Homework 4 Solutions

Due: Thursday 2/20/20 by 10:00am

We're going to keep working with the `broc` data for a little while longer. Again, it is posted on the course website, which contains the average price of one pound of broccoli in urban areas each month, from July 1995 through December 2019. Throughout this problem, we're just going to work with the residuals from fitting a linear model with a linear time trend and month effects to all but the last 12 months of data. We'll call them `y`, because we'll be thinking of them as our observed time series.

```
load("~/Dropbox/Teaching/TimeSeries2020/stat697/content/data/broc.RData")
set.seed(1)
broc$fdate <- as.Date(broc$date, "%Y-%m-%d")
broc$month <- format(broc$fdate, "%m")
broc$dayssincestart <- as.numeric(broc$fdate) - min(as.numeric(broc$fdate))
n.sub <- nrow(broc) - 12
linmod <- lm(price~dayssincestart+factor(month), data = broc,
             subset = 1:(n.sub))
pred <- predict(linmod, broc)
y <- broc$price - pred
```

In last week's homework, we saw evidence of substantial dependence across time in the residuals `y`. Now we're going to try modeling that dependence over time, using an autoregressive model of order p:

$$y_t = \mu + \sum_{i=1}^{p} \phi_i y_{t-i} + w_t, \quad w_t \overset{i.i.d.}{\sim} \text{normal}\left(0, \sigma_w^2\right) \tag{1}$$

Note: for this assignment, only assume what is written above! Do *not* assume that y_t is a stationary process.

(a) Write down what you can of the likelihood for $y_1, \ldots, y_{n-12}$ Indicate whether or not there are any values y_t which you cannot write down the likelihood for and/or need to condition on.

Based on the model given in (1), we can write down the likelihood of $y_{p+1}, \ldots, y_{n-12}$ conditional on the first p values $y_1, \ldots, y_p$.

$$p\left(y_{p+1}, \ldots, y_{n-12} | y_1, \ldots, y_p\right) = \prod_{t=p+1}^{n-12} p\left(y_t | y_{t-1}, \ldots, y_{t-p}\right) \tag{2}$$

We cannot write down the likelihood function for the remaining observations $y_1, \ldots, y_p$ because (1) only describes the probability distribution of observations y_t given all p previous values.

Based on the model given in (1), we *can* say that $y_t | y_{t-1}, \ldots, y_{t-p} \sim \text{normal}\left(\mu + \sum_{i=1}^{p} \phi_i y_{t-i}, \sigma_w^2\right)$, so we can simplify the conditional likelihood from (2):

$$\begin{aligned} p\left(y_{p+1}, \ldots, y_{n-12} | y_1, \ldots, y_p\right) &= \prod_{t=p+1}^{n-12} \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp\left\{ -\frac{1}{2\sigma_w^2} \left(y_t - \left(\mu + \sum_{i=1}^{p} \phi_i y_{t-i} \right) \right)^2 \right\} \\ &= \frac{1}{\sqrt{2\pi\sigma_w^2}^{(n-12-p)}} \exp\left\{ -\frac{1}{2\sigma_w^2} \sum_{t=p+1}^{n-12} \left(y_t - \left(\mu + \sum_{i=1}^{p} \phi_i y_{t-i} \right) \right)^2 \right\} \end{aligned}$$

(b) The likelihood in (a) looks like a linear regression model. Clearly indicate how you would construct the response vector and matrix of covariates for a fixed value of p.

For a specific value of p, let's define $\boldsymbol{z}$ to be the response vector and $\boldsymbol{X}$ to be the matrix of covariates. Based on what (a), $\boldsymbol{z}$ will have $n-12-p$ elements and $\boldsymbol{X}$ will have $n-12-p$ rows.

For $p=1$, we would have

$$\boldsymbol{z} = \begin{pmatrix} y_2 \\ y_3 \\ \vdots \\ y_{n-12} \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} 1 & y_1 \\ 1 & y_2 \\ \vdots & \vdots \\ 1 & y_{n-13} \end{pmatrix}$$

For $p=2$, we would have

$$\boldsymbol{z} = \begin{pmatrix} y_3 \\ y_4 \\ \vdots \\ y_{n-12} \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} 1 & y_2 & y_1 \\ 1 & y_3 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & y_{n-13} & y_{n-14} \end{pmatrix}$$

For $p \geq 2$, we will have:

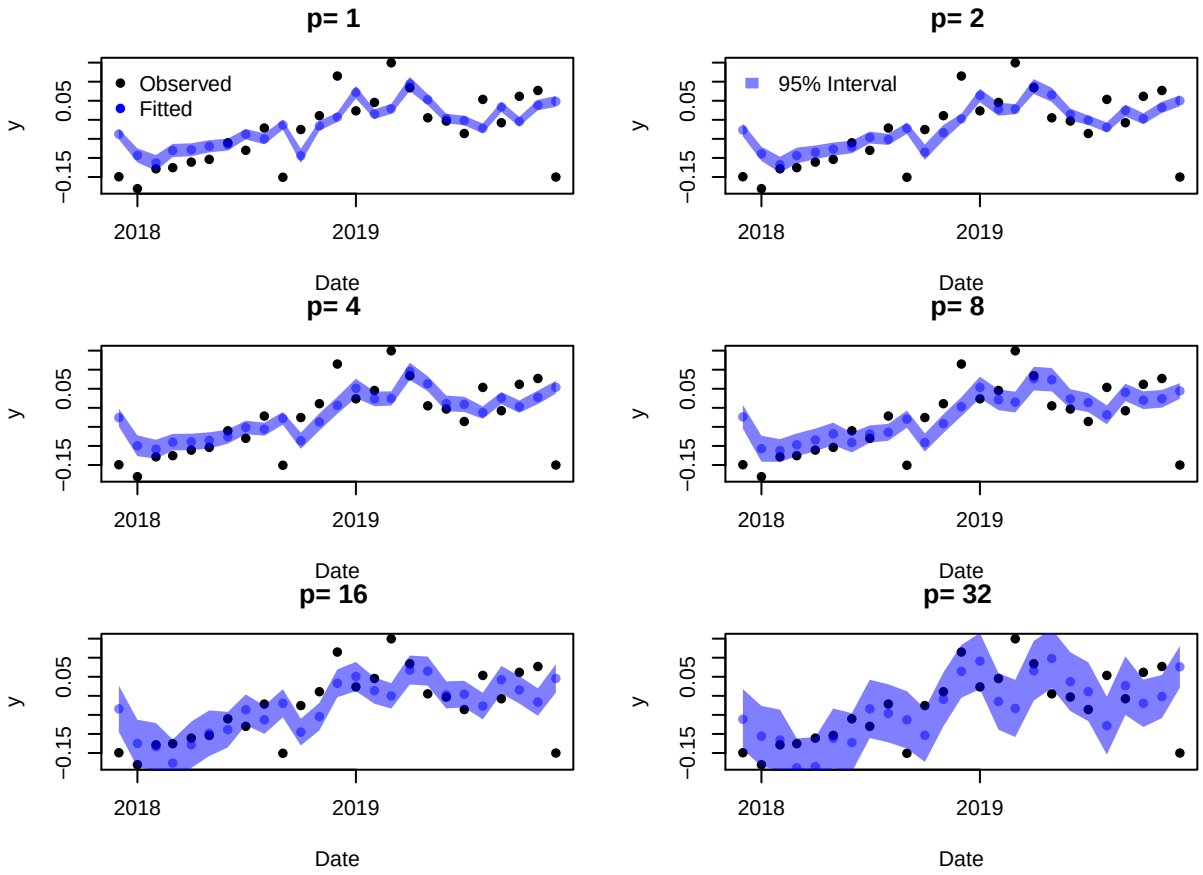
$$z_i = p+i, \quad x_{i1} = 1, \quad x_{ij} = y_{p+i-(j-1)} \text{ for } i = 1, \ldots, n-12-p \text{ and } j = 1, \ldots, p$$

