

Homework 4 Solutions

Due: Thursday 2/28/19 by 12:00pm (noon)

Although this homework assignment is not due until Thursday 2/28/19, there may be related material on your exam on Friday, 2/22/19.

Since this homework had more parts than usual, we have graded it out of 10, however it will be equally weighted with the rest of the homeworks when your total grade is computed. Since 1. (a)-(b) were on the exam, we just graded 1. (c)-(i) on a 0-3 scale as follows:

- 0 points for nothing written.
- 1 point for issues with code and interpretation.
- 2 points for either code correct but interpretations off or interpretations correct but code off.
- 3 points for pretty much everything correct.

Questions 2. and 3. were each worth 2 points, distributed as follows:

- 0 points for incorrect order, incorrect estimation of parameters.
- 1 point for correct or off by one order order, incorrect estimation of parameters.
- 2 points for correct order and estimation of parameters.

Question 4. (a)-(e) was worth 3 points in total, distributed as follows:

- 0 points for nothing correct.
- 1 point for very little correct.
- 2 points for almost everything correct, e.g. everything but final plot is correct.
- 3 points for everything correct.

Forecasting

1. In class, we derived the **forecasting equation** for computing $\hat{x}_{n+1} = \sum_{j=1}^n c_{nj}x_{n+1-j}$ based on x_1, \dots, x_n by minimizing

$$v_n = \mathbb{E} \left[\left(x_{n+1} - \sum_{j=1}^n c_{nj}x_{n+1-j} \right)^2 \right].$$

- (a) Write down the quantity we should minimize if we want to forecast values further into the future, i.e. if we want to compute \hat{x}_{n+k} for any $k > 0$.

We want to find the values c_1, \dots, c_n that minimize

$$\mathbb{E} \left[\left(x_{n+k} - \sum_{j=1}^n c_j x_{n+1-j} \right)^2 \right].$$

Note that you can use whatever notation you want, i.e. we could equivalently say that we want to find the values $c_{(n+k-1)1}, \dots, c_{(n+k-1)n}$ that minimize

$$\mathbb{E} \left[\left(x_{n+k} - \sum_{j=1}^n c_{(n+k-1)j} x_{n+1-j} \right)^2 \right],$$

or we could say that we want to find the values c_k, \dots, c_{k+n-1} that minimize

$$\mathbb{E} \left[\left(x_{n+k} - \sum_{j=k}^{n+k-1} c_j x_{n+k-j} \right)^2 \right].$$

The main idea remains the same regardless of what notation we use - we want to find the linear combination of observed values x_1, \dots, x_n that minimizes expected squared error loss for predicting x_{n+k} .

- (b) Write out the forecasting equation that corresponds to (a). Hint: In class we talked about minimizing v_n . Defining $a_{n,ij} = \gamma_x(i-j)$ and $b_{n,i} = \gamma_x(i)$, the corresponding forecasting equation was:

$$\mathbf{A}_n \mathbf{c}_n = \mathbf{b}_n.$$

Let's stick with the notation \mathbf{c} to refer to the prediction of x_{n+k} as we did first in (a). Remember, we got the forecasting equation in the one-step-ahead case by expanding out the expected squared forecast error v_n and differentiating with respect to \mathbf{c}_n . Let's try that again approach again here, first expanding the expected squared forecast error:

$$\begin{aligned} \mathbb{E} \left[\left(x_{n+k} - \sum_{j=1}^n c_j x_{n+1-j} \right)^2 \right] &= \mathbb{E} \left[x_{n+k}^2 - 2x_{n+k} \left(\sum_{j=1}^n c_j x_{n+1-j} \right) + \left(\sum_{j=1}^n c_j x_{n+1-j} \right)^2 \right] \\ &= \gamma_x(0) - 2 \left(\sum_{j=1}^n c_j \mathbb{E}[x_{n+1-j} x_{n+k}] \right) + \left(\sum_{j=1}^n \sum_{l=1}^n c_j c_l \mathbb{E}[x_{n+1-j} x_{n+1-l}] \right) \\ &= \gamma_x(0) - 2 \left(\sum_{j=1}^n c_j \gamma_x(k-1+j) \right) + \left(\sum_{j=1}^n \sum_{l=1}^n c_j c_l \gamma_x(l-j) \right) \\ &= \gamma_x(0) - 2\mathbf{c}'\mathbf{b} + \mathbf{c}'\mathbf{A}\mathbf{c}, \end{aligned}$$

where $b_j = \gamma_x(k-1+j)$ and $a_{jl} = \gamma_x(l-j)$ for $j = 1, \dots, n$ and $l = 1, \dots, n$.

