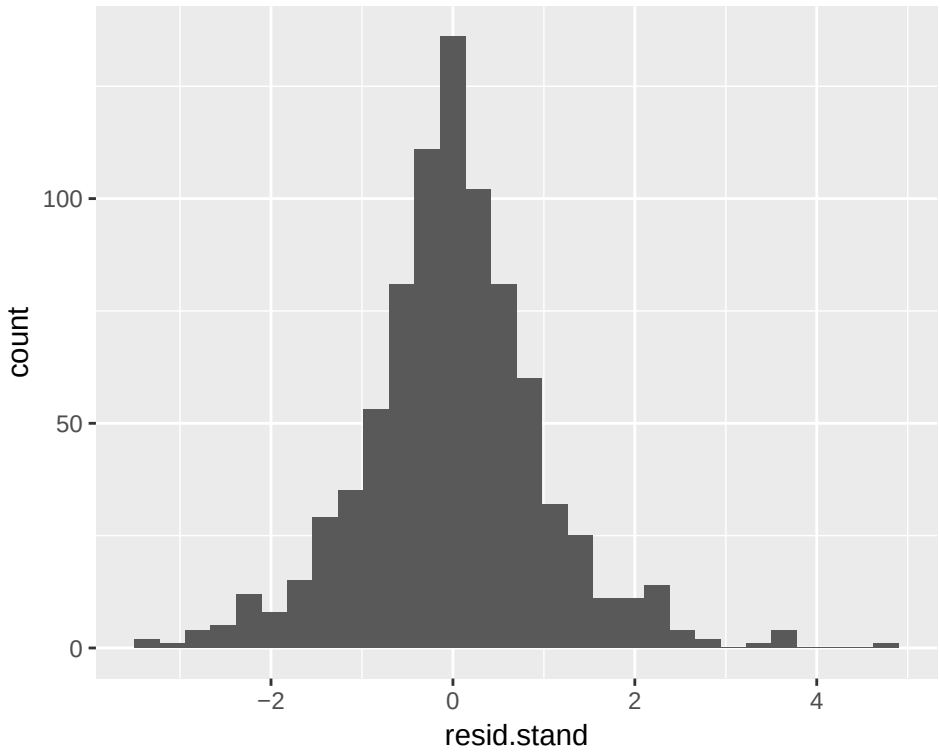



```
pacf(y2.df$resid.stand)
```




```
ggplot(y2.df, aes(x=resid.stand)) +
  geom_histogram(bins = 30)
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```




```
shapiro.test(y2.df$resid.stand)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  y2.df$resid.stand
## W = 0.976, p-value = 1.582e-10
```

```
Box.test(y2.df$resid.stand, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  y2.df$resid.stand
## X-squared = 0.14474, df = 1, p-value = 0.7036
```

Unfortunately, we have some significant autocorrelation in the residuals, so this model is not perfect, but performing a Shapiro-Wilk and Ljung-Box test suggest that they are iid normal.

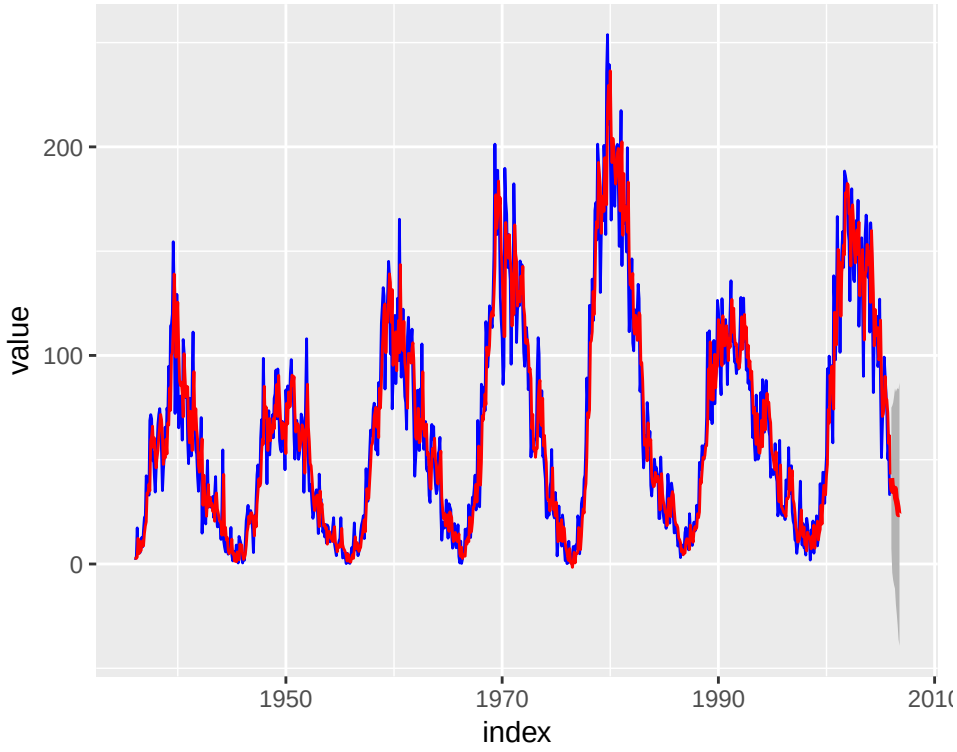
Now we will predict the next 10 observations with 95% prediction limits:

