Homework 7 Solutions

Due: Friday 4/3/20 by 5:00pm

This week, we're going to wrap things up with ARMA models and the broccoli data. We'll try modeling dependence over time using an autoregressive moving average model of orders p and q.

Again, 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. In this problem, we're going to consider two different ways of modeling the `broc` data. Letting z_t refer to the observed broccoli prices, the two approaches we will consider are:

i. First regressing out a linear time trend and month effects obtained from fitting a linear regression model to all but the last 12 months of data. Letting y_t refer to the corresponding residuals, we will model the residuals as an ARMA(p, q) process:

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

where $y_t - \mu$ is a stationary process. We will treat the estimated regression coefficients from the initial linear regression model as fixed and known throughout.

ii. Directly incorporating the linear time trend and month effects:

$$z_t = \boldsymbol{x}_t'\boldsymbol{\beta} + \sum_{i=1}^{p} \phi_i \left(z_{t-i} - \boldsymbol{x}_{t-i}'\boldsymbol{\beta}\right) + \sum_{j=1}^{q} \theta_j w_{t-j} + w_t, \quad w_t \overset{i.i.d.}{\sim} \text{normal}\left(0, \sigma_w^2\right), \tag{2}$$

where $z_t - \boldsymbol{x}_t'\boldsymbol{\beta}$ is a stationary process.

```
load("~/Dropbox/Teaching/TimeSeries2020/stat697/content/data/broc.RData")
source("~/Dropbox/Teaching/TimeSeries2020/stat697/content/code/arma_autocovariance.R")
source("~/Dropbox/Teaching/TimeSeries2020/stat697/content/code/diy_prediction.R")
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$fdate))
n <- nrow(broc)
m <- nrow(broc) - 12
linmod <- lm(price~dayssincestart+factor(month), data = broc,
             subset = 1:m)
pred <- predict(linmod, broc)
y <- broc$price - pred
z <- broc$price
X <- model.matrix(~dayssincestart+factor(month), data = broc)
```

Order Selection

In this part, our goal is to choose the order of the ARMA(p, q) model for approaches i. and ii.. Consider $p = 0, 1, 2, 3$ and $q = 0, 1, 2$, and the following measures of model performance:

- One-step-ahead moving window style cross-validation, using the mean squared error of the forecasts
- Twelve-steps-ahead moving window style cross-validation, using the mean squared error of the forecasts
- AIC
- BIC
- Box-Pierce Test Result
  - This is the test you implemented on Homework 3, see parts (e) and (f) of the second problem

(a) Describe any choices you need to make when computing these measures of model performance, e.g. number of moving windows, and justify your choices.

In order to implement one- and twelve-step-ahead moving window style cross validation, we need to choose the number of moving windows, and the length of each one. To implement the Box-Pierce test, we need to define the number of samplem autocorrelations we will square and sum over to construct a test statistic.

Here, I will use 50 moving windows for one-step-ahead cross-validation, and 50 moving windows for twelve-step-ahead cross validation. Each window I use for one-step cross-validation will contain 232 consecutive time points, whereas each window I use for twelve-step-ahead cross-validation will contain 221 consecutive time points. These choices are somewhat arbitrary, but 50 moving windows should be enough for the average across windows to be stable, and length 232 and 221 time-series should be long enough to be reasonable representative of the observed data we are fitting the model to.

I will sum over the first 20 squared sample autocorrelations to conduct the Box-Pierce test. Again, this choice is somewhat arbitrary. The choice is motivated by the idea that most dependence across time is likely to be between observations that are close together in time, and that choosing too many sample autocorrelations could yield a test with poor power.