Now we can easily differentiate with respect to \mathbf{c} . Setting the derivative equal to zero, we get:

$$-2\mathbf{b} + 2\mathbf{A}\mathbf{c} = \mathbf{0} \implies \mathbf{A}\mathbf{c} = \mathbf{b}.$$

- (c) In class, I showed you the following function for computing the values \mathbf{c}_n and v_n for a **ARMA**(1,1) model.

```
solve.direct <- function(n, phi1 = 0, theta1 = 0, sig.sq.w = 1) {
  A.n <- matrix(nrow = n, ncol = n)
  b.n <- numeric(n)
  for (i in 1:n) {
    b.n[i] <- gamma.x(i, phi1 = phi1, theta1 = theta1, sig.sq.w = sig.sq.w)
    for (j in 1:n) {
      A.n[i, j] <- gamma.x(h = i - j, phi1 = phi1, theta1 = theta1, sig.sq.w = sig.sq.w)
    }
  }
  c.n <- solve(A.n)%*%b.n
  v.n <- gamma.x(0, phi1 = phi1, theta1 = theta1, sig.sq.w = sig.sq.w) +
    t(c.n)%*%A.n%c.n - 2*t(c.n)%*%b.n
}
```

```

  return(list("c.n" = c.n, "v.n" = v.n))
}

```

Modify this function to take an additional argument, k , and return the coefficients that give the forecast \hat{x}_{n+k} as well as the expected squared error loss of \hat{x}_{n+k} . Note that you will need to use the `gamma.x` function from class.

```

gamma.x <- function(h, phi1, theta1, sig.sq.w) {
  h <- abs(h)
  if (h == 0) {
    g.x <- (theta1^2 + 2*phi1*theta1 + 1)*sig.sq.w/(1 - phi1^2)
  } else {
    g.x <- sig.sq.w*phi1^(h - 1)*((1 + theta1*phi1)*(phi1 + theta1)/(1 - phi1^2))
  }
  return(g.x)
}

```

The only thing that changes is computation of the entries of \mathbf{b} , they will depend on k now.

```

solve.direct <- function(n, phi1 = 0, theta1 = 0, sig.sq.w = 1, k = 1) {
  A.n <- matrix(nrow = n, ncol = n)
  b.n <- numeric(n)
  for (i in 1:n) {
    b.n[i] <- gamma.x(i + k - 1, phi1 = phi1, theta1 = theta1, sig.sq.w = sig.sq.w)
    for (j in 1:n) {
      A.n[i, j] <- gamma.x(h = i - j, phi1 = phi1, theta1 = theta1, sig.sq.w = sig.sq.w)
    }
  }
  c.n <- solve(A.n)%*%b.n
  v.n <- gamma.x(0, phi1 = phi1, theta1 = theta1, sig.sq.w = sig.sq.w) +
    t(c.n)%*%A.n%*%c.n - 2*t(c.n)%*%b.n
  return(list("c.n" = c.n, "v.n" = v.n))
}

```

(d) Use the function you wrote in (c) to plot the expected squared error loss of \hat{x}_{3+k} for the following models for $k = 1, \dots, 5$, all with $\sigma_w^2 = 2$:

- i. $\phi_1 = 0.5, \theta_1 = 0$;
- ii. $\phi_1 = -0.5, \theta_1 = 0$;
- iii. $\phi_1 = 0, \theta_1 = 0.57735$;
- iv. $\phi_1 = 0, \theta_1 = -0.57735$.

