Time Series Modelling and Forecasting with Applications in R

Prof. Sudeep Bapat

Shailesh J. Mehta School of Management

Indian Institute of Technology Bombay

Week 06

Lecture 30: Practical Session in R-6

Hello all, welcome to this course on time series modeling and forecasting using R. Now, as you see in front of you, we have a practical session today, and the focus of this session would be to work out the details using real data or perhaps simulated data and try to connect that with the theory we have learned so far this week. Now, again, if you are watching this video after a long time, just quickly review what we have covered this week so far. So, the main focus this week was a different kind of time series model called the ARFIMA process. Right.

As we discussed earlier, ARFIMA stands for autoregressive fractionally integrated moving average. Right. Now, first, you should understand the difference between an ARFIMA process compared to ARIMA or SARIMA. When we studied ARIMA, ARIMA stands for autoregressive integrated moving average. In ARIMA, the differencing operator or the differencing order, which is D, must be an integer. We difference the series once, twice, thrice, and so on, where d stands for the number of times we difference the series.

When we transition to ARFIMA processes, the differencing operator can also take decimal values. As seen in previous lectures this week, the value of d indicates how persistent a time series is. The whole idea behind discussing ARFIMA processes is to capture the persistence in the underlying time series. And what exactly is persistence? Persistence is whether a time series possesses long-memory properties, right? For example, if you analyze the stock price of Reliance or TCS or any company, a random shock can cause a sharp movement in the stock price, right?

Now, does that movement last for a while, or does the series revert back to the mean very quickly? So, if that sort of effect lasts for a while in the future as well, then we say that

the underlying time series is persistent. Or, exactly opposite to that, if the series or if the stock price sort of reverts back to its mean, then we say that it has some anti-persistent nature. Alright, so before we start with the actual coding, you require—as always—some packages, and then you want to import some libraries. And if you are following our

practical sessions, which we take every week in R, then you must remember that you obviously have to install all these packages if not done already—forecast, fGarch, tseries, and urca, okay? Now, by the way, these packages would be required to sort of manipulate or implement the underlying functions that will soon run in this particular code, right? And I think, as of now, you must be familiar with forecast and tseries, right? So, forecast and tseries are kind of predominant packages that one has to apply or import when you're trying to analyze any practical time series data, okay? So, again, let's do that.

Now, again, just a very quick refresher. If you sort of forgot how to install the packages, again, one can go to tools here and then click on install packages. And then a couple of things to sort of take care of here—you want to make sure that this This thing says repository. So, if not, then you want to select repository.

Because the repository stands for searching online from its online repository. Provided that you have a valid internet connection, it will download the package and then install it. Okay. Then, whatever package name you want, you can input it here. For example, forecast, frag div, or T-series, etc.

Then simply click on install. Once you install, don't forget to load the packages into the R environment using the library command. So we'll do exactly that now. So let's say, library forecast, then library frag div. Again, notice that since this is a console, once you type library for any package, R runs a simple command to load the package into the R environment.

Okay. Then the second one is FRAGDIF. FRAGDIF stands for fractional differencing because the idea behind ARFIMA is that the differencing order D can also be a fraction. Hence, we call it a fractional differencing order or fractional differencing parameter. So we require this FRAGDIF library and then T-series.

So, T-series stands for time series, and then URCA is for some of the other functions that are there in this code. Alright, so once you make sure that all the libraries have been installed, the next thing is you want to run this. So, set seed 1, 2, 3. So, set.seed is a function that tells you that every time you try to replicate this code or reiterate the code, it

should give you some random values and not the exact values from the first iteration, right? So, this just tells R that every time you simulate the same code again and again, it has to randomize and start the entire process again, okay?

And the number you input here really does not matter. So, this is just a random number. So, 1, 2, 3—we are just putting some random number inside this set.seed. And here, you can also see that for reproducibility, if you want to reproduce the code again and again, you have to ensure that every time it should be a completely random environment. Alright, so set.seed 1, 2, 3, and then the next thing we will do is fix some orders now.

So, the first thing is we will fix some n. So, n in this case is 1000, which is the number of observations, right? And then the value of D. So, D we are taking as 0.3 here, right? And then D stands for nothing but the fractional differencing parameter that you have in the ARFIMA structure, okay? So, D is 0.3. Now, again, you can play around with this code, right?

