Advanced Algorithmic Trading and Portfolio Management

Prof. Abhinava Tripathi
Department of Management Sciences
Indian Institute of Technology, Kanpur

Lesson-2

We will discuss Data Handling and Data Claiming with R.

As a best practice, I would recommend that we set our working directory where we want to read and write data. To start with, go to this session, set working directory, choose directory and then select the appropriate folder where we would like to set our working directory and click open. Notice that a command will appear on your console window. You can copy paste it for future references. So, whenever you are working in future, you can run this compound simply to set the working directory at appropriate place.

Now as an example, we will read an CSV file. We have already seen how to read and write data in here. So, we will simply write this read.csv, give the name of the file which is salary.csv, run this command and data will be read. Another way to read this data is go to this import dataset, go to from text. You can click on this salary file, click on open and it will read the data for you.

So that is another way to read the data. You can see the format, the way in which data is read. You can select the correct options if header is yes or row names, separator and all. You can see the permutations and combinations. The way the file will be appearing, you can see here in the data frame. If you click on import, the data will be read. So this is another way to read the data. In case it was an Excel file, you can import dataset from Excel and then again with same procedure browse and read the data. Again select the data and read it. So, this is how we will read the data.

As a next step, we would like to see the dimensions of data. So there are 14,017 rows or observations and seven columns or variables. So this is the dimension of data frame. Most of these commands we have already seen in the fundamentals of our module.

As a next step, we will summarize the data.
So we can see all the variables, their nature and various other aspects. You can also use brief command to have a look at this data. Brief command will give you broad overview of data. You can see that how data looks from starting to end. It starts from first three rows that are visible and last two rows.

You can also have a look at the head of the data with head command. You can see initial six elements using head commands. If you want to see more or less, you can select head data, maybe eight and you'll see the initial eight elements. Moreover, you can also see the structure of data with str command. It will give you the names of variables like name, their nature, which is character or numeric if it is, agents, all the variables we can see here, gross pay, annual salary and so on. So, we get the names of the data.

You can also check the column name. So simply by writing call names, you can check all the column names that are there. So you can see there are seven columns name, job title, agency ID and so on up till gross pay. As a first step, you'd like to change the names of some of these columns. So, if you want to change the names of these columns, let's see how to do that. So let's say I want to change the, so we'd like to change the name of the first column from name in small to name in caps like this. Also, probably you would like to change the names of some other columns starting from two to four, multiple columns and you'd like to change the name maybe from job title to title. So earlier name was job title for the column number two. I want to keep it title. Then there's a column called agency ID, which I would like to convert to ID and then there is agency column, which I would like to convert to agency name. So I'll run this command. Notice now the new names of the data variable. So all the names are changed as we wanted. You can also see the head command and you will see that the names are changed as you wanted them to be.

So with this, we have understood how to input the data, see its structure, summarize it and change the column names. In the next video, we'll talk about cleaning the observations in the data frame.

Now, we’ll talk about cleaning the observations, cleaning the observations, and cleaning the data frame. Let's consider two vectors. We have already seen what a vector of variable is.
It contains NA value also. So it contains certain values along with NA observation. So if I print this variable, I'll get those variables, including the NA observation. Now if I run this command X greater than two, notice that second element of the console is NA, which means R is not able to compare the second value with two. How to go about it? So while comparing X greater than two, we'll also use an interesting command and which is given by ampersand symbol and then we'll use is.na

What is is.na? is.na checks whether a value is NA or not. Let's look at this.

So if I put is.na along with NA, it will give true and if I use exclamatory mark, which is a symbol for no, it will give false. So I want to make this comparison of X greater than two, but I want to do it only with those values that are not NA value. So I'll put exclamatory mark, that means the value should not be NA and then run this along with ampersand. It says that the value has to be not NA and then the comparison with two has to be made and notice now that NA value is ignored. So instead of that NA, it is compared with two and we can get all the false, false, false. So, it has considered that NA is also not equal to two and it gives me a false value.