(b) Summarize your results in a table which shows the best choice of p and q for each measure of of model performance.

	i.	ii.
CV-1	$p = 3, q = 1$	$p = 3, q = 1$
CV-12	$p = 3, q = 2$	$p = 3, q = 2$
AIC	$p = 2, q = 1$	$p = 2, q = 1$
BIC	$p = 1, q = 0$	$p = 1, q = 0$
BP	$p = 1, q = 0$	$p = 1, q = 0$

```
# First, implement for approach i.
ps <- 0:3
qs <- 0:2
results <- array(dim = c(length(ps), length(qs), 5))
num.cv <- 50
for (p in ps) {
  cat("p = ", p, "\n")
  for (q in qs) {
    l1mse <- rep(NA, num.cv)
    l12mse <- rep(NA, num.cv)

    arma.fit <- arima(y[1:m], order = c(p, 0, q),
                          method = "ML")
    acf.fit <- acf(arma.fit$residuals, lag.max = 20,
                   plot = FALSE)$acf[-1, 1, 1]
    cr <- qchisq(0.95, df = 20)
    bpt <- m*sum(acf.fit^2)

    for (i in 1:num.cv) {
```

```
      linmod.i <- lm(price~dayssincestart+factor(month), data = broc,
             subset = i:(m - num.cv + i - 1))
      pred.i <- predict(linmod.i, broc)
      y.i <- broc$price - pred.i

      l1arma.fit <- arima(y.i[i:(m - num.cv + i - 1)], order = c(p, 0, q),
                          method = "ML")
      l1mse[i] <- (y.i[m - num.cv + i] - c(predict(l1arma.fit, n.ahead = 1)$pred))^2

      linmod.i <- lm(price~dayssincestart+factor(month), data = broc,
             subset = i:(m - num.cv + i - 12))
      pred.i <- predict(linmod.i, broc)
      y.i <- broc$price - pred.i

      l12arma.fit <- arima(y.i[i:(m - num.cv + i - 12)], order = c(p, 0, q),
                           method = "ML")
      l12mse[i] <- sum((y.i[m - num.cv + i - 12 +  1:12] -
                          c(predict(l12arma.fit, n.ahead = 12)$pred))^2)
    }


    results[which(p == ps), which(q == qs), ] <- c(mean(l1mse),
                                                   mean(l12mse),
                                                   AIC(arma.fit),
                                                   BIC(arma.fit),
                                                   bpt > cr)
  }
}
which(results[, , 1] == min(results[, , 1]), arr.ind = TRUE)
which(results[, , 2] == min(results[, , 2]), arr.ind = TRUE)
which(results[, , 3] == min(results[, , 3]), arr.ind = TRUE)
which(results[, , 4] == min(results[, , 4]), arr.ind = TRUE)
results[, , 5]
# Repeat for ii
for (p in ps) {
  cat("p = ", p, "\n")
  for (q in qs) {
    l1mse <- rep(NA, num.cv)
    l12mse <- rep(NA, num.cv)
    arma.fit <- arima(z[1:m], order = c(p, 0, q),
                          method = "ML",
                       xreg = X[1:m, ], include.mean = FALSE)
    acf.fit <- acf(arma.fit$residuals, lag.max = 20,
                   plot = FALSE)$acf[-1, 1, 1]
    cr <- qchisq(0.95, df = 20)
    bpt <- m*sum(acf.fit^2)
    for (i in 1:num.cv) {
      l1arma.fit <- arima(z[i:(m - num.cv + i - 1)], order = c(p, 0, q),
                          method = "ML",
                          xreg = X[i:(m - num.cv + i - 1), ],
                          include.mean = FALSE)
      l1mse[i] <- (z[m - num.cv + i] -
                     c(predict(l1arma.fit, n.ahead = 1,
```

```
                                newxreg = X[m - num.cv + i, ,
                                            drop = FALSE])$pred))^2
      l12arma.fit <- arima(z[i:(m - num.cv + i - 12)], order = c(p, 0, q),
                           method = "ML",
                           xreg = X[i:(m - num.cv + i - 12), ],
                           include.mean = FALSE)
      l12mse[i] <- sum((z[m - num.cv + i - 12 +  1:12] -
                          c(predict(l12arma.fit,
                                    newxreg = X[m - num.cv + i - 12 +  1:12, ,
                                                drop = FALSE],
                                    n.ahead = 12)$pred))^2)
    }


    results[which(p == ps), which(q == qs), ] <- c(mean(l1mse),
                                                   mean(l12mse),
                                                   AIC(arma.fit),
                                                   BIC(arma.fit),
                                                   bpt > cr)
  }
}
which(results[, , 1] == min(results[, , 1]), arr.ind = TRUE)
which(results[, , 2] == min(results[, , 2]), arr.ind = TRUE)
which(results[, , 3] == min(results[, , 3]), arr.ind = TRUE)
which(results[, , 4] == min(results[, , 4]), arr.ind = TRUE)
results[, , 5]
```

(c) For this problem, does the best choice of p and q depend much on the approach used?