(c) For $p = 1, 2, 4, 8, 16, 32$, compute estimates of μ, $\phi_1, \ldots, \phi_p$ using `lm` or any other approach to computing regression coefficients and residual standard errors for a linear regression model. Make a plot with 6 panels. Using one panel for each value of p, plot the last 24 observations, the fitted values from corresponding fitted model, and the approximate 95% confidence intervals obtained by treating this as a standard regression problem.

```
ps <- 2^(0:5)
data <- y
for (i in 1:max(ps)) {
  data <- cbind(data, c(rep(NA, i), y[1:(length(y) - i)]))
}
data <- data.frame(data)
names(data) <- c("y", paste("y.lag", 1:max(ps), sep = ""))
fits <- matrix(NA, nrow = length(y), ncol = length(ps))
ses <- matrix(NA, nrow = length(y), ncol = length(ps))
for (p in ps) {
  linmod <- lm(as.formula(paste("y~", paste(paste("y.lag", 1:p, sep = ""),
                                            collapse = "+"),
                                sep = "")),
               data = data, subset = (p + 1):(length(y) - 12))
  preds <- predict(linmod, newdata = data, se.fit = TRUE)
  fits[, which(p == ps)] <- preds$fit
  ses[, which(p == ps)] <- preds$se.fit
}
```

```
par(mfrow = c(3, 2))
par(mar = c(4, 4, 2, 2))
for (p in ps) {
plot(broc$fdate[(length(y) - 24):length(y)],
     y[(length(y) - 24):length(y)], pch = 16,
     xlab = "Date", ylab = "y",
```

```
     main = paste("p=", p, "\n"))
points(broc$fdate[(length(y) - 24):length(y)],
       fits[, which(p == ps)][(length(y) - 24):length(y)],
       col = rgb(0, 0, 1, 0.5), pch = 16)
polygon(c(broc$fdate[(length(y) - 24):length(y)],
          rev(broc$fdate[(length(y) - 24):length(y)])),
        c(fits[, which(p == ps)][(length(y) - 24):length(y)] +
            qnorm(0.025)*ses[, which(p == ps)][(length(y) - 24):length(y)],
          rev(fits[, which(p == ps)][(length(y) - 24):length(y)] +
                qnorm(0.975)*ses[, which(p == ps)][(length(y) - 24):length(y)])),
        border = FALSE, col = rgb(0, 0, 1, 0.5))
if (p == min(ps)) {
  legend("topleft", pch = c(16, 16),
         col = c("black", "blue"),
         legend = c("Observed", "Fitted"),
         bty = "n")
}
if (p == ps[2]) {
  legend("topleft",
         fill = rgb(0, 0, 1, 0.5),
         legend = c("95% Interval"),
         bty = "n", border = "white")
}
}
```



(d) Recall that AIC, AICc, and BIC/SIC are only appropriate when all of the models being compared were

fit to the same data, i.e. the likelihoods all corresponded to the likelihood of the same set of data points. Are AIC, AICc, or BIC/SIC appropriate for comparing these models fit in (c) with different values of p? If yes, pick a criterion (AIC, AICc, or BIC/SIC), justify your choice, and indicate which model you would choose.

AIC, AICc, and BIC/SIC are all inappropriate for comparing the models fit in (c) because the different models were fit to different data. For $p = 1$, the model was fit to $y_2, \ldots, y_{n-12}$, for $p = 2$ the model was fit to $y_3, \ldots, y_{n-12}$, and so on.

(e) For $p = 1, 2, 4, 8, 16, 32$, compute estimates of μ, $\phi_1, \ldots, \phi_p$ using only $y_{33}, \ldots, y_{282}$ as values of the response. Make a plot with 6 panels. Using one panel for each value of p, plot the last 24 observations, the fitted values from corresponding fitted model, and the approximate 95% confidence intervals obtained by treating this as a standard regression problem.