Include a horizontal dashed line at $\gamma_x(0)$ (Note: $\gamma_x(0)$ is the same for all of the models).

```

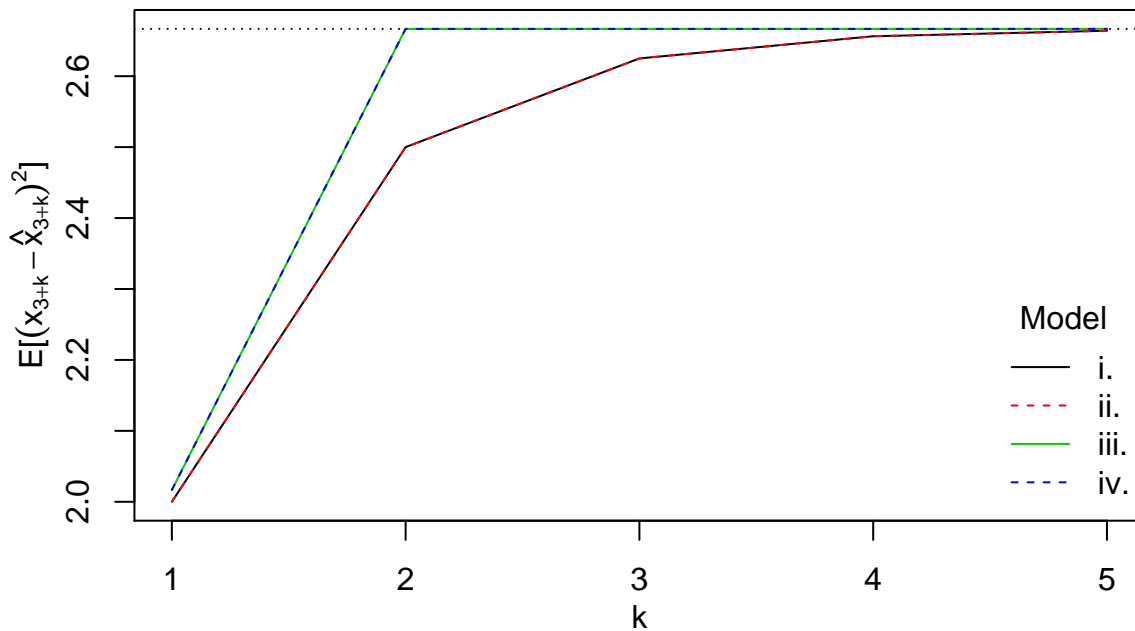
n <- 3
sig.sq.w <- 2
pars <- data.frame("phi1" = c(0.5, -0.5, 0, 0),
                  "theta1" = c(0, 0, 0.57735, -0.57735))
v <- matrix(nrow = nrow(pars), ncol = 5)
gamma.x.0 <- numeric(nrow(pars))
for (i in 1:nrow(pars)) {
  for (k in 1:ncol(v)) {
    v[i, k] <- solve.direct(n = n, phi1 = pars[i, "phi1"], theta1 = pars[i, "theta1"],
                          sig.sq.w = sig.sq.w, k = k)$v.n
  }
  gamma.x.0[i] <- gamma.x(0, phi1 = pars[i, "phi1"], theta1 = pars[i, "theta1"],
                        sig.sq.w = sig.sq.w)
}

```

```

}
plot(1:ncol(v), v[1, ], ylim = c(min(v), gamma.x.0[1]), type = "n",
     xlab = "", ylab = "")
mtext(expression(paste("E", "[", (x[3+k]-hat(x)[3+k])^2, "]", sep = "")), 2, line = 2)
mtext("k", 1, line = 2)
abline(h = gamma.x.0[1], lty = 3)
for (i in 1:nrow(pars)) {
  lines(1:ncol(v), v[i, ], col = i, lty = i %in% c(2, 4) + 1)
}
legend("bottomright", col = c(1:4), lty = c(1, 2, 1, 2),
      legend = c("i.", "ii.", "iii.", "iv."),
      title = "Model", bty = "n")

```



(e) In one sentence, interpret how the forecast error relates to the sign of ϕ_1 or θ_1 .

The expected squared forecast error doesn't depend on the sign of ϕ_1 or θ_1 at all, which suggests that how well we can forecast future values depends on the absolute magnitudes of the correlations over time, as opposed to their signs.

(f) In one sentence, interpret how the forecast error changes as k increases for an **AR**(1) model versus an **MA**(1) model.

The expected squared forecast error increases as k increases, because as k increases we have to predict further into the future without any new data.