It depends on the measure of model performance. When BIC or the Box-Pierce test is used, the best choice of p and q does not depend on the approach used. However, when other measures of model performance are used, larger models tend to be favored when approach ii. is used.

(d) Choose one measure of model performance to base your choice of p and q on, and justify your choice.

I tend to prefer the simplest plausible model, so I would use the Box-Pierce test to assess model fit. I also find it reassuring that the Box-Pierce test leads to the same conclusions as BIC, and that the conclusions do not depend on the approach used.

Forecasting

Using the best ARMA(p, q) model based on your choice in part (d) of the previous problem, compute predicted values of the rest of the time series (just the last 12 observations) using the observed data using both of the two approaches. You can use the `get.preds` function I have provided to get these predictions, with `h` set to the number of remaining observations.

(a) Make a table that shows estimates of the coefficient of the linear trend term, the autoregressive and/or moving average parameters, and noise variance fit using both approaches. Does the approach used affect the estimates much?

	i.	ii.
$\hat{\beta}$	1.231×10^{-4}	1.234×10^{-4}
$\hat{\phi}_1$	0.625	0.625

	i.	ii.
$\hat{\sigma}_w^2$	1.125×10^{-2}	1.125×10^{-2}

The approach used does not affect the estimates very much at all.

```
p <- 1
q <- 0
arma.fit.full <- arima(z[1:m], order = c(p, 0, q),
                       method = "ML",
                    xreg = X[1:m, ], include.mean = FALSE)
pred.full <- predict(arma.fit.full, n.ahead = length(y) - m,
                     newxreg = X[(m + 1):length(y), ])$pred
pred.full.se <- predict(arma.fit.full, n.ahead = length(y) - m,
                     newxreg = X[(m + 1):length(y), ])$se
arma.fit.part <- arima(y[1:m], order = c(p, 0, q),
                       method = "ML")
pred.part <- predict(arma.fit.part, n.ahead = length(y) - m)$pred +
  pred[(m + 1):length(y)]
pred.part.se <- predict(arma.fit.part, n.ahead = length(y) - m)$se
```

```
linmod$coef["dayssincestart"]
arma.fit.full$coef["dayssincestart"]
arma.fit.part$sigma2
arma.fit.full$sigma2
arma.fit.part$coef["ar1"]
arma.fit.full$coef["ar1"]
```

(b) Compare the predicted values for the last 12 observations obtained using each approach by plotting them on the same plot as the last 24 observations. Does the approach used affect the predicted values much?

See the plot after (e). The approach used does not affect the predicted values much.

(c) Use `predict` to obtain the variances of the predictions of the last 12 observations, and add 95% prediction intervals to your plot for each method. Does the approach used affect variances of the predictions much?

