

**Essentials of Data Science with R Software - 2**  
**Sampling Theory and Linear Regression Analysis**  
**Prof. Shalabh**  
**Department of Mathematics and Statistics**  
**Indian Institute of Technology Kanpur**

**Sampling Theory with R Software**  
**Lecture - 35**  
**Bootstrap Methodology**  
**Bootstrap Analysis Using boot Package in R**

Hello friends, welcome to the course Essentials of Data Science with R Software - 2, where we are trying to learn the topics of Sampling Theory and Linear Regression Analysis. In this module on the Sampling Theory with R Software we are going to continue with the topic on Bootstrap Methodology with R Software.

So, in the last lectures lecture we had considered the aspect of bootstrap methodology and I had given you a couple of examples, that how you can very easily compute different statistical function using the bootstrap methodology. And particularly in those situation, where the mathematics becomes too complicated to get the finite sample properties or the exact form of the for example, standard error or confidence interval etc., so, there good strap methodology can help us a lot.

So, up to now we have considered the estimation of estimator mean the value of the estimator, its bias and standard error. So, first let me explain you today that how these things can be implemented in the R software and after that I will try to explain you how we can construct the confidence interval using the bootstrap methodology and after that I will give you how to compute them in the R software, ok. So, now let us begin our discussion on how to use the bootstrap methodology in R software, ok.

(Refer Slide Time: 01:55)

**Bootstrapping using R with "boot" package:**  
**Description**

We would like to draw samples by bootstrap resampling.  
First we need to install and load the **boot** package.

```
install.packages("boot")  
library(boot)
```

Generate R bootstrap replicates of a statistic applied to data.

Both parametric and nonparametric resampling are possible.

For more details, see help on boot package, use help(boot).

2

So, in order to use the bootstrapping methodology first we need to install and load a package, this package is boot b o o t. So, this name is coming from bootstrapping and first you need to install the package using the command install dot packages and within the parenthesis in double quotes you have to write b o o t and then you have to use the command library and inside the parenthesis you have to write boot to load the package in the R console.

Now, these packages helps us in generating the R bootstrap replicates of a statistic which I want to entertain and this statistics is applied to the generated data and various types of quantities are estimated, right. And when we are trying to use this boot package then both parametric and non parametric resampling methods are possible.

Means, I would like to say confess one thing that well bootstrapping is a very vast area and many many developments have been made from the parametric statistical inference from the nonparametric statistical inference point of view. But, in this lecture I am trying to concentrate on the only on that much which can boost your confidence and make you more confident so, that you can use it, right.

And I will try to give you here the brief details of the important aspect, but I would request you that you please like try to look into the help on the boot package using the

command help and within the parenthesis boot and this will give you more details actually, ok.

(Refer Slide Time: 03:52)

```
Bootstrapping using R with "boot" package:  
Usage  
boot(data, statistic, R, sim = "ordinary",  
stype = c("i", "f", "w"), strata = rep(1,n), L  
= NULL, m = 0, weights = NULL, ran.gen =  
function(d, p) d, mle = NULL, simple = FALSE,  
..., )  
Arguments  
data  
The data as a vector, matrix or data frame. If it is a matrix or data  
frame then each row is considered as one multivariate  
observation.
```

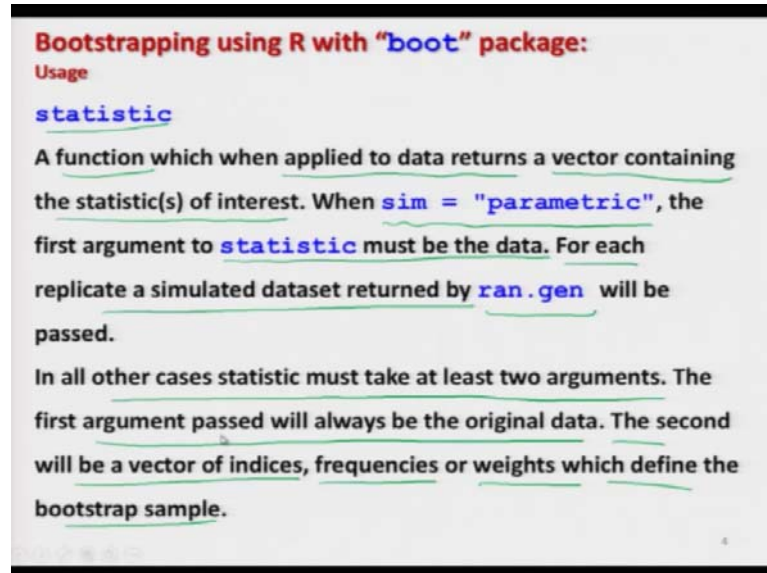
So, now, when you want to use this boot package the command here is boot and inside the parenthesis you have to there are many many options which can be used. So, first important aspect is data. So, here you need to give the data and this data has to be in the form of a vector matrices or a data frame, right and in case if this data set is in the form of a matrix or a data frame then each row is considered as one multivariate observation.

