Advanced Financial Instruments for Sustainable Business and Decentralized Markets

Prof. Abhinava Tripathi

Department of Management Sciences

Indian Institute of Technology, Kanpur

Week 10

Lesson 30

In this lesson, we will use equal-weighted portfolio objects from UA futures, S&P 500, ESG fund, and Bitcoin data to perform return and volatility modeling. First, we will perform ARMA modeling and then find the model across several competing models that offers best out of sample prediction accuracy. Then we will also model volatility using GARCH models and find the most suitable model using AIC and BIC information criteria. In this video, we will do a quick recap and show how to construct our variable portfolio equal weighted portfolio variable for analysis. To begin with, recall in the previous lesson, we had four data series and we had saved this object by the name of final data. Let me show you how to read this.

So I need to click on this final data object. If I click on OK, a command line will be run. Data is already read, but for record, I'll just keep this command line. In future, if I may forget the location and all, I can simply just press control enter and this data will be read.

In this data, let me show you what is inside this data. This carries four data points. One is time series of Bitcoin, S&P 500, ESG fund and carbon market that is in way. Now from this data, I'll first construct the return series. Let's see how.

So first I'll create a zoo object. Zoo is a time series object which requires application of zoo libraries. Now I'll type this zoo command final data and out of this final data, the last four data which is Bitcoin, S&P, ESG and these are the objects of my use. So I'm only selecting column number two to five. Column number one is date and now I'm defining because this is a zoo object.

It needs a data date observation to construct the time series. So I'm giving the date object here and this is now my zoo object. If I print this zoo object, let me show you. This is a very interesting object. So if I create this zoo object, you would notice there is no

separate column for time variable.

The time is stamped, marked as a stamp on each set of observation. Like for example, here you can see 2013, April 2013 is marked 29th marked for all these four observations. So now I have my zoo object. Let us create the return. So let me call it as final rate and I'll take a log and then take the difference.

So very useful command difflog. So this command first takes the log of the object inside which is final zoo which has four price series. First it will take the log, natural log here and then difference. So it becomes like natural log of PT upon PT minus one price at time T divided by price at time T minus one which is a property of log. That log of PT upon PT minus one is same as natural log of PT minus natural log of PT minus one.

So this is my return object. Now in this return object, I will construct my equal weighted portfolio by simply taking row means. For example, I'll run this final return dollar equal weight portfolio. So I'm creating this equal weight portfolio which is nothing but row means. So I'm taking these means of these corresponding rows.

Let me show you what is inside my final return object. I haven't shown that yet. So if you take the head, these are the return object. So notice these are all return observations for each row constructed from only the price variable. So now I can take row means and inside this row means I can type this.

So essentially what will happen is all these returns, their mean will be taken which is essentially equal weighted portfolio object is created. I'm assigning equal weights to them. So this is my portfolio object. I can separately sort of save this equal weighted portfolio object as final return dollar. So now I'm extracting that equal weighted portfolio object from my original data frame final rate.

In this process, some NA observations not available observations may have been created. For example, when you construct return, the first observation is eliminated. To avoid such issues in one go, just using NA.omit, this will help me removing all those NA observations. And now we have created our equal portfolio object.

So to summarize this video, we have first read the data which contained all the four time series prices. Then in this data, we constructed a time series object, created a time series object out of the price data. Then we constructed the return from this data and took the equal weighted averages of these return which led to a series of returns which is sort of equal weighted portfolio. Then we cleaned the data and now we are good to start with the operations. In this video, we start the exercise by setting the working directory, load the

relevant packages and read the data.

So let us first set the working directory by clicking on session tab here, set working directory, choose directory. Now I have already set my working directory where all my data and required files are there. So I will just click on open and a command will appear on my console window here. For future reference, I can copy paste this command here so that even if I may forget the exact location, I can see this and just run control enter or by running this command or by pressing run either way I can press control enter on this command and then all the relevant files will appear on my file tab here. Next we will load the relevant packages.

So in this particular lesson, we are going to use library forecast. Then we are going to use library through Garch for Garch and all. Then we are going to use XPS. Lot of data we are going to convert into extensible time series. We are also planning to use tidyverse for some of its applications.