(g) Add two more lines to your plot in (d) corresponding to the following models with $\sigma_w^2 = 0.2533333$:

v. $\phi_1 = 0.9$, $\theta_1 = 0$;

vi. $\phi_1 = 0$, $\theta_1 = 2.064742$.

```

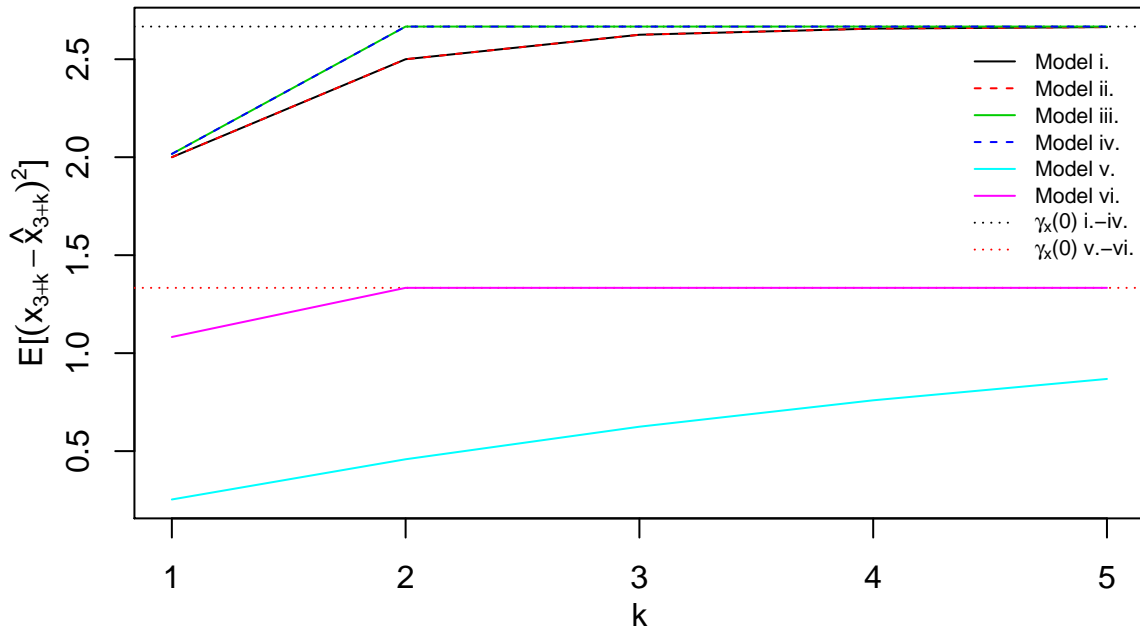
n <- 3
pars <- data.frame("phi1" = c(0.5, -0.5, 0, 0, 0.9, 0),
                  "theta1" = c(0, 0, 0.57735, -0.57735, 0, 2.064742),
                  "sig.sq.w" = c(2, 2, 2, 2, 0.2533333, 0.2533333))
v <- matrix(nrow = nrow(pars), ncol = 5)
gamma.x.0 <- numeric(nrow(pars))

```

```

for (i in 1:nrow(pars)) {
  for (k in 1:ncol(v)) {
    v[i, k] <- solve.direct(n = n, phi1 = pars[i, "phi1"], theta1 = pars[i, "theta1"],
                           sig.sq.w = pars[i, "sig.sq.w"], k = k)$v.n
  }
  gamma.x.0[i] <- gamma.x(0, phi1 = pars[i, "phi1"], theta1 = pars[i, "theta1"],
                           sig.sq.w = pars[i, "sig.sq.w"])
}
plot(1:ncol(v), v[1, ], ylim = c(min(v), gamma.x.0[1]), type = "n",
     xlab = "", ylab = "")
mtext(expression(paste("E", "[", (x[3+k]-hat(x)[3+k])^2, "]", sep = "")), 2, line = 2)
mtext("k", 1, line = 2)
abline(h = gamma.x.0[1], lty = 3)
abline(h = gamma.x.0[5], lty = 3, col = 2)
for (i in 1:nrow(pars)) {
  lines(1:ncol(v), v[i, ], col = i, lty = i %in% c(2, 4) + 1)
}
legend("topright", col = c(1:6, 1, 2), lty = c(1, 2, 1, 2, 1, 1, 3, 3),
      legend = c("Model i.", "Model ii.", "Model iii.",
                  "Model iv.", "Model v.", "Model vi.",
                  expression(paste(gamma[x], "(", 0, ") i.-iv.", sep = "")),
                  expression(paste(gamma[x], "(", 0, ") v.-vi.", sep = ""))),
      title = "", bty = "n", cex = 0.7)

```



(h) Is the variance $\gamma_x(0)$ still the same as it was for the models first plotted in (c)?

