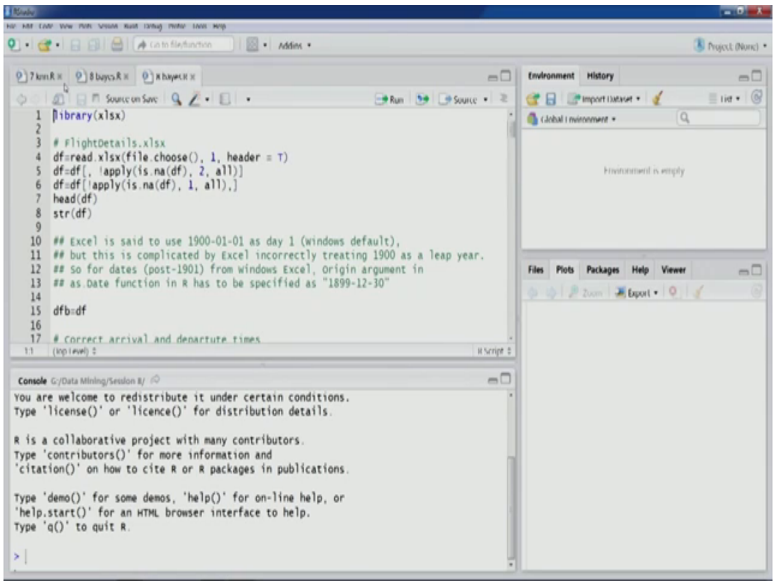
Business Analytics & Data Mining Modeling Using R
Dr. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology, Roorkee

Lecture - 34
Naive Bayes – Part IV

Welcome to the course business analytics and data mining modelling using R. So, in the previous lecture we were doing an exercise for Naive Bayes we did an exercise for Naive Bayes in excel. Now we also talked about a particular data set on a flights the on flights, and we are going to use that to do a modelling exercise in R. So, let us open R studio so on.

(Refer Slide Time: 01:00)

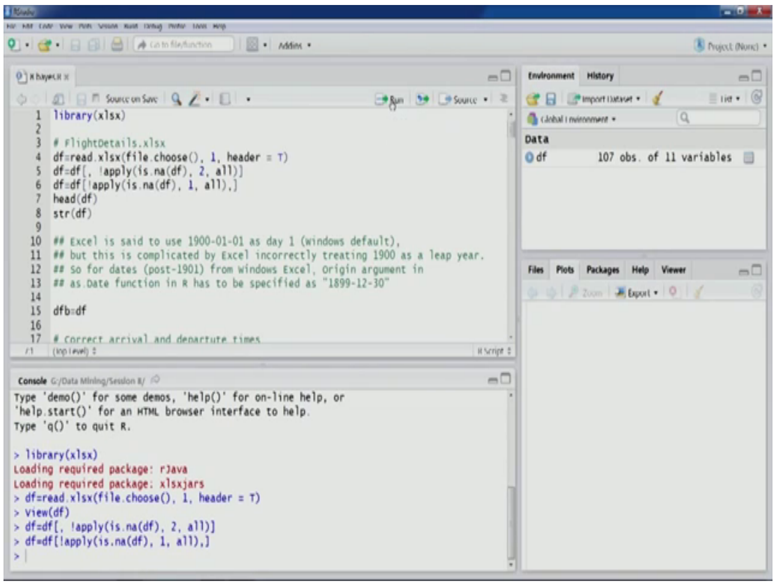


```
library(xlsx)

# FlightDetails.xlsx
df=read.xlsx(file.choose(), 1, header = T)
df=df[, !apply(is.na(df), 2, all)]
df=df[!apply(is.na(df), 1, all),]
head(df)
str(df)

## Excel is said to use 1900-01-01 as day 1 (Windows default),
## but this is complicated by Excel incorrectly treating 1900 as a leap year.
## So for dates (post-1901) from Windows Excel, origin argument in
## as.Date function in R has to be specified as "1899-12-30"

dfb=df

# Correct arrival and departute times
```

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

So first let us reload this particular library xlsx because we have data in excel format. So, we would like to utilize this particular library functions available in this to import the data set as we I am been doing before.

(Refer Slide Time: 01:20)



```
library(xlsx)

# FlightDetails.xlsx
df=read.xlsx(file.choose(), 1, header = T)
df=df[, !apply(is.na(df), 2, all)]
df=df[!apply(is.na(df), 1, all),]
head(df)
str(df)

## Excel is said to use 1900-01-01 as day 1 (Windows default),
## but this is complicated by Excel incorrectly treating 1900 as a leap year.
## So for dates (post-1901) from Windows Excel, origin argument in
## as.Date function in R has to be specified as "1899-12-30"

dfb=df

# Correct arrival and departute times
```

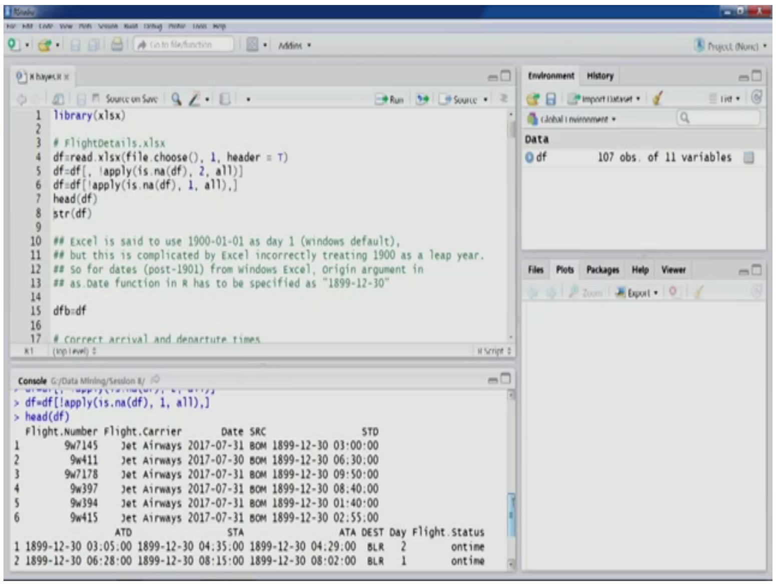
```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(xlsx)
Loading required package: rJava
Loading required package: xlsxjars
> df=read.xlsx(file.choose(), 1, header = T)
> View(df)
> df=df[, !apply(is.na(df), 2, all)]
> df=df[!apply(is.na(df), 1, all),]
> |
```

Data
df   107 obs. of 11 variables

So, this so file is flight details dot xlsx. So, let us import the data set. So, here you can see 108 observations of 13 variables , but we do not have those many variables and there are some NA columns and NA rows in this actually. So, you can see in the data set, and these are some NA columns, and we might also have yes and there are some and I wrote as well. So, we would like to get rid of these NA rows and na columns because we do not have those many variables and rows.

So, first particular code as we have been talking about in previous lectures as well previous our exercise as well that the first one will remove the NA columns. So, let us says execute this. The second code would actually remove the NA rows. So, let us execute this. So, how it works we have discussed in previous lectures as well. Let us look at the first 6 observations. So, these are the observations first we have flight number the flight number is given there for different carriers flight carriers also there.

