Final Exam

STA 5362: Time Series Analysis

December 9, 2015

1 Instructions

You may refer to your notes and *R* documentation to complete the exam. However, you may not speak to anyone about this exam other than me (Dr. Harvill). You may not use any external resources.

Complete the problems on the following pages. If you choose to write your solutions by hand, that is fine. However, please keep them organized, and in the order they are assigned.

Your completed work is due by 1:00 PM on Tuesday, December 15th, 2015.

2 The Exam

There are four parts to the exam.

2.1 Investigating Estimators of the Coefficients of an AR(p) Process.

Let the process $\{X_t,\ t \in \mathbb{Z}\}$ represent a stationary autoregressive process of order p, AR(p); that is, X_t satisfies the stationary stochastic difference equation,

$$X_t + \alpha_1 X_{t-1} + \cdots + \alpha_p X_{t-p} = \varepsilon_t,$$

where $\{\varepsilon_t,\ t \in \mathbb{Z}\}$ is a white noise process with variance $\sigma^2 < \infty$.

There are many methods for estimating the coefficients $\alpha_1, \ldots, \alpha_p$ of an AR(p). The four most popular are the Yule-Walker estimators, the ordinary least squares estimators, the maximum likelihood estimators, and the Burg estimators. The question is, "Which of the four is 'best'?"

You will use the function `ar_est` in the file `arestsim.R` (on Canvas) to investigate the properties of the four estimators. The model you will study is

$$X_t - 0.3X_{t-1} + 0.8X_{t-3} = \varepsilon_t, \qquad \sigma^2 = 1,$$

so that $\alpha_1 = -0.3, \alpha_2 = 0$, and $\alpha_3 = 0.8$. Properties of the estimators will be investigates for series lengths $n = 50, 100, 250$, and 500. For each series length, take the number of replications to be `nreps` = 500. Documentation for that function can be found in Part 3.1 on page 7.

A. For each coefficient, use the `summary` function (in Base R) to obtain a numerical description of the estimates. Construct and complete a set of tables to organize the results.

B. For each coefficient, create a 2×2 array of plots where the top left plot is a probability histogram of the estimates of the coefficient for $n = 50$; the top right plot is the same, but for $n = 100$; the bottom left for $n = 250$; and the bottom right for $n = 500$.

   (i) The lengths of the horizontal axes should be equal for all graphs for the same coefficient. Likewise, the heights of all vertical axes for all four histograms should be equal. Entitle each histogram with the type of estimator and series length.

   (ii) On each histogram, superimpose a red vertical line at the (true) value of α_j.

   (iii) On each histogram, superimpose a blue vertical line at the mean value of $\hat{\alpha}_j$ across all replications.

   (iv) On each histogram, place brackets on either side of the mean at $\pm$ one standard error of $\hat{\alpha}_j$ computed using the standard deviation of $\hat{\alpha}_j$ across the replications.

C. For each coefficient, create a 2×2 array of plots where the top left plot is four side-by-side boxplots of the estimates of the coefficient for $n = 50$; the top right plot is the same but for $n = 100$; the bottom left for $n = 250$; and the bottom right for $n = 500$. Label each boxplot with the name of the estimate it represents. Be sure the series length is indicated somewhere on the plot.

D. If there are other numerical or graphical summaries that you believe would assistance in the investigation of the estimators, include that. (This part is optional.)

Turn in the plots for α_2 for series lengths `n = 100` and `n = 500`.

Answer the following questions.

1. What is the shape of the sampling distribution of the estimators?

2. Which, if any, of the estimators appear to be unbiased? If an estimator is biased, does the increasing series length appear to have an affect on the bias?

3. As n is increasing, what is happening to the standard error of the estimators?

4. Provide one- to two-paragraph summary of the properties you've observed based on this simulation study.

5. If you were to select one of the four estimation methods, which would you chose, and why?

2.2 Investigating Estimators of the Coefficients of an MA(q) Process.

Let the process $\{X_t,\ t \in \mathbb{Z}\}$ represent an invertible moving average process of order q, MA(q); that is, X_t is a linear filter of order q white noise, given by

$$X_t = \varepsilon_t + \beta_1 \varepsilon_{t-1} + \cdots + \beta_q \varepsilon_{t-q},$$

where $\{\varepsilon_t,\ t \in \mathbb{Z}\}$ is a white noise process with variance $\sigma^2 < \infty$.