See the plot after (e). The approach used does not affect the variances of the predictions much.

```
nboot <- 100

pred.full.boot <- matrix(NA, nrow = nboot, ncol = length(y) - m)
pred.part.boot <- matrix(NA, nrow = nboot, ncol = length(y) - m)
beta.full <- arma.fit.full$coef[(p + q + 1):length(arma.fit.full$coef)]
beta.part <- linmod$coefficients

for (i in 1:nboot) {

  # Simulate data using the full model
  z.sim.full <- arima.sim(model = list("ar" = arma.fit.full$coef["ar1"]),
                          n = m,
                          sd = sqrt(arma.fit.full$sigma2)) + X[1:m, ]%*%beta.full
  z.sim.part <- arima.sim(model = list("ar" = arma.fit.part$coef["ar1"]),
                          n = m,
                          "sd" = sqrt(arma.fit.part$sigma2)) + X[1:m, ]%*%beta.part
```

```
  y.sim <- z.sim.part - X[1:m, ]%*%beta.part

  arma.fit.full.sim <- arima(z.sim.full[1:m], order = c(p, 0, q),
                             method = "ML",
                            xreg = X[1:m, ], include.mean = FALSE)
  beta.full.sim <- arma.fit.full.sim$coef[(p + q + 1):length(arma.fit.full.sim$coef)]
  pred.full.sim <- get.preds(z[1:m] - X[1:m, ]%*%beta.full.sim,
                             phi = arma.fit.full.sim$coef["ar1"],
                             theta = 0,
                             sig.sq.w = arma.fit.full.sim$sigma2,
                             h = length(z) - m)$pred[(m + 1):length(z)] +
    X[(m + 1):length(z), ]%*%beta.full.sim
  arma.fit.part.sim <- arima(y.sim[1:m], order = c(p, 0, q),
                             method = "ML", include.mean = FALSE)
  pred.part.sim <- get.preds(y[1:m],
                             phi = arma.fit.part.sim$coef["ar1"],
                             theta = 0,
                             sig.sq.w = arma.fit.part.sim$sigma2,
                             h = length(y) - m)$pred[(m + 1):length(y)] + pred[(m + 1):length(y)]

  pred.full.boot[i, ] <- pred.full.sim
  pred.part.boot[i, ] <- pred.part.sim

}
pred.full.boot.se <- apply(pred.full.boot, 2, sd)
pred.part.boot.se <- apply(pred.part.boot, 2, sd)
```

(d) Using the parametric bootstrap, obtain approximate standard errors of the predicted values of the last 12 observations. Remember - in approach i. we are treating the fitted values from the regression as fixed, so just simulate new residuals and continue to use the same fitted values. Add 95% confidence intervals to your plot for each method. Does the approach used affect the confidence intervals much? How do the confidence intervals compare to the prediction intervals?

See the plot after (e). The approach used does affect the approximate standard errors of the predicted values, which makes sense because the approximate standard errors for approach ii. account for uncertainty about β whereas the standard errors for approach i. do not. Regardless of the approach used, the confidence intervals tend to be much smaller than than the prediction intervals from (c).

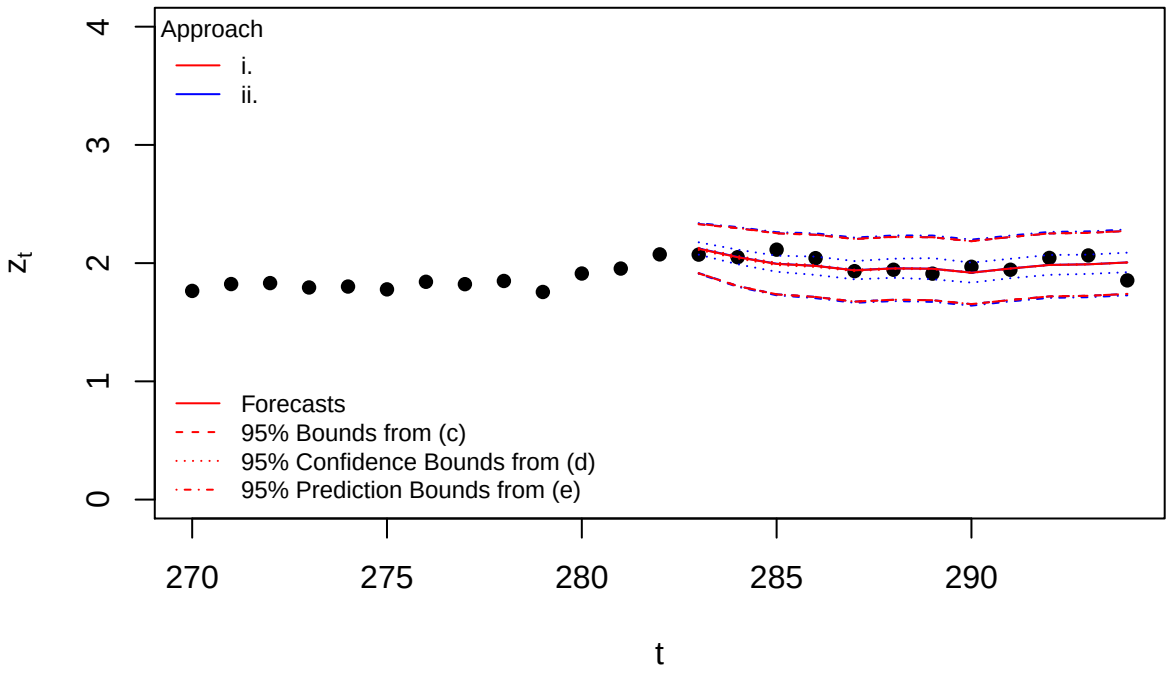
(e) The prediction intervals for the last 12 observations obtained in (c) account for variability due to uncertainty regarding future values of the time series but do not account for uncertainty regarding the values of the parameters of the model. We can obtain a variance estimate that accounts for both sources of uncertainty by taking the sum of the variances obtained in (c) and (d). Prediction intervals that account for both sources of uncertainty can be obtained using this variance new estimate. Add these prediction intervals for the last 12 observations to your plot for each approach. Does the approach used affect variances of the confidence intervals much? Which source of variability appears to dominate?