That means for example, if that is the patient number 1 then the first column is suppose height, second column is weight, third column is age, so, all the 3 variables will be considered, when you try to give the data in the form of matrix or data frame. After this the second option is statistic. So, statistics here you have to give the details that what really you want to compute. Suppose in the earlier examples well I had shown you how to compute the coefficient of variation.

So, here you need to write a small function which can compute the required statistics. So, here you need to write down a small function, and then here the third value here is R, R is trying to R is the number of bootstrap samples which are required to compute the required statistical inference values, right, and after that there are some other command

like sim, stype, strata, weights, d, mle etc. So, I try to give you a quick review of this thing.

(Refer Slide Time: 05:45)



But again once means I would request you please try to go into the help and try to read it. And one thing I can share with you my own experience that whenever you are trying to use this packages, there are many options and it is practically very difficult to keep all the things in your mind.

So, what I do personally that yes, I have to remember that what is the command or packages for the boot, but once, but whenever I am working I always first go to the help menu and try to see what are the different inputs and what are the different requirements because, if you violate those conditions or requirement the output will be something else and that may not be the same what you wanted to have.

So, always make a rule that before you start try to go to the help menu because, that is the most authenticated say information and that is the biggest advantage of R software also that this information has been provided by the person who has created the package. So, that is the best person who can give you the details, right ok.

So, now first I give you an idea about the statistics. So, as I said `statistic` has to be given in the form of a function and the use of this when this `statistic` is applied to the data, then it will return a vector containing the function or the formula of and when along with this

statistics if I try to use one more option `sim` is equal to `parametric`; that means, we are trying to do the parametric bootstrap then the first argument to `statistics` must be the data that may first you give to try to give the data ok.

And when we try to execute it then for each replicate a simulated data set is returned by the `ran dot gen`. What is this thing? I will try to show you, and if you are not using the parametric bootstrapping then in all other cases `statistics` must take at least two arguments.

The first argument passed will always be the original data and the second argument will be the vector of indices, frequencies or weights which define the bootstrap sample. Well, what is the interpretation of this thing? What is the meaning of this statement? That will get cleared in a couple of minutes when I try to take an example, right.

(Refer Slide Time: 08:17)

**Bootstrapping using R with "boot" package:**  
Usage

**R** The number of bootstrap replicates. Usually this will be a single positive integer.  $B = R$

**sim** A character string indicating the type of simulation required. Possible values are "ordinary" (the default), "parametric", "balanced", "permutation", or "antithetic".

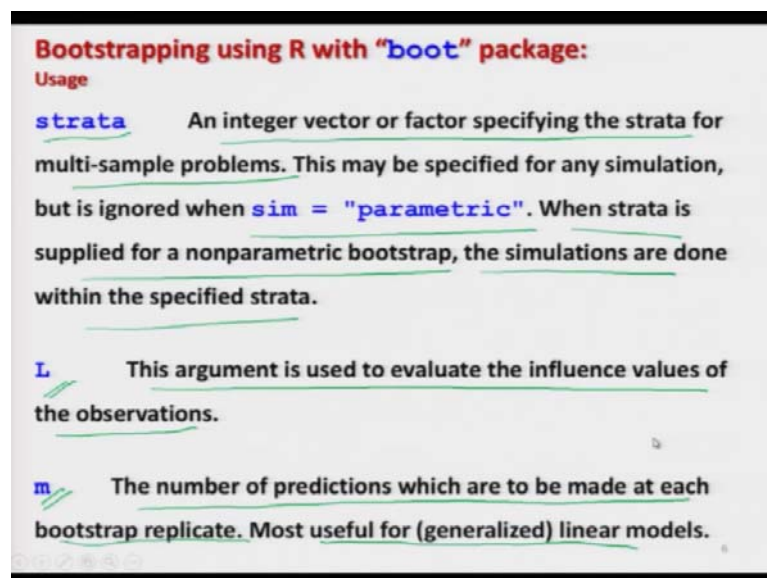
**stype** A character string indicating what the second argument of `statistic` represents. Possible values of `stype` are "i" (indices - the default), "f" (frequencies), or "w" (weights). Not used for `sim = "parametric"`.