We will use libraries do and because we are going to measure some errors, RMSE, MSE and so on, we are also planning to use matrix. So this is we have loaded all the relevant packages for the current working session. So now we are ready to read the data and you notice as soon as I run my command on the script here, I am seeing the same execution on my console window as well. So let us now read the data. I have the data file at my equal weighted underscore portfolio variable.

Let's use this read.DS and by the same name, I have the same file. This file you can see here, this file is there. I could have directly click on it. Okay.

And the file would be read. That is one way to read it. I can just paste that command for future reference. For example, in future, I don't want to exactly remember the name or anything. I can just press control enter and the file will be run. So this is how I read the data.

Let me examine the data. So if I check the class of this equal weight portfolio, the class of this data is you. So it's a time series object. I can also look at the head of this few initial observations. So these are my time series of portfolio observation, which is my equal weighted portfolio that I have created from my cryptocurrency, EAG fund, S&P 500 and carbon market data. So this is equal weighted portfolio of those data points, which I have created in the previous lesson, the equal weighted portfolio.

Let me also show you the plot of this equal weighted. So we get a good idea how it appears. So if I plot this, I can run this plot command and so if I run this plot command,

you can see the plot of this equal weighted portfolio. It's the some shock somewhere around 2020 due to COVID and it's a fluctuating series. It's relatively less fluctuating because it's a portfolio.

So to summarize in this video, first we had set the working directory for our current session. So these are the relevant packages. These packages were already installed. So I need not install them again. I have simply added them in the current working environment only.

Then we read the data, which is equal weighted portfolio. This data we created in the previous session. This essentially is a combination of four series, which is one carbon market data, then cryptocurrency data, EAG fund data and a broad market index like S&P 500 and I have taken their equal weighted that is assigned them 25%, 25% gain in the previous lesson we constructed this data and now we are ready to model with the Arma Arima kind of models. In this video, we'll start with our Arma Arima modeling process. So we'll start with the Arma Arima modeling.

First we'll convert our time series object into extensible time series by using this as .xts command. This has very nice and desirable properties and that's why we installed this XTS library. If I show you the class now, see the class that has changed. If you see the class of this new variable that is XTS Zoo extensible time series.

The plot will also be very nicely appearing. If I show you the plot, the new plot of this data, you can see very nice looking plot with its time. Let me zoom it for you. So very nice plot with all its properties is there. You can see some of the shocks in the series.

Now in Arma Arima modeling, if you recall our discussion about critical concepts, the first and foremost important part is construction of ACF-ACF plot that is autocorrelation function and partial autocorrelation function. This gives us the property or highlights the first visual examination about their Arma Arima properties. So first let me show you the ACF plot here. ACF plot of ACF plot very nicely highlights. You can see the first bar 0 which is ideally 1.

The next bars are smaller. We show that there is some minor autocorrelation though it is significant because it is outside the dotted blue lines. So there are few of them that are significant indicating there is some kind of AR structure. We will have more details to see about that but there is some AR structure that is visible here. Similarly, we can see if there is PACF.

PACF highlights that very clearly because it. So now this PACF structure has multiple

lines which shows that not only there is AR there is some kind of MA structure also happening. So let us look at this AR, ACF and PACF structure in conjunction. So we can see there is a sort of declining kind of structure in the PACF diagram and if you look at PACF there are few bars. So there is it seems to be the presence of both AR and MA because if there is AR then that AR structure is nicely captured by PACF plot in the form of declining this kind of declining bars. As you can see here similarly if there is MA you can see in the ACF plot this kind of declining structure.

So this is somewhat small MA but predominant you can see. So you can see there is some kind of some AR. If there is strong AR then you would have strongly large bars across the legs but some AR but at the same time if you look at PACF that confirms the presence of MA structure. So we have some intuition that there is MA AR structure taking place. Next we will try to see the box plot, the loom box, loom box and box pierce plot for test of autocorrelation.

So we will test with loom box and box pierce test to see if there is some kind of autocorrelation structure. So first is box dot test and I will run this equal weight portfolio object. So let us say I have some intuition that there is some like structure so I am putting a lag of 4 you can change the lag size. My hunch is because this is these are these markets run at least a week so even if there is some information structure that should be captured in a weekly interval.