There are many methods for estimating the coefficients $\beta_1, \ldots, \beta_q$ of an MA(q). The two most popular are the maximum likelihood estimators and the conditional sums of squares estimators. The question is, "Which of the two is 'better'?"

You will use the function `arma_est` in the file `armaestsim.R` (on Canvas) to investigate the properties of the two estimators. The model you will study is

$$X_t = \varepsilon_t - 0.3\varepsilon_{t-1} + 0.8\varepsilon_{t-3}, \qquad \sigma^2 = 1,$$

so that $\beta_1 = -0.3, \beta_2 = 0$, and $\beta_3 = 0.8$. Properties of the estimators will be investigates for series lengths $n = 50, 100, 250$, and 500. For each series length, take the number of replications to be `nreps` $= 500$. Repeat the same exercises as you did for the AR(3) process in 2.1. Turn in the plots for β_2 for series lengths `n = 100` and `n = 500`.

2.3 Data Analysis.

There are seven data sets on Canvas that you described as a part of an early homework assignment. (Please download those data sets again to be sure you have the correct versions. One of the data sets was corrupted.)

1. Modify the `descplot` function in *Timeslab* so that (the natural logarithm of the standardized) smoothed periodogram is plotted in the place of the "raw" periodogram. For help on this, see Part 3.3. Create graphs of the descriptive statistic for each series, using `m = 30` lags. For the optimal value of `M`, choose the number $\texttt{M} = 4n^{1/5}/3$, or the value closest to that so that $0.05 \leq \texttt{M}/n \leq 0.10$. Turn in this set of plots. Do **not** summarize the graphs in writing. (You already did that in a homework assignment. I do not want you to repeat that here.)

2. Use the plots you created in part 1, use the sample autocorrelation function and sample partial autocorrelation function to make an educated guess at the process that resulted in the realization.

   (a) If you determine the process is white noise, estimate the process variance.

   (b) If you determine the process is a random walk, take the first difference. Analyze the differenced data. Does the differenced data help to confirm or refute that the process is a random walk? Explain.

   (c) If you determine the process is a harmonic process, determine the frequency components, and estimate the corresponding amplitudes. (You may have already done this on your homework. Repeat that here.)

   (d) If you determine the process is an AR(p), MA(q), or ARMA(p, q), complete the following.

      (i) Determine the optimal order or orders. You will be given the correct answer in a few more sentences. The point of this exercise is to compare the data-driven results with the correct orders to see which order selection criteria is best. Let the maximum orders be 5 (`max.p = 5` and `max.q = 5`).
      Use the following statistics to help make your determinations: (1) sample autocorrelation function and sample partial autocorrelation functions; (2) Akaike's information criteria (AIC); (3) corrected version of AIC (AICC); and (4) BIC. See part 3.4. You will need the library `forecast` to complete this part of the exam. Create a table that specifies the optimal orders using each of the sample autocorrelation structures, AIC, AICC, and BIC. Compare those results with the true orders of the three process which are as follows: for Series 5, $p = 2, q = 0$; for Series 6, $p = 0, q = 3$; for Series 7, $p = 2, q = 3$. The true orders should be included in your table.

      (ii) Fit the model using the maximum likelihood estimates using the optimal orders based on AICC. If the true orders differ from the optimal orders, fit the model using the true order too (using maximum likelihood). State the fitted model(s).

2.4 Predicting ARMA(p, q) models.

Series 8 (on Canvas) is a new data set. The true orders of the model are $p = 1$ and $q = 2$.

1. Fit an ARMA(1,2) model to all but the last ten observations of the data using the `arima` function (see part 3.5).
2. Use the function `predict` (see part 3.6) to predict the last ten values and to compute the standard error of prediction for each.
3. Plot the entire time series, including the final ten observations that were not used in fitting the model. Connect the first 240 values with a solid line, and the last 10 with a dashed line (`lty = 2`).
4. On the time plot, superimpose the fitted values onto the plot using a blue line. Use the standard errors of the predicted values to superimpose ± 1.645 standard error confidence bounds around the predicted values using red lines. How many prediction intervals contain the realized data value?