And as I said `R` is here the number of bootstrap replicates. Usually that is a single positive integer. Obviously, means you mean you would like to generate capital as we had discussed we are generating the capital `B` number of bootstrap samples, right. So, `B` here is denoted by `R` and then there is an option here `sim`, this is a character string and this indicates the type of simulation what we want? The possible values are ordinary this is the default what we have actually done, right.

Then there is parametric, then there is balanced, then there is permutation, then there is antithetic and so on. Well I am not going into those details because, I have not covered it in the theory part and similarly there is another option here `stype`; `stype` is a character string which indicates that what the second argument of statistics represent and the possible values of this options are say `i` that is the default and it indicates the indices.

And similarly we have another option here `f` which is indicating the frequencies, and then we have `w` which is indicating the weights and this is not used when we are trying to use the option `sim` is equal to `parametric`.

(Refer Slide Time: 09:45)



And similarly we have one more option here is `strata` that is an integer vector or a factor that specifies the strata for multi sample problem. And this may be specified when you are trying to conduct any simulation, but it is ignored when we are trying to use the parametric option for the same, right, and when we are trying to create the strata, which is supplied for a nonparametric bootstrap then the simulations are done within the specified strata, right.

Then, we have another here argument say here capital `L` and this argument is used to evaluate the influence values of the observations. Well, we have not discussed this thing, but I am simply telling you and if you want to learn more detail that should be actually

your objective I would request you that you try to go to the book of say Bradley Efron that I shown you in the last lecture.

Then, we have another option here `m` and which is the number of prediction which are to be made at each bootstrap replicate, actually this option is more useful when we are trying to consider the bootstrapping in the generalized linear models, ok.

(Refer Slide Time: 11:03)

**Bootstrapping using R with "boot" package:**  
Usage

`ran.gen` This function is used only when `sim="parametric"` when it describes how random values are to be generated. It should be a function of two arguments. The first argument should be the observed data and the second argument consists of any other information needed.

`mle` The second argument to be passed to `ran.gen`. Typically these will be maximum likelihood estimates of the parameters.

`simple` <sup>TRUE</sup> logical, only allowed to be **TRUE** for `sim="ordinary"`, <sup>FALSE</sup>  
`stype = "i", n = 0` (otherwise ignored with a warning). By default a `n` by `R` index array is created.  $n \times R$

Then we have here another option `ran dot gen` and this function is used only when `sim` is equal to `parametric` option is used. And it describes how random values are to be generated? And this is a function of two arguments. The first argument should be the observed data and the second argument consists of any other information whatever is needed ok.

And then the next type is `mle`, this is the second argument which has to be passed here in the `ran dot gen` and actually this will be the maximum likelihood estimate of the parameters. And then, we have another option here `simple` which takes two possible values the logical operators `TRUE` and here `FALSE`.

And this is only allowed to be `TRUE` for `sim` is equal to `ordinary` `stype` is equal to `i` `n` is equal to `0` otherwise, it is ignored with a warning and you do not have to care for those things, right. And by this thing default and by `R` index array is created of order `n` by `R` ok.



(Refer Slide Time: 12:16)

**Bootstrapping using R with "boot" package:**  
**Value**  
The returned value is an object of class "boot", containing the following components:

- t0** The observed value of **statistic** applied to **data**.
- R** The value of **R** as passed to **boot**.
- t** A matrix with **sum(R)** rows each of which is a bootstrap replicate of the result of calling **statistic**.
- data** The **data** as passed to **boot**.
- seed** The value of **.Random.seed** when **boot** started work.

And once you try to execute this package then there are several options which can be called. How they can be called? I will try to explain you with an example. So, first value will be here say, for example, t0 this is actually the observed value of the statistics applied to the data.

For example, when you are trying to generate different bootstrap samples then in every sample you are trying to estimate the statistics of your interest and then you are trying to take the arithmetic mean of all those statistics which are computed in each and every bootstrap replicate and this is actually the bootstrap estimator of the parameter.

So, this t0 is indicating the value of the bootstrap estimator of the parameter theta which we want to estimate, then we can also know about this R which is the value of the R which is passed to the boot that is the that how many bootstrap samples have been generated to compute the value of t0. And similarly here, there is another option here t which is a matrix with sum of actually R rows and each of the row is a bootstrap replicate of the results which have been obtained in statistics in this statistics function.