(Refer Slide Time: 02:40)



```r
library(xlsx)

# FlightDetails.xlsx
df=read.xlsx(file.choose(), 1, header = T)
df=df[, !apply(is.na(df), 2, all)]
df=df[!apply(is.na(df), 1, all),]
head(df)
str(df)

## Excel is said to use 1900-01-01 as day 1 (windows default),
## but this is complicated by Excel incorrectly treating 1900 as a leap year.
## So for dates (post-1901) from Windows Excel, Origin argument in
## as.Date function in R has to be specified as "1899-12-30"

dfb=df

# Correct arrival and departute times
```

```
> df=df[!apply(is.na(df), 1, all),]
> head(df)
  Flight.Number Flight.Carrier       Date SRC                 STD
1        9W7145    Jet Airways 2017-07-31 BOM 1899-12-30 03:00:00
2         9W411    Jet Airways 2017-07-30 BOM 1899-12-30 06:30:00
3        9W7178    Jet Airways 2017-07-31 BOM 1899-12-30 09:50:00
4         9W397    Jet Airways 2017-07-31 BOM 1899-12-30 08:40:00
5         9W394    Jet Airways 2017-07-31 BOM 1899-12-30 01:40:00
6         9W415    Jet Airways 2017-07-31 BOM 1899-12-30 02:55:00
                  ATD                 STA                 ATA DEST Day Flight.Status
1 1899-12-30 03:05:00 1899-12-30 04:35:00 1899-12-30 04:29:00  BLR   2        ontime
2 1899-12-30 06:28:00 1899-12-30 08:15:00 1899-12-30 08:02:00  BLR   1        ontime
```

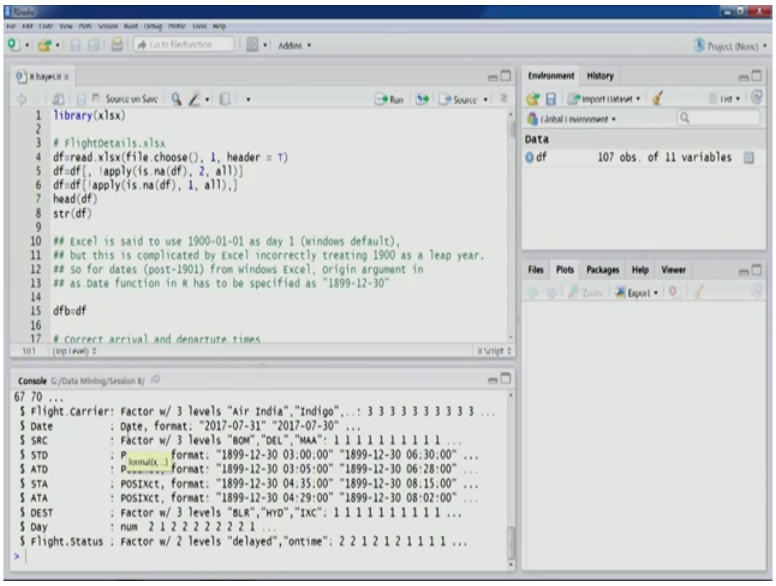
So there are as we will see there are three carriers we have information on data on three carries we have and date these are the 2 dates specifically. Then we have source, then we have a schedule time of departure here you would see this number is actually you can see 899 12 30 how and then after this we have schedule departure time.

So, how this number is coming in the why this particular date is coming in there so this is important detail that we will discuss. Now all the variables which are storing time actual time of departure schedule time of arrival or actual time of the depa arrival you would see that this particular date is also appended before the time right.

After this we have a destination variable destination variable, we have actually 3 variables in this. So, we will discuss them 2 days, flights on 2 days a one represented by one is representing Sunday and 2 is indicating Monday and then we have flight status on time or delayed.

So, as we discussed in the previous class if the actual time of arrival of a flight is a less than or equal to schedule time or arrival time than it is on time if it is more than that more than schedule time of arrival then it is delayed it has been classified as delayed. So, let us run this structure function.

(Refer Slide Time: 04:30)



```r
library(xlsx)

# FlightDetails.xlsx
df=read.xlsx(file.choose(), 1, header = T)
df=df[, !apply(is.na(df), 2, all)]
df=df[!apply(is.na(df), 1, all),]
head(df)
str(df)

## Excel is said to use 1900-01-01 as day 1 (Windows default),
## but this is complicated by Excel incorrectly treating 1900 as a leap year.
## So for dates (post-1901) from Windows Excel, Origin argument in
## as.Date function in R has to be specified as "1899-12-30"

dfb=df

# Correct arrival and departute times
```

```
67 70 ...
 $ Flight.Carrier: Factor w/ 3 levels "Air India","Indigo",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ Date          : Date, format: "2017-07-31" "2017-07-30" ...
 $ SRC           : Factor w/ 3 levels "BOM","DEL","MAA": 1 1 1 1 1 1 1 1 1 1 ...
 $ STD           : P        format: "1899-12-30 03:00:00" "1899-12-30 06:30:00" ...
 $ ATD           : P        format: "1899-12-30 03:05:00" "1899-12-30 06:28:00" ...
 $ STA           : POSIXct, format: "1899-12-30 04:35:00" "1899-12-30 08:15:00" ...
 $ ATA           : POSIXct, format: "1899-12-30 04:29:00" "1899-12-30 08:02:00" ...
 $ DEST          : Factor w/ 3 levels "BLR","HYD","IXC": 1 1 1 1 1 1 1 1 1 1 ...
 $ Day           : num  2 1 2 2 2 2 2 2 2 1 ...
 $ Flight.Status : Factor w/ 2 levels "delayed","ontime": 2 2 1 2 1 2 1 1 1 1 ...
> 
```

You can see that flight carrier it has 3 levels Air India, Indigo, and then Jet Airways as you saw in the first 6 observation. So, these are the three carriers we have information on these 3 carriers then the date, is there are 2 dates 30 and 31st of July 2017.

And then the next variable is source that is factor variable 3 levels “BOM” that is for a Mumbai and “DL” that is for Delhi and then m a “MAA” that is for Chennai airports. And then we have scheduled time of departure and the actual time of departure schedule time of arrival schedule time of arrival.

So, these are actually this POSIXc t, format that is there are and the times are given all right. So, we will discuss this why this particular date is coming there then destination we have three levels so these are the 3 airports, “BLR” for Bangalore, as “HYD” for Hyderabad and “IXC” for Chandigarh. So, these are the 3, these are the 3 airports.

Then this information on days so that is 2 days of flight on 2 days we have in our data set. So, one representing Sunday as we talked about 2 and then we need to this is shown here as the numerical variables. So, we would be required to convert it into a factor variable or categorical variable. then we have flight status 2 levels in delayed and on time. So, 2 levels as you can see here.

So, let us talk about the this peculiar date that is being appended in the in this particular arrival or departure time. If we look at the excel file that we have here if we look at the excel file we do not have this information here you can see that scheduled time of departure as execute time of departure and the excel file we do not have this information.

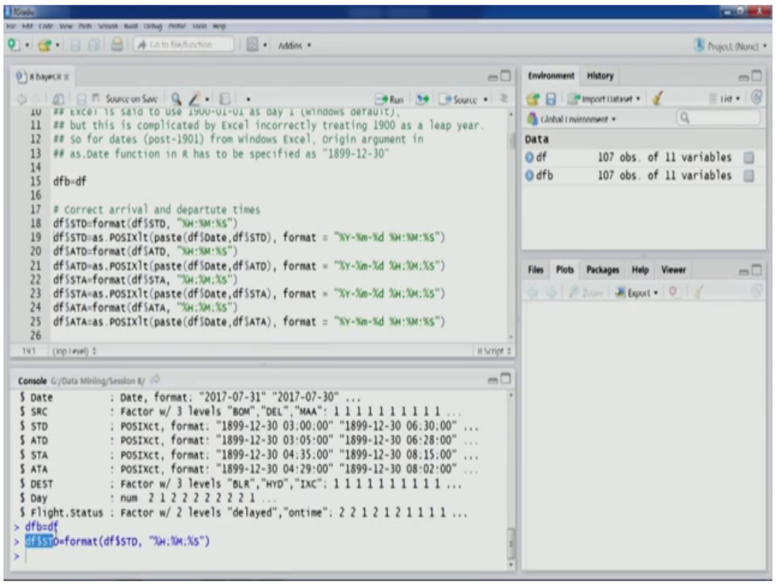
But when this flight data set is actually loaded into our environment are actually on the format in which are imports this particular data it requires it to also have the calendar date and the of course, the data is being imported from excel into R environment it is the date default calendar date that is available in excel that is being taken and the default date that is being taken is actually in the calendar date of day one that is there for in the windows excel.

So, in the windows excel we have nineteen hundred one and one that is first January of 1900 that as the day one so that is the default value ah, but this particular value is not shown here instead 18, 99, 12, 30 that is 30th of December 1899 that is shown here that is again because of that the excel incorrectly treats 1900 as a leap year and this has been this being the case all the dates which are come after 1901 so they are for those dates the we have to do the adjustment.

So, the so this particular this particular date has been taken the after this adjustment for post 1901, you can see this particular date which was 1901 and on which was meant to be in windows excel as 1901 1 now becomes 1899 1230 right. So, so this is the particular date that is taken by default when we import data from excel windows excel we are running windows operating system and we are importing this data from the windows system in and the excel data into our environment so, this date is being appended here. Now how do we correct this.