So because of that I am putting as 4 day lag. If I run this you can see it is significant so that means there is some autocorrelation in the process. Autocorrelation structure is indeed present. I will make another text which is loom box text, loom box test in this same lag 4. So even if there is 4 lag obviously the 1 lag also would have been significant so let me show you. See it is also significant so 1 lag structure is also significant whether you use 1 or 4 it is turning out to be significant.

So there is indeed some kind of ARMA ARIMA structure. In the next video we will run auto dot ARIMA command and simulate that structure. In this video to summarize we saw through ACF PACF diagram that there is some visual intuition that we have some kind of ARMA structure there. There is some particularly from PACF diagram we found that there is some YA structure and from ACF diagram we found there was some ME structure presence of some significant AME though the sizes are small. Based on this we also tested them for autocorrelation of through box test box, pierce and loom box. We have found there was some evidence of significant autocorrelated lags.

In this video we will start with the examination of data and then segregate the data into training and test data. First with the help of auto dot ARIMA command which is an

inbuilt command it is available in the R package auto dot ARIMA it is a very useful command. So if you run this command you will see this is available in the forecast package it helps simulate or find the inherent structure in the process. So if I run this auto dot ARIMA to understand this information structure.

So if I run this auto dot ARIMA command. In this auto dot ARIMA command if I run this equal portfolio you can see it tells me that there is some kind of MA 5, MA 3, AR 5 autoregressive 5 autoregressive terms and 3 MA terms are there in the process. So that gives me some idea that it is AR 5 and MA 3 that is ARMA 5, 3 process. So if that is the case then while simulating our process we will take care of that. Now let us first break the data into test and training data set this is very important. So you first train the model on the training data set and then you see whether it has good out of sample prediction for testing data set.

So first we will check the accuracy of model in sample forecast by using training data set. So let us see let us create the training data set for that I am segregating this equal portfolio object into two segments one is starting from 2017, January 2020, December 2020. This is my training data, this is my training data from 3 year data, it is 3 year data training data. So I will train my data on this object, equal to portfolio training, so this is my training object. Similarly, I can also construct my testing object from there whenever I have to test or test the model for out of sample accuracy.

So this is in the training data field test it may fit very well which is the problem of overfitting. So I need to test my data outside that or through a new data or new sample on which it was not trained, so it is not aware about that data and therefore its power or accuracy will be tested. So I am using for 2021 and 2022, 2 years for sorry I should use testing data test. So first is my training and this is my testing data. So now I have my training data set available with me and I need to do the forecasting and testing test them.

So first I will simulate the model, simulate the model using this training data. So I will start with a very simple kind of ARMA model, let us call it ARMA, M1, M1 model and this in this ARMA model, I am giving a very simple kind of specification, we will increase the complexity. First I am giving it order c equal to c, 1 0 1. So on this, in this model we are only using 1 lag of AR which is this one and 1 lag of MA, so this is a very simple model. If I print it for you, you can see ARMA, it is a simulated model given this data, so we simulated this, there you can see the output.

Similarly you can simulate M2, how do we simulate M2? This is let us by assuming 2 AR and 2 MA terms, so this is M2, I can print M2 for you, let us print M2 now, so you can see M2. So this is 2 AR and 2 MA terms, earlier we started with MA1, then I can also do

with M3, where we will use 3, now the model will become more complex, so this is M3, let us simulate M3. Similarly, so my plan is to simulate 4 such models, so this M4 will have 4 AR term and 4 MA term and I can just, you can see this and then 5th model, 5th model, this 5, 5 term, now this has become extremely complex. The 6th term we will use which we simulated from, so recall we simulated through auto dot ARIMA and in this auto dot ARIMA we found that there are 5 AR and 3 term if you recall, let me show you again auto dot ARIMA. So when we run that auto dot ARIMA, you would notice a 5 AR and 3 MA terms appear, so now the last model M6 would have, the last model M6 would have 5 AR and 3 MA term as we simulated in the auto dot ARIMA command.