So, since you have access to all the codes, a very strong suggestion is to not just run the same code that we are discussing in class, but also to play around with it. So, let us say, change this value. So, let us say 0.6 or 0.7, something like that. Or you can even take a negative value, right? If you remember, one of the lectures this week was predominantly based on what values of D give you, let us say, long memory property or short memory property, stationarity, non-stationarity, persistence, or anti-persistence, right?

So, by playing around with the value of D, you can actually pinpoint the particular case that we studied in the theoretical section this week, alright? Okay, so now developing further, since our FEMA process requires the AR components and the MA components as well, we can actually fix some AR coefficients. So, the AR coefficients in our case would be 0.5, and this is nothing but an AR1 structure, right? So, you can run this and then the MA coefficients. So, the MA coefficients we are keeping as 0.2.

Now again, you can play around with all these values. So, probably change this 0.5 to, let us say, 0.9 and then see what different result you get, right? So, this is a very basic idea if you are really starting off with some hands-on exercise, right? I mean, whatever code you have been given, Do not just blindly run it and then try to emulate the results.

I mean, try to play around with some of the values, right? So, for example, what would happen if you changed this 1000 to, let's say, 5000? What would happen if you changed

this 0.3 to, let's say, 0.7? Because then 0.7 would exceed 0.5, right? And then it might happen that you have an entirely different sort of ARFIMA fit.

Okay. Or what might happen if you changed the coefficients here? So, let's say from 0.5, you take a very, very small coefficient. Let's say 0.001. Right.

So, all these things could be tried out. So, due to time constraints, we can only focus on one particular case, right? But then it is your task to emulate the same results by mixing and matching the parameter values. So, I hope this point is clear. And again, this is not just for this particular code, but also for all the previous codes and the codes to follow.

Okay, so once we fix all these values, the next thing is to simulate a particular ARFIMA model, right? So, we will name it ARFIMA_Sim, and in R, once you import some of the libraries we saw earlier, this is an inbuilt function. So, fracdiff.sim. Now, again, as discussed earlier, if you simply hover over a particular function, right? It tells you the entire syntax.

So, can you see that? So, what exactly goes inside the fracdiff.sim function, right? And then you have to specify the AR component, MA component, etc. So, you have a lot of things, right? So, here we will specify N, which is the sample size, then D equals D. So, D is the fractional differencing parameter in our case, and then AR would be the AR coefficients, and MA would be the MA coefficients, right.

So, needless to say, even though you see a lot of different things inside that fracdiff.sim command, we have to keep it simple, right. So, what are the ingredients required to model the ARFIMA process? These are the specific ones. So, N, D, AR coefficients, and then MA coefficients, okay. So, now, we will run this line, okay.

So, again, it says that object MA coefficients were not found. So, probably, we should go back here again. Now, again, to make sure that if you get some error like that. So, again, make sure that that particular line has been run, okay. Alright.

So, now, the MA coefficients have been run. Now, again, if you go back to ARFIMA _sim, it should work now, okay. So, this is again, sort of, what do you call it? So, looking at the errors in the console and then working accordingly. So, once you fix or once you model or fit this ARFIMA process with these parameters, then we can actually see a plot of that.

So, a plot of the fitted ARFIMA series, and this is exactly how the plot looks. Now, again, let me zoom in for you. So, this is a zoomed plot. So, this is nothing but a simulated RFIMA process with those parameters. So, D is 0.3, then AR coefficient, MA coefficient, etc.

Now, again, if you repeat the same exercise a second time, you should get a completely different picture. I mean, it should be similar, but all the values will be different because this is a simulated sample. So, every time you run the code, you should actually hope that in the next iteration, you will get a different sort of sample. Alright? So, I hope this makes sense.

Now, let me close this and come back to the code again. Now, what we want is to estimate the ARFIMA model. Right? We want to estimate the ARFIMA model. So, again, you have a command called ARFIMA.

So, we will apply this ARFIMA command on the fitted ARFIMA series. So, ARFIMA applied on ARFIMA underscore sim dollar series. So, let me run this now, and then we will give it a name as ARFIMA underscore model. So, this structure here—ARFIMA underscore model—contains the fitted or the estimated ARFIMA model. Now, how do you extract, let us say, the parameter estimates or something like that?

I mean once you are estimating anything or once you are estimating a model, the first thing is that you want to extract the estimated parameters, is not it? So, for that you can actually use this summary command. So, summary applied on the fitted model. So, RFEMA underscore model and you can actually see all the fitted coefficients in the console. So, this is what the output is.