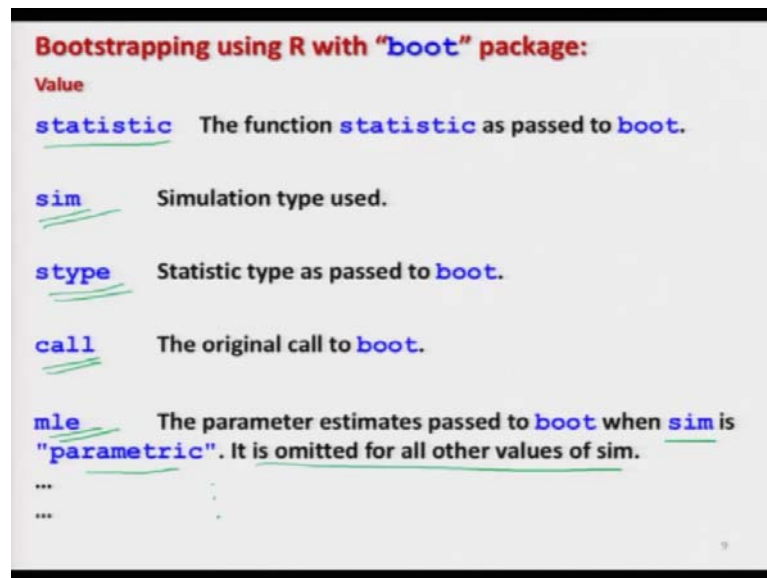
Then there is another option here data that will give you that what data has been passed to the boot function, and similarly here there is here seed this is the value of the random seed this is the value which was used to generate the bootstrap sample. So, as I had discussed earlier in a lecture, that what is the meaning of the seed so, this value will give



you that what is the value of the seed from which the random observations were started to be generated.

So, if you want to repeat the same experiment then you can fix this value and every time you repeat the function you will get the same outcome, otherwise what will happen? Every time you generate a bootstrap sample the sample is going to be different and it will give you a entirely different value ok.

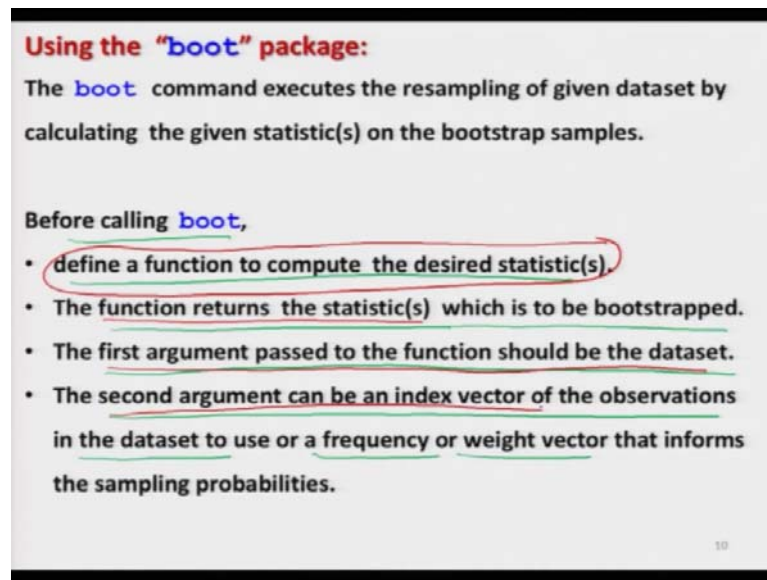
(Refer Slide Time: 14:52)



So, next as we have discussed statistics means, if you want to know this is the function which is passed to the boot and sim will give you an idea that what type of simulation has been used? S type will give you an idea that what is the statistics type? That has been passed to the boot function and call here is another option that is the original call to the boot and then there is here mle.

So, this will give you the parameter estimate, which have been passed to the function boot when sim is chosen to be parametric and it is omitted for all other values of sim, right. So, and then there are different types of values I am not going into that detail, but I will request you that you please try to go to the help and you will see at the end of the help the authors usually take couple of examples to illustrate how the things are working.

(Refer Slide Time: 15:44)



**Using the "boot" package:**

The `boot` command executes the resampling of given dataset by calculating the given statistic(s) on the bootstrap samples.

Before calling `boot`,

- define a function to compute the desired statistic(s).
- The function returns the statistic(s) which is to be bootstrapped.
- The first argument passed to the function should be the dataset.
- The second argument can be an index vector of the observations in the dataset to use or a frequency or weight vector that informs the sampling probabilities.