So, let us look at the few code if you lines up code so first let us take a backup. So, this was the data originally has it was imported. So, we taken the backup now what we will do we will format these particular for these particular times you know arrival and departure times into this format just this format so that the end date is we are able to get rid of the date information.

(Refer Slide Time: 09:39)



```
## Excel is said to use 1900-01-01 as day 1 (Windows default),
## but this is complicated by Excel incorrectly treating 1900 as a leap year.
## So for dates (post-1901) from Windows Excel, origin argument in
## as.Date function in R has to be specified as "1899-12-30"

dfb=df

# Correct arrival and departute times
df$STD=format(df$STD, "%H:%M:%S")
df$STD=as.POSIXlt(paste(df$Date,df$STD), format = "%Y-%m-%d %H:%M:%S")
df$ATD=format(df$ATD, "%H:%M:%S")
df$ATD=as.POSIXlt(paste(df$Date,df$ATD), format = "%Y-%m-%d %H:%M:%S")
df$STA=format(df$STA, "%H:%M:%S")
df$STA=as.POSIXlt(paste(df$Date,df$STA), format = "%Y-%m-%d %H:%M:%S")
df$ATA=format(df$ATA, "%H:%M:%S")
df$ATA=as.POSIXlt(paste(df$Date,df$ATA), format = "%Y-%m-%d %H:%M:%S")
```

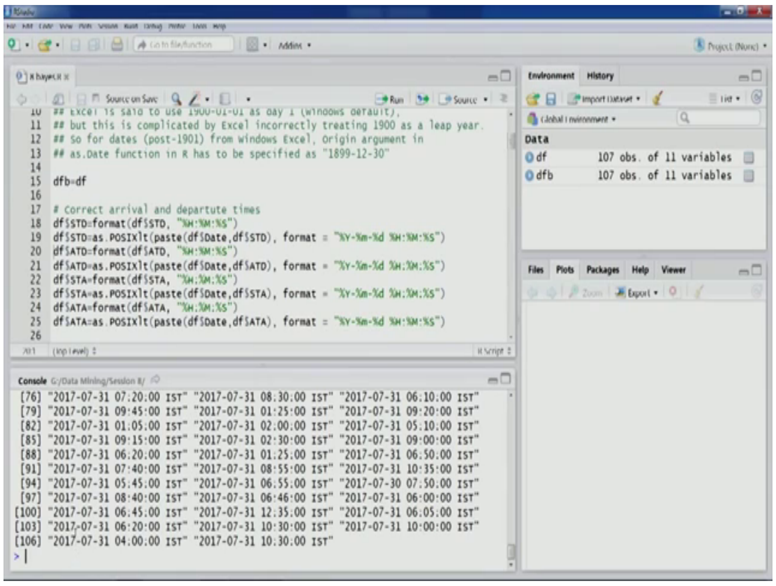
```
 $ Date          : Date, format: "2017-07-31" "2017-07-30" ...
 $ SRC           : Factor w/ 3 levels "BOM","DEL","MAA": 1 1 1 1 1 1 1 1 1 1 ...
 $ STD           : POSIXct, format: "1899-12-30 03:00:00" "1899-12-30 06:30:00" ...
 $ ATD           : POSIXct, format: "1899-12-30 03:05:00" "1899-12-30 06:28:00" ...
 $ STA           : POSIXct, format: "1899-12-30 04:35:00" "1899-12-30 08:15:00" ...
 $ ATA           : POSIXct, format: "1899-12-30 04:29:00" "1899-12-30 08:02:00" ...
 $ DEST          : Factor w/ 3 levels "BLR","HYD","IXC": 1 1 1 1 1 1 1 1 1 1 ...
 $ Day           : num  2 1 2 2 2 2 2 2 2 1 ...
 $ Flight.Status : Factor w/ 2 levels "delayed","ontime": 2 2 1 2 1 2 1 1 1 1 ...
> dfb=df
> df$STD=format(df$STD, "%H:%M:%S")
>
```

So, let us execute this. So, once this is has been formatted if you are interested in the value you can type here and you would see that all the values are in this format. Now and you are interested in the class of this now and that also you can see this is now character vector of 107 length right. So, this has been extracted now we will convert it into a POSIXlt format that is available in R.

So, we use the paste function where we will first paste this particular date that is stored in the another variable in the data frame which is having the corrected information as we saw in the previous output.

Let us look at again so this is the date column where we have the correct dates. So, that particular date is going to be used now and then we will have stored in this format. So, let us execute this know once this is done.

(Refer Slide Time: 10:51)



```
## Excel is said to use 1900-01-01 as day 1 (Windows default),
## but this is complicated by Excel incorrectly treating 1900 as a leap year.
## So for dates (post-1901) from Windows Excel, Origin argument in
## as.Date function in R has to be specified as "1899-12-30"

dfb=df

# Correct arrival and departute times
df$STD=format(df$STD, "%H:%M:%S")
df$STD=as.POSIXlt(paste(df$Date,df$STD), format = "%Y-%m-%d %H:%M:%S")
df$ATD=format(df$ATD, "%H:%M:%S")
df$ATD=as.POSIXlt(paste(df$Date,df$ATD), format = "%Y-%m-%d %H:%M:%S")
df$STA=format(df$STA, "%H:%M:%S")
df$STA=as.POSIXlt(paste(df$Date,df$STA), format = "%Y-%m-%d %H:%M:%S")
df$ATA=format(df$ATA, "%H:%M:%S")
df$ATA=as.POSIXlt(paste(df$Date,df$ATA), format = "%Y-%m-%d %H:%M:%S")
```

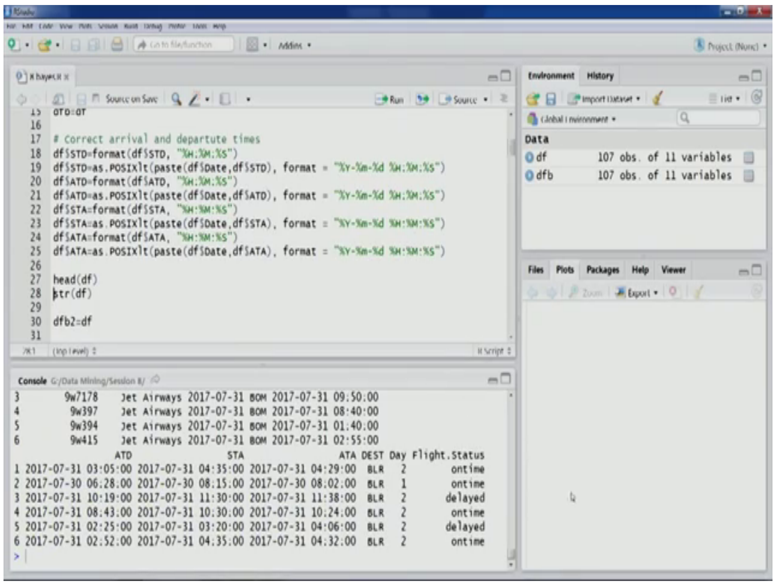
```
 [76] "2017-07-31 07:20:00 IST" "2017-07-31 08:30:00 IST" "2017-07-31 06:10:00 IST"
 [79] "2017-07-31 09:45:00 IST" "2017-07-31 01:25:00 IST" "2017-07-31 09:20:00 IST"
 [82] "2017-07-31 01:05:00 IST" "2017-07-31 02:00:00 IST" "2017-07-31 05:10:00 IST"
 [85] "2017-07-31 09:15:00 IST" "2017-07-31 02:30:00 IST" "2017-07-31 09:00:00 IST"
 [88] "2017-07-31 06:20:00 IST" "2017-07-31 01:25:00 IST" "2017-07-31 06:50:00 IST"
 [91] "2017-07-31 07:40:00 IST" "2017-07-31 08:55:00 IST" "2017-07-31 10:35:00 IST"
 [94] "2017-07-31 05:45:00 IST" "2017-07-31 06:55:00 IST" "2017-07-30 07:50:00 IST"
 [97] "2017-07-31 08:40:00 IST" "2017-07-31 06:46:00 IST" "2017-07-31 06:00:00 IST"
[100] "2017-07-31 06:45:00 IST" "2017-07-31 12:35:00 IST" "2017-07-31 06:05:00 IST"
[103] "2017-07-31 06:20:00 IST" "2017-07-31 10:30:00 IST" "2017-07-31 10:00:00 IST"
[106] "2017-07-31 04:00:00 IST" "2017-07-31 10:30:00 IST"
> 
```

Now let us look at the values. So, you can see correctly specified the dates right where it is supposed to be 30, it is 30 or 30th of July 2017, where it is supposed to be 31st of July 2017 it is that and the time format is also now IST.