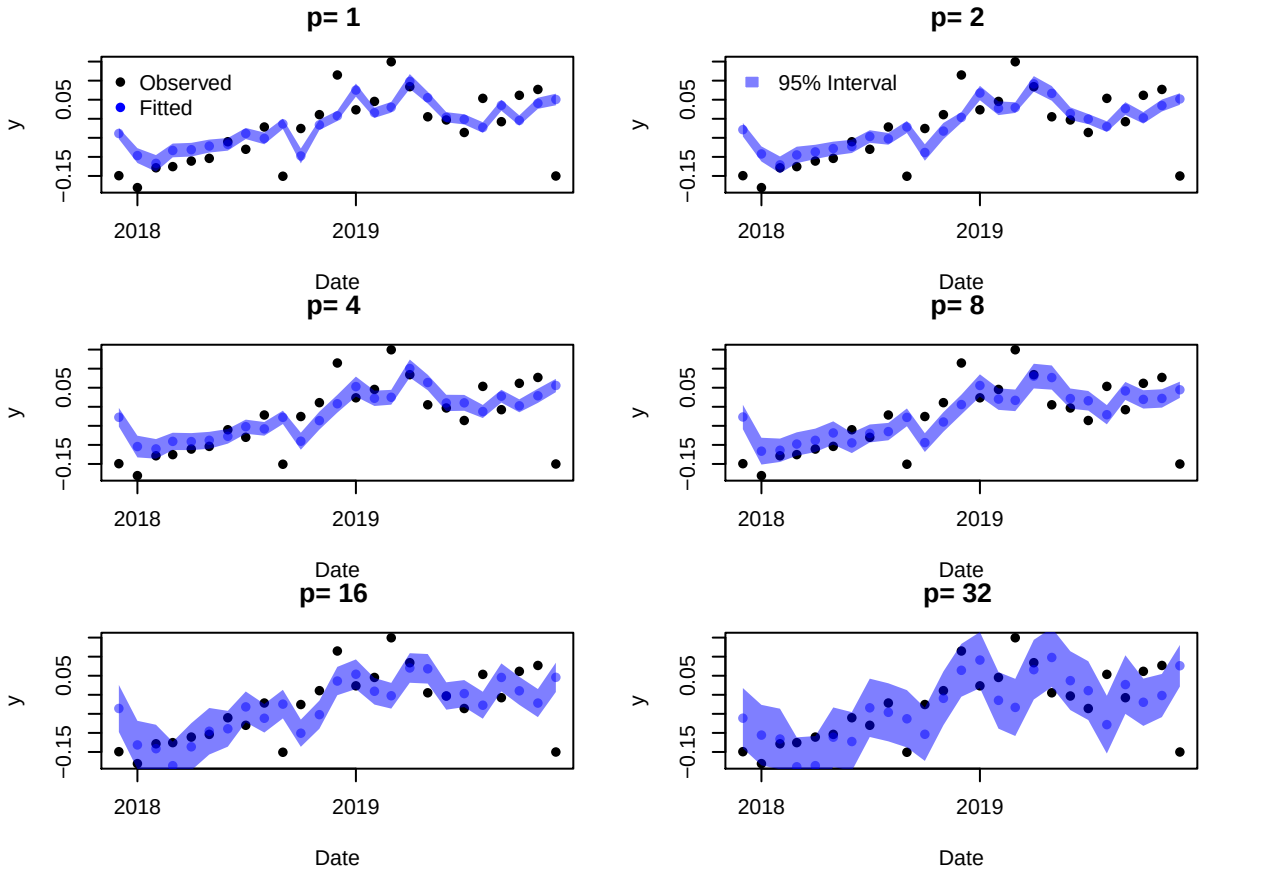
```
fits <- matrix(NA, nrow = length(y), ncol = length(ps))
ses <- matrix(NA, nrow = length(y), ncol = length(ps))
bics <- rep(length(ps))
sims <- array(NA, dim = c(length(y), length(ps), 100))
for (p in ps) {
  linmod <- lm(as.formula(paste("y~", paste(paste("y.lag", 1:p, sep = ""),
                                            collapse = "+"),
                                sep = "")),
               data = data, subset = (max(ps) + 1):(length(y) - 12))

  sims[1:p, which(ps == p), ] <- y[1:p]
  mu.hat <- linmod$coef[1]
  phi.hat <- linmod$coef[-1]
  ssw <- summary(linmod)$sigma^2
  for (i in (p + 1):length(y)) {
    sims[i, which(ps == p), ] <- mu.hat +
      t(phi.hat)%*%sims[i - (1:p), which(ps == p), ] +
      rnorm(dim(sims)[3], 0, sd = sqrt(ssw))
  }

  preds <- predict(linmod, newdata = data, se.fit = TRUE)
  fits[, which(p == ps)] <- c(preds$fit)
  ses[, which(p == ps)] <- c(preds$se.fit)
  msr <- mean(linmod$residuals^2)
  bics[which(p == ps)] <- log(msr) +
    log(length(y) - 12 - max(ps))*length(linmod$coef)/(length(y) - 12 - max(ps))
}
```

```
par(mfrow = c(3, 2))
par(mar = c(4, 4, 2, 2))
for (p in ps) {
plot(broc$fdate[(length(y) - 24):length(y)],
     y[(length(y) - 24):length(y)], pch = 16,
     xlab = "Date", ylab = "y",
     main = paste("p=", p, "\n"))
points(broc$fdate[(length(y) - 24):length(y)],
       fits[, which(p == ps)][(length(y) - 24):length(y)],
       col = rgb(0, 0, 1, 0.5), pch = 16)
polygon(c(broc$fdate[(length(y) - 24):length(y)],
          rev(broc$fdate[(length(y) - 24):length(y)])),
        c(fits[, which(p == ps)][(length(y) - 24):length(y)] +
            qnorm(0.025)*ses[, which(p == ps)][(length(y) - 24):length(y)],
```

```
          rev(fits[, which(p == ps)][(length(y) - 24):length(y)] +
                qnorm(0.975)*ses[, which(p == ps)][(length(y) - 24):length(y)])),
        border = FALSE, col = rgb(0, 0, 1, 0.5))
if (p == min(ps)) {
  legend("topleft", pch = c(16, 16),
         col = c("black", "blue"),
         legend = c("Observed", "Fitted"),
         bty = "n")
}
if (p == ps[2]) {
  legend("topleft",
         fill = rgb(0, 0, 1, 0.5),
         legend = c("95% Interval"),
         bty = "n", border = "white")
}
}
```