10

Now, I come to the main part that how you are going to use this boot package and whatever commands I have used how they are going to be implemented. Well, I am taking here a simple example.

My objective is to explain you that how you can start, after that how long you will go, how long you want to go, how many things you want to do, how you want to improve it further, how you want to say implement the latest methodology, that depends only on you that is your job, ok.

So, the one basic difference in using the boot package and all other package is that, that you have to write the a small function or the function which will compute the value of the statistics that you want to use. For example, if you want to find out the, say standard error of coefficient of variation. So, your statistics is coefficient of variation and you need to write a small program how you can compute the coefficient of variation.

Sometime there will be inbuilt package to compute the function and sometime you will have to write it. So, before you start the first step is this you have to define a function to compute the desired statistic. And this function has to be written in such a way such that this function returns the statistics which is to be bootstrapped ok.

And the first argument in this function passed to the function should be the data set, that which is the data set which you want to bootstrap or from where you want to draw the

sample by SRSWR. And the second argument can be an index vector of the observation in the dataset to be used or this can be a frequency or a weight vector that inform the sampling probabilities.

(Refer Slide Time: 18:03)

**Using the "boot" package: Example**

Suppose the marks of students depend upon number of hours per week of study.

Suppose we want to estimate the correlation coefficient between the marks of students and number of hours per week of study by bootstrap.

*Corr.*

x	y	x <sub>1</sub>	x <sub>2</sub>	...	x <sub>n</sub>
		y <sub>1</sub>	y <sub>2</sub>	...	y <sub>n</sub>

*measure of degree of linear relationship between x and y*

*0 ≤ r ≤ 1*

$$r = \frac{\text{Cov}(x, y)}{\text{sd}(x) \cdot \text{sd}(y)}$$

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

So, now let me take a simple example, suppose we want to find out correlation coefficient, right. So, we know that if I have got here 2 random variables say x y on which you have got data x<sub>1</sub>, x<sub>2</sub>,..., x<sub>n</sub> y<sub>1</sub>, y<sub>2</sub>,...,y<sub>n</sub>; then the correlation coefficient is defined as covariance between x and y upon say standard deviation of x into standard deviation of y.

Or this is actually  $\frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$ . This is your here r and r this is a measure

of degree of linear relationship between x and y values, and this r lies between 0 and 1. So, that we know.

So, suppose I take an example that the marks of students at any exam they depend upon how many hours the student has studied. So, we have some data set in which the marks of the students have been obtained and the corresponding number of hours of study per week by the student they are also obtained and suppose we want to find out the

correlation coefficient between marks and the number of hours this is our objective, right.

So, this is what we want that we want to estimate the correlation coefficient between the marks of the student and the number of hours per week of study using the bootstrap. Why boot step? Because, if you ask me what is the variance of correlation coefficient because, correlation coefficient is also a statistics, this is a function of sample value, this is the; so  $r$  so, the correlation coefficient itself is a random variable and that is statistics.

So, this will also have some variance or standard error, but we do not know the exact form of the variance or standard error. So, we would like to compute the say this bias of correlation coefficient, the value of correlation coefficient as well as the estimate of variance of correlation coefficient that is the standard error.

(Refer Slide Time: 20:36)

Using the "boot" package:

**Example**

Observations on 20 students are collected

Let

$y$ : Marks of students (Max. marks: 250)

$X$ : Number of hours per week of study

*Corr(x,y)*

Student no.	y	X
1	180	34
2	116	12
3	118	15
4	139	33
5	195	31
6	152	24
7	218	40
8	170	31
9	179	21
10	210	37
11	178	29
12	104	15
13	145	17
14	203	38
15	163	17
16	216	36
17	106	13
18	216	39
19	191	36
20	197	34

So, now this is our data set. So, you can see here I have here 20 students and the marks of the student they have been denoted by  $y$  and the maximum marks are 250 and capital  $X$  is denoting the number of hours per week. So, it is something like this. Student number 1 has got 180 marks out of 250 and the student has studied 34 hours in a week.

And similarly, the 2nd student has got 116 marks and the student has studied for 12 hours in a week and so on. So, this data is for 20 student, and we want to now compute the correlation coefficient between  $X$  and  $y$  using the bootstrap.