3 Documentation for *Timeslab* and *R* Functions

The following pages contain help on using the *R* and *Timeslab in R* functions that you will need to complete this exam. The information here is an abbreviated version of the help files that focuses only topics relevant to the exam.

3.1 The `ar_est` Function

The `ar_est` function is written specifically to aid in completing 2.1. The call to the function is

```
ar_est(n = c(50, 100, 250, 500), alpha, rvar = 1, nreps = 500, seed = 0)
```

The function generates `nreps` time series from a stationary AR(p) process, $p \leq 3$. For each series length, the function computes estimates of the AR coefficients using each of the Yule-Walker, maximum likelihood, ordinary least squares, and Burg methods. For each method an array of dimension (p, `length(n)`, `nreps`) is returned containing the estimates.

Example: Consider the process

$$X_t - 0.3X_{t-1} + 0.8X_{t-3} = \varepsilon_t, \qquad \sigma^2 = 1.$$

To generate `nreps = 500` series from this process, and compute estimates of the AR coefficients for each of the replicates for series lengths `n = c(50, 100, 250, 500)`, the call to the function is

```
example1 <- ar_est(alpha = c(-0.3, 0, 0.8))
```

To list the Yule-Walker estimates of the AR coefficients for the first five replications for series length `n = 100`, type

```
example1$yw.est[,2,1:5]
```

A matrix of dimension 3×5 is returned. The first row are the Yule-Walker estimates of α_1 for the first five replications of length 100 of the process. The second row are the Yule-Walker estimates of α_2 for the first five replications of length 100 of the process. And the third row are the Yule-Walker estimates of α_3 for the first five replications of length 100 of the process. Estimates from the other methods are contained in the similarly defined objects `example1$ols.est`, `example1$mle.est`, and `example1$burg.est`.

One option available for summarizing the estimates for a particular α_j, $j = 1, 2, 3$, across the specified series lengths is using the `apply` function. For example, to summarize the maximum likelihood estimates for α_2 across (all) `nreps = 500` replications by series lengths, use

```
apply(example1$mle.est[2,,1:500],1,summary)
```

This returns

```
             [,1]       [,2]       [,3]       [,4]
Min.    -0.27670 -1.601e-01 -0.0967900 -0.0944500
1st Qu. -0.04454 -4.063e-02 -0.0275300 -0.0195300
Median   0.01037  4.637e-05  0.0008977  0.0002438
Mean     0.02133  8.768e-03  0.0041140  0.0011510
3rd Qu.  0.07950  5.290e-02  0.0332500  0.0208900
Max.     0.51760  2.575e-01  0.1931000  0.0811900
```

The first column `[,1]` is the summary statistics for the maximum likelihood estimates of α_2 for `n = 50`; the second column is for `n = 100`; the third is for `n = 250`, and the fourth column is for `n = 500`.

3.2 The `arma_est` Function

The `ar_est` function is written specifically to aid in completing 2.2. The call to the function is

```
arma_est(n = c(50, 100, 250, 500), alpha, beta, rvar = 1, nreps = 500,
                 seed = 0)
```

The function generates `nreps` time series from an invertible, stationary ARMA(p, q) process, $p \leq 3$ and $q \leq 3$. For each series length, the function computes estimates of the ARMA coefficients using both of the maximum likelihood and conditional sum of squares methods. For each method an array of dimension ($p + q$, `length(n)`, `nreps`) is returned containing the estimates. The first p matrices are the AR coefficient estimates; the final q are the MA coefficient estimates.