Let us print it also. So we can see the object here, so this is the output, there are some NANs produced that is slightly, it is giving me slightly poor fit that usually happens when mathematically a model is very good fit but theoretically it is not justified, then this kind of situations may appear where mathematically it was giving me a good fit but when I fitted the 5, 3 model AR5 and MA3, it is giving me NANs for my standard error, so that means there is some challenge while fitting this kind of structure but we will go ahead. So to summarize in this video, we broken our data into test and training set, then we simulated 6 different models including for each lab of ARMA 1, 2, 3, 4, 5 and lastly we also simulated fitted structure, mathematically fitted structure, so auto dot ARIMA command, we named it M6, so we have 5 model that we have trained and in the next video we will examine their out of sample accuracy. In this video, we will examine the in sample fitting accuracy of our model and for that we have already trained our 6 models, we will examine which one of them fit in sample data very well. Please recall AIC, BIC measures are very useful measures for such examination as they penalize addition of new variable unless until these new variables are giving us a very good information content or adding to the information. So I will create this output object to store these fitted AIC, BIC values, so I will it is a data frame object with 2 rows each for AIC, BIC and 6 columns for all the 6 models respectively.

Now I will give the names of these columns as M1, M2, these are my column names that I am giving M3, for each column there is one M4, then M5, then M6, so these are my column names similarly I can give my row names also I can give my row names which are AIC, BIC. So let me show you how the printed of print this object and show you how this object has is appearing on my console. So there are 6 columns each named M1, M2 and 6 rows each named AIC, BIC, 2 rows basically each for AIC, BIC and now I will start assigning the values in each of these row columns. So first the simple way to do this is pick a column let's say first column is M1 then I extract the AIC value from M1 model and also BIC value from M1 model. If I run this command the first column M1 column is assigned these AIC, BIC values.

In identical fashion I can do the same for M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15, M16, M17, M18, M19, M19, M20, M21, M22, M22, M23, M24, M25, M26, M26, so now I have my AIC, BIC values nicely assigned to this output object that I had created you can see here if I print this you can see these values are printed. Now I would not like to do to sort of manually compare them so what I will do is I will create a rank object which will rank these values so let me show you how it is done it's very simple this is a very interesting rank command which will help me choose the right set of objects for example if I do rank output based on AIC so this will give me the rank based on AIC so this gives the first rank to M5 notice this rank the way it works it is giving the highest rank to the lowest the one to the lowest one which is minus 5307 this is the lowest value and it is giving the rank first and this is the most complex of all it contains five sort of five AR and M8 terms and if I run BIC, BIC gives the rank 1 to M1 so that is interesting on AIC we are getting the first rank with M1 while which is the simplest model while AIC is giving the highest rank to M5.

Now this may be due to the fact that BIC penalizes addition of variable more heavily so the key difference between AIC, BIC in their formula is that BIC penalizes additional of new variables more heavily unless and till they contribute very significantly to information so this tells us that contrasting results we are getting A as per AIC M5 which is 5's ARMA M5 which is ARMA 5 model is good which is more complex while as per BIC we are getting that M1 which is ARMA 1, 1 model is best. So more testing is needed and to summarize in this video we created an object that stored our AIC, BIC values we examined AIC, BIC values for in sample fitness and we found very contrasting result while AIC showed that M1, M5 is the best and M6 is M1 is the worst which was the most simple while the most complex M5 was the best BIC showed that M1 was the best so we ascribe this to the fact that BIC penalizes addition of new variables more heavily unless and till they contribute to the model. However we must note that often in sample fitting is officiated with the out over fitting process that is a given model in which it is trying to fit the data often more complex models they fit over fit the given sample in which they are trained but their out of sample performance is not so good.

So in the next set of videos we will try to conduct out of sample fitting of the data and see which model performs well. In this video we will construct out of sample forecast. Why this is important is because often a model which fits well in a given sample has a poor out of sample forecasting accuracy and particularly the more complex model are prone to this problem. So we will conduct the out of sample forecast and then we will visualize them also.

So visual comparison will see how they are performing. So for this we will let's create this M1 underscore predict object and we will assign in this predict command. We will predict for 50 periods ahead. Now this is a critical choice what period or what length you

want to predict a very small period would result in very few observations where you cannot very effectively test the performance of your model. In contrast if you choose very large number of observations then after a certain point your forecast will become meaningless because no new information is added. So I feel that 50 steps ahead forecast is a reasonable compromise a reasonable trade off between synchronicity and accuracy.

So a period that is not too small neither too large and effective to judge the model. So then similarly for M2. For all the models we will predict next 50 steps. So we have these six models that we have employed for prediction M1, M2, M3, M4, M5, M6.