So, you are trying to fit an Arfima structure, then what are the estimates of let us say D, then the AR coefficient, MA coefficient, etc. And further it also tells you let us say log likelihood, then AIC, etc. Make sense? So this is more like a fitting exercise. So given a simulated data coming from an Arfima process, how do you fit or how do you estimate the underlying parameters of the Arfima model?

Now here one important thing to note here is that the estimate of D, if you look closely, is 0.2824. Which is very very close to the value we are fixing already. So if you remember what was d that we fixed. So the value of d was 0.3. Isn't it?

So this 0.2824 is close to the assumed value. The assumed value, which makes sense, which makes sense, alright. Okay, so now that we have fitted a particular ARFIMA

model, we have to check and then proceed with the diagnostic checking. So, once you fit any time series model—be it AR, MA, ARIMA, ARFIMA, or whatever—the next step is to ensure that all the model's assumptions are met, particularly regarding the residuals. So, what exactly are these assumptions?

We covered this in depth in earlier weeks, right? If you recall, the whole idea of diagnostic checking revolves around this. So, normality of the errors, independence of the errors, and constant variance of the residuals. These are the underlying assumptions we need to verify now, alright. So, the command you see in line 27, This is PAR and then MF row equals 1 comma 2.

If you want to plot multiple plots in the same window, you need to use such a command. This tells R that I want both the ACF and PACF plots in the same window—or rather, in the same row—because this is 1 comma 2. So, a single row and two columns. You will essentially get plots side by side. So, we will run this, and then we will run the ACF and PACF of the residuals.

Now again, let me zoom the plots for you. So, these are the ACF. So, ACF on the left-hand side and the PACF on the right-hand side. Now, again, interestingly, you do not see any correlations that are highly significant. I mean, of course, apart from this right here, right.

But apart from that, all the other correlations are in between the bounds, which means that they are not significant. Which also means that there is no autocorrelation among the residuals. And then again, the other part to note is that you do not see any patterns among the spikes also. So, this is one very important way of checking if the assumptions are being met or not by simply plotting the ACF and the PACF of the residuals. Okay, and then the next thing is you want to check for stationarity of the residuals using the KPSS test.

So, again, if you remember the KPSS test, we have covered this long back, but then one can actually implement this KPSS test to check, in fact, whether our residuals are stationary or not, because we do not want any non-stationarity among the residuals, right, because residuals should be completely random. So, they should be, in other words, they should be stationary. So, how do you check that? We can actually implement this KPSS test on the residuals and then let us see what the output is. Now, again here it says that it could not find the function UR dot KPSS.

Now, again, let me go back and try running this URCA one more time. And again, come down to the KPSS test here, and now it has run. And why is that? Because this command UR dot KPSS is there in the URCA package. So, if for some reason your URCA package did not load correctly the first time, now again go back and try installing the package and loading it in the R environment once more.

Make sense? So, now this has passed, and here is the summary of the KPSS test. So, this is exactly the KPSS test, and here it gives you the value of the test statistic. So, 0.0461, and all these are the critical values. So, 0.347, 0.463, 0.574, and then 0.739.

So, here, if you see, this 0.0461 is less than any of the critical values that you see here. Which means that the value of the test statistic is less than the critical value. So, you fail to reject the null hypothesis. If the value of the test statistic exceeds the critical value, then you would reject the null hypothesis. But if the value of the test statistic is less than any of the critical values here at whatever percentage points you take, you fail to reject the null.

And the null hypothesis in KPSS is that the underlying residual series or whatever series you are testing for is indeed stationary. So here, the conclusion is since you are failing to reject the null, you actually go with the fact that the residuals are indeed stationary. Okay, which makes sense also. Okay, so once you sort of do these preliminary diagnostic checks, then you can actually go ahead and do the forecasting. So, the first thing is you specify some forecast horizon.

So, let us say I want the forecast 20 time points down the line in the future. So, the forecast horizon is 20 in my case; let me run that. And then again, you sort of bring this command forecast from the forecast package. You apply it on which model? You apply it on the ARFIMA fitted model. And then here, you specify the forecast horizon.

That's it. Right. And then we'll give it a name as ARFIMA underscore forecast. Makes sense. So, running forecasting is not a tedious task.

So just a couple of lines. Right. Now the whole idea is how you visually check if the forecasts make sense, so you can actually plot the forecasts. So these are exactly what the forecasts look like. So again, if you notice here, something strange is happening.