Example: Consider the process

$$X_t = \varepsilon_t - 0.3\varepsilon_{t-1} + 0.8\varepsilon_{t-3}, \qquad \sigma^2 = 1.$$

To generate `nreps = 500` series from this process, and compute estimates of the AR coefficients for each of the replicates for series lengths `n = c(50, 100, 250, 500)`, the call to the function is

```
example2 <- arma_est(alpha = 0, beta = c(-0.3, 0, 0.8))
```

To list the maximum likelihood estimates of the MA coefficients for the first five replications for series length `n = 250`, type

```
example2$mle.est[,3,1:5]
```

A matrix of dimension 3×5 is returned. The first row are the maximum likelihood estimates of β_1 for the first five replications of length 250 of the process. The second row are the maximum likelihood estimates of β_2 for the first five replications of length 250 of the process. And the third row are the maximum likelihood estimates of β_3 for the first five replications of length 250 of the process. The conditional sum of squares estimates contained in the similarly defined array `example2$css.est`.

3.3 Smoothing the Periodogram

The periodogram, $\hat{f}(\omega_j)$ is an estimator of the spectral density function $f(\omega)$ at the natural frequencies $\omega_j = (j-1)/n,\ j = 1, 2, \ldots, n$. As discussed in class, and seen in homework, the periodogram is not a consistent estimator of $f(\omega)$. In fact, it turns out that, if a process $\{X_t : t \in \mathbb{Z}\}$ has a spectral density $f(\omega)$ that is continuous at frequency ω, and the cumulant function is absolutely summable, then

$$\lim_{n\to\infty} \text{Var}\left(\hat{f}(\omega)\right) = \begin{cases} f^2(\omega) & \omega \neq 0, 0.5 \\ 2f^2(\omega) & \omega = 0, 0.5. \end{cases}$$

In other words, the limit of the variance of the estimator $\hat{f}(\omega)$ is *not zero*, and so $\hat{f}(\omega)$ is not consistent.

Moreover, if $\{X\}$ follows a general linear process, then for frequencies $\omega_1, \omega_2, \ldots, \omega_M$ in the (open) interval (0, 0.5), the random variables

$$\frac{2\hat{f}(\omega_1)}{f(\omega_1)}, \frac{2\hat{f}(\omega_2)}{f(\omega_2)}, \cdots, \frac{2\hat{f}(\omega_M)}{f(\omega_M)}$$

converge in distribution to the M independent random variables $Q_1, Q_2, \ldots, Q_M$ each having a χ^2 distribution with two degrees of freedom. (The result holds for the natural frequencies). The asymptotic distributions of

$$\frac{\hat{f}(0)}{f(0)} \quad \text{and} \quad \frac{\hat{f}(0.5)}{f(0.5)}$$

are χ^2 with one degree of freedom.

If $\{X\}$ is Gaussian white noise, then the periodogram ordinates are independent and identically distributed with

$$\begin{array}{ll} \frac{2\hat{f}(\omega_j)}{\sigma^2} \sim \chi^2_2 & \omega_j \neq 0, 0.5 \\ \frac{\hat{f}(\omega_j)}{\sigma^2} \sim \chi^2_1 & \omega_j = 0, 0.5. \end{array}$$

Although the "raw" periodogram is not a consistent estimator of the spectral density function, a smoothed version of the periodogram is a consistent estimator of $f(\omega)$. The

Timeslab function `windowf` computes a smoothed version of the periodogram. The call to the function `windowf` is

```
windowf(rho, R0, Q, ioptw, M, n, alpha = 0.05)
```

The arguments are

`rho`: a real vector containing the sample autocorrelation function of the time series.

`R0`: a real scalar containing an estimate of the time series variance.

`Q`: an integer scalar indicating how many frequencies are to be computed. Usually `Q` is the length of the series.

`ioptw`: an integer scalar between one and eight, indicating the window used to smooth the periodogram. There are eight possible windows from which to choose. We will use `ioptw = 4`, the Parzen window.

`M`: an integer scalar determining the window width. For the Parzen window, the suggested optimal value for `M` is the greatest integer value of $4n^{1/5}/3$.

`n`: an integer scalar containing the length of the original time series (for the purposed of a confidence band).

`alpha`: a real scalar containing the confidence level for confidence bands.