The earlier time format if you are interested in but we will not look at the time zone does not look at the time zone. So, we can still do it use using by looking at the this particular the backup that we had taken. So, there you can see the by default not just the calendar date that was appended was the day one of windows excel and that too in you know incorrect. And then the time zone is also taken by default at GMT right.

So, we have been able to change this and now the correct dates are appended and then IST. So, you might be thinking why we have date calendar date information at all in time. So, this is the our format where it requires if you do not if you just try to create a POSIX lt or ct object without the calendar date it will automatically take the systems date and append it to that time information. So, let us do the same exercise for other columns other date columns times column that we have arrival departure columns. So, let us execute these lines of code.

(Refer Slide Time: 12:30)



```r
dfb=df

# Correct arrival and departute times
df$STD=format(df$STD, "%H:%M:%S")
df$STD=as.POSIXlt(paste(df$Date,df$STD), format = "%Y-%m-%d %H:%M:%S")
df$ATD=format(df$ATD, "%H:%M:%S")
df$ATD=as.POSIXlt(paste(df$Date,df$ATD), format = "%Y-%m-%d %H:%M:%S")
df$STA=format(df$STA, "%H:%M:%S")
df$STA=as.POSIXlt(paste(df$Date,df$STA), format = "%Y-%m-%d %H:%M:%S")
df$ATA=format(df$ATA, "%H:%M:%S")
df$ATA=as.POSIXlt(paste(df$Date,df$ATA), format = "%Y-%m-%d %H:%M:%S")

head(df)
str(df)

dfb2=df
```

```
3       9w7178    Jet Airways 2017-07-31 BOM 2017-07-31 09:50:00
4        9w397    Jet Airways 2017-07-31 BOM 2017-07-31 08:40:00
5        9w394    Jet Airways 2017-07-31 BOM 2017-07-31 01:40:00
6        9w415    Jet Airways 2017-07-31 BOM 2017-07-31 02:55:00
                  ATD                 STA                 ATA DEST Day Flight.Status
1 2017-07-31 03:05:00 2017-07-31 04:35:00 2017-07-31 04:29:00  BLR   2        ontime
2 2017-07-30 06:28:00 2017-07-30 08:15:00 2017-07-30 08:02:00  BLR   1        ontime
3 2017-07-31 10:19:00 2017-07-31 11:30:00 2017-07-31 11:38:00  BLR   2       delayed
4 2017-07-31 08:43:00 2017-07-31 10:30:00 2017-07-31 10:24:00  BLR   2        ontime
5 2017-07-31 02:25:00 2017-07-31 03:20:00 2017-07-31 04:06:00  BLR   2       delayed
6 2017-07-31 02:52:00 2017-07-31 04:35:00 2017-07-31 04:32:00  BLR   2        ontime
>
```

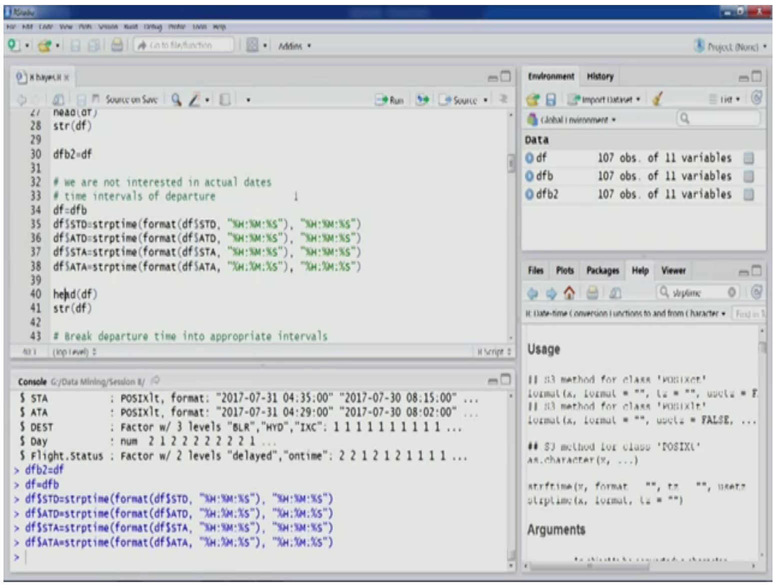
And now let us look at the first 6 observations. Now if you see 30 dates are being correctly specified and we have also seen already that time zones are have also been appropriately corrected we can also have a look at the structure of this particular data frame you can see structure now and you would see in the structure as well the correct dates are now there.

Now let us also take a backup of this particular data frame as well, but as we have discussed that a particular task is to predict the flight delays therefore, we are not and the predictors information that we would like to model is not actually incorporating the actual dates rather than we are going to use the time intervals of departure.

So, therefore, we are not interested in actual date so it does not matter us whether the date information is appended in the in the departure and arrival times and what that date is we are not interested in that, but we are interested in the time intervals during a particular day. So, therefore, having different date appended in a particular you know arrival or departure time will complicate the thing and complicate the process as you will see in the later part of the code.

We would like to have to simplify that coding process that we are going to follow later on we would like to have the same date not even though the data is on 2 days 31st of July 2017 and 30 th of July 2017 we would like to have just one date here. So, that our computation which we are going to do later on are easy because we are interested in time intervals in on any calendar date right.

(Refer Slide Time: 14:30)



```
head(df)
str(df)

dfb2=df

# we are not interested in actual dates
# time intervals of departure
df=dfb
df$STD=strptime(format(df$STD, "%H:%M:%S"), "%H:%M:%S")
df$ATD=strptime(format(df$ATD, "%H:%M:%S"), "%H:%M:%S")
df$STA=strptime(format(df$STA, "%H:%M:%S"), "%H:%M:%S")
df$ATA=strptime(format(df$ATA, "%H:%M:%S"), "%H:%M:%S")

head(df)
str(df)

# Break departure time into appropriate intervals
```

```
$ STA           : POSIXlt, format: "2017-07-31 04:35:00" "2017-07-30 08:15:00" ...
$ ATA           : POSIXlt, format: "2017-07-31 04:29:00" "2017-07-30 08:02:00" ...
$ DEST          : Factor w/ 3 levels "BLR","HYD","IXC": 1 1 1 1 1 1 1 1 1 1 ...
$ Day           : num  2 1 2 2 2 2 2 2 2 1 ...
$ Flight.Status : Factor w/ 2 levels "delayed","ontime": 2 2 1 2 1 2 1 1 1 1 ...
> dfb2=df
> df=dfb
> df$STD=strptime(format(df$STD, "%H:%M:%S"), "%H:%M:%S")
> df$ATD=strptime(format(df$ATD, "%H:%M:%S"), "%H:%M:%S")
> df$STA=strptime(format(df$STA, "%H:%M:%S"), "%H:%M:%S")
> df$ATA=strptime(format(df$ATA, "%H:%M:%S"), "%H:%M:%S")
>
```

```
Data
df      107 obs. of 11 variables
dfb     107 obs. of 11 variables
dfb2    107 obs. of 11 variables
```

```
Usage

## S3 method for class 'POSIXct'
format(x, format = "", tz = "", usetz = F
## S3 method for class 'POSIXlt'
format(x, format = "", usetz = FALSE, ...

## S3 method for class 'POSIXt'
as.character(x, ...)

strftime(x, format = "", tz = "", usetz
strptime(x, format, tz = "")

Arguments
```

We are not bothered about the specific date. So, let’s restore this particular data frame they are be the first backup that we have taken and we are again going to use this particular function strip time is the t R p time so this is going to format the information that we have into the in the format that we want we will see just now.