No, so the models will not converge to the same expected squared forecast error as $k \rightarrow \infty$.

(i) In one sentence, interpret what you see in (g). How does increasing ϕ_1 or θ_1 affect how the forecast error changes as k increases?

Increasing θ_1 does not change the fact that the forecast error increases to $\gamma_x(0)$ immediately when $k > 1$ because only the lag-one autocorrelation $\rho_x(1)$ is nonzero under an **MA**(1) model, whereas increasing ϕ_1 makes the forecast error increase to $\gamma_x(0)$ more slowly.

Estimation

Let's look back at the chicken data one more time. We're going to consider **ARMA**(p, q) time series models for the residuals from a linear regression of the chicken prices on an intercept and time. I've computed them for you here and named them `r`, to make sure everyone starts out on the same page.

```
library(astsa)
data(chicken)
r <- lm(chicken~time(chicken))$res
```

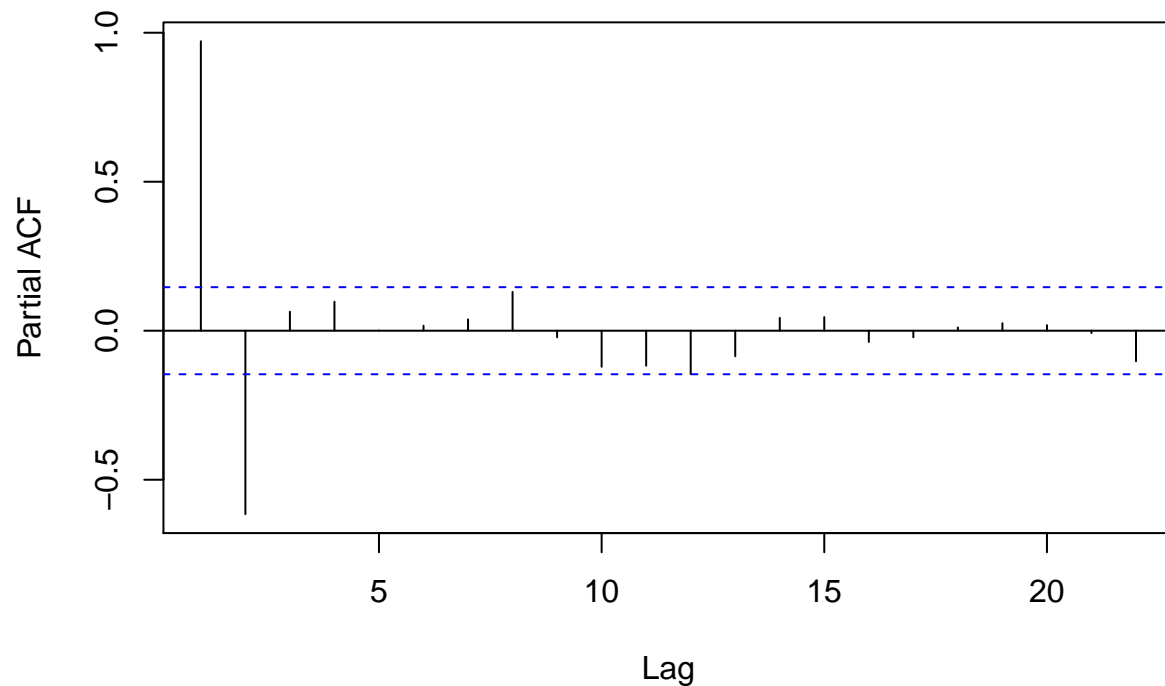
2. First, we're going to consider setting $q = 0$ and fitting an **AR**(p) model to `r`.

- (a) Using the `acf` function, plot the sample partial autocorrelations. Based on the sample partial autocorrelation function, what would you select for p ? Accompany your choice with at most one sentence of reasoning.