(Refer Slide Time: 21:26)

```
Using the "boot" package: Example

Entering the data
> studentno=c(1:20)

> marks=c(180,116,118,139,195,152,218,170,179,
210,178,104,145,203,163,216,106,216,191,197)

> hours=c(34,12,15,33,31,24,40,31,21,37,29,15,
17,38,17,36,13,39,36,34)

> student_marks_hours_data =
data.frame(studentno, marks, hours)
```

So, first I try to do I have to create my data sets this table has to be converted into a data frame. So, I try to create the 3 variables student number from 1 to 20 and I have entered these marks manually here I have typed it. So, the marks are given in the data vector called as marks and the number of hours per week of study they are indicated by the variable hours.

And all these 3 variables they have been combined in a data frame using the command data dot frame and this data frame has been stored in a with the name student underscore marks, underscore hours, underscore data, right. So, you have to just keep in mind this example because, I will be using it in the case of a confidence interval also.

(Refer Slide Time: 22:20)

### Using the "boot" package: Example

```
> studentno=c(1:20)
> marks=c(180,116,118,139,195,152,218,170,179,210,178,104,145,203,163,216,106,216,191,197)
> hours=c(34,12,15,33,31,24,40,31,21,37,29,15,17,38,17,36,13,39,36,34)
> student_marks_hours_data=data.frame(studentno,marks,hours)
> @student_marks_hours_data
  studentno marks hours
1          1  180   34
2          2  116   12
3          3  118   15
4          4  139   33
5          5  195   31
6          6  152   24
7          7  218   40
8          8  170   31
9          9  179   21
10         10  210   37
11         11  178   29
12         12  104   15
13         13  145   17
14         14  203   38
15         15  163   17
16         16  216   36
17         17  106   13
18         18  216   39
19         19  191   36
20         20  197   34
```

So, you can see here this is the screenshot, I will try to show you later, but you can see here that this is the data frame that we have obtained ok.

(Refer Slide Time: 22:30)

### Using the "boot" package: Example

First, define a function to compute the desired statistic which is the correlation coefficient between marks and hours using `cor()`.

The following function returns the correlation coefficient which is to be bootstrap for the given data on marks and hours -

```
return(cor(d2$marks, d2$hours)).
```

```
> marks hours <- function(d, i) {
  d2 <- d[i,]
  return(cor(d2$marks, d2$hours))
}
```

Execution of bootstrap

```
> corrboot <- boot(data = student_marks_hours_data,
  statistic = marks_hours, R = 500)
```

Now, let me explain you how would you like to find out the bootstrap value of r, its bias as well as standard error, right, ok. So, as I said, before you do the bootstrapping you have to write a function which is going to compute the values of the desired statistic, right.

So, in this case we are interested in the correlation coefficients, so, we would like to define a function for the correlation coefficient between marks and hours and there is a built in command say `cor` in this `r` package to find out the correlation coefficient between two variables. Means, if you want to have more information on the `cor` function you can look into the help menu, but this is the built in command which is available in the base package itself.

So, I write the program here. First you try to look here. So, I am writing a program whose name is `marks_underscore_hours`, this is going to be a function which depends on two values `d` and here `i`. So, you can see here this is my 1st argument and this is my here 2nd argument.

So, the 2nd argument will supply the information on the unit that which unit has been used. And then I try to create it call it call the value of `d` and `i` using this command `d` inside the square bracket `i`. So, you know that what does this mean and whatever is the outcome this is stored in `d2`.

So, this is essentially going to read the data. And whatever the data has been stored in `d2` from this data set I try to call the data on marks and data on hours. So, this can be obtained by say `d2` dollar that is the name of the database and then dollar and then the name of the variable and similarly the hours are called by `d2` dollar hours.

And then I try to compute the correlation coefficient between these 2 variables marks and hours, and whatever is the value I want that value to come out of the function, so, I use here the command here `return`, right. So, this function will take the value from the data set and then it will compute the correlation coefficient and then it will return the value of the correlation coefficient for the given data set.

So, this is my function, this is what I said here. You can see here in this slide I had said that you first need to define a function to compute the desired statistics, and this function returns the value of the statistic which has to be bootstrap. So, I have used the command `return` and the first argument passed to the function should be the data set and the second argument can be an index vector.



So, this is exactly now, I have done here and I have shown you what do I mean by writing that slide in writing that slide. Now, after this I have to use the boot package. So, I try to use here boot and then the first argument is the data set, so, I am giving here an option data is equal to what?

I am asking that please read the data from the data frame whose name is student\_marks\_hours\_data, this is the same data set which you have created here you can see here, right. You can see here this is the name of the data frame in which you have stored the given data, ok.