Example. Let `x` contain a time series. To obtain a smoothed window periodogram, and plot the natural logarithm of the standardized version of the smoothed periodogram, use the following.

```
n <- length(x)
M <- floor((4/3)*n^(1/5))
r <- corr(x, M)
fhat <- windowf(rho = r$corr, R0 = r$var, Q = n, ioptw = 4, M, n = n)$f
plotsp(fhat, n = n, div = r$rvar, main = "Smoothed Spectral Estimate")
```

In fact, when smoothing is used for estimating the spectral density, Parzen himself suggests using three values of `M`, M_1, M_2, and M_3, where

$$0.05 \leq \frac{M_1}{n} \leq 0.10, \qquad 0.10 \leq \frac{M_2}{n} \leq 0.25, \qquad 0.25 \leq \frac{M_3}{n} \leq 0.75.$$

The *Timeslab* function `windsp3` will compute a smoothed periodogram estimate for three values of `M` and superimpose the natural logarithm of the standardized version of then on the same set of axes. The call to `windsp3` is

```
windsp3(x, ioptw = 4, Q = length(x), M1 = 10, M2 = 20, M3 = 40,
            main = "Log Standardized Window Spectral Estimates")
```

3.4 Determining ARMA Orders

The *R* library `forecast` contains the function `auto.arima` which returns optimal orders for a ARIMA(p, d, q) using one of three order selection information criteria. The three criteria are the Akaike's information criteria (AIC), corrected AIC (AICC), and Bayesian information criteria (BIC), respectively defined by

$$\begin{aligned} \text{AIC} &= -2\log(L) + 2(p+q+1) \\ \text{AICC} &= AIC + \frac{2(p+q+1)(p+q+2)}{n-p-q-2} \\ \text{BIC} &= AIC + (\log(n)-2)(p+q+1), \end{aligned}$$

where L is the likelihood of the data (the errors are assumed Gaussian). The "best" set of orders is that set that minimizes the chosen information criteria. AICC and BIC are typically preferred over AIC, as AIC tends to result in a model with orders that are too large.

To use the function `auto.arima`, you will need the library `forecast`. The call to `auto.arima` has a large number of arguments. This discussion is restricted to only those arguments required to complete this exam. With that in mind, the call to the function is

```
auto.arima(x, d = NA, D = NA, max.p = 5, max.q = 5,
   stationary = FALSE, seasonal = TRUE, ic = ("aicc", "aic", "bic"))
```

The arguments above are

`x`: a real vector containing the time series.

`d`: an integer scalar containing the order of first-differencing. If missing (the default), the function will choose a value based on the KPSS test.

`D`: an integer scalar containing the order of seasonal differencing. If missing (the default), the function will choose a value based on the OCSB test.

`max.p`: an integer scalar giving the maximum value for the AR order. The default value is `max.p = 5`.

`max.q`: an integer scalar giving the maximum value for the MA order. The default value is `max.q = 5`.

`stationary`: A logical argument; if `TRUE`, the search is restricted to stationary models. The default is `stationary = FALSE`.

`seasonal`: A logical argument; if `FALSE`, the search is restricted to non-seasonal models.

`ic`: A character indicating which information criteria is to be used in model selection. The default is `ic = "aicc"`.

There are some defaults in the call to `auto.arima` that we need to change. Specifically, we will use `d = 0`, `D = 0`, `stationary = TRUE`, and `seasonal = FALSE`. Secondly, the auto.arima function should be applied to a demeaned series; that is to $x - \bar{x}$. Let `x` represent the time series we want to fit. Then the series of calls (to complete the problem in this exam) is as follows.

```
library("forecast")  # Only necessary once per R session.
xx <- x - mean(x)    # Demean the series
auto.arima(xx, d = 0, D = 0, stationary = T, seasonal = F, ic = "aic")
auto.arima(xx, d = 0, D = 0, stationary = T, seasonal = F, ic = "aicc")
auto.arima(xx, d = 0, D = 0, stationary = T, seasonal = F, ic = "bic")
```