```
pred <- forecast(y2, model = fit, h = 10)

pred.df <- data.frame(index=index(pred$upper), mean=coredata(pred$mean), lower=coredata(pred$lower[,2])

pred.df
```

```
##       index     mean      lower    upper
## 1  2006.000 41.19051   7.432847 74.94817
## 2  2006.083 35.83900  -4.156458 75.83445
## 3  2006.167 34.81234  -7.708356 77.33304
## 4  2006.250 35.14334 -10.045301 80.33199
## 5  2006.333 36.17997 -11.795991 84.15594
```

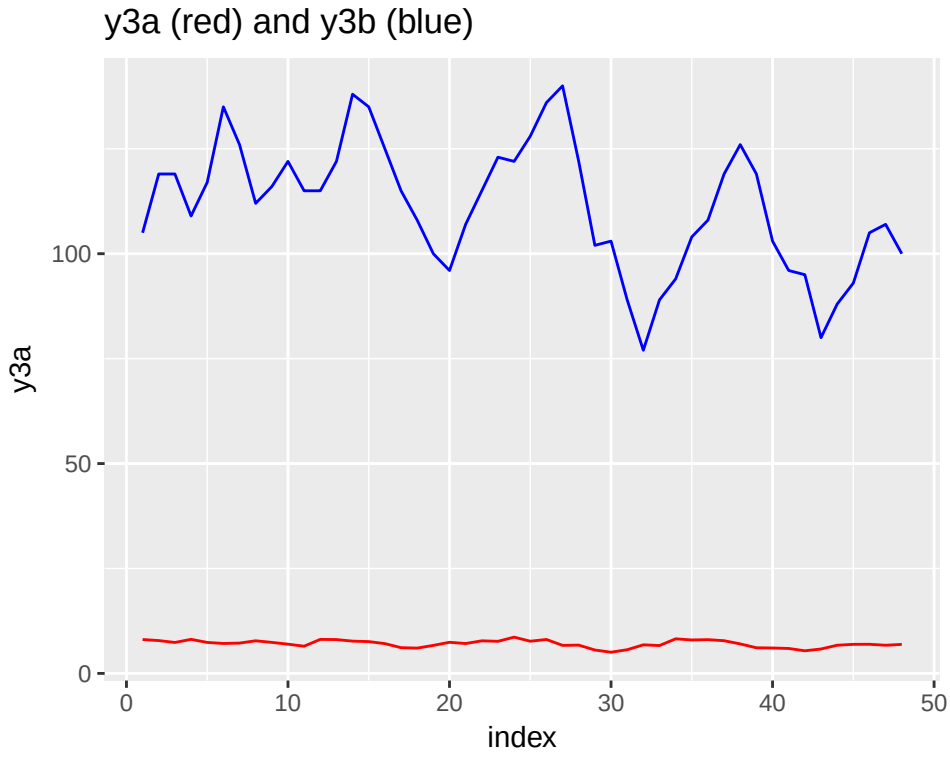
```
## 6  2006.417 31.63459 -19.227504 82.49668
## 7  2006.500 29.89408 -23.935044 83.72321
## 8  2006.583 27.44471 -29.416923 84.30633
## 9  2006.667 23.79128 -36.155041 83.73759
## 10 2006.750 23.84236 -39.229451 86.91416
```

```
ggplot(y2.df, aes(x=index)) +
  geom_line(aes(y=value), col="blue") +
  geom_line(aes(y=fitted), col="red") +
  geom_ribbon(data = pred.df,
              aes(ymin=lower, ymax=upper),
              fill = 'grey70') +
  geom_smooth(data = pred.df,
              aes(y=mean),
              colour='red',
              stat='identity')
```



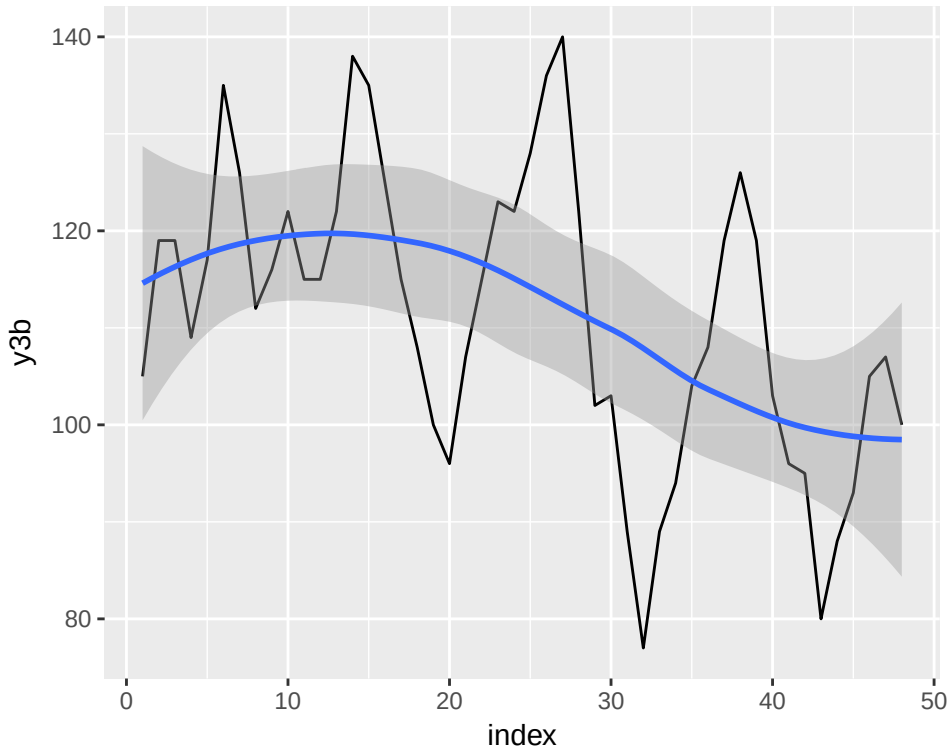
(c) y3 is a data frame containing two time series y3a and y3b.

```
y3 <- cbind(index=index(ts(y3)), y3)