I would select $p = 2$, because the sample partial autocorrelations for lags greater than two \hat{c}_{hh} for $h > 2$ are within the approximate 95% intervals for \hat{c}_{hh} if the true value $c_{hh} = 0$.

```
acf(r, type = "partial",
    main = "Sample Partial Autocorrelations of Residuals")
```

Sample Partial Autocorrelations of Residuals



- (b) Setting the order based on (a), fit the **AR**(p) model to `r` using the Yule-Walker equations. I do not want you to use the `ar.yw` function, but you can base your code on the `solve.direct` function. Give the **AR**(p) parameter estimates.

```
gamma.hat <- acf(r, lag.max = 3, plot = FALSE,
    type = "cov")$acf[, 1, 1]

A.hat <- matrix(nrow = 2, ncol = 2)
A.hat[1, 1] <- A.hat[2, 2] <- gamma.hat[1]
A.hat[1, 2] <- A.hat[2, 1] <- gamma.hat[2]
```

```

b.hat <- gamma.hat[2:3]

phi.hat <- solve(A.hat)%*%b.hat
sig.sq.w.hat <- gamma.hat[1] - t(phi.hat)%*%gamma.hat[2:3]

```

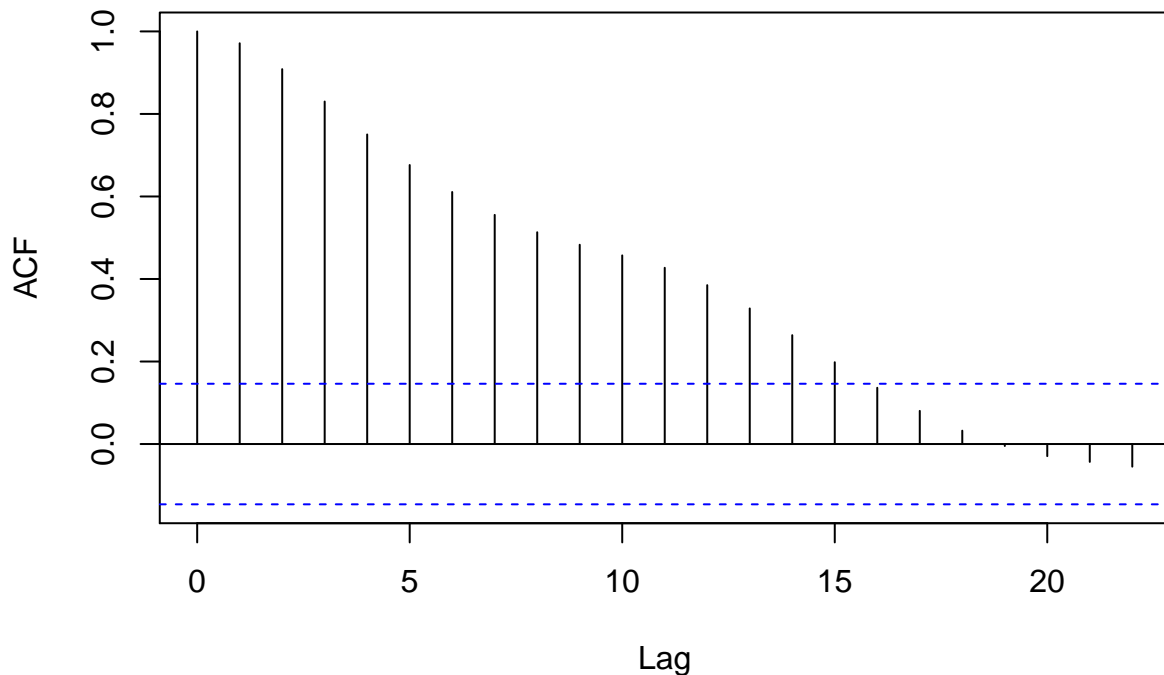
We obtain $\hat{\phi}_1 \approx 1.57$, $\hat{\phi}_2 \approx -0.62$, and $\hat{\sigma}_w^2 \approx 0.77$.

