Stat 5170: Homework 2

due Jan 31

1 R Tutorial

Now we will learn to write our own functions in R. (If you would like more information about writing functions, see Chapter 10 of *An Introduction to R.*) One important reason for learning about functions is to allow us to run simulations. In this section, we will write a function to simulate AR processes. For the homework, you will write your own function to simulate MA processes.

1. We will start by writing a very simple function, so we can learn how things work in R. Functions take arguments, do something to the arguments, and return a result. Start a new script titled "add.R". Here is a short function to enter at the top of your script that will simply add two numbers and return the result.

   ```
   add<-function(a,b){
   result<-a+b
   result}
   ```

   Now, run your script. Then, type `add(2,3)`, and notice the result. This function has two arguments, `a` and `b`. The result is returned by simply typing the object that you would like to have returned, in this case, the numerical variable `result`.

2. Another example would be to write a function to simulate a random walk process. We will need to take arguments for the length of the series and the variance of the white noise process and return a vector representing the random walk. Here is one possible implementation:

   ```
   randomwalk<-function(sigsq,T){
   x<-rep(0,T)
   w<-rnorm(T,sd=sqrt(sigsq))
   for ( i in 2:T){
   x[i]<-x[i-1]+w[i]
   }
   x
   }
   ```

(Note that I am assuming an initial condition of zero.) Try a few sample paths, and calculate the ACF of those sample paths, e.g. `acf(randomwalk(1,10))`.

3. Now, we will write a function to simulate an $AR(p)$ process of length T. We will need the AR parameters $\phi_1, ..., \phi_p$, the variance of the white noise σ_w^2, and the length of the time series we would like to generate T. These will be the arguments for our function.

   It would be nice if we could write one function that will generate an AR time series regardless of p. We will accomplish this by entering the AR parameters as a vector. Our function will be as follows:

```
arsim<-function(phis, sigsq, T){
  p<-length(phis)
  #find the number of lags in our AR
  noise<-rnorm(T+p, sd=sqrt(sigsq))
  #generate the white noise plus a few to get started
  x<-c(noise[1:p],rep(0,T))
  #put the initial noise terms in and set the rest to zero
  for (i in (p+1):(T+p)){
  #this loop generates the AR series with the recursive formula
    x[i]<-phis %*% x[i-(1:p)] +noise[i]
    }
  x<-x[(p+1):(T+p)]  #throw away those initial starting points
  x #return the time series
}
```

   We should walk through this function definition to make sure we understand things. First, we see that there are three arguments. The first argument is `phis` which is a vector containing the AR parameters in order: $\phi_1, ..., \phi_p$. The second is `sigsq`, the variance of the white noise, and the last is how many observations we want, `T`.

   The first line of the function finds the length of the `phis` which is p. The second line generates $T + p$ independent normal random variables with variance `sigsq`. In terms of how we have been writing our models, this command generates the w_t. We generate p extra white noise terms to use as our initial values for the first p values of our time series. We have to start our series in some way.

   We are going to use the variable `x` to store our time series. So, the next command sets up the vector `x` with the first p white noise terms and the rest zeros. The next set of statements is a `for` loop. This loop generates the time series using our recursive definition of an AR series. Note the operator `%*%`. This take the inner product of two vectors. In other words, the first element of vector one times the first element of vector two plus the second element of vector one times the second element of vector two plus and so on. The command after the for loop simply gets rid of those first p white noise

terms leaving us with the AR process only. The final `x` returns our time series.

4. Now that we have a function to generate an AR series, let us put it to work. Generate an $AR(1)$ time series of length 200 and with $\phi_1 = 0.5$ and $\sigma_w^2 = 1$ using the command

   ```
   x1<-arsim(c(0.5), 1,200)
   ```

   Take a look at a plot of the data and the ACF of this time series. Is this what you would expect?

   Also, generate a similar time series but with $\phi_1 = -0.5$. How do the two time series look different? How about the ACF's?

5. We see with a $|\phi_1| < 1$ the process appears reasonable and stationary, and the last simulations you did confirmed that. Try a ϕ_1 of 1.1, 1.01, and 1.001. What do you notice?

2 Homework 2

Directions:

- Turn in your answers via Assignments on Collab.
- Please be sure to clearly label the question you are answering.
- If you are submitting multiple files, giving proper descriptive names will be helpful to the grader.
- Please avoid submitting word documents.
- R code should be submitted as R scripts.
- Handwritten work is fine as long as your handwriting is legible to the grader. You may upload a clear scan or picture of your handwritten work.

1. Write a function called `masim` to generate a $MA(q)$ series of length T. Be sure to turn in the code for your function as a separate R script; do not put your code in the pdf document (the grader should be able to open your R script and test run your code easily).

2. Make sure that your function works. Simulate a model with $\sigma_w^2 = 1$, $\theta_1 = 0.5$, and $\theta_2 = 2$ with $T = 10,000$. Make an ACF plot. Is this consistent with the model that you generated? Try generating a model with the same parameters but only 200 observations. Make an ACF plot. What do you notice about the autocorrelations and the dotted blue lines when comparing both models? (If you have a difficult time seeing what's going on, repeat a few times and see if you notice a consistent pattern going on.)

3. (No R required). Let $\{w_t\}$ be iid $N(0,1)$. Show the following $\{x_t\}$ processes are weakly stationary. Be sure to derive the mean and autocovariance (or autocorrelation) functions as well.

   (a) $x_t = w_t - w_{t-3}$.

   (b) $x_t = w_t w_{t-2}$.

4. (No R required.) Show that the $MA(3)$ model $x_t = w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \theta_3 w_{t-3}$ is weakly stationary. You need to show that the mean is zero and the covariance function depends only on distance. This will be very similar to what was done in class for the $MA(2)$ model.

5. Load the data in "dailyibm.dat" using the command `ibm<-scan("dailyibm.dat", skip=1)`. This series is the daily closing price of IBM stock from Jan 1, 1980 to Oct 8, 1992.

   (a) Make a plot of the data and an ACF plot of the data. Does the time series appear to be stationary? Explain. Interpret the ACF plot in this situation.

   (b) Difference the data. Plot this differenced data, and make an ACF plot. What is your opinion of whether the series is stationary after differencing?

   (c) Another option for attempting to obtain stationary data when there is something similar to an exponential trend is to take the logarithm. Use the R command `log()` to take the logarithm of the data. Plot this transformed data. Does the transformed data appear stationary? Explain.

   (d) Perhaps some combination of differencing and the logarithmic transform will give us stationary data. Why would `log(diff(ibm))` not be a very good idea? Try the opposite, difference the log transformed data `difflogibm<-diff(log(ibm))`. Except for a few extreme outliers, does this transformation succeed in creating stationary data?

   (e) Delete the extreme outliers using the following command:

   ```
   difflogibm<-difflogibm[difflogibm> -0.1]
   ```

   Plot this data and the ACF for this data. Sometimes with very long time series like this one, portions of the series exhibit different behavior than other portions. Break the series into two parts using the following commands:

   ```
   difflogibm1<-difflogibm[1:500]
   difflogibm2<-difflogibm[501:length(difflogibm)]
   ```

   Plot both of these and create ACF plots of each. Do you notice a difference between these two sections of the larger time series?

   (f) Assume the model for the data that we have called `difflogibm2` is of the following form:

   $$d_t = \delta + w_t$$

where $w_t, t = 1, ..., T$ is Gaussian white noise with variance σ_w^2. Is this reasonable from what you now know of this time series? How would you estimate δ and σ_w? What are the numerical values of these estimates?