Now let's say I wanted to check whether X is not only equal to zero, but also I want to put another condition which is with or operator, or operator is either one of them can be true along with X equal to two. So if I run this, I get seven true, falses including one NA, we know because this second observation is NA. To account for that NA observation, we will write an additional command which is again same as and exclamatory mark is not NA and this will account for that NA observation and we will get false instead of NA. Similarly, there are some other operators, for example, many times the value is NA and not a number.

For example, something like zero by zero would be an NA and for R. So if I run zero by zero, that is true. Sometimes the value is infinite. So if the value is infinite, again similar operation, the same procedure we can select is not infinite. Let's say one by zero which is infinite value.

So it will give me true. So this way we can handle NA observations, observations that are not available and so on. Let's take one example. So originally we had that data which we read. Now let me make a copy of this data so that we do not disturb the original data. So, we read it in data underscore one. Now let's play around with this data underscore one a little bit. So for this data underscore one, let's put some values, maybe thousand row, this value represents thousand row element at fifth column. Let's set it to NA. Similarly, I'll also set 3000 positions in the second column as NA and then again, I will set maybe 4000 element on the third column as NA.

Let's run these commands. So there are three NAs that I have introduced in the original data. If you want to check whether there were any NAs in the original data, how do we check? A very simple way would be to check is dot NA data underscore one. First we'll check original and I will add an exclamatory mark or sum. So all the NA observations are taken here as proof. As we discussed earlier, true is equal to one.

So all the NA observations will be extracted from data and notice if I run the sum, it is zero. That means there are no NA observations in the original data. However, now that we have introduced three NA observations in the new data, if I check the same command, I'll get the summation as three because three NAs have been introduced. So now we have three NAs in this data.

There are other ways to check that. So for example, if I write all is dot NA, it checks whether all the observations are not NA and it tells me no. That means there are some observations that are NA. If I would have run this on the original data, this would have given me true. That means none of the observations were NA in the original data. The new data which is data underscore one carries those NA observations.

Now let's say you want to replace all those NA observations with zero. A very simple solution to do that is this form that you have already seen inside data underscore one. I filter out those observations that are in it with this simple command and all those observations I set as zero. That is one way to do that. When I do that, all those observations will be set to zero.
And if I take the summation, now I take the summation of is dot NA, it will be zero. So all those observations are zero or I can also check this one. All observations are not NA and yes, they are not NA. So all the observations are replaced by zero. You can replace it by any other notation if you want or in similar manner, we can process.

So this was some of the data cleaning observations. We will further move to some of the other ways to handle the data in next set of videos. You can see some of the examples on how to handle NA observations. Consider this data frame, we will create a data frame for the vector. We will create this data frame with the data dot frame command where element one is a vector which contains a new value and two numeric.

The next element B is another character vector which contains two characters, one NA and another character. So this is another vector and combination of these vectors is provided in BF data frame. So we can see what is BF data frame. It carries two NAs, one is the numeric variable NA and one is the character variable NA. Now one way to deal is individually remove these NAs from individual columns.

For example, I can subset BF. There are different ways to do it. As you see, we use the subset command and I will select only those NA observations in variable A in the data frame BF. If I do that, notice the resulting output removes NA row, row which contained NA in variable A which was a numeric variable and the remaining variable or remaining data frame is a 2 x 2 data frame instead of 3 x 3 original data frame. Similarly, we can do the treatment for column B and if I do that for B, again a 2 x 2 kind of data frame will emerge where the NA observation, the row corresponding to NA observation in column B which is a character vector is removed. However, if you want to remove all the NA observations in the entire data frame, a more comprehensive treatment is required, although many times it is not advised.

So you can run this subset DF and then you can write complete dot cases which will only select complete cases in the data frame and if you run this, notice only one row is left and NA's rows that contained NA in variable A and B both are removed. Another command which have similar effect which is NA.Tomit, it will also have similar effect you can use. So this will remove NA observations in our data frame.