3. Now, we're going to consider setting $p = 0$ and fitting an $\mathbf{MA}(q)$ model to \mathbf{r} .

- (a) Using the `acf` function, plot the sample autocorrelations. Based on the sample autocorrelations, what would you select for q ? Accompany your choice with at most one sentence of reasoning.

```
acf(r, main = "Sample Autocorrelations of Residuals")
```

Sample Autocorrelations of Residuals



I would select $q = 15$, because the sample autocorrelations for lags greater than 15 $\hat{\rho}_x(k)$ for $k > 15$ are within the approximate 95% intervals for $\hat{\rho}_x(k)$ if the true value $\rho_x(k) = 0$.

- (b) Fit an $\mathbf{MA}(1)$ model to \mathbf{r} by computing the innovation coefficients \mathbf{d}_n using the function given in class, and treating the innovation coefficient estimates \mathbf{d}_n as estimates of ψ_1, \dots, ψ_n . Give the $\mathbf{MA}(1)$ coefficient estimate.

```

est.innov <- function(x) {
  n <- length(x)
  gamma.hat <- acf(x, type = "cov", plot=FALSE, lag.max = n-1)$acf
  D <- matrix(nrow = n-1, ncol = n-1)
  v <- rep(NA, n)
  gamma.x.0 <- gamma.hat[1]
  v[1] <- gamma.x.0
  for (i in 1:(n - 1)) {
    for (j in 0:(i-1)) {
      D[i, i - j] <- gamma.hat[i - j + 1]
      if (j > 0) {

```

```

    for (k in 0:(j - 1)) {
      D[i, i - j] <- D[i, i - j] - D[j, j - k]*D[i, i - k]*v[k + 1]
    }
  }
  D[i, i - j] <- D[i, i - j]/v[j + 1]
}
v[i + 1] <- gamma.x.0 - sum(D[i, 1:i]^2*v[1:i], na.rm = TRUE)
}
return(list("D"=D, "d.n" = D[nrow(D), ],
           "v" = v, "v.n" = v[length(v)]))
}
innov.r <- est.innov(r)

```

This gives $\hat{\theta}_1 = 1.48$ and $\hat{\sigma}_w^2 = 6.83$.

4. Finally, let's to consider fitting an **ARMA**(p, q) model to r .

- (a) Can we use the sample autocorrelations or sample partial autocorrelations to select p or q ? Just give a yes or no.

No.

- (b) Using the `arima` function, fit **ARMA**(p, q) models with $p = 0, \dots, 3$ and $q = 0, \dots, 3$, excluding the case where $p = q = 0$. Compute AIC, AICc, and BIC according to the equations that were given in class early on by hand.

```

n <- length(r)
ps <- 0:3
qs <- 0:3
aics <- aiccs <- sics <- matrix(nrow = length(ps), ncol = length(qs))
for (p in ps) {
  for (q in qs) {
    if (!(p == q & p == 0)) {
      fit <- arima(r, order = c(p, 0, q))
      aics[which(p == ps), which(q == qs)] <- log(fit$sigma2) +
        (n + 2*(p + q + 1))/n
      aiccs[which(p == ps), which(q == qs)] <- log(fit$sigma2) +
        (n + p + q + 1)/(n - p - q - 1 - 2)
      sics[which(p == ps), which(q == qs)] <- log(fit$sigma2) +
        (p + q + 1)*log(n)/(n)
    }
  }
}

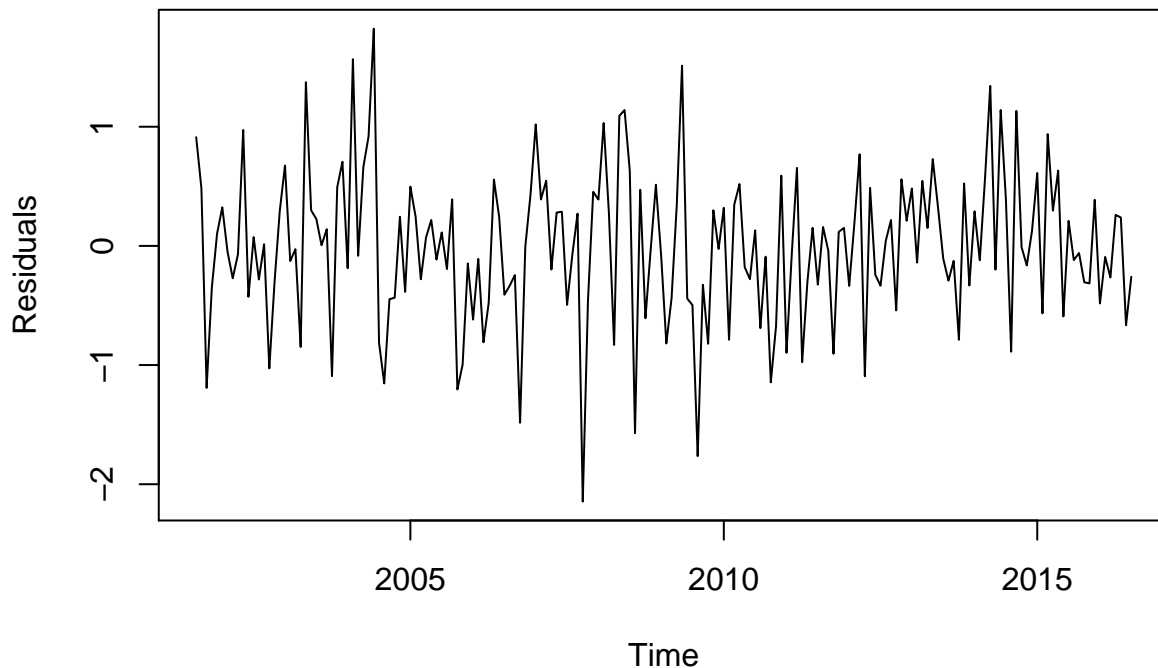
```