ggplot(y3, aes(x=index, y=y3a)) +
  geom_line(col="red") +
  geom_line(aes(x=index, y=y3b), col="blue") +
  labs(title="y3a (red) and y3b (blue)")
```

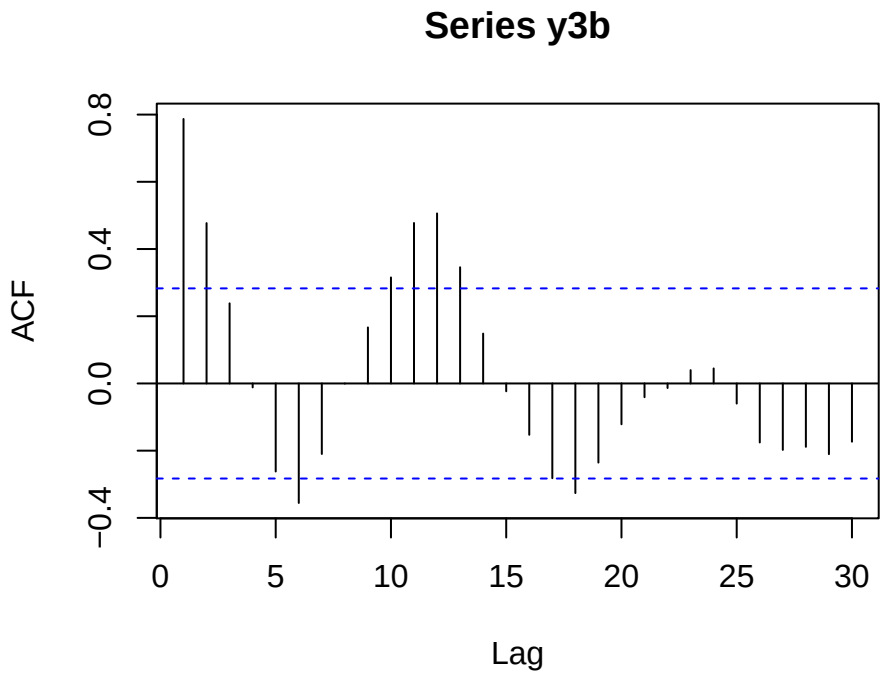


i. Ignore `y3a` and fit an appropriate model to `y3b`. Forecast the next 4 observations wtih 95% prediction limits.

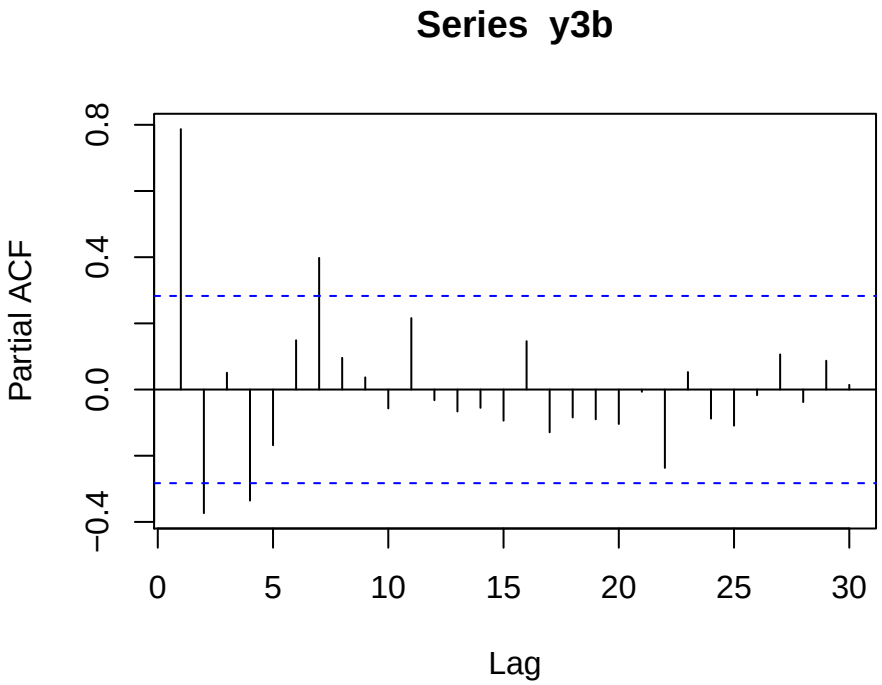
```
y3b <- ts(data = y3$y3b, start=1, end=length(y3$y3b))
ggplot(y3, aes(x=index)) +
  geom_line(aes(y=y3b)) +
  geom_smooth(aes(y=y3b), method = "loess")
```




```
acf(y3b, lag.max=30)
```




```
pacf(y3b, lag.max=30)
```

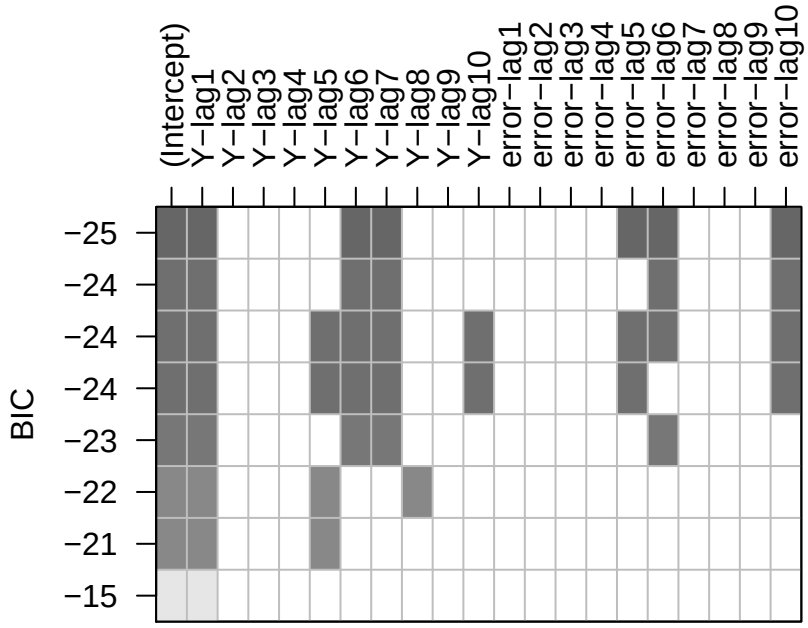



```
adf.test(y3b)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y3b
## Dickey-Fuller = -5.0832, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