Let us start with the visualization. So I will create this equal weight portfolio predict object which will contain it's a zoo object it will contain all the predicted values and inside this I need to have a c-mine. So first is the actual set of observations what are these observations? So these are from test dataset and we are using the first 50 observations to compare. So first 50 observations then I have prediction from my M1 model M1 which is equal to M1 underscore predict and inside predict the object is stored with predict so here I have to use this predict object I store them then M2 in the M2 column I am giving the column name as M2 in M2 column the predicted values from the M2 model will be there.

So I am filling those values then M3 will have so I can do that I need not type the entire thing again and again so then we have M3. So this consistency name is very useful you don't have to write the full thing again you just copy paste and make use of your already written command then similarly you can have M5 and then M6 so these are this is the data frame now I this is my data frame object and now I need to provide the date index that will be used so I am going to use the index object from my equal weight portfolio test data and initial 10 dates as we are testing against the first 10 observations in the test data so this will be run let us run it.

So once I run it I can check the object produced it should be a column so let's say head of this let us run the head of this this should be a new object with 50 observations I can check the dimension of this object also the 50 observation with 7 columns first column is the actual observation and then remaining 6 columns are predicted. So now very easily let us plot this object so for that we will make use of plot command equal weight portfolio predicted object will give the central heading as visual comparison across models we want to make this heading a slightly higher in size so 6.main will give us 2 so that larger headings will result each line each plot should be of different color so this col object line width has 2 and type has b so both line and point type both point and line will appear lastly I am giving this pch very useful argument pch as 126 so each graph will have a separate symbol so the points will be of separate design or separate object for point let me show you what exactly are in there. So if I run this notice a very nice looking plot has appeared in each plot you can have model m1 m2 m3 and so on m5 m4 m6 their

predictions and the actual object and you can see all of them are different color and not only different color the point object the point that is there in side line is also different so it's a very nice plot and later on we will see which is more accurate here you can see m1 m2 being very simple after a few steps they converge to a straight line while m3 m4 m5 m6 which are more complex are giving some fluctuations have some fluctuations which gradually die away. So to summarize in this video we first constructed 50 out of sample forecast and then we visualize the predicted objects from different m1 to m6 model that we have simulated earlier we trained earlier and those predictions and we also compared those predictions that were made using these simulated and trained models.

In this video we will examine the out of sample forecasting efficiency of our trained models. For this we will first create an error object to store a number of errors you can have as many errors as possible we will take a certain number of errors that are most often employed error object to compare the models. So first let me create this error object which is a data frame kind of object it contains four rows each row for kind of error will also create a complex error object using our computed error so and then we have six column each column for a model first we will give the column names as we did earlier so the column names would include corresponding to each I need not type this I can simply borrow this from the previous. So this is my these are my columns each column represents one model similarly I can have error value so I am planning to use RMSE which is very important as a measure root mean square then MAE mean absolute error MAE you can examine their construction on net or you can go to this matrix package for example you can go to this matrix pack or further you can do this RMSE and you can see the construction of each it is part of matrix package so you can check the documentation here also you can have a look at the matrix package go to the package tab search here the matrix package and click in this web symbol and see whatever all the errors you will find there their explanation and detail computation so we are not doing that right now then you have RMSE for example this RMSE you can you can just press control enter here and see root mean square log error its construction and everything lastly we will also create a complex error object in such algorithmic systems you often have an error which is a combination of all the errors which is desirable to you and you see the performance of all the models on this particular error object so you examine all the models their performance on this complex error object so let us create this error object let us print it what do we have in this object so if I print it let me enlarge the console window you can see here it has 5 6 columns each corresponding to a model and each row corresponding to an error object so next step would be assign the error values computed from our prediction models and store here so we will start with the first model which is dollar m1 and first three errors we are computing in the fourth error we are taking a simple average as a simple system we are taking a complex error this complex error as a simple average of these to avoid any such complexity right now so first is RMSE in this RMSE we will