To illustrate, I generated data from a ARMA(2,2), and followed the procedure above. The resulting output is below. They are summarized in Table 1.

```
> auto.arima(xx, d = 0, D = 0, stationary = T, seasonal = F, ic = "aic")
Series: xx
ARIMA(2,0,2) with zero mean

Coefficients:
          ar1      ar2      ma1      ma2
      -0.0072  -0.8146  -0.6309  -0.1751
s.e.   0.0443   0.0377   0.0766   0.0781

sigma^2 estimated as 1.098:  log likelihood=-368.48
AIC=746.96   AICc=747.21   BIC=764.57
> auto.arima(xx, d = 0, D = 0, stationary = T, seasonal = F, ic = "aicc")
Series: xx
ARIMA(2,0,2) with zero mean

Coefficients:
          ar1      ar2      ma1      ma2
      -0.0072  -0.8146  -0.6309  -0.1751
s.e.   0.0443   0.0377   0.0766   0.0781

sigma^2 estimated as 1.098:  log likelihood=-368.48
AIC=746.96   AICc=747.21   BIC=764.57
> auto.arima(xx, d = 0, D = 0, stationary = T, seasonal = F, ic = "bic")
Series: xx
ARIMA(3,0,1) with zero mean

Coefficients:
         ar1      ar2     ar3      ma1
```

```
      0.1951  -0.8255  0.1630  -0.8466
s.e.  0.0829   0.0360  0.0818   0.0488

sigma^2 estimated as 1.101:  log likelihood=-368.83
AIC=747.66   AICc=747.91   BIC=765.27
```

Table 1: Summary of `auto.arima` function to determine the optimal orders of an ARMA(p, q) process. The demeaned simulated data were generated from the model $X_t + 0.8X_{t-2} = \varepsilon_t - 0.65\varepsilon_{t-1} - 0.2\varepsilon_{t-2},\ t = 1, 2, \ldots, 250$.

	Order	
Method	AR order	MA order
AIC	2	2
AICC	2	2
BIC	3	1
Model	2	2

3.5 Fitting an ARMA(p, q) Model

The function `arima` is a function the `stats` library of *R*. The call to the function is below. As in previous sections, the only items included are those that are relevant to completing the tasks in this exam. There is much more to the `arima` function than what is presented here.

```
arima(x, order = c(0L, 0L, 0L), include.mean = TRUE,
          method = c("CSS-ML", "ML", "CSS"))
```

The arguments are

`x`: a real vector containing the time series.

`order`: an integer vector of length three containing value of the (p, d, q) orders; p is the AR order, d the order of differencing, and q the MA order.

`include.mean`: a logical indicating whether or not to include a mean in the fit. The default is `include.mean = TRUE` for undifferenced series, and is ignored for ARIMA models with differencing.

`method`: a character variable indicating the type of method to use in fitting the model. The three methods are maximum likelihood (ML) or conditional sum of squares (CSS). The default (unless there are missing values) is to use CSS to find initial values, and then ML.

Before using the `arima` function, you should subtract the sample mean from the time series.

Example. Suppose the original series in `x` is to be fitted using an ARIMA(2,0,2) model using CSS to get initial coefficient estimates and then ML to fit the model. To accomplish this task using the `arima` function, you need the following.

```
xx <- x - mean(x)
fit.xx <- arima(x, order = c(2, 0, 2), include.mean = F, method = "CSS-ML")
summary(fit.xx)
```

It should be noted that it is not absolutely necessary to assign the result of applying the `arima` function to an object. However, it will make predicting from that series a little easier.

3.6 Predicting an ARMA(p, q) Model

The `predict` function is part of the `stats` library in *R*. The call to the function (for a time series) is

```
predict(ojbect, n.ahead = 1)
```

The arguments are

`object`: The result of using the function `arima` to fit a demeaned time series.

`n.head`: An integer scalar indicating how many steps ahead to predict. The default is `n.ahead = 1`.

Example. Suppose we want to predict the original series in `x` ten steps ahead using a fitted ARIMA(2,0,2) model. To accomplish this task using the `predict` function, you need the following.

```
xx <- x - mean(x)
fit.xx <- arima(x, order = c(2, 0, 2), include.mean = F, method = "CSS-ML")
pred.xx <- predict(fit.xx, n.ahead = 10)
#  To get predicted values use
pred.xx$pred[1:10]
#  To get standard error of the predicted values use
pred.xx$se[1:10]
```

It should be noted that it is not absolutely necessary to assign the result of applying the `arima` function to an object. However, it will make predicting from that series a little easier.