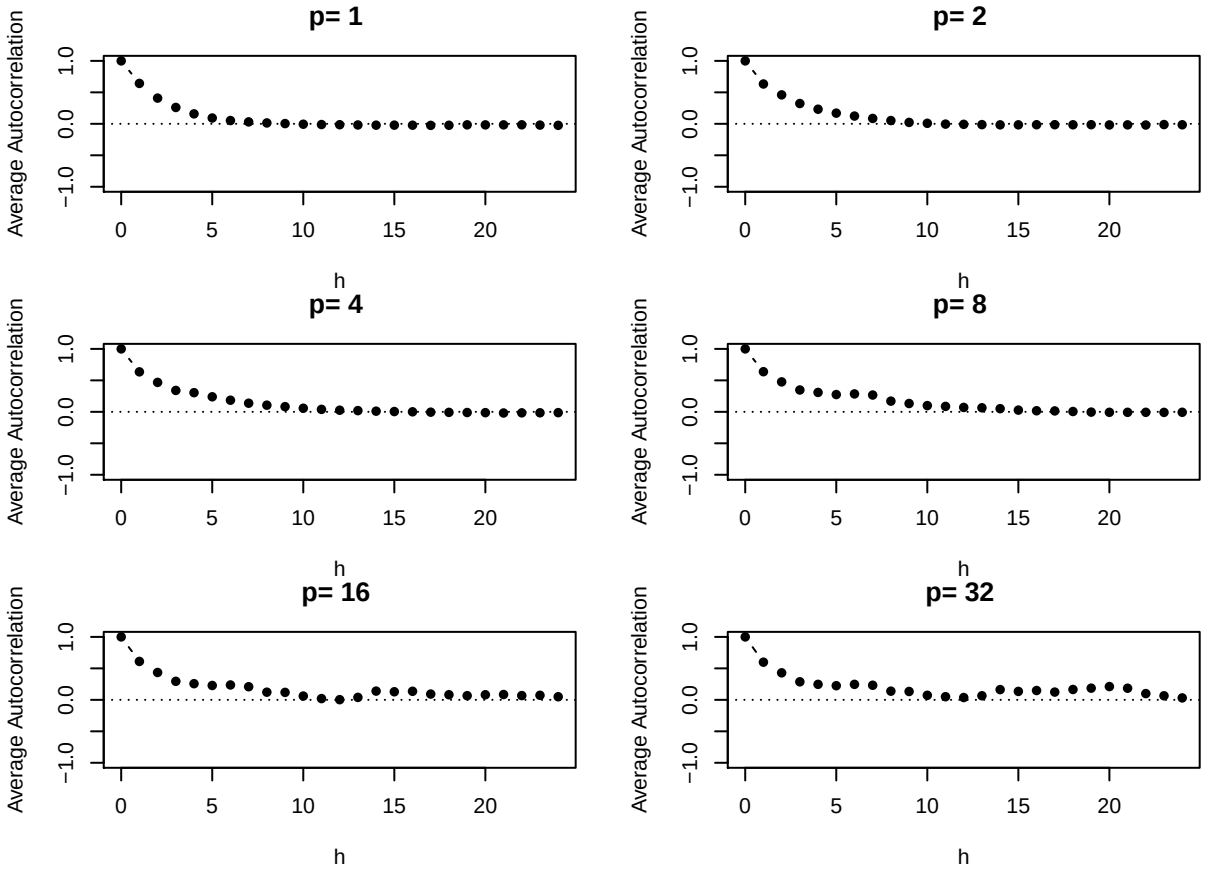
(f) Are AIC, AICc, or BIC/SIC appropriate for comparing these models fit in (e) with different values of p? If yes, pick a criterion (AIC, AICc, or BIC/SIC), justify your choice, and indicate which model you would choose.

If we fit all of the models to the same data, $y_{33}, \ldots, y_{n-12}$, then it is appropriate to use AIC, AICc, or BIC/SIC to choose a model. I tend to prefer smaller models, so I would use BIC to choose. BIC would suggest that we should choose the smallest model, with $p = 1$.

(g) For each value of $p = 1, 2, 4, 8, 16, 32$, simulate 100 synthetic time series $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(100)}$ according to the model given by (1), with $\hat{\mu}$, $\hat{\phi}_i$, and $\hat{\sigma}_w^2$ given by the linear model fits from (e).Condition on or hold

constant whatever you need to condition on. Make a plot with six panels, one for each value of p. In each panel, plot the average lag-$h = 0, \ldots, 24$ autocorrelations across all simulations. Comment on how the autocorrelations change with p.

```
sim.acfs <- apply(sims, c(2, 3), function(x) {
  acf(x, plot = FALSE)$acf[1:25]
})
mean.acfs <- apply(sim.acfs, c(1, 2), mean)
par(mfrow = c(3, 2))
par(mar = c(4, 4, 2, 2))
for (p in ps) {
  plot(0:24, mean.acfs[, which(p == ps)], pch = 16, type = "b",
       xlab = "h", ylab = "Average Autocorrelation",
     main = paste("p=", p, "\n"), ylim = c(-1, 1))
  abline(h = 0, lty = 3)
}
```