We will work it a little bit more on NA. So let's use a library car. It contains a number of useful databases. So we use this library car. It contains a data called Friedman.

We will work on this Friedman data. This Friedman data seems to carry some NA observations. Again we will do all the similar commands like str, Friedman and so on to check the structure and other properties of the data. We can summarize it also. As when I summarize it, these are the observations. Now if I compute the median of this Friedman, kindly have a look at the summary data.

Look at the density variable. Notice that there are 10 NA observations. Let us see what is the impact of these NA's. Let us compute the median of Friedman density data. We know how to compute that.

If I compute the median, I will get an NA. A simple treatment to this kind of problem is to use median and then R provides this functionality to use NA.RM equal to 2. If I do that, R ignores the NA observations, not available observations and gives us the median value without considering both of them. Similarly, if you compute the mean of density, again you will get an NA because there is a NA observation.

So you compute mean with na.rm equal to true and you get the mean. So this is one good solution, one shortcut to handle NA in your observations. A more drastic way to handle this problem is to use Friedman data, Friedman.good and remove all the NA observations as we have seen earlier na.omit. While this kind of treatment removes all the NA observations, as you will see if I compute the summary measure of this Friedman.good data. However, it is a rather more drastic treatment because despite the fact that there may be only one NA in a particular variable, all the variable rows will be removed. All those rows there are no even single NA observation is there, those rows will be removed. Another way to do the same kind of treatment is to let's say have this Friedman data.

Let's create a Friedman underscore not available variable and in this variable we will use Friedman data. Again we will use exclamatory mark and that are not complete or any observation. This is the procedure to exactly notice what are the observations that can, what are the observations that carry NA values. So, we will use complete. cases since we want to extract all the observations that carry NA, we will put an exclamatory mark and in this variable notice in this command, in this not available data frame, we have all the variables that have some kind of NA observation.

Let me print that. So notice two columns, population and density column, 10 NA observations are up there. Because there are some other columns like non-white, crime and all, they carry some value. So depending upon your requirement, whether you are interested in retaining these values from non-white and crime variables, you may decide to use NA.omit. If you use NA.omit, then all these rows will also be removed from the table. We will take one more example on how to work around with the NA.na values. We will use using R library for that.

So we will make use of using R library. In this library, there is a babies database. So let's extract this babies database. From this database, there is a DWT column, which is the weight column, DWT, which is dad's weight. We will assign this to a variable x.

And in this variable, there are certain values which are outliers. Let's summarize this x variable and there we will see there are certain values which are outliers, which are coded as 999. Now this 999 may appear to be a numeric, but from a priori knowledge, we know that this is outlier. So how to handle this outlier? Let's say you want to decide, you decide to add NA or replace these 999s with NA. A very simple way to use this is this kind of command and then assign NA values.

So this will assign NA values to all the 999 values. Now once I do that, the values will be replaced by NA. So again, similar problem with the use of NA variable will emerge. So if I compute range of x, it will be NA. If I compute some summary of x, notice there are some NA values, but still we get some of the good summary measures like minimum, median, mean and so on. For range also, we can write or make use of our NA dot rm equal to 2, which is very useful.

So we get the range. So in this way, we handle the NA values in our data frame and vector of variables. Examine how to remove non-unique values. Recall that we had original data, salary data, which was saved into data variable.

We can check the head of this variable again. So this was our salary data. Let's create a copy of data as data underscore 2 and save the values here. Simple assignment operation. Now what we'll do is and notice the dimension of this data, it should be same as original data, 14017 observations. Now let's create another data which carries data underscore 3, which carries not only this data underscore 2, but also some values which are there in data underscore 2, starting from 1 to 500.

So all the columns and row numbers 1 to 500. So we are adding row wise with rbind dot data dot frame to create a new variable data underscore 3. Now it should be quite obvious to us that this data underscore 3 will carry those redundant or non-unique values at the end of it, which are exactly the values of row number 1 to 100, 500 from data underscore 2 data frame. So if you look at the data dimensions of this data underscore 3, they are 500 more than the original data frame, which is data underscore 2.