Even if you run this single plot command, again R thinks that there will be one more plot here we should follow because remember that we applied this par command earlier and specified that we want two plots in a single row. But now we do not want that, right? I

mean, now we do not want it. So, what one can do is change this two to, let us say, one again, alright? So, I want a single plot in a single row and a single column.

So, run this. And then again, if you come down and run the Arfima forecast plot. Now it should give you the complete plot in a single window. Does it make sense now? Hopefully.

So, let me zoom in now. Alright. So, this is the ARFIMA fit or the fitted model. And then, the line in blue that you see. So, let me zoom in again.

So, the line in blue that you see towards the end is nothing but the forecast. Now, a couple of conclusions one can make is that it is at least capturing the trend aspect. I mean, you are not seeing a horizontal line for sure. So, it has a tendency of capturing the trend aspect, of course. So, in a way, it might make sense.

Alright, and then now the next thing is, if you want to save the results to some CSV file, let us say, right, so we can do it. So, let us say results, and then dataframe, and then you mention the timestamps. So, 1 to whatever length of the series you have, and then the value that has to be outputted as a second column in that CSV file would be nothing but the actual ARFIMA fitted series. Alright. So, probably I will not run this because I don't want a CSV file to be created on my desktop.

But if you want a separate CSV file where you have two columns, the first column tells you the timestamps, and the second column tells you the actual values of the fitted ARFIMA model. Then you can actually run this results and then write dot CSV command. So, write dot CSV command, then inside you input the results comma, you have to give it a name. So, I will give it a name as ARFIMA underscore simulated underscore data dot CSV, right, and then row names equal to false, right. So, this is the syntax, okay.

Now, let us say, what if you want to simply print out the fitted coefficient? So, if you are not interested in any of the other stuff that the summary command gave us earlier, if you are only interested in seeing the fitted coefficients of the parameters, then you can actually do something like this. So, you can give it a name, let us say estimated ARFIMA model coefficients, and then you want to simply print out the coefficient part from the ARFIMA model. So, this is exactly what you will see. So, this is what your D is, the estimated D is, right?

This is the estimated AR coefficient, and this is the estimated MA coefficient. Make sense so far? Okay, so now the last thing we will do in this code is to sort of implement the ARFIMA process on some real data because otherwise, how would you figure out if the ARFIMA process is indeed making sense or not, right? And for that matter, we will again use the air passengers data that we discussed earlier. So, we have been again using the same air passengers data again and again.

Just to sort of keep some consistency. So if I bring in some random data every time, then there would not be any connection as to what is going on in that particular session as compared to what we studied in some earlier practical session and so on. So if you keep a practical data which is consistent throughout, let's say, explaining all sorts of models. Then at least you will try to find out that what model is better to fit that or what model might not be very good model to fit that and so on. Make sense?

So again air passengers data I think probably almost all of you should have some idea about what this data means. Right? Okay. But again, if not, then as discussed in the last code also, you can always type in, let's say question mark and then air passengers. Okay.

Or probably we'll do it now also. So let's say, so you load in the data first, right? So data air passengers, then you give it a name. So let's say Y is the data, which is air passengers. Okay.

And then again, like I said, if you want more information about any function or any underlying data set, then you have to actually do question mark and then type in the name of the data. So air passengers and then hit run. And over here, you see the description of the data set. So, do you remember this data? So, we have been using this data multiple times already.

So, this is the monthly airline passenger numbers from 1949 to 1960 and then sort of gives you a small description and then the source of the data. So, this data set is from Box and Jenkins airline data and then again between 1949 to 1960. And then this is a monthly time series in thousands, right? And this is exactly the source. So, it sort of dates back to 1976 from this book by Box, Jenkins and Rinzel.

So, time series analysis, forecasting and control, the third edition. Okay. And then down the line, it sort of gives you some examples that you can sort of implement using this dataset. So, here let us say Arima model. So, if you want to fit some Arima model, right.

And then, this is sort of an example that tells you some hands-on ways to start coding using this air passengers data. Make sense? Now again, we will use this data here for our purpose. Now, the very first thing to do is to plot the data. So, let us say you have a simple plot of the data, and then this is exactly how the plot looks.

Now again, remember that this is a very classical dataset people use for explanation purposes because it contains a trend, seasonality, and changing variance. So, all sorts of non-stationary behaviors can be seen in this air passengers data. Make sense? Okay, now the very first thing is we want to check if the dataset is stationary or not. So, we again apply the KPSS test to check for stationarity, and then this is the summary.