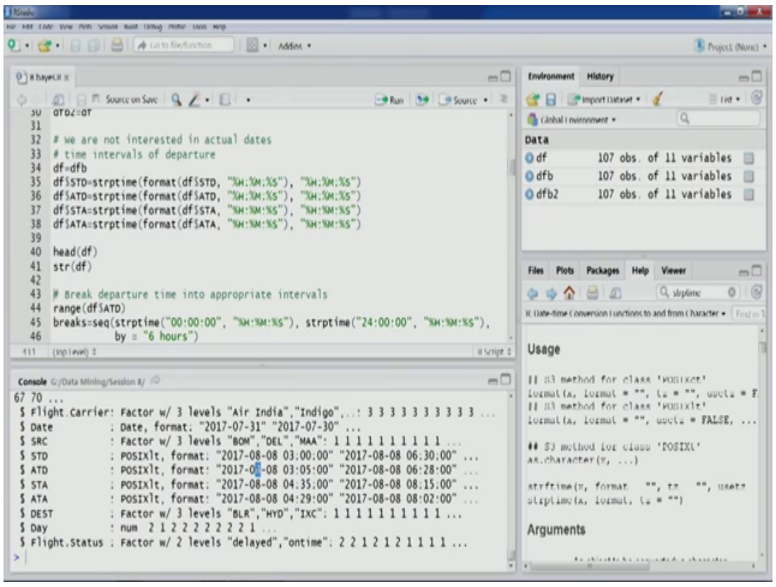
So, if you are interested in finding more detail on this particular function you can type the name of the function here in the help section and you can find more information your date time conversion functions to and from characters.

So, this is these what we will take the character vectors character vector in the x and will format as per the specified format and the time zone if it is null not specified then the systems time zone would actually be taken and this is what has happened if we look at the first 6 observation you would see that all the times now they have been appended with the this particular days systems date here you can see 8th of August 2017.

So, all the time, time related variables departure and arrival times they have been appended with this particular date even though the actual date as per the date columns were different. But since we are not interested in dates we do not mind because we would be calculating time intervals.

So, there we would like to have the same date instead of different dates, can have a look at this structure variable as well you can see the different date has been mentioned here.

(Refer Slide Time: 16:30)



```
dfb2=df

# we are not interested in actual dates
# time intervals of departure
df=dfb
df$STD=strptime(format(df$STD, "%H:%M:%S"), "%H:%M:%S")
df$ATD=strptime(format(df$ATD, "%H:%M:%S"), "%H:%M:%S")
df$STA=strptime(format(df$STA, "%H:%M:%S"), "%H:%M:%S")
df$ATA=strptime(format(df$ATA, "%H:%M:%S"), "%H:%M:%S")

head(df)
str(df)

# Break departure time into appropriate intervals
range(df$ATD)
breaks=seq(strptime("00:00:00", "%H:%M:%S"), strptime("24:00:00", "%H:%M:%S"),
          by = "6 hours")
```

```
67 70 ...
 $ Flight.Carrier: Factor w/ 3 levels "Air India","Indigo",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ Date          : Date, format: "2017-07-31" "2017-07-30" ...
 $ SRC           : Factor w/ 3 levels "BOM","DEL","MAA": 1 1 1 1 1 1 1 1 1 1 ...
 $ STD           : POSIXlt, format: "2017-08-08 03:00:00" "2017-08-08 06:30:00" ...
 $ ATD           : POSIXlt, format: "2017-08-08 03:05:00" "2017-08-08 06:28:00" ...
 $ STA           : POSIXlt, format: "2017-08-08 04:35:00" "2017-08-08 08:15:00" ...
 $ ATA           : POSIXlt, format: "2017-08-08 04:29:00" "2017-08-08 08:02:00" ...
 $ DEST          : Factor w/ 3 levels "BLR","HYD","IXC": 1 1 1 1 1 1 1 1 1 1 ...
 $ Day           : num  2 1 2 2 2 2 2 2 2 1 ...
 $ Flight.Status : Factor w/ 2 levels "delayed","ontime": 2 2 1 2 1 2 1 1 1 1 ...
>
```

```
Data
df     107 obs. of 11 variables
dfb    107 obs. of 11 variables
dfb2   107 obs. of 11 variables
```

So, we had been converting data into one format to other another because of this issue wherein the departure and arrival times are being appended with the calendar date as well.

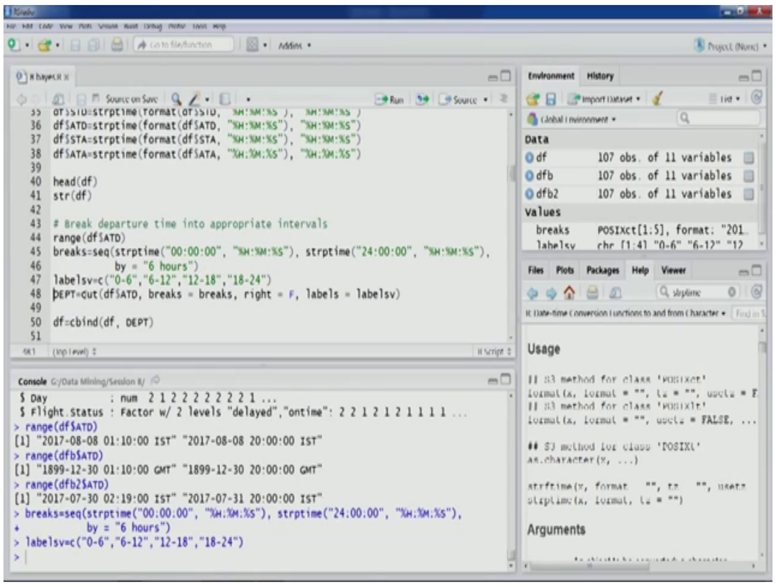
So, we saw that by default it takes the 18 99 30 12 value then we change it to the correct values at for the as per the information the data set that we have 30th July and 31st July then because the later computations and the modelling that be required we do not actually focus on these specific dates and the computation would simplify if we have the same date for all the times it would be easier for us to compute the time intervals later on.

So, we have appended the correct if we have appended the current data of the system so as you can see here. Now as we discussed what we are going to do is we are going to take the actual departure time and this particular variable ATD and we are going to break it into appropriate time intervals.

So, we are converting a variable converting particular date variable or time variable rather time variable into a categorical variable. So, we are going to do binning for this. So, let us look at the range.

So, here it was for this range that we changed the calendar design the calendar date for arrival and departure times. Now had it been the had it been the same dates the actual dates as you would see that as in the, this particular data frame I guess. So, we would see that ATD if I run a range for this not this one the, they where we had the actual dates. So, I think this one.

(Refer Slide Time: 18:34)



```
df$STD=strptime(format(df$STD, "%H:%M:%S"), "%H:%M:%S")
df$ATD=strptime(format(df$ATD, "%H:%M:%S"), "%H:%M:%S")
df$STA=strptime(format(df$STA, "%H:%M:%S"), "%H:%M:%S")
df$ATA=strptime(format(df$ATA, "%H:%M:%S"), "%H:%M:%S")

head(df)
str(df)

# Break departure time into appropriate intervals
range(df$ATD)
breaks=seq(strptime("00:00:00", "%H:%M:%S"), strptime("24:00:00", "%H:%M:%S"),
           by = "6 hours")
labelsv=c("0-6","6-12","12-18","18-24")
DEPT=cut(df$ATD, breaks = breaks, right = F, labels = labelsv)

df=cbind(df, DEPT)
```

```
 $ Day            : num  2 1 2 2 2 2 2 2 2 1 ...
 $ Flight.Status : Factor w/ 2 levels "delayed","ontime": 2 2 1 2 1 2 1 1 1 1 ...
> range(df$ATD)
[1] "2017-08-08 01:10:00 IST" "2017-08-08 20:00:00 IST"
> range(dfb$ATD)
[1] "1899-12-30 01:10:00 GMT" "1899-12-30 20:00:00 GMT"
> range(dfb2$ATD)
[1] "2017-07-30 02:19:00 IST" "2017-07-31 20:00:00 IST"
> breaks=seq(strptime("00:00:00", "%H:%M:%S"), strptime("24:00:00", "%H:%M:%S"),
+            by = "6 hours")
> labelsv=c("0-6","6-12","12-18","18-24")
> 
```

So, this is what you see that it is also in this particular range it is also counting the calendar date which is 30th of July comes first. So, the first date that we have is this particular information on 30th July to 19 and then last one is 31st July and 20. So, this is this we do not want and do not want because we want to create time intervals irrespective of the day.