Now, after this I am saying ok read that data for what? So, I am saying ok, my statistics is here marks underscore hours so that means, you have to use these statistics through the function which is given here, right. And you also have to specify that how many bootstrap samples you want to generate based on that the value of the correlation coefficient will be estimated. So, this I am giving by here R is equal to 500.

So, now what will happen? This boot command will use the data set which you have supplied here on the function which is which you have supplied here and it will try to generate R number of bootstrap samples and it will try to compute the value of the statistics which is given here under the statistics and it will supply it; and what I try to do? Because, I want to use this outcome at a later stage also to extract different type of information, so, I am storing the outcome of this boot command as corrboot.

So, you will also remember the name this is the correlation coefficient which has been found through boot package using bootstrap methodology.

(Refer Slide Time: 28:23)

```
Using the "boot" package: Example
> corboot

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = student_marks_hours_data,
      statistic = marks_hours, R = 500)

Bootstrap Statistics :
  original      bias    std. error
t1* 0.877904 -0.009871274 0.07328005

> cor(marks, hours) #True correlation
[1] 0.877904
```

Now, once you try to execute it here you will get an outcome like this one, right. So, it is now giving you that you have used here the ordinary non parametric bootstrap, this is the same thing what we have actually done, right, if you want to go with parametric and other things then you need to study them yourself they are not difficult at all.

And this is giving you that what function you have supplied, right. So, this is the same command which you have supplied to obtain the data stored under this function this outcome corboot. Now, here is your main result you can see here. So, it is trying to say first thing here is original. So, original means, here that if you simply try to use the entire data on x and y, which data? Which you have supplied here, we this is your here the original data, right, this is here the original data.

So, if you try to use this data and if you try to find out the correlation coefficient then this then the original value or the value of the correlation coefficient in the original sample will come out to be 0.877904 which is stored under t1., right, then, it is trying to give you here the value of bias that is the bootstrap bias of the correlation coefficient. So, you can see here it is like this minus 0.009871274.

And this third column is giving you the value of the standard error of the correlation. So, you can see here this is here like this, right. Now, just in order to verify whether this t1 statistics is giving us the value of the correlation coefficient based on the original data I

have computed here the correlation coefficient directly from the data; and you can see here this is coming out to be the same value as this value, right. So, this is how you can obtain the outcome.

Well, after this you can use different types of options which I explained you, but here I will keep myself limited to only the basic of this bootstrap methodology ok.

(Refer Slide Time: 30:56)

```

# R Console
> marks_hours <- function(d, i){
+   d2 <- d[i,]
+   return(cor(d2$marks, d2$hours))
+ }
>
> corrboot <- boot(data=student_marks_hours_data, statistic=marks_hours, R=500)
> corrboot

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = student_marks_hours_data, statistic = marks_hours,
      R = 500)

Bootstrap statistics:
  original      bias   std. error
t1* 0.877904 -0.009871274 0.07328005
>
> cor(marks_hours) #true correlation
[1] 0.877904
> |
  
```

So, this is here the outcome or screenshot of the outcome which I shown you and now, I try to get the details from this output.

(Refer Slide Time: 31:05)

```

Using the "boot" package: Example
Command summary provides the summary statistics of the
bootstrap parameters.

> summary(corrboot)
  Length Class      Mode
t0      1 -none-    numeric
t       500 -none-    numeric
R        1 -none-    numeric
data     3 data.frame list
seed    626 -none-    numeric
statistic 1 -none-    function
sim      1 -none-    character
call     4 -none-    call
stype    1 -none-    character
strata   20 -none-    numeric
weights  20 -none-    numeric
  
```

So, remember we have stored our outcome under the variable name `corrboot`. So, if you try to use here the command `summary` on this outcome that is `summary(corrboot)`, then you will get here this type of data. Well this data is not directly useful to us, but it is giving us the different types of parameters or different types of information which has been used in the computation of the correlation coefficient.

So, `t0` it has got here length 1 class is here none mode here is numeric and so, you can see here that, right, and this is here the seed value which is which has been used by default as 626. So, if you want to repeat it you can fix your seed value at 626 and you will get the same outcome which I have shown you here, right and then here are the different values of the option which have been by default passed on to the `boot` function, ok.

(Refer Slide Time: 32:13)

**Using the "boot" package: Example**  
 Using the `t` vector and `t0`, we can calculate the bootstrap estimator of correlation coefficient, its bias and standard error.