They are now 14,570. Column number will remain same of course. Now what we want to do is we want to remove these non-unique values and a very easy way to do that, let's say we create a new unique data frame with a data underscore 4 and unique function can be used which underscore 3 will run this command. And now if you notice dimension of data underscore 4, it should be same as 14017. So it carries only the unique observations and all those non-unique redundant observations that were repeated, because at the end of it, we added row 1 to 500 from the data underscore 2 data frame, all those redundant non-unique observations are removed. So with this, we conclude the discussion on removal of non-unique columns, handling the data frames, data handling.

Let's start with selection of columns and rows. Selection of column and rows in R. Although we have already seen this in bits and pieces earlier. So let's say there is an iris data in R, there are certain columns and each column carries certain observations which comprise rows. So let's say I want to select column number 3, a very simple command like this will extract the column number 3 for me. I can use this head to see the initial elements that I have extracted.

So these are the initial elements. If I want to extract multiple column elements, let's say column 3 and 5, I can do it like this. Column number 3 and 5 are extracted. If I want to extract all the columns from 3 to 5, a very simple command, 3 to 5, we have seen this notation, will extract column number 3, 4, 5.

I can check the head of this column number 3, 4, 5. It gives me 3 columns. Now you want to extract not only specific columns but specific row number elements also from those specific columns. Let's say you want to extract row number 4 to 10 for column number 3 to 5, then this kind of notation or this kind of command will extract row number 4 to 10 for columns 3, 4, 5, which is petal length, width and species. Although you can also extract columns with their names, but generally that is not so advisable. You can extract, let's say you want to extract column number species and another is petal width. You can also do that, however, it has problems because you need to remember the spellings and there you may make some mistakes.

So it is better to use numeric notation, but you can use this. So for example, you have species,

species, I need to remember the name exactly, so I need to use species and petal dot width exactly the same name. I need to remember with exact spelling and then I can extract this, these two. Better would be to use head so that we can see the initial elements, so I can extract that. Now, we will come to the next step, which is creation of new variables inside data frame. So the way to do it, let's say you want to create a new variable called petal dot ratio and this variable is equal to ratio of petal dot length divided by petal width and you want to create another variable called sepal dot ratio, sepal length divided by sepal width and you run that as well.

Now if I check the header, I will find that new columns are added, if I check that you have sepal ratio and petal ratio available. So there is a minus spelling mistake which I need to correct, so you get the sepal ratio and petal ratio variable here. Extracting observations based on conditions and summarizing the observations. So let's start with extracting observations.

Let's say in the iris data, I want to extract those observations petal width of more than 0.5 along with another condition and that condition should hold true where the species is it should have, let me check the head of this and I want a data where petal width is greater than 0.5 and species should be, I am going to use m percent to create that effect, species should be equal to equal to setosa. So now with this and I need to add a comma that all the columns are deselected. So with this all the variables where petal width is greater than 0.5 and species equal to setosa will be selected, pardon me for the spelling mistake, it should be setosa.

And so now if I run this, I can see the row that has been extracted for which petal width is greater than 0.5 and species equal to setosa. Similarly, I can create same effect with subset command as well, with subset I will specify the data, I will specify the petal width to be greater than 0.5 and I will use m percent operator to create the and effect where I am saying that species should be equal to equal to setosa. So with this I will have the similar effect and the same row although it seems there is only one row which will be extracted.

So now we will move on to some rising observations. So a very basic summary measure for any data frame is summary, we can see the nature of all the variables, their summary measure depending upon whether they are numeric, character or so on. We can also use structure command to get a sense of variables and we can also use brief command which will give us a sense of variables. Some initial three rows and closing two rows 149 and 140 we get a sense of the variable. Now you can create a user defined summary also let us say you want to summarize in this fashion.