- (c) Plot the residuals from the AIC-minimizing model.

```

which.min <- which(aics == min(aics, na.rm = TRUE), arr.ind = TRUE)
fit <- arima(r, order = c(ps[which.min[1]], 0, qs[which.min[2]]))
plot(as.numeric(time(chicken)),
     fit$residuals,
     xlab = "Time", ylab = "Residuals", type = "l")

```

- (d) Fit the AIC-minimizing model to `r` using the `arima` function. What algorithm does the `arima` function use to estimate $\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_p$ and σ_w^2 by default? (Unconditional maximum likelihood, unconditional least squares, conditional least squares...)

The `arima` documentation indicates that it computes starting values using conditional least squares, and then applies unconditional maximum likelihood to compute the estimates that are returned to the user.

- (e) On a plot with the x -axis ranging from $2001 + 7/12$ to 2020 and y -axis ranging from 60 to 130, plot:
- The observed chicken time series.
 - The linear model fit of the chicken time series, i.e. the fitted values you get from `lm(chicken~time(chicken))`.
 - Predicted mean chicken prices from $2016 + 6/12$ to 2020 based on the model you fit in (d). Remember - you fit the model in (e) to the residuals, but we want predicted chicken prices. So you'll have to add the linear fit to the predictions you get from the ARMA model.
 - Predicted mean chicken prices from $2016 + 6/12$ to 2020 plus or minus one standard error based on the model you fit in (e). Again, you'll have to add the linear model fit to the prediction standard errors you get from the ARMA model.

You can get the predictions and standard errors by applying the `predict` function to your `arima` object from part (d).

```
plot(chicken, type = "l", xlim = c(min(time(chicken)), 2020),
     ylim = c(60, 130), xlab = "Time", ylab = "Chicken Prices")
lm <- lm(chicken~time(chicken))
abline(a = lm$coef[1], b = lm$coef[2], col = "purple")
n.ahead <- 12*3 + 6
pred <- predict(fit, n.ahead = n.ahead, se.fit = TRUE)
t.new <- max(time(chicken)) + 1:length(pred$pred)/12
lines(t.new, lm$coef[1] + lm$coef[2]*t.new + pred$pred, col = "blue")
lines(t.new, lm$coef[1] + lm$coef[2]*t.new + pred$pred + qnorm(0.975)*pred$se,
     col = "blue", lty = 2)
lines(t.new, lm$coef[1] + lm$coef[2]*t.new + pred$pred - qnorm(0.975)*pred$se,
     col = "blue", lty = 2)
legend("bottomright", col = c("black", "purple", "blue", "blue"),
     lty = c(1, 1, 1, 2), legend =
```

```
c("Observed Values",  
  "Linear Model Fit",  
  "Predicted Values",  
  "Prediction Standard Errors"),  
bty = "n")
```