See the plot after (e). The approach used does affect the approximate standard errors of the predicted values, which makes sense because the approximate standard errors for approach ii. account for uncertainty about β whereas the standard errors for approach i. do not. The variability due to future values being unobserved appears to dominate the variability due to uncertainty about the parameter estimates.

```
plot(c((m - 12):length(y)), z[c((m - 12):length(y))],
     ylim = c(0, 4), pch = 16,
     xlab = "t", ylab = expression(z[t]))
legend("topleft", lty = c(1, 1), col = c("red", "blue"),
       legend = c("i.", "ii."), title = "Approach", cex = 0.75,
```

```
       bty = "n")
legend("bottomleft", lty = c(1, 2, 3, 4), col = c("red"),
       legend = c("Forecasts",
                  "95% Bounds from (c)",
                  "95% Confidence Bounds from (d)",
                  "95% Prediction Bounds from (e)"), cex = 0.75,
       bty = "n")
lines((m + 1):length(y), pred.full, col = "blue")
lines((m + 1):length(y), pred.full + qnorm(0.975)*pred.full.se,
      col = "blue", lty = 2)
lines((m + 1):length(y), pred.full + qnorm(0.025)*pred.full.se,
      col = "blue", lty = 2)
lines((m + 1):length(y), pred.full + qnorm(0.975)*pred.full.boot.se,
      col = "blue", lty = 3)
lines((m + 1):length(y), pred.full + qnorm(0.025)*pred.full.boot.se,
      col = "blue", lty = 3)
lines((m + 1):length(y), pred.full + qnorm(0.975)*sqrt(pred.full.boot.se^2 +
                                                       pred.full.se^2),
      col = "blue", lty = 4)
lines((m + 1):length(y), pred.full + qnorm(0.025)*sqrt(pred.full.boot.se^2 +
                                                       pred.full.se^2),
      col = "blue", lty = 4)

lines((m + 1):length(y), pred.part + qnorm(0.975)*pred.part.se,
      col = "red", lty = 2)
lines((m + 1):length(y), pred.part + qnorm(0.025)*pred.part.se,
      col = "red", lty = 2)
lines((m + 1):length(y), pred.part + qnorm(0.975)*pred.part.boot.se,
      col = "red", lty = 3)
lines((m + 1):length(y), pred.part + qnorm(0.025)*pred.part.boot.se,
      col = "red", lty = 3)
lines((m + 1):length(y), pred.part, col = "red")
lines((m + 1):length(y), pred.full + qnorm(0.975)*sqrt(pred.part.boot.se^2 +
                                                       pred.part.se^2),
      col = "red", lty = 4)
lines((m + 1):length(y), pred.full + qnorm(0.025)*sqrt(pred.part.boot.se^2 +
                                                       pred.part.se^2),
      col = "red", lty = 4)
```



Final Project

If you have not yet submitted a proposed data set for the final project, please do so as part of this assignment. At minimum, provide a link to the data set, and plot of the time series you would like to analyze, and the number of observations. You may work with your classmates to find a dataset, and you multiple students can use the same data set for their final project.