You can use this summarize command and you can tell R that you want to summarize iris data frame. Inside iris data frame you want to create a mean variable let us say petal dot length dot mean which is the mean of petal length. So you can create a mean of petal length variable. So we have a petal length variable for which we want to create a mean petal length mean. So maybe I want to create a mean variable for sepal length again maybe you also want standard deviation for these two variables so I will use this d for standard deviation. Now if I run this command notice mean notice the output in console you have mean of sepal length and petal length you have standard deviation of sepal length and petal length they are produced.

So in this way I can create more variables with a more user defined or user desired summary. How to work with data frames. We will install library car which we have already used. As a first step we will make use of Davis data which is already inbuilt inside the car data frame. There are 200 rows and 5 columns. If we check the head of tables these are some of the elements gender column, weight, height, reported weight, reported height.

Now as a first step we will create another variable which is a data frame element. This data frame is expected to have same dimensions as Davis data. So we will use the following command matrix and number of rows same as Davis data and number of column also same as Davis data. So we will create this variable.

Let's see what is inside this data. This data carries 200 rows and 5 columns. We can check the demo for. So this is a new variable that we have created to store the observations for practice. Also we'll give the name to the variables inside Davis. So currently if you look at the head output you will notice that there are no names automatically by default x1 x2 are provided.

So we'll create some names for this. Since we are planning to use Davis data to fill this variable we'll use similar names that are gender, weight, height, reported weight, reported height. So this is our new variable. You can see the head now it will be changed with the new names that we have created.

Now let's assign the values of Davis variable inside this. So we'll assign the values. So this is our output variable dollar gender. So we'll assign the gender variable from Davis data. Similarly we'll assign the weight variable, height variable, reported weight variable, and we'll also assign the height variable. So in this fashion we have created this output variable which has similar dimensions as Davis data and for practice we have assigned this output variable same data or same variables as Davis variable like this.

So it has gender, weight, height, reported weight and reported height variables. So in this video we have learned how to create a data frame, how to assign values to that data frame, set its dimensions and then assign values from different sources. Learn how to deal with factor variables and we'll also perform some operations on data frames. So we'll learn working with factor variables. We'll make use of library using R in which there is a cars 93 variable which we’ll make use of.

We'll make use of this cars 93 variable which carries various dimensions of cars. Now as a starting point let's create a sub data frame or extract certain elements of cars 93. Let's first three rows and out of all the first four columns. So this is a three four three cross four kind of data. Let's see what is inside this. It carries four columns manufacturer, model, type and minimum price and three elements for each column or variable.

Now we can see the structure of this new data small data frame. We can also summarize this small data frame. As a starting point let's assign some NA values to this. So I'll assign the third row fourth column. I'll assign a new value and first row first column also I'll assign a new value. Now if you print this data frame D notice the first column manufacturer first value is NA.

Similarly fourth column minimum price last value is NA. In this sub data frame if you want to add let's say some new elements. Let's say you want to add to the third list you want to change the values that are there in column two and four and you want to change them to new elements maybe A3 to the second column and 30 to the fourth column should be easy right. However notice that it gives you a NA message and if you print D notice on the second column third row is created. The reason being if you notice the class of D model it's a factor variable so it has some levels what are these levels.

We can check those levels. Levels D $ model. So it has some levels. There are a number of levels which is because these levels have a stick because we extracted it from original class data so original levels are sticking. So let's first remove the unused levels which is quite simple. I'll use this drop level command drop levels from D $ model and if I run this command all the unused levels will be removed and then I will run this level command to see the remaining levels which are currently used. So if I run this you will find two levels are currently used one is integer one is legit so only two levels are remaining all the unused levels have been removed now. The problem is because this is a factor variable which is using certain levels which are specified if I add new levels like A3 they are not added and instead they create NA values because R is confused that this level is not specified.

So how to specify new levels. So we'll try and specify some new levels to this model variable. Please note we'll not ignore the earlier levels so we'll first create the existing levels with the model variable. So these are the existing levels that will make use of and in addition we'll add certain more levels. We'll add probably A3. We'll also add A4 and we'll maybe adding A6 also. Now that we have assigned these levels if I check the levels of model variable it will have three levels added and now we have five levels.