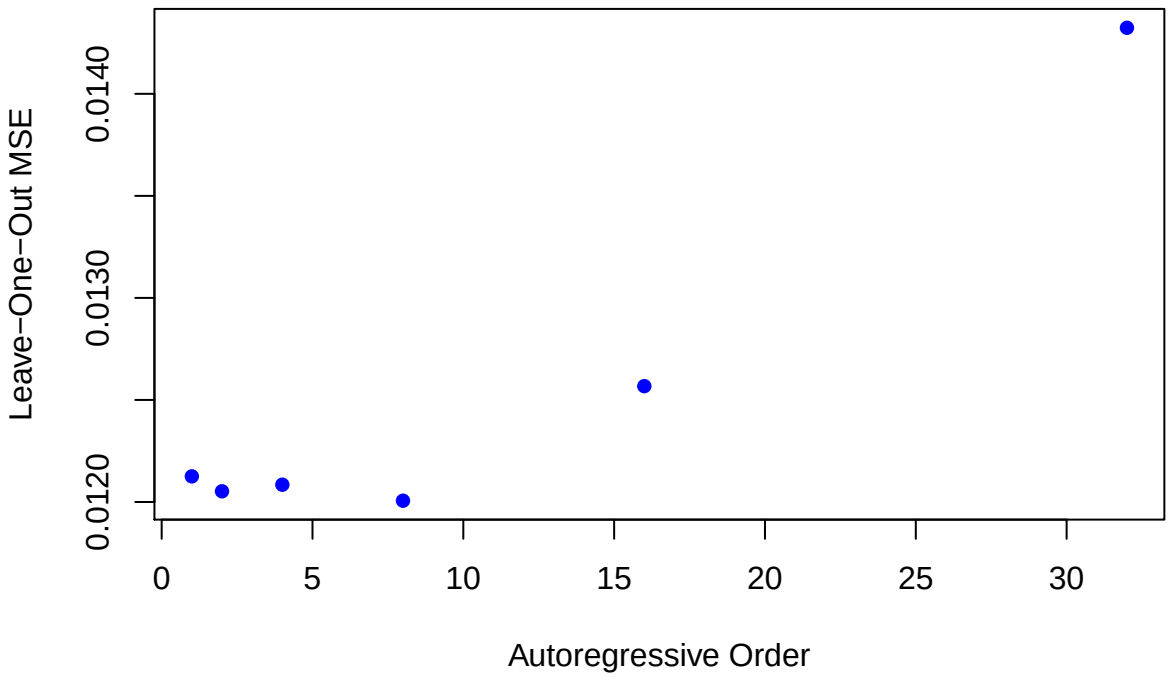


Regardless of the autoregressive order p, the average autocorrelations across simulations are start out large and positive for small h, and decrease towards 0 as h increases. Larger values of p, i.e. higher autoregressive orders, produce more flexible average autocorrelation functions. This suggests that as p increases, an autoregressive model can accommodate more complex types of time series dependence.

(h) For $p = 1, 2, 4, 8, 16, 32$, perform leave-one-out cross-validation. Leaving out observations $y_{33}, \ldots, y_{282}$ from one at a time and using as much data as possible as the response for each value of p. Record the squared error loss on the left out observations and plot the average it as a function of p. Indicate which model performs best according to this metric. Note any complications you encounter. Hint: does leaving out a single value y_t only affect the response?

```
tests <- (max(ps) + 1):(282 - max(ps))
mses <- matrix(NA, nrow = length(tests), ncol = length(ps))
for (p in ps) {
  for (i in tests) {
    samps <- (p + 1):(length(y) - 12)
    toss <- i:(i + p)
    toss <- toss[toss <= (length(y) - 12)]
  linmod <- lm(as.formula(paste("y~", paste(paste("y.lag", 1:p, sep = ""),
                                            collapse = "+"),
                                    sep = "")),
               data = data,
               subset = samps[!samps %in% toss])
  mses[which(i == tests),
       which(p == ps)] <- mean((y[toss] - predict(linmod,
                                                  newdata = data)[toss])^2)

  }
}
plot(ps, colMeans(mses), xlab = "Autoregressive Order",
     ylab = "Leave-One-Out MSE",
     pch = 16, col = "blue")
```