Bootstrap estimator  
`mean(corrboot$t)`  
`[1] 0.8680327`

Bootstrap bias  
`mean(corrboot$t) - corrboot$t0`  
`[1] -0.009871274`  
 which is obtained by  $0.8680327 - 0.877904 = -0.009871274$

Bootstrap standard error  
`sd(corrboot$t)`  
`[1] 0.07328005`

Handwritten notes:  
 $B = 500$   
 $\hat{\theta}_b = \lambda_1 \dots \lambda_{500}$   
 bootstrap estimator of corr. coeff.  
 $\hat{\theta}_b = \frac{1}{B-1} \sum_{b=1}^B (\lambda_b - \lambda_b)^2$

Now, I try to extract different type of information from this outcome. So, using the `t` vector and `t0` vector we can calculate the bootstrap estimator of the collision coefficient, its bias and standard error. How? Whatever is your here this outcome `corrboot` you try to use the command here say `corrboot` and you obtain the value of your `t` and try to join it with the dollar sign.

So, now if you try to find out the mean of this `corrboot` dollar `t` it will give you the value of the bootstrap estimator of correlation coefficient, this means here what? This means,

according to our theory you have generated 500 bootstrap samples, so, you will have here  $r_1, r_2, \dots, r_{500}$  values of correlation coefficient which have been computed in each of the bootstrap replication.

So, there are 500 samples, so, there are 500 values of correlation coefficient, and then we try to find out the mean of these values. And this is going to be your here, so, called here only earlier you had denoted by  $\hat{\theta}_b$  which is the bootstrap estimator of correlation coefficient. So, this will give you the bootstrap estimator of correlation coefficient and this is obtained here.

So you can see here. Now, if you want to really verify for example, you can see here this outcome is giving you here bias. So, if you really want to verify whether this result is correct or not I have done this thing here. For example, I am trying to compute the bootstrap bias manually. So, I am using here this value here the value of the bootstrap estimator minus the true value, right.

So, you can see here this is here 0.8680327 and what is here this thing? This thing you can see here in this outcome also it is giving you here the value the original value of the correlation coefficient or the correlation coefficient for the original sample. So, this is the original sample which has been treated as a population value and if you try to subtract it this is giving you this value and this is the same value which has been obtained here.

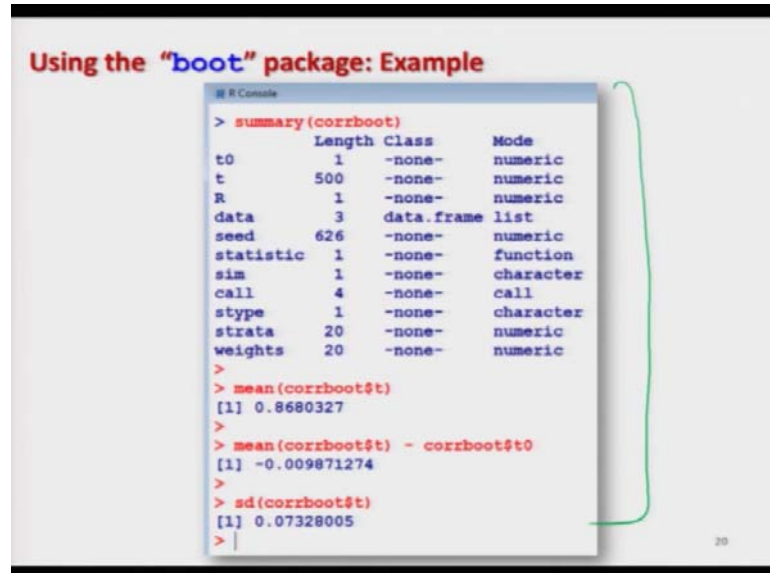
So, just by doing this thing you can verify that whatever theory you had developed earlier this is holding correct through this software also. And whatever are the values which you have obtained here the 500 values of the bootstrapped value of correlation

coefficient we are simply trying to find out the  $\frac{1}{B-1} \sum_{b=1}^B (r_b - \bar{r}_b)^2$ .

$\bar{r}_b$  is the bootstrap estimator something like  $\hat{\theta}_b^2$  and then you try to take here the positive square root, then this is going to give you the standard error this is the bootstrap standard error. So, this can be obtained by the command `sd corrboot dollar t`. So, this is here the standard deviation of the estimator correlation coefficient, and you can see here that this is the same value which has been given by the package also in the outcome. So, you can now believe that whatever you wanted that has been given by the software and