have we will compare the m1 predict object this is our predicted object and we will compare it with the actual value what is our actual value in that test data set so in our test data set the first 50 values we are comparing with the predicted first predicted 50 values similarly like RMSE I can have my other error object which is MAE so it makes the job simple the similarity of codes makes it easier to copy paste so the next error object is RMSE so we have three error objects we have created and this is for model m1 so let me run it let me see if it is error free so if I have run it correctly I can print my error object now let's see yes so first three errors are assigned this means I have run the command correctly similar fashion now before moving to the next model I need to also generate the last value what is the last value value number four which is nothing but the complex error or the mean of all the error objects which is mean of this object that I have created mean of this so if it is running error free let me show you the error object yet all the four values are filled so we are good so the same command will I need not read write again I'll just copy paste it for the other objects simply by replacing m1 from m2 and if I replace it the model similarly I'll again paste it and replace m1 by m3 it this makes coding very easy otherwise typing would have taken a lot of time here so I'm just copy pasting the same command only changing and if you maintain consistency with the symbols this is the great advantage that you have you don't have to write everything again you can just simply copy paste due to similarity and this aspect four with five now replacing this five then lastly six so we'll we have replaced now all six the next step is to run these commands let's run them again so starting with one I'll run them again so now my error object will be completely filled with all the three errors that is RMSE MAE and RMSE along with the complex error object let's see if it is done or they some error so all the error objects are filled so in this video we computed our error object to fill the performance or performance error of all the six models all the six models on RMSE MAE RMSE error and we also created one complex error object which is nothing but simple average of all the objects one can create a complex error object with much more complex function of the errors computed in the next video we'll compare the models their performances across these error objects in the previous video we have computed the error measures of our fitted object in this video we'll compare the errors and thus the performance of m12 m16 models so we'll create a rank object this will rank the performance on all the error objects that we have created so this object will have four rows corresponding to each error measure that is RMSE MAE RMSE and complex error and columns and call that is equal to six so six columns for each models one column for each model so let's have again this is the same scheme that we had earlier call names we'll have the rank object and we'll use the same call name so I'll just copy it from our previous code I have these six models and similarly I will have row names and each row corresponds to a error measure so I'll just use this previous command so now if I run this the rank object that I have created will have row columns where each column is the model and each row is the error now all I need to do is fill this rank object with ranks so

now we'll fill this rank objects with the ranks computed from the error object so we have error object which will use to rank these models and if I send this the rank object is signed so let's compare the rank values produced from the with the error value so let me show you so let us compare notice that on RMSE if you look at RMSE m5 performs best so on RMSE this 0.10195 is the best across all the models m5 is best on RMSE similarly if you look at m3 it is worst on all the errors m3 is worst on all the errors across all the models so that way we can see different different objects and particularly object m2 on our desired objective function which is complex error m2 works best so you can see here m2 0.0183 is better than all the objects here and therefore it is our model of choice m2 which is second order AR and second order me process to summarize in this video we compare the ranks of our error objects errors from different models and we found that different models work or rate performed differently in different error objects but on our desired complex object which was the average of all the three errors RMSE, ME, ER and SLE m2 works well we have intuition for m2 working well as it is a very sort of parsimonious object it has only two lakhs of AR and ME structure as compared to m3 m4 m5 and m6 which are more complex at the same time it is not as simplistic as m1 so it is slightly better than m1 and performs well on our complex error object in fact it performs well on ME object also in the previous video we discussed the modeling of return and out of sample prediction and forecasting in this video we'll talk about modeling the volatility and for that we'll employ the GARCH models GARCH modeling for volatility in particular we are going to use standard GGR and EGARCH that exponential GARCH and two kinds of assumption we'll assume normal and generalized error distribution GED.

So we'll do the GARCH modeling with these specifications for interpretation and theoretical underpinnings we can refer to the theoretical lectures that we did in the previous weeks so we would be needing for comparison of these models we'll be using again information criteria in particular we'll be using AIC, BIC, Shibata, HENQING and a complex information criteria object so let's create this information criteria object that we are going to make use of in this we'll have five rows each row representing one information criteria like AIC, BIC and so on and six columns one for each of the models let me explain what are these models for all the theoretical underpinnings we have already discussed in the previous week so in the interest of brevity I'll not repeat it again so we'll give them the names so first we have standard GARCH with normal distribution we'll assume the normal distribution then we'll have GJR GARCH with normal distribution then we'll have E exponential GARCH again a very powerful model with normal distribution assumption similarly we will change the distribution to generalize error distribution which is a more realistic assumption so the same set of models standard GARCH with GD generalized distribution GJR GARCH with generalized distribution and exponential GARCH with generalized distribution so these are the three others versions so let me also set up the row name so in this case the row names are according to

the information criteria that we are going to use which is AIC chi K Bayesian BIC then we have Shibata we'll use Hanquin HQ which is HQ IC or HQ Hanquin and a complex error of complex information object which simply is the same in this case we'll be using a simple average you can have any complexity that you want a more complex function of the remaining four criterias in this case I'll be using for simplicity the simple average now I'm going to plot my predicted volatility so I'm setting my plot window as M for equal to C T to this command we ensure that my plotting is done nicely so all the six volatility fitted volatility from all the six models are nicely fitted so let's start.