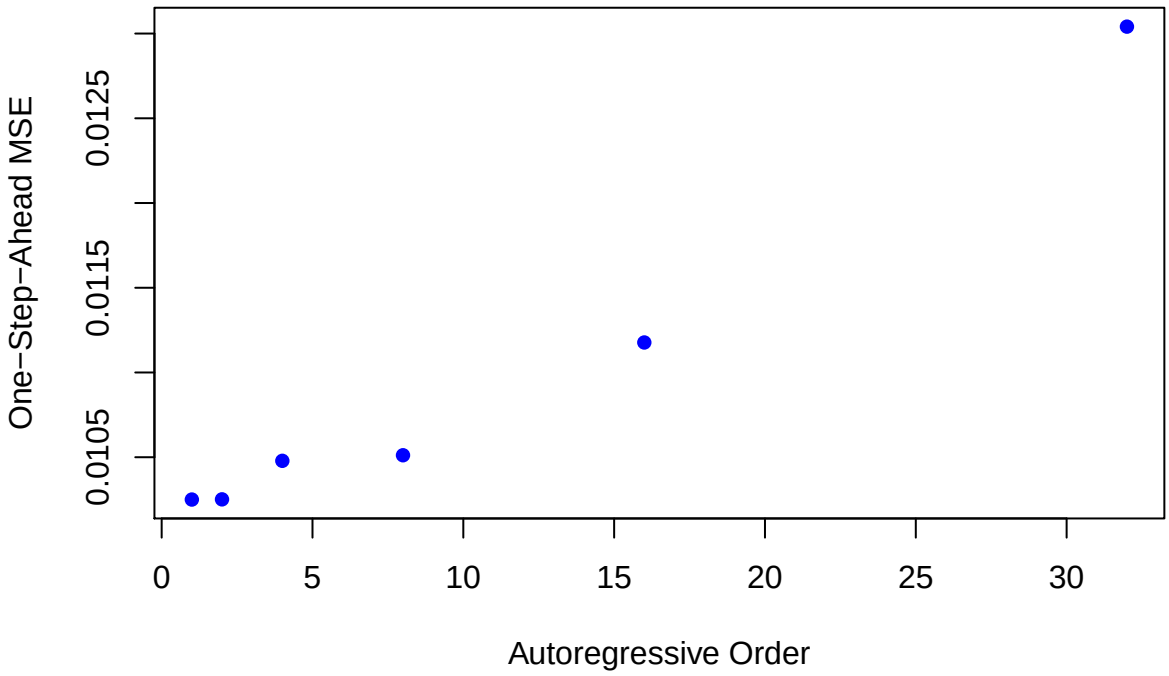


The model with $p = 8$ performs best according to this metric.

(i) For $p = 1, 2, 4, 8, 16, 32$, perform leave-one-out cross-validation. Leaving out observations y_{184}, y_{282} from the response one at a time, fit the model to the previous 150 values of the time series. Record the squared error loss on the left out observation and plot the average it as a function of p. Indicate which model performs best according to this metric.

```
tests <- 184:282
mses <- matrix(NA, nrow = length(tests), ncol = length(ps))
for (p in ps) {
  for (i in tests) {
```

```
    samps <- (i - 150):(i - 1)
  linmod <- lm(as.formula(paste("y~", paste(paste("y.lag", 1:p, sep = ""),
                                            collapse = "+"),
                                  sep = "")),
               data = data,
               subset = samps)
  mses[which(i == tests),
       which(p == ps)] <- mean((y[i] -
                                  predict(linmod, newdata = data)[i])^2)
  }
}
plot(ps, colMeans(mses), xlab = "Autoregressive Order",
     ylab = "One-Step-Ahead MSE",
     pch = 16, col = "blue")
```