And again here, surprisingly, if you see the value of the test statistic is 2.739, which is much higher than any of the critical values you see here. So, the maximum critical value is 0.739. So, this 2.739 is way above any of the critical values. So, what do you do? So, you reject the null, right?

And then you go with the alternative, and then the alternative hypothesis under the KPSS test is that the series is not stationary, right? So, the null hypothesis is that the series is stationary, and then the alternative is that the series is not stationary. So, if you are able to reject the null, then it sort of means that the underlying series is not stationary to start with, all right? Okay, then how do you handle it? So, the next thing is you sort of difference the series.

So, if something is not stationary, the very first thing you should do is apply some differencing. So, we can difference the series once, right? And here again, you can check for the KPSS test, right? So, you can again apply the KPSS test on the differenced series now to check if the differenced series is stationary or not. Or, as we have done earlier, we can actually go for creating the ACF plots and the PACF plots of the actual differenced series.

So, ACF of the differenced Y and then PACF of the differenced Y. Now again, here I want two plots side by side. So, I have to specify using a PAR command. So, first is ACF, second is PACF. Let me zoom in both the plots for you. And these are the ACF structures and the PACF structures.

Now, here clearly you can see that even after the first difference, the series is not stationary, right? Because you see some repetitions here. Can you see that? So, you have lots and lots of correlations that are outside the bounds, which means that you have some

autocorrelation among the lags as well, right? And then there is some seasonality also, right?

So, because you know that from the initial plot of the series, we can spot the seasonality, and also from these two plots, we can actually plot or spot the seasonality, okay. So, can we go ahead and try to model this in an ARFIMA model? So, this is the question. So, we can try. So, let us say ARFIMA underscore model would be ARFIMA applied on Y. Now, one thing to note here is that we are applying ARFIMA not on the differenced series, but on the actual series.

Which is y, is it? Now, the summary of the ARFIMA model. So, this is the summary of the ARFIMA model. So, what is your d? Then what is the AR coefficient, right?

So, here if you notice, R is estimating an ARFIMA model fitted on the actual air passenger data, and then it indicates that the value of d should be 0.47. There should be two orders in the AR part. So, AR1, AR2, and then there should be a single order in the MA part, and then these are the coefficients. Again, it gives you the AIC value, log-likelihood value, etc. And now, diagnostic checking.

So, we repeat all the steps that we have done with the simulated data. So, let us say ACF of the residuals and PACF of the residuals. Now again, let me zoom in. So, these are the ACFs, and these are the PACFs. Now again, clearly, you can see there are some repetitions, right?

So, can you see the repetitions—or not the repetitions—but can you see significant spikes at lags 12, 24, probably 36, etc.? So, these are very strong indications that there is seasonality present, and you have a monthly seasonality, right. So, whenever you have a monthly seasonality, such tendencies could be seen in the ACFs and the PACFs, all right. And then, the KPSS test is applied to the residuals. Now, we will see what happens, right.

Now, here the value of the test statistic is 0.72, and surprisingly, 0.72 is bigger than this 2.5 percentage point, right. Even though it is slightly less than 1 percentage point, it is still bigger than this. So, we can actually reject the null and conclude that the series is indeed non-stationary to start with. But nevertheless, since we are fitting an ARFIMA model, we can actually go ahead and try to forecast that. So, the forecast horizon would be 12, which means for the entire next year or 12 months.

And then you implement the forecast command on the fitted model and plot the forecast. Let us say if you want a single plot comprising or spanning the entire window, then you

again go back here and try to change this to 1, 1. Now, come back here again and try to plot this one more time. So, this is the forecast plot. And here, you can clearly see that the forecasts are not making much sense.

Because you would want to preserve the seasonality aspect, but you are seeing that you are getting almost a straight line. So, in a way, the ARFIMA model is not performing very well. So, this is the conclusion, but again, this is just to show you how we implement a fitted ARFIMA model on some real data. So, if you think that there is some long-term or long memory underlying in the series you have, or there is some persistence in the underlying series, then you should go with fitting the ARFIMA model, and hopefully, the forecast should make sense. But for this dataset with the air passengers, it may happen that the tendency of monthly air travel is not persistent.

Because again, clearly, you can see that even if you have a trend, you also have seasonality. So, whenever you have a spike, it sort of gets back to its mean very quickly due to seasonality. So, you do not have any persistence in the time series, but this was just an example to show you how one can fit an ARFIMA model on simulated data and real data.

Thank you.