So, in this case if we have the same date then you would see that we do not focus on the date then you would see the first time is 1 10 and then the last one is 20. So, this is the difference we have to be on to create the time intervals you have to focus on this.

So, this is what we are going to do it here is we will create different intervals. So, this is a function that we are going to use breaks then we are going to create a sequence of time intervals for time intervals and as you typical the strip time function can be used to convert the character vector into a appropriate you know time and date and time format in R.

So, as per this format specified format it will be converted 0 0 value and up to 24 and we are going to create 6 by 6 hours so we are going to create 4 time intervals if for within 24 hours for a day.

So, first is going to be as you can see in the next code labels 0 to 6. So, between 0 to 6 a m that is first time interval for us from 6 to 12, and that is the second time interval first for us, and then 12 to 18 the third time interval, and then 18 to 24. So, 24 hours of a particular day we have broken we are trying to break them into 4 time intervals and then later on as you will see we will class we will break the departure time departure, departure time, actual departure time information into one of these intervals right.

So, if right if it has an actual departure time which lies between 0 and 6 am it would be given the value 0 to 6 a particular flight has the actual departure time and between 6 and 12 between 6 am and 12 pm then it would be given this particular this particular class this particular category and then similarly for other flights as well. So, this is what we are trying to do here.

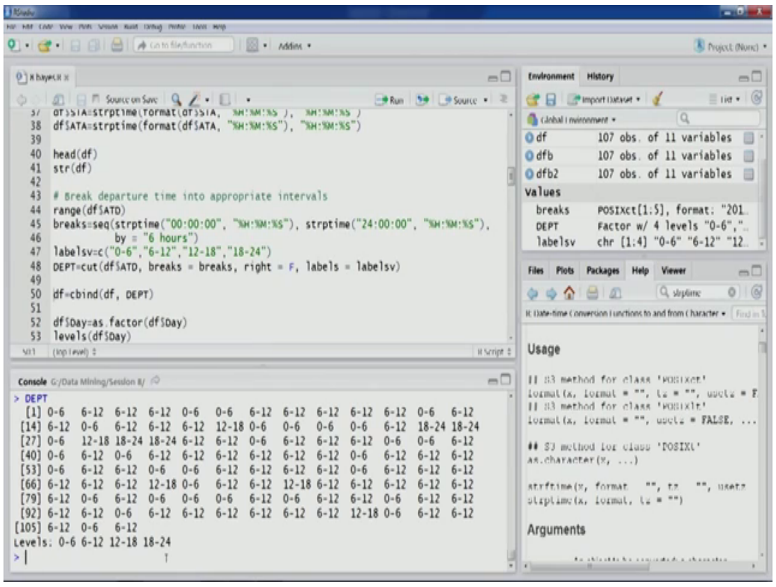
So let us execute this code you can see breaks this these this particular variable has been created labels and we can use the cut function and use the breaks information that we have just created and the first argument in the cut function is going to be the variable which we want to cut which we want to bin.

So, we want to bin actual departure time into 4 bins then we have already defined labels name are appropriately defined and as per the breaks it is going to be different observation is going to be binned. Now there is one more argument that you would see that right as false so this would indicate whether the weather in a range the right side value is going to be counted or not so because this is false. So, right value is not going to be counted.

So, the range is actually going to be open on the first value of the these interval and being closed being open or open on the open on the last value of the interval and closed on this particular values it will take the value it can take the value0, but not 6 6 will come here in the in the 6 to 12 category.

Similarly 12 will come here in the 12 to 18 in category and similarly flight at departuring at 18 would come into this category. So, in this fashion we would be able to create this bin this particular variable. So, you can see in the environment section the departure this function has been created you are interested in the actual values you can see.

(Refer Slide Time: 23:10)



```
df$STA=strptime(format(df$STA, "%H:%M:%S"), "%H:%M:%S")
df$ATA=strptime(format(df$ATA, "%H:%M:%S"), "%H:%M:%S")

head(df)
str(df)

# Break departure time into appropriate intervals
range(df$ATD)
breaks=seq(strptime("00:00:00", "%H:%M:%S"), strptime("24:00:00", "%H:%M:%S"),
           by = "6 hours")
labelsv=c("0-6","6-12","12-18","18-24")
DEPT=cut(df$ATD, breaks = breaks, right = F, labels = labelsv)

df=cbind(df, DEPT)

df$Day=as.factor(df$Day)
levels(df$Day)
```

```
> DEPT
  [1] 0-6   6-12  6-12  6-12  0-6   0-6   6-12  6-12  6-12  6-12  6-12  0-6   6-12
 [14] 6-12  0-6   6-12  6-12  6-12  12-18 0-6   0-6   0-6   0-6   6-12  18-24 18-24
 [27] 0-6   12-18 18-24 18-24 6-12  6-12  0-6   6-12  6-12  6-12  0-6   0-6   6-12
 [40] 0-6   6-12  0-6   6-12  6-12  6-12  6-12  6-12  6-12  0-6   6-12  6-12  6-12
 [53] 0-6   6-12  6-12  0-6   0-6   6-12  6-12  6-12  6-12  0-6   6-12  6-12  6-12
 [66] 6-12  6-12  6-12  12-18 0-6   6-12  6-12  12-18 6-12  6-12  6-12  6-12  6-12
 [79] 6-12  0-6   6-12  0-6   0-6   0-6   6-12  0-6   6-12  6-12  0-6   6-12  6-12
 [92] 6-12  6-12  0-6   6-12  6-12  6-12  6-12  6-12  6-12  12-18 0-6   6-12  6-12
[105] 6-12  0-6   6-12
Levels: 0-6 6-12 12-18 18-24
>
```

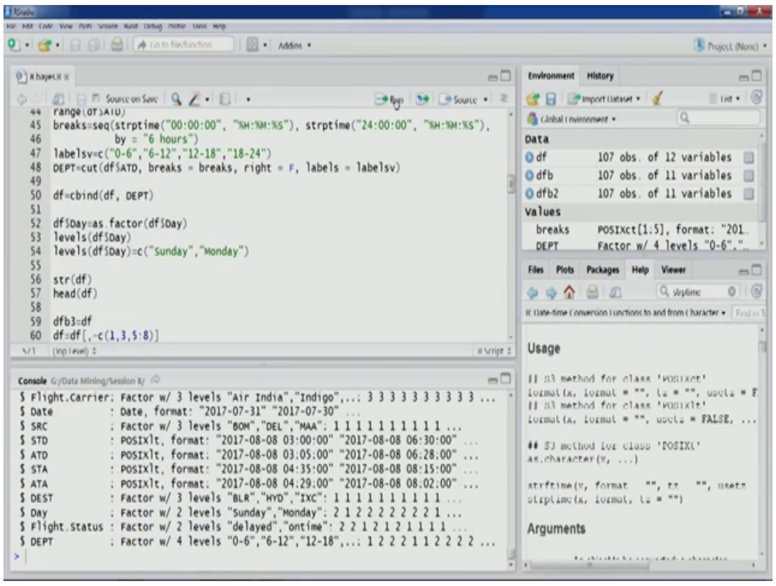
```
df       107 obs. of 11 variables
dfb      107 obs. of 11 variables
dfb2     107 obs. of 11 variables
Values
  breaks   POSIXct[1:5], format: "201…
  DEPT     Factor w/ 4 levels "0-6","…
  labelsv  chr [1:4] "0-6" "6-12" "12…
```

So for all the 107 values you know all the values they have been given a class or they have been assigned a bin to which they belong. From the output of this particular variable categorical variable that we have just created you can see the labels as well.

So, using the cut function by default we are creating a factor variable you can see the labels have also been appropriately specified in this. So, once this particular departure time the actual departure time once we have doing this particular variable we can add this into our existing data frame and as we talked about the day variable that we had let us also look at more details on this day and you can see this is numerical variable even though this is categorical stored stored as numerical here.