Now if we run the original command which created NA which is this D3C24 where we are trying to assign A3 to the third element of second column and 40 to the third element of fourth column there is no NA creation and you can see A3 has been assigned to model because we already specified it as a level. So this is how you add level. Let's say now you want to add a fourth column and there are multiple ways to do it. One way is to use this index notation where I use D4 and then say I simply add all the four elements first element let's say name of the car which is oddy then one level which is A4 since we have already specified A4 as a level this will not create any trouble then type as midsize and price minimum price is 35.

So if I do that a new row is created fourth row and we can see the elements oddy A4 midsize 35. Another way to do the same thing is to use rbind and I can create D equal to rbind and with rbind I can again write the same elements rbind D original D and with this original D I'll add the new set of variables oddy A4 and 35 this will also have the similar effect. So if I run this command I'll again get the new variable D which has the next element which is oddy A4 midsize since I have added already added the fourth row this will be added in the form of fifth row. So this is another way to do it. Now let's say you want to create a new column fifth column which is a multiple of minimum price let's say D dollar minimum price multiplied by 1.3 a new column will be created we can print D V5 is created if you want to give it a name you can select a name that you find useful let's say you pick a name of column D fifth column and call it mod price.

So a mod price name will be assigned if I print D instead of V5 we have now modified price or mod price. Another more simpler way to do that would be simply use D dollar mod price I give it a name mod price and then assign the same value which is D dollar minimum price into 1.3 this will also have the same effect and a new column mod price will be created. Another easy way to do the same effect is within function within function transformation will be made inside data frame D and within D we are assigning or creating a new mod price equal to minimum price this is also quite simple mod price into 1.3. So if I run this again I'll get a new variable D which is mod price is created. So this is how you transform the value work out with vectors and transform variables inside a data frame. Transforming data frames between long and byte format. So we'll transform the data frames across long and byte format. So let's start with a simple construction of data frame. Let's create a number of variables first let's say variable speed dot 1 this may be first observation for different speed let's have number of values here.

So these are hypothetical values of speed observations for different vehicles A, B, C, D, E, F. So these are some of observations. Similarly we'll have another set second set of observations as speed 2 with hypothetical values here may be. Similarly we'll have third observation for the speed variable again some random values like 800. We'll have fourth set of. So objective here is to create a rather wide format and then we'll see whether we are able to in the interest of time we'll keep the value the same.

So there are five speed observations speed 1 dot speed 2 speed 3 4 speed 5. Another there is ID variable which identifies the units 1, 2, 3, 4, 5, 6. Then we have one variable which may give the name. Let's call it A, B, C, D, E, F.

So these are speed variable 6, 5 speed variables ID run. Now let's combine them under the variable speed. And give it a name. Let's combine them first C with the command C by dot dot data frame first ID variable then run variable then speed dot 1 speed dot 2 speed dot 3 speed dot 4 and speed dot 5. So, these are the variables that we have created speed variable.

Then you have head of speed. We can see the variables ID and their runs. We can see the summary. Structure. We can see the variable. It's a rather wide data frame. In order to make this a long data frame we'll take the help of package reshape2.

So this reshape2 package will be added with the library command. We'll add this reshape2. And now we are going to go. So the way it works, we'll create a long data format by using melt function. Melt function will help us create this wide data format which is speed and then we’ll give the ID of variables that are to be fixed.

So we'll give the names speed. So first two variables. So we are giving the name of first two variables. We want them to be fixed. These are ID and run variables and the variable which we. The variable which we want to create a more long data frame.

This is the speed variable. So all the five speeds we want to put them in one variable called speed. So if I run this command. Now if I run the command notice how long variable appears now. So in this all the values are put in the value column and all the speed variables speed.1 speed.2 and so on are separated now. So you can see that speed.1 speed.2 and so on they are combined into one column which is speed and their values are put in value. So now this is rather long data frame. So now also we get the interpretation when we are saying long and wide.