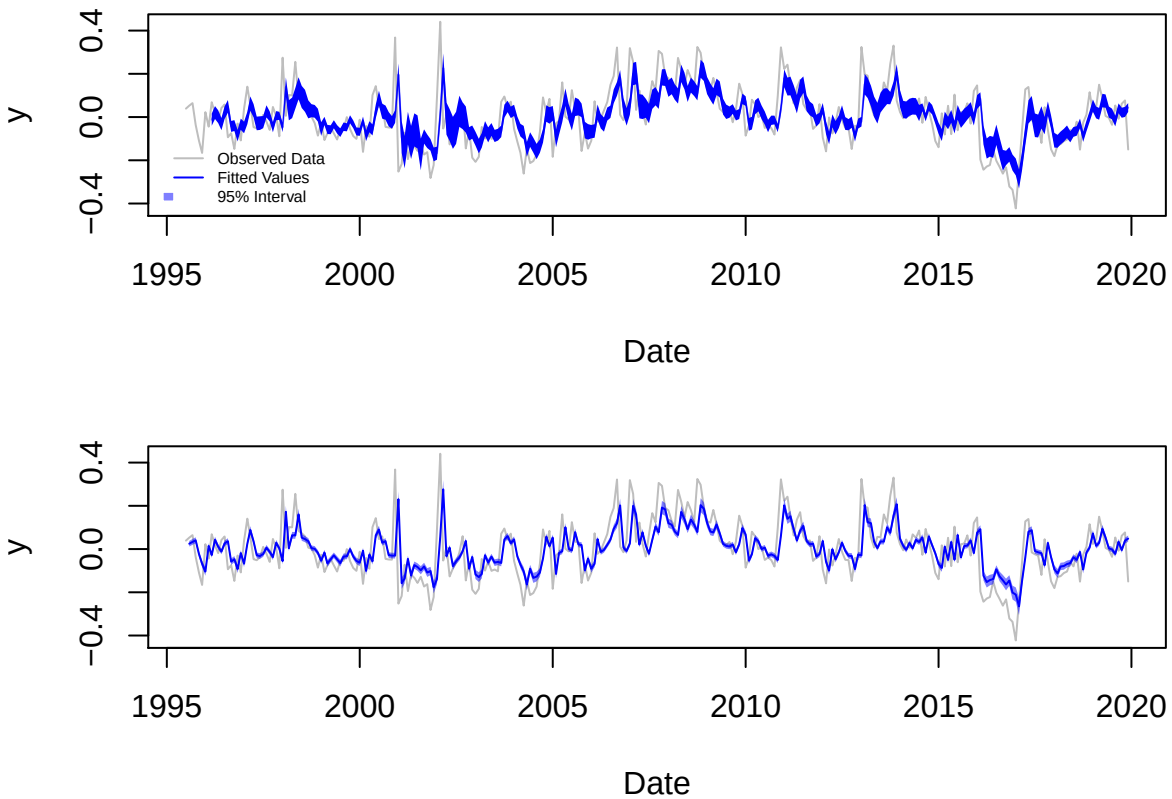
The model with $p = 1$ performs best according to this metric.

(j) Make a plot with two panels. In the first panel, plot all of the values of $\boldsymbol{y}$ along with the fitted values and approximate 95% confidence intervals using the model identified in (h). In the second panel, plot all of the values of $\boldsymbol{y}$ along with the fitted values and approximate 95% confidence intervals using the model identified in (i). Comment on how the fits from the two models compare.

```
par(mfrow = c(2, 1))
par(mar = c(4, 4, 2, 2))
plot(broc$fdate, y, type = "l", col = "gray",
     xlab = "Date")
linmod <- lm(as.formula(paste("y~",
                              paste(paste("y.lag", 1:8, sep = ""),
                                    collapse = "+"),
                                sep = "")),
               data = data, subset = 1:(length(y) - 12))
fits <- predict(linmod, data, se.fit = TRUE)
lines(broc$fdate, fits$fit, col = rgb(0, 0, 1, 0.5))
```

```
polygon(c(broc$fdate, rev(broc$fdate)),
        c(fits$fit + qnorm(0.025)*fits$se.fit,
          rev(fits$fit + qnorm(0.975)*fits$se.fit)),
        border = FALSE, col = rgb(0, 0, 1, 1))
legend("bottomleft",
       lty = c(1, 1, NA), col = c("gray", "blue", NA),
       fill = c(NA, NA,
                rgb(0, 0, 1, 0.5)),
       legend = c("Observed Data", "Fitted Values", "95% Interval"),
       bty = "n", border = "white", cex = 0.5)


plot(broc$fdate, y, type = "l", col = "gray",
     xlab = "Date")
linmod <- lm(as.formula(paste("y~", paste(paste("y.lag", 1:1, sep = ""),
                                          collapse = "+"),
                                  sep = "")),
             data = data,
             subset = 1:(length(y) - 12))
fits <- predict(linmod, data, se.fit = TRUE)
lines(broc$fdate, fits$fit, col = rgb(0, 0, 1, 1))
polygon(c(broc$fdate, rev(broc$fdate)),
        c(fits$fit + qnorm(0.025)*fits$se.fit,
          rev(fits$fit + qnorm(0.975)*fits$se.fit)),
        border = FALSE, col = rgb(0, 0, 1, 0.5))
```



Both models produce similar fitted values, however the smaller model chosen using one-step-ahead cross validation produces fitted values with less uncertainty. Additionally, the larger model chosen using leave-one-

out cross validation seems to fit the overall curve of the observed data slightly better, which makes sense because it minimizes the deviation of observed values from the fitted value at random points in the process, as opposed to just in the future as one-step-ahead cross validation does.