So, we would like to convert it into factor or categorical variable using as dot factor function. So, let us execute this code now let us look at the labels of this we can see labels are 1 and 2. So, to bring more clarity to these labels will change the layer and labels name as Sunday and Monday because one is representing Sunday and 2 representing Monday.

(Refer Slide Time: 24:41)



```
range(df$ATD)
breaks=seq(strptime("00:00:00", "%H:%M:%S"), strptime("24:00:00", "%H:%M:%S"),
           by = "6 hours")
labelsv=c("0-6","6-12","12-18","18-24")
DEPT=cut(df$ATD, breaks = breaks, right = F, labels = labelsv)

df=cbind(df, DEPT)

df$Day=as.factor(df$Day)
levels(df$Day)
levels(df$Day)=c("Sunday","Monday")

str(df)
head(df)

dfb3=df
df=df[,-c(1,3,5:8)]
```

```
$ Flight.Carrier: Factor w/ 3 levels "Air India","Indigo",..: 3 3 3 3 3 3 3 3 3 3 ...
$ Date          : Date, format: "2017-07-31" "2017-07-30" ...
$ SRC           : Factor w/ 3 levels "BOM","DEL","MAA": 1 1 1 1 1 1 1 1 1 1 ...
$ STD           : POSIXlt, format: "2017-08-08 03:00:00" "2017-08-08 06:30:00" ...
$ ATD           : POSIXlt, format: "2017-08-08 03:05:00" "2017-08-08 06:28:00" ...
$ STA           : POSIXlt, format: "2017-08-08 04:35:00" "2017-08-08 08:15:00" ...
$ ATA           : POSIXlt, format: "2017-08-08 04:29:00" "2017-08-08 08:02:00" ...
$ DEST          : Factor w/ 3 levels "BLR","HYD","IXC": 1 1 1 1 1 1 1 1 1 1 ...
$ Day           : Factor w/ 2 levels "Sunday","Monday": 2 1 2 2 2 2 2 2 2 1 ...
$ Flight.Status : Factor w/ 2 levels "delayed","ontime": 2 2 1 2 1 2 1 1 1 1 ...
$ DEPT          : Factor w/ 4 levels "0-6","6-12","12-18",..: 1 2 2 2 1 1 2 2 2 2 ...
>
```

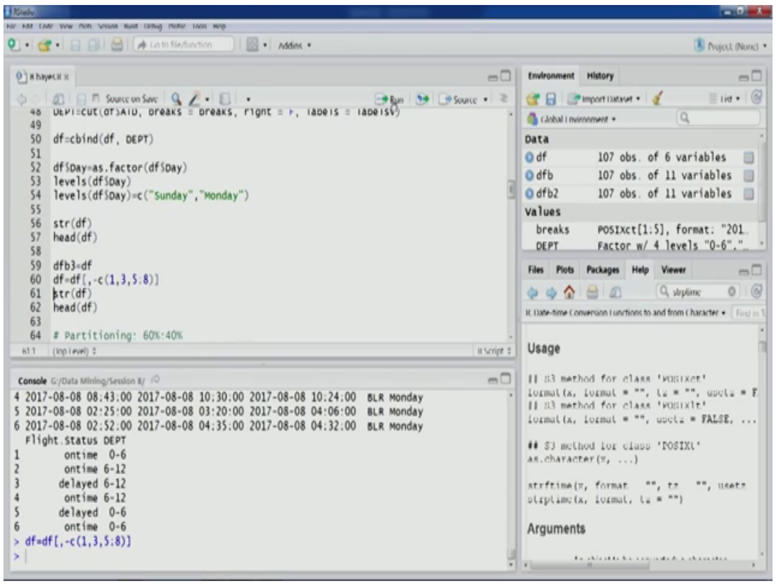
```
Data
df      107 obs. of 12 variables
dfb     107 obs. of 11 variables
dfb2    107 obs. of 11 variables
Values
breaks  POSIXct[1:5], format: "201…
DEPT    Factor w/ 4 levels "0-6","…
```

So, let us change the labels as well so once this is done we can have a look at the again the structure. So, you can see now the day has also been appropriately being mentioned here in the output Sunday, Monday here and the destination also now you can also see the another variable departure DEPT with 4 levels.

So, now we have been able to create the variables have the variables in the format that we want. Let us look at the first of the observation as well one more variable as you can see DEPT and they also you can see it has changed has now become Monday and Sunday.

Because we have changed the labels also this has also changed now certain variables in this we are not interested so we will get rid off them for example, flight number this are not going to use the date column this also we are not going to use now the departure and arrival times and different columns that we have that also we do not require the 5 to 8 the 4 columns that we have. So, we are not interested in that we have appropriately transformed the created the required variables.

(Refer Slide Time: 25:57)



```
DEPT=cut(df$ATD, breaks = breaks, right = F, labels = labels)

df=cbind(df, DEPT)

df$Day=as.factor(df$Day)
levels(df$Day)
levels(df$Day)=c("Sunday","Monday")

str(df)
head(df)

dfb3=df
df=df[,-c(1,3,5:8)]
str(df)
head(df)

# Partitioning: 60%:40%
```

```
4 2017-08-08 08:43:00 2017-08-08 10:30:00 2017-08-08 10:24:00  BLR Monday
5 2017-08-08 02:25:00 2017-08-08 03:20:00 2017-08-08 04:06:00  BLR Monday
6 2017-08-08 02:52:00 2017-08-08 04:35:00 2017-08-08 04:32:00  BLR Monday
  Flight.Status DEPT
1        ontime  0-6
2        ontime 6-12
3       delayed 6-12
4        ontime 6-12
5       delayed  0-6
6        ontime  0-6
> df=df[,-c(1,3,5:8)]
>
```

```
Data
df      107 obs. of 6 variables
dfb     107 obs. of 11 variables
dfb2    107 obs. of 11 variables
Values
breaks  POSIXct[1:5], format: "201…
DEPT    Factor w/ 4 levels "0-6","…
```

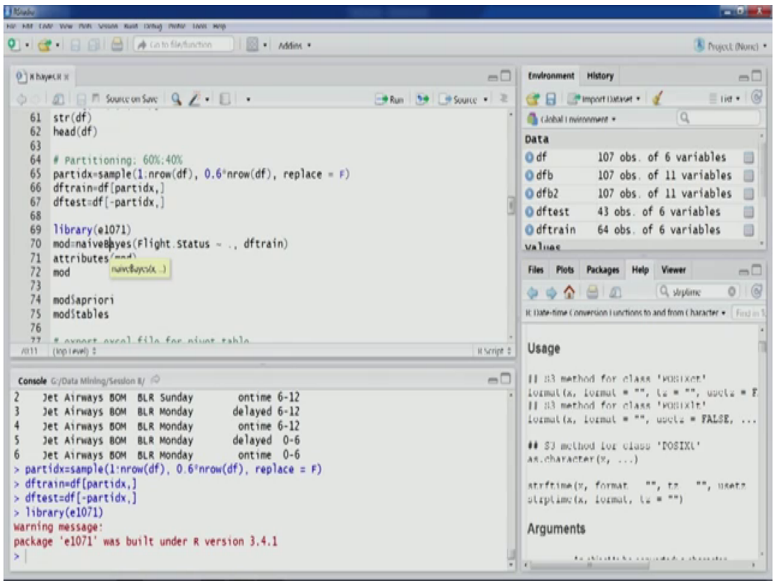
```
Usage

## S3 method for class 'POSIXct'
format(x, format = "", tz = "", usetz = F
## S3 method for class 'POSIXlt'
format(x, format = "", usetz = FALSE, ...

## S3 method for class 'POSIXt'
as.character(x, ...)

strftime(x, format = "", tz = "", usetz
strptime(x, format, tz = "")

Arguments
```

So, let us get rid of those columns now let us look at the variables that we are interested in now we are reduced to a data set of 5 variables, data frame of 5 variables, 6 variables, 5 predictors, and one the outcome variable of interest is right status and the other variables you can see first one fight carrier now 3 levels and then the source then we have destination 3 3 air force each then we have day that is Sunday or Monday. Then we flight status at the outcome variable and then we have departure that is different time intervals or part for a particular flight you know the time interval for the departure.