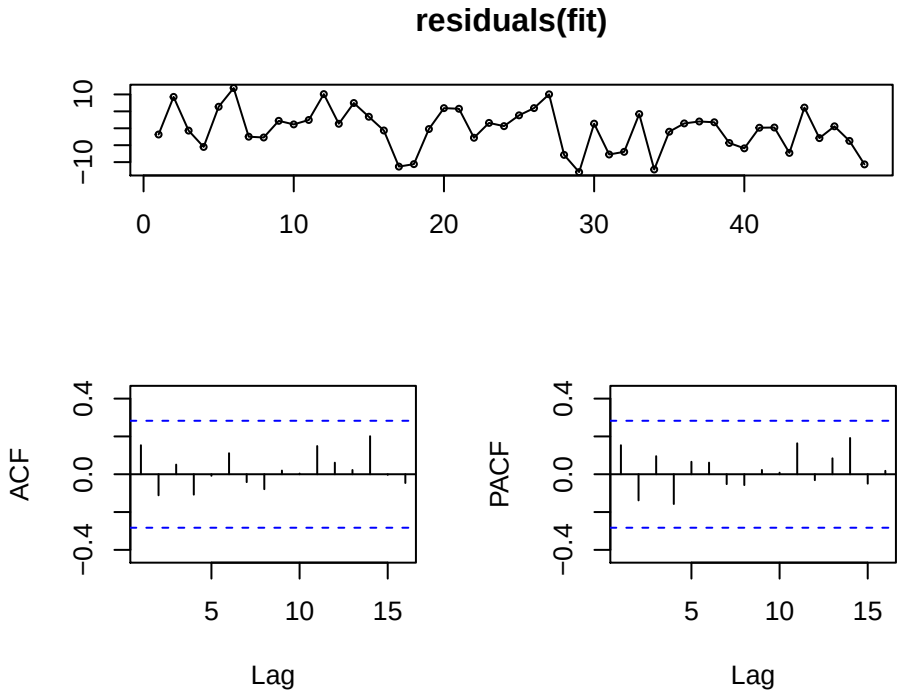
Although the ADF test shows it to be stationary, it doesn't immediately look like it is.

```
subs <- armasubsets(y3b, nar = 10, nma = 10)
plot(subs)
```

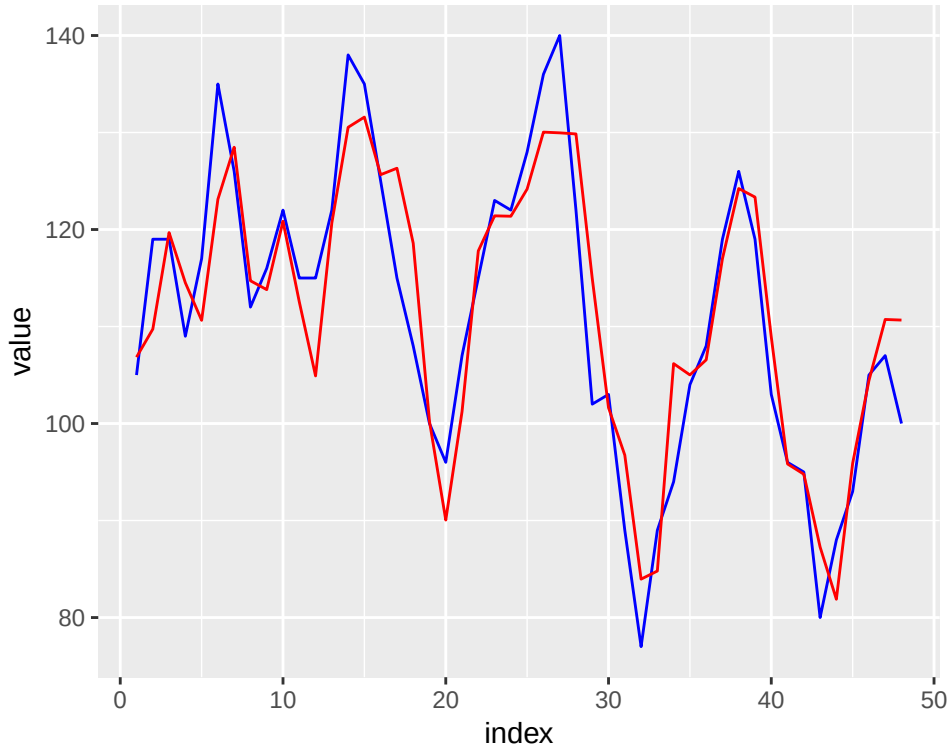


The `armasubsets()` plot suggests an ARMA(7, 10).

```
ar1 <- NA
ar2 <- 0
ar3 <- 0
ar4 <- 0
ar5 <- 0
ar6 <- NA
ar7 <- NA

ma1 <- 0
ma2 <- 0
ma3 <- 0
ma4 <- 0
ma5 <- NA
ma6 <- NA
ma7 <- 0
ma8 <- 0
ma9 <- 0
ma10 <- NA
intercept <- NA

fit <- Arima(y3b,
             order = c(7, 0, 10),
             method = "ML",
             fixed = c(ar1, ar2, ar3, ar4, ar5, ar6, ar7,
                       ma1, ma2, ma3, ma4, ma5, ma6, ma7, ma8, ma9, ma10,
                       intercept),
             transform.pars = FALSE)

tsdisplay(residuals(fit))
```



```
y3b.df <- data.frame(index=index(y3b),
                     value=coredata(y3b),
                     fitted=fitted(fit),
                     resid.stand=rstandard(fit))

ggplot(y3b.df, aes(x=index, y=value)) +
  geom_line(col="blue") +
  geom_line(aes(x=index, y=fitted), col="red")
```