So first I'll start with the simple standard got first we'll model the normal distribution assumption three models are there first one is the standard GARCH so first one is the standard GARCH model this requires application of rule GARCH package that we added to our current working session earlier so all you need to do is type this GARCH spec so this will create your spec function and spec object you have GARCH spec GARCH spec object so this GARCH spec function will ensure that your mean model so our mean model is list and already we have found the right fit ARMA model so ARMA order equal to noted note that we already found that two comma two ARMA model fits well so we'll be using that then we also need to supply with the variance model variance model which is in this case we are going to use standard GARCH so list model equal to S GARCH standard GARCH model in this case the distributional assumption is normal so we are providing distribution as normal so later on we'll also be using generalized error distribution so we'll run this command to see if it is running error fee yes it does so our GARCH spec object is there you can also print your GARCH spec object so you can if you want you can print your GARCH spec object it gives you a lot of information for example it gives you the kind of model you have run and basic contours of the model next we'll fit our data in this specification and we'll create a new object called GARCH fit which will comprise of U GARCH fit we'll fit our data so you this U GARCH fit and data we already had which is equal portfolio test object equal weight portfolio test object.

So this is our test data which we are going to fit and then we'll use our spec as GARCH spec so this spec we have created it may take a few second to fit this object once the object is fitted you can print it also to see some what is inside this object for example it contains a lot of information for us starting from it gives us the parameters mu ar1a may1ma2 omega alpha all those parameters for interpretation you can go and check the lectures what are the interpretations of these estimates it is also there with robust standard errors so you can check the interpretation of these parameters and their implication for the model we have in these information criteria that we are going to extract there are some other tests for example lumbop text that we have already discussed about the autocorrelation structure arch lm takes for arch effects you can check whether arch effects are there particularly in this case seems that not many artifacts are there which may be the case that we are using a very diversified portfolio so probably some of the

arch effects are eliminated so all the proper for all the properties you can have a look at the theoretical discussion so we are not we are not going to discuss that the next step is to extract the fitted wall so let's extract the volatility conditional volatility that we have computed with the sigma command guard fit i can extract it for now you can print it also so this will be fitted volatilities.

If you want you can print this as well plot fit so here i can plot this object with plot fit i can give the central heading as main equal to standard norm and i'll give i'll change the colors so that plot is differentiable so you can have this fit for and it is plotted here so that way you can plot so what i'll do is i'll from from this command so that because we have six plot that are going to come up so should be there so this is one plot i'll plot them simultaneously the next step is to save our fitted object the aicbic criteria from one to four in our information criteria info criteria function and i can fit the guard fit object this will save my information criteria here let me show it to you let me print it so if i print it you can see here so now i have shown you how to create my fitted ais object with aic basic criteria extraction and we have also visualized the volatility here i'll create the remaining five models and show you do summarize the findings in the next video now in this video we'll construct the remaining five objects but before that if you wanted to forecast using this guard fitted model all you need to do is you need to you can create a guard forecast kind of object by the name of guard forecast and you can use this u guard forecast provide your fitted model guard fit and whatever the step you want to forecast ahead you can specify five period and then you can print the forecasted object so you'll get the forecasted objects here the forecasted series of sigma so now we'll fit the remaining five model the second one is gjr garch model so let's do the gjr gjr garch model i'll not rewrite the entire code so i'll just copy not rewrite the entire code so i'll just copy paste here i'll just copy and in this all i need to do is in this instead of this this i can mention this gjr the rest of the things will be taken care of automatically in the name also here i'll write gjr then here i need to change as gjr norm similarly we have next exponential guard or e got exponential guard for that again and here i need to write e for exponential guards again here i'll write exponential and i need to change here also the genome so this is my exponential gosh model detail that are used then the next after this i have created three specifications so now i'll use generalized error distribution.