So, with this we are reading for ready for our modelling. So, we have appropriately perform over data processing. So, you can see first 6 observation also so these are the observations now we are ready for our modelling exercise. So, what we are going to do here is we will first start with the partitioning of this particular data set. So, we will keep 60 percent of the observations in the training partition and 40 percent of the observation in the test partition.

So, let us do the partitioning using this sample function. Let us create the training partition then test partition. Now the library that we require the package that we require to perform Naive Bayes modelling in R is e 10 7. So, let us load this particular let us load this particular library.

(Refer Slide Time: 27:45)



```
str(df)
head(df)

# Partitioning: 60%:40%
partidx=sample(1:nrow(df), 0.6*nrow(df), replace = F)
dftrain=df[partidx,]
dftest=df[-partidx,]

library(e1071)
mod=naiveBayes(Flight.Status ~ ., dftrain)
attributes(mod)
mod

mod$apriori
mod$tables
```

```
2    Jet Airways BOM  BLR Sunday      ontime 6-12
3    Jet Airways BOM  BLR Monday     delayed 6-12
4    Jet Airways BOM  BLR Monday      ontime 6-12
5    Jet Airways BOM  BLR Monday     delayed  0-6
6    Jet Airways BOM  BLR Monday      ontime  0-6
> partidx=sample(1:nrow(df), 0.6*nrow(df), replace = F)
> dftrain=df[partidx,]
> dftest=df[-partidx,]
> library(e1071)
Warning message:
package 'e1071' was built under R version 3.4.1
>
```

```
Data
df        107 obs. of 6 variables
dfb       107 obs. of 11 variables
dfb2      107 obs. of 11 variables
dftest     43 obs. of 6 variables
dftrain    64 obs. of 6 variables
```

Now let us look at the Naive Bayes functions in the Naive Bayes function the first argument is going to be the formula that we have to express in this form as we have been doing in other functions in previous lectures. So, you can see if flight dot status is the outcome variable of interest here.

So, you can see appropriately specified then the tilde and dot that is representing that all other variables are going to be counted as predictors. Now as you have seen that data frame that we have now we have just the outcome variable and the predictors no other variable there.

So, we can use the dot here and the training partition that we have here is the day of train let us execute this code, let us look at the attributes of this particular variable mod. So, this is so these are the attributes classes Naive Bayes and this is a Naive Bayes object and so these are the attributes these are the other information that we have right labels call tables a priory.

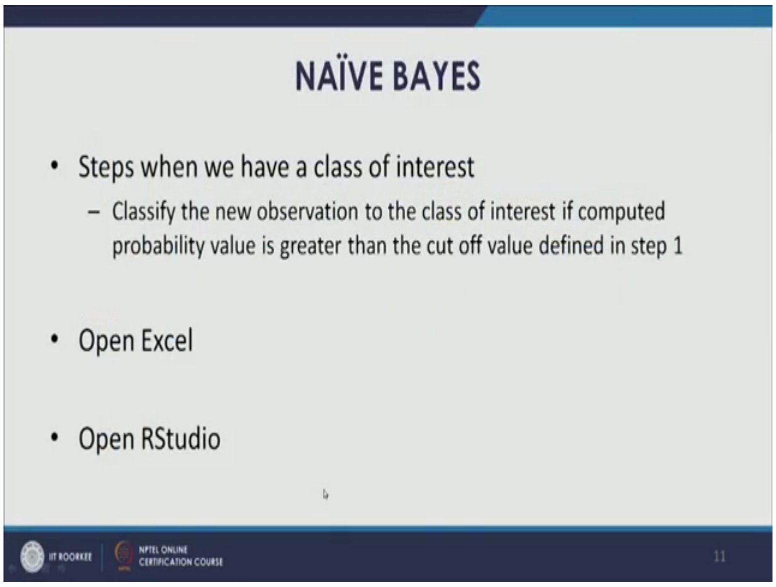
So, let us look at the output in more detail. So, first we have calls so that is the naive call that we made then we have prior probabilities. So, these are the actual proportion of records belonging to delayed class and the outcome variable or on time class in the outcome variable.

So, as you can see 37.5 percent of the records belong to the delayed class in the data set and in the 62.5 record belong to the on time class of outcome variables this is with respect to training partition and not the full data set that we have. Now after this we see a table on conditional probabilities as we can see here that why that is nothing, but flight status and flight dot carrier and for different labels we have the probabilities values right.

So, just like the exercise that we have done on using excel. So, here we have more than 1 or 2 predictors in this exercise and you can see for first predictor that is flight carrier and the outcome variable for different classes we have the probabilities values.

So, this would essentially mean that the Naive Bayes formula that we have talked about so these are the probabilities right. So, these are the probabilities values for each class and for each class of that predictor and given that whether they belong to the delayed or on time class about come variable so these probabilities values you can see here for other predictors as well.

(Refer Slide Time: 30:54)



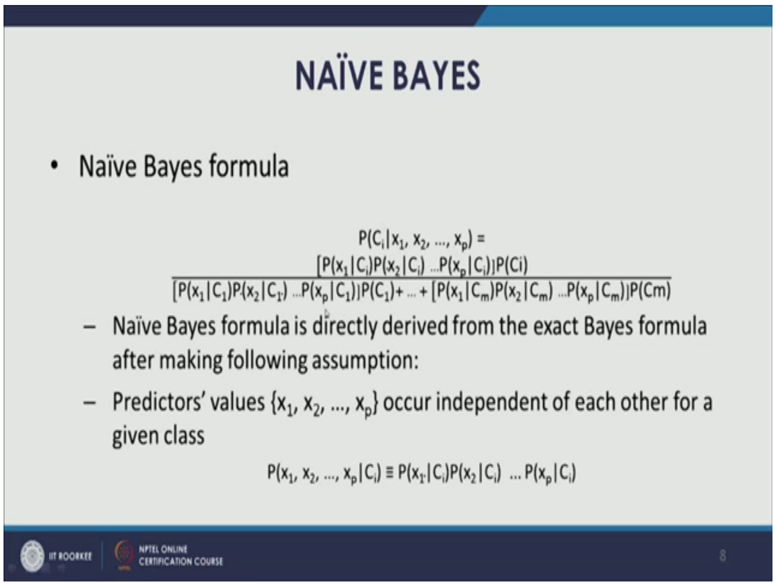
NAÏVE BAYES

- Steps when we have a class of interest
  - Classify the new observation to the class of interest if computed probability value is greater than the cut off value defined in step 1
- Open Excel
- Open RStudio

IIT ROORKEE | NPTEL ONLINE CERTIFICATION COURSE 11

If you want to have a relook at the Naive Bayes formula we can again go back to the formula you can see.

(Refer Slide Time: 30:59)



NAÏVE BAYES

- Naïve Bayes formula

$$P(C_i|x_1, x_2, ..., x_p) = \frac{[P(x_1|C_i)P(x_2|C_i) ... P(x_p|C_i)]P(Ci)}{[P(x_1|C_1)P(x_2|C_1) ... P(x_p|C_1)]P(C_1) + ... + [P(x_1|C_m)P(x_2|C_m) ... P(x_p|C_m)]P(Cm)}$$

  - Naïve Bayes formula is directly derived from the exact Bayes formula after making following assumption:
  - Predictors' values $\{x_1, x_2, ..., x_p\}$ occur independent of each other for a given class

$$P(x_1, x_2, ..., x_p|C_i) \equiv P(x_1|C_i)P(x_2|C_i) \; ... P(x_p|C_i)$$

IIT ROORKEE
NPTEL ONLINE CERTIFICATION COURSE
8

So, these are the values we are interested in x 1 that is predictor given the that particular class that is Ci whether delayed or on time and the probability of belonging to probability of that particular value predictors value.

So, So these are so these predictors being categorical. So, they can take these values for example, SRC it can take 3 values so for each of these values with respect to different class in the outcome variable the variable value. So, we have so this table actually has the all the information that we required to compute these Naive Bayes probabilities and then the numerator or denominator or the actual Naive Bayes probability.

So, the attributes that we saw if we you want to access these attributes you know one by one this is all we can do. So, these are the a priori one the first has give you the other one tables that we have already gone through.

So, with this we will stop here and we will continue our discussion we will continue perform doing our modelling using the same data set in next lecture.

Thank you.