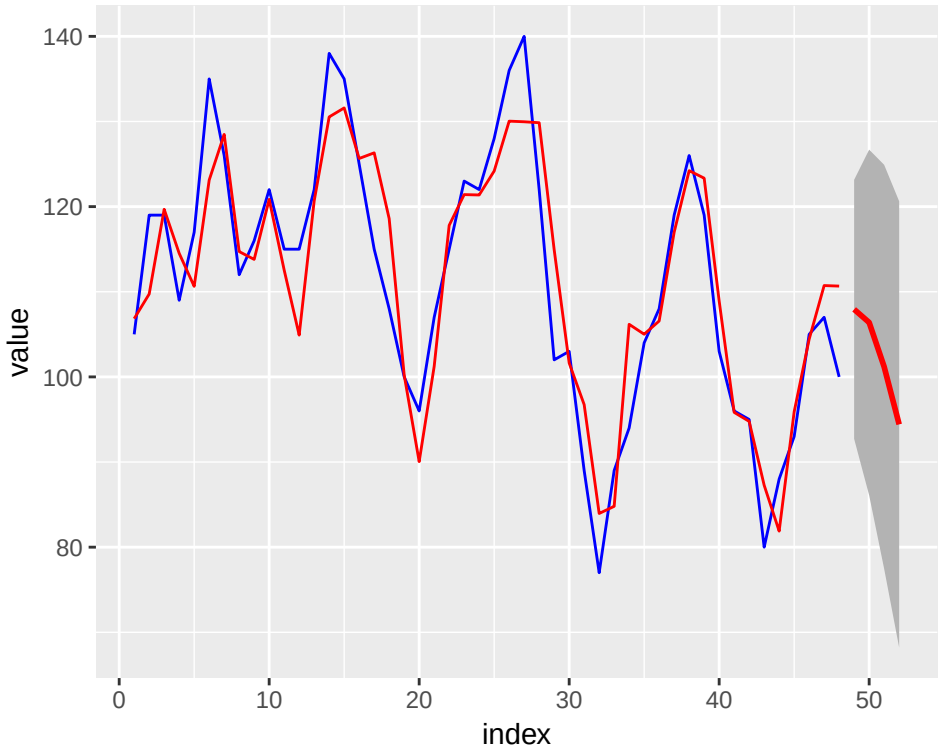



```
pred <- forecast(y3b, model=fit, h=4)

pred.df <- data.frame(index=index(pred$upper), mean=coredata(pred$mean), lower=coredata(pred$lower[,2])

pred.df
```

```
##   index      mean    lower    upper
## 1    49 107.93528 92.71982 123.1507
## 2    50 106.39132 86.09181 126.6908
## 3    51 101.18869 77.46686 124.9105
## 4    52  94.42164 68.21584 120.6274
```

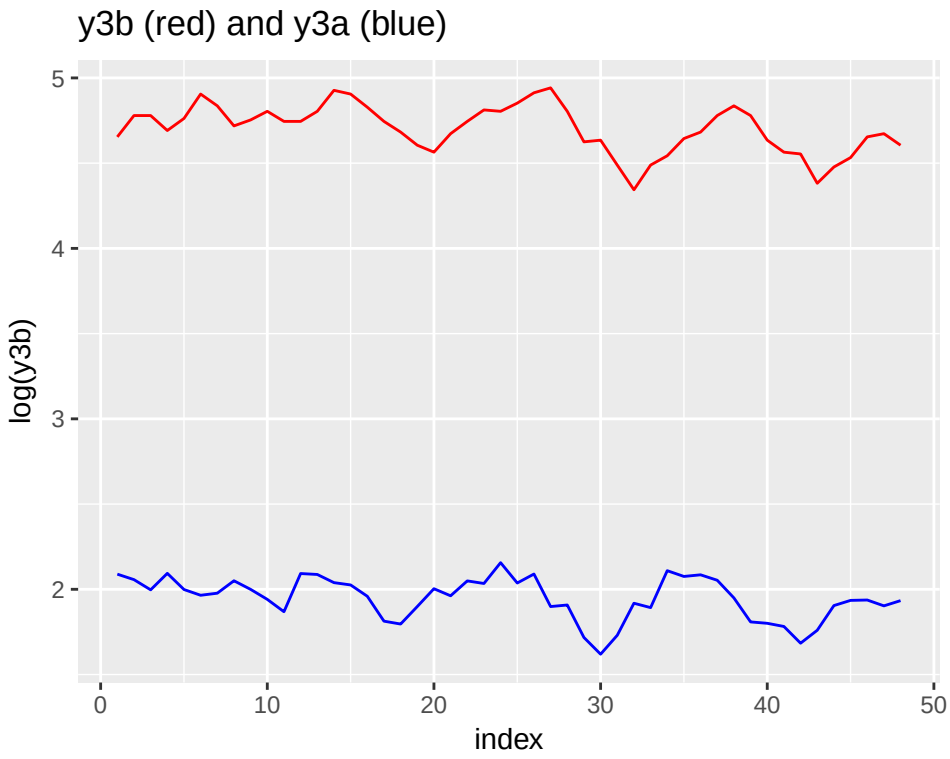
```
ggplot(y3b.df, aes(x=index)) +
  geom_line(aes(y=value), col="blue") +
  geom_line(aes(y=fitted), col="red") +
  geom_ribbon(data = pred.df,
              aes(ymin=lower, ymax=upper),
              fill = 'grey70') +
  geom_smooth(data = pred.df,
              aes(y=mean),
              colour='red',
```

```
            stat='identity')
```

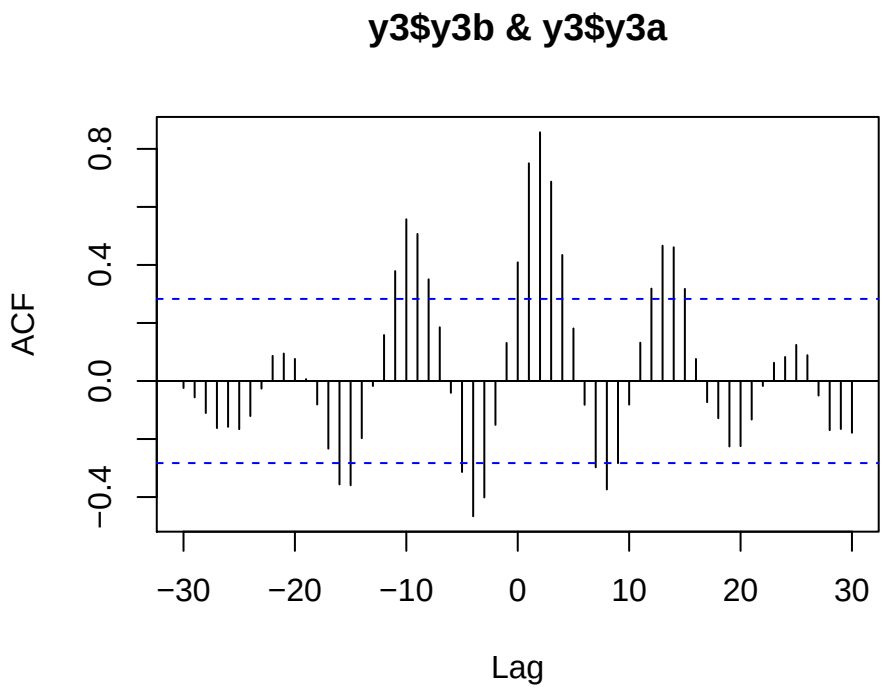


ii. Are `y3a` and `y3b` correlated? If so, at what lags?

```
ggplot(y3, aes(x=index)) +
  geom_line(aes(y=log(y3b)), col="red") +
  geom_line(aes(y=log(y3a)), col="blue") +
  labs(title="y3b (red) and y3a (blue)")
```




```
ccf(y3$y3b, y3$y3a, lag.max = 30, type = c("correlation"))
```



The cross-correlation is strongest at lag 2.