And now we'll try to get back to our original wide data frame. How to do that for that we'll create a new data frame called wide. We'll use this dcast. dcast and we'll specify that we want the long data frame. The variables ID plus run they need to be fixed and the variable speed need to be adjusted. If I do that look at the head of wide data frame.

So this is how we work upon wide and long data frames different context required different formats maybe long or wide.

In this video we'll talk about merging data frames. Merging different data frames and various properties of the merge command. So we'll talk about merging data frames. Let's create two data frames. We create variable v1 and these are movies, and they are domestic collections. First we'll create that. So, first movie is the Avengers, Avengers Dark Knight, The Hobbit games, Skyfall, Hobbit. And then v2 which is their foreign collection. Maybe let's put some hypothetical numbers.

So we are putting some hypothetical numbers here. Now let's combine them. So we'll combine them. We'll give them a name domestic v1 and v2. So we have combined them. Let's see their names. Use head domestic. If you want to create more appropriate names you can use col names with Name, Domestic. Now we have created the name. So now if I run the head command, I will get the new data frame.

Probably I'll adjust a little bit so that it is visible. Next we'll create foreign collections variable. So again we'll start with the same process. We'll create v3 which is equal to and we'll use the movie's name. Probably we'll change the movie's names a little bit. So this time around we'll use a little bit difference we'll make.

So probably we'll remove the one of the Hunger Games and add Ice Age. Probably we'll change this. And then we'll add the collections. The idea is to create two data frames with slightly different movie names. And their collections and then try to merge them and see how it works out. So, this time around we'll add again since it is hypothetical case so we'll for collections we’ll use some again same hypothetical numbers, so it doesn't matter the numbers don't matter here.

So, then we have foreign. So we'll give it a name foreign. Equal to v3 comma v4. So these are foreign we can check the head. Again we'll switch the names. Please notice this time around while giving the names I will be using a slightly different notation so instead of exact name earlier we'll use the small cap not in caps. So notice instead of name variable earlier we are I'm using small name with small n and then foreign. So this name will be our joining variable but I'm using a different syntax.

Let's see the head. So now this is our name variable so let's create the final variable which is merge and notice how I create this variable so final variable, which is using merge command. I'm merging the domestic variable with foreign variable and notice by dot x so first is domestic variable so by dot x equal to name. And the second variable is foreign so by dot y equal to name but please notice here name I'm giving with n as a small and I'll add them up. Now this is a rather difficult case here if to make it simpler it is always advisable to use the same syntax so like this capital name again.

And if I do that now the name is foreign, I need not give the second thing. I can simply put it like this or rather I'll use for instead of this I will write in fact I'll write a new command if I would have written it like this capital name I'll execute this. It will be better if I keep this command. So now if I run this and now in order to add this I need things are more simple now I need not give two names I can simply use this. And head final so you can see it has merged the data let's see what exactly has happened.
So if you notice if I do this kind of command, it has merged all the movie names, and this kind of merge is sort of intersection. So if you notice two movie names are missing one is Ice Age and one is Hunger Games. Now since one of the movies was not present in one data frame while other was present in other, so they are excluded it's the sort of intersection of merge or inner merge. Let's do the outer merge that is also doable. So, for that we'll use instead of this command I will use all = t and now if I run this command then notice there is Ice Age movie there and NA is in the domestic because it was not there in domestic.

Similarly Hunger Games NA is in foreign, but all the movie names are present now. So this is called outer merge. Then you can also decide whether you want to merge based on one variable or the for example if I write all dot x equal to t in that case all the domestic will be taken and those that are not present in domestic will be ignored. Similarly if I do it for all dot y then all values that are available in foreign will be considered and domestic will be ignored. So now we conclude with the merging aspect of data frames and in this complete module we learned how to clean and handle data and a more complex form of data which is data frame.

Thank you.