So i'll use generalized error distribution assumption and i'll fit the remaining all three models all i need to do is in the same set of model i need to here change sdd as gd also i need to change here as plotting and then here gd similarly here i need to use a generalized distribution and here also instead of normal i'll use gd and here also i need to use gjr gd lastly here also i'll change to a gd and this also norm also also the plot colors that i have used i need to change them slightly for now i have used the plot here also i'll change to gd now the plot this is the sixth part so color six this is the fifth the color five then color four color three color two and then last color one so now i'll run two and then last color one so

now i'll run them all at one go and i'll clean my plotting window so that nice plot is there so let me run the full thing in one go starting from here let's see if it runs error free so yep it seems the graphs are being produced you can see very nice graphs are being produced the error is not there it may take a few time because it takes some time in now let me zoom it for you so this is my complete plot all the six models fitted volatility which is first is standard normal then gjr guards with normal distribution exponential guard with normal standard diode guard gjr with generalized error exponential with generalized error now there are some minor differences and we get slightly different different volatility predictions here we need to make use of aicbic information criteria while ranking them which we'll do in the next video in this video we'll at the end now we'll just compile that last complex error object and please note our complex error object ic dollar the five the fifth row we said that it should be the mean of all the aicbic criteria column means column means for all the four so all the four rows first four rows i'll compute it so let's see if it is done correctly if i print this object we can see yes we get all the error objects for our different different models so you can see here all the aicbic shibata hankwin complex and all the objects are printed here we can also see the visual comparison of them so in this video we created and fitted our garch model for four six different models particularly three distributions normal three two distributions normal and generalized error and three different models garch model that is standard exponential and gj.

So we have six models we fitted we extracted their information criterias and we also visualize it visualize the fitted volatility we also saw how to predict in future steps five steps how to predict volatilities for next five steps in the next video we'll compare the rank of all these six models on their prediction on their fitted volatility how good they are fitting using these information criterias and conclude the exercise in this video we'll compare the performance of our fitted six garch models and rank them based on their information criteria so we'll create a rank object we need to create a rank object so this rank object will be a data frame as we are doing rank object and it should have five rows for each information criteria kic bic shibata and quinine complex and six columns for each of the six models i'll borrow the names from the previous command i'll not rewrite everything again so i can borrow it here the call names and i'll just change it to rank variable so rank and rank so let me just print it out for you so if i print it you can see there are columns for each of the model as norm gjr norm and so on and information criteria ac bic sic and so on all i need to do is just put the rank and use the rank function apply this rank function using this apply command apply this rank function on the information criteria object on its all columns and print it so first i'll print this rank variable and i'll also print the ic variable so we can understand this ranking so here the lower the information criteria the better so for example here it seems that exponential gd is working best let's compare so if you look at any version the exponential gd seems to be the lowest across all there so the lowest information aic bic criterias are found in egd so in this case the lower

the information criterias the better and we can see the lowest is being produced by exponential gd guard so that is the best model that we have.

So to summarize this video we have compared the performance of all the gaatsch fitted models on different information criteria we find that the lowest value lowest information criteria value that is the best fitted values are obtained for all the five models including aic bic shibata and quinine complex on gd exponential gd so which is an axisymmetric model which gives us the best fit to summarize this lesson first we prepared our equal weighted portfolio object using the return series data of carbon market bitcoin asg fund and snp 500 securities then we examined the object with acf and pacf plots to find its arma properties next we break the data into two segments namely test and training data sets subsequently we fit various arma models with different lags of ar and ma tabs we examined the sample goodness of it for all these models by training data set using aic and bic information criteria then we also examined the out of sample forecasting accuracy of these trained models on the test data with several error measures we find that the bulky and less parsimonious models that attained high in sample fit are not necessarily good fit in our out of sample predictions lastly we model the volatility of our equal weight portfolio object using standard gjr and exponential gaatsch objects and assuming normal and generalized error distributions for the data we compared the goodness of it across six gaatsch models that is three gaatsch models and two distributional assumptions we found that the exponential gaatsch model with generalized error distribution offers the best fit we also plotted the fitted conditional volatility and performed volatility forecasting for our fitted volatility models.