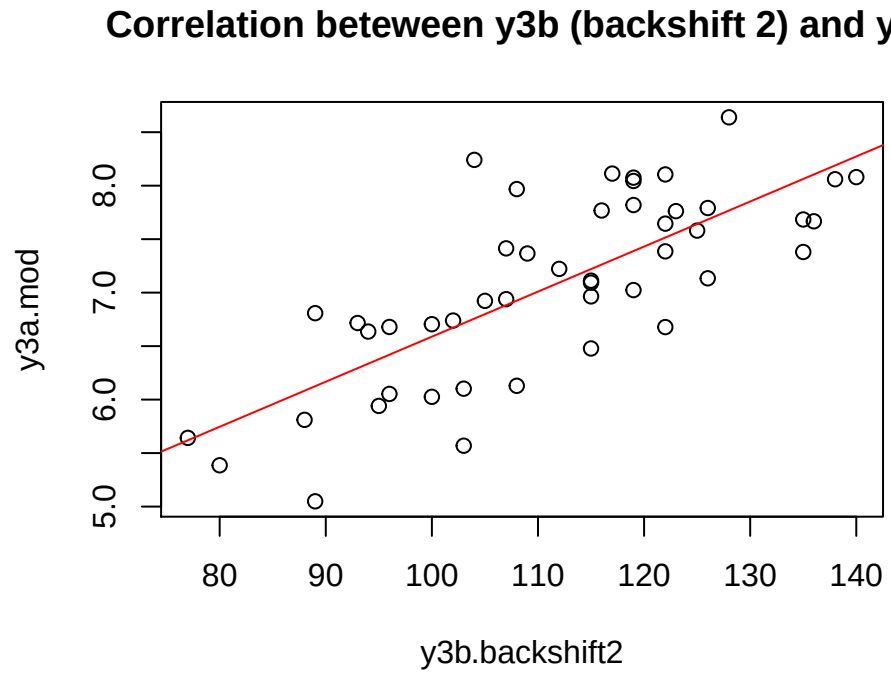
```
y3b.backshift2 <- y3$y3b[2:length(y3$y3b)]
length(y3b.backshift2)
```

```
## [1] 47
```

```r
y3a.mod <- y3$y3a[1:(length(y3$y3a) - 1)]
y3a.mod
```

```
##  [1] 8.075 7.819 7.366 8.113 7.380 7.134 7.222 7.768 7.386 6.965 6.478
## [12] 8.105 8.060 7.684 7.580 7.093 6.129 6.026 6.679 7.414 7.112 7.762
## [23] 7.645 8.639 7.667 8.080 6.678 6.739 5.569 5.049 5.642 6.808 6.636
## [34] 8.241 7.968 8.044 7.791 7.024 6.102 6.053 5.941 5.386 5.811 6.716
## [45] 6.923 6.939 6.705
```

```r
fit <- lm(y3a.mod ~ y3b.backshift2)
plot(y3b.backshift2, y3a.mod, main="Correlation beteween y3b (backshift 2) and y3a")
abline(fit, col="red")
```




```r
summary(fit)
```

```
##
## Call:
## lm(formula = y3a.mod ~ y3b.backshift2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.14576 -0.37273  0.02229  0.41047  1.48412
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.376677   0.618673   3.842  0.00038 ***
## y3b.backshift2 0.042117   0.005505   7.651 1.12e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5723 on 45 degrees of freedom
## Multiple R-squared:  0.5654, Adjusted R-squared:  0.5557
## F-statistic: 58.54 on 1 and 45 DF,  p-value: 1.115e-09
```

iii. (Extra bonus) Fit an appropriate model to `y3b` using `y3a`. Forecast the next 4 observations with 95% prediction limits.

In order to get `y3a` to look like `y3b`, we need to get them both on the same scale and do some transformation.

We'll first take the log of each, which will bring them on the same scale, then we'll add the mean of the difference to `y3a` to bring them to nearly equal values.

Let X_t represent `y3b` and Y_t represent `y3a`.

$$Y_t' = log(Y_t) + \frac{\sum_{i=1}^{n}(log(X_t) - log(Y_i))}{n}$$

We know that `y3a` is correlated with `y3b` at lag 2, so we just need to shift `y3a` forward by 2. There are various ways we can handle this, but we'll prepend the first two values of `y3b` to `y3a` and discard the last 2.

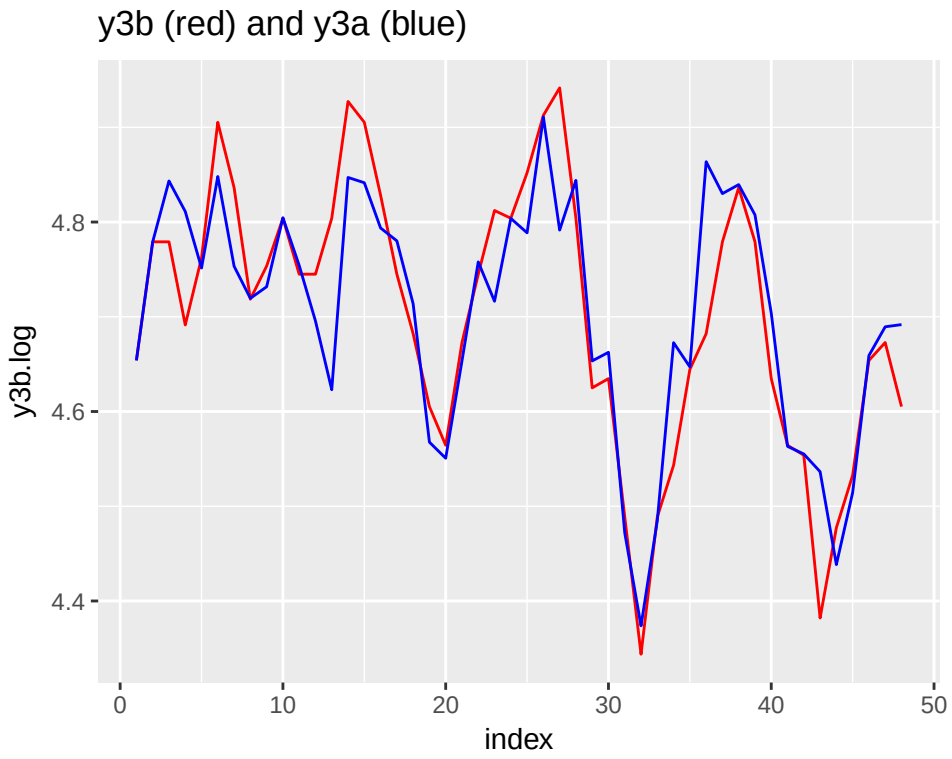
```
y3$y3b.log <- log(y3$y3b)
y3$y3a.log <- log(y3$y3a)
y3$y3a.mod.log <- y3$y3a.log + mean(y3$y3b.log - y3$y3a.log)

y3$y3a.mod.log[1:length(y3$y3a.mod.log)]
```

```
##  [1] 4.843358 4.811142 4.751460 4.848053 4.753359 4.719458 4.731717
##  [8] 4.804598 4.754172 4.695483 4.622997 4.847067 4.841499 4.793726
## [15] 4.780099 4.713694 4.567617 4.550669 4.653554 4.757956 4.716369
## [22] 4.803826 4.788637 4.910872 4.791511 4.843977 4.653404 4.662497
## [29] 4.471801 4.373776 4.484824 4.672684 4.647095 4.863707 4.830019
## [36] 4.839512 4.807555 4.703918 4.563202 4.555140 4.536463 4.438388
## [43] 4.514338 4.659078 4.689435 4.691743 4.657439 4.688134
```

```
y3$y3a.mod.log[1:(length(y3$y3a.mod.log) - 2)]
```

```
##  [1] 4.843358 4.811142 4.751460 4.848053 4.753359 4.719458 4.731717
##  [8] 4.804598 4.754172 4.695483 4.622997 4.847067 4.841499 4.793726
## [15] 4.780099 4.713694 4.567617 4.550669 4.653554 4.757956 4.716369
## [22] 4.803826 4.788637 4.910872 4.791511 4.843977 4.653404 4.662497
## [29] 4.471801 4.373776 4.484824 4.672684 4.647095 4.863707 4.830019
## [36] 4.839512 4.807555 4.703918 4.563202 4.555140 4.536463 4.438388
## [43] 4.514338 4.659078 4.689435 4.691743
```

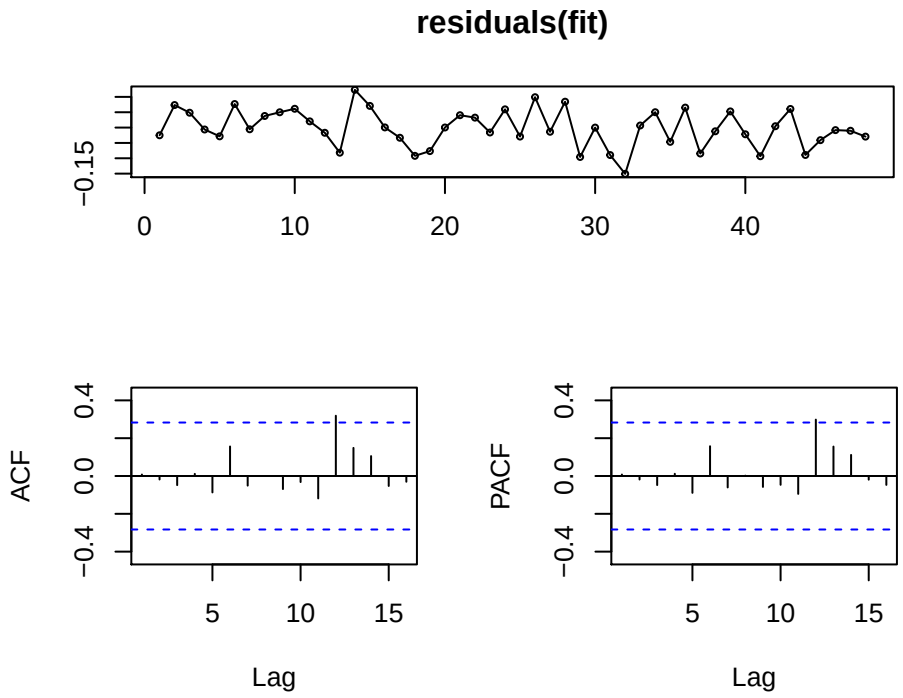
```
y3$y3a.mod.log.shift <- c(y3$y3b.log[1:2], y3$y3a.mod.log[1:(length(y3$y3a.mod.log) - 2)])

ggplot(y3, aes(x=index)) +
  geom_line(aes(y=y3b.log), col="red") +
  geom_line(aes(y=y3a.mod.log.shift), col="blue") +
  labs(title="y3b (red) and y3a (blue)")
```



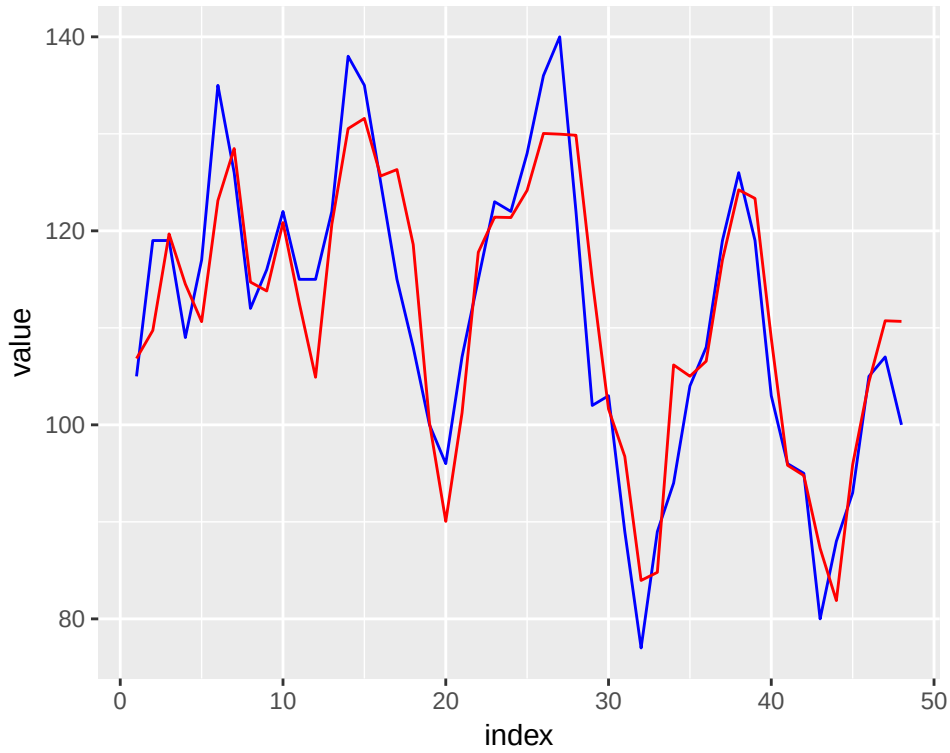
This looks pretty good. Now we can fit a model to our transformed `y3a` and overlay it on `y3b`.

```
fit <- armasubsets(ts(y3$y3a.mod.log.shift), nar=10, nma=10)

ar1 <- NA
ar2 <- 0
ar3 <- 0
ar4 <- 0
ar5 <- 0
ar6 <- NA
ar7 <- NA
ar8 <- NA
ar9 <- NA
ar10 <- NA

ma1 <- 0
ma2 <- 0
ma3 <- 0
ma4 <- 0
ma5 <- NA
ma6 <- NA
ma7 <- 0
intercept <- NA

fit <- Arima(ts(y3$y3a.mod.log.shift),
             order = c(10, 0, 7),
             method = "ML",
             fixed = c(ar1, ar2, ar3, ar4, ar5, ar6, ar7, ar8, ar9, ar10,
                       ma1, ma2, ma3, ma4, ma5, ma6, ma7,
                       intercept),
```

```
              transform.pars = FALSE)

tsdisplay(residuals(fit))
```



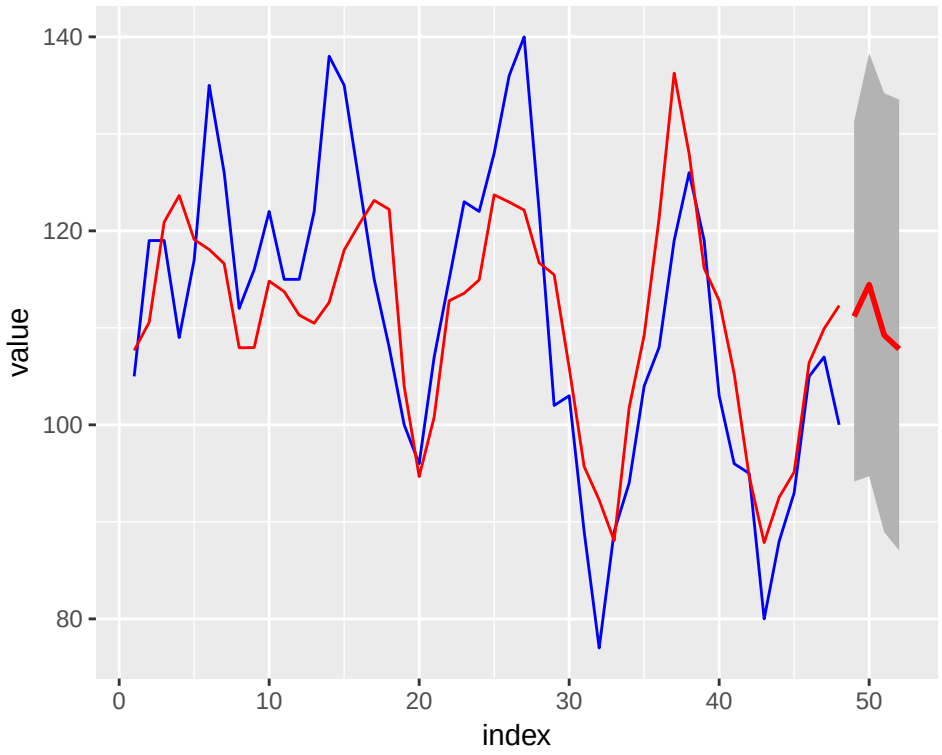

```
y3a.df <- data.frame(index=index(y3$index),
                     value=y3$y3b,
                     fitted=fitted(fit),
                     resid.stand=rstandard(fit))

ggplot(y3b.df, aes(x=index, y=value)) +
  geom_line(col="blue") +
  geom_line(aes(x=index, y=fitted), col="red") +
  labs("y3b fitted from a transformed y3a model")
```




```
pred <- forecast(ts(y3$y3b.log), model=fit, h=4)

pred.df <- data.frame(index=index(pred$upper), mean=coredata(pred$mean), lower=coredata(pred$lower[,2])

exp(pred.df[,-1])
```

```
##       mean    lower    upper
## 1 111.1786 94.15982 131.2735
## 2 114.4562 94.70648 138.3244
## 3 109.2575 88.95321 134.1965
## 4 107.8155 87.05191 133.5317
```

```
ggplot(y3a.df, aes(x=index)) +
  geom_line(aes(y=value), col="blue") +
  geom_line(aes(y=exp(fitted)), col="red") +
  geom_ribbon(data = pred.df,
              aes(ymin=exp(lower), ymax=exp(upper)),
              fill = 'grey70') +
  geom_smooth(data = pred.df,
              aes(y=exp(mean)),
              colour='red',
              stat='identity')
```


140
120
100
80
value
0
10
20
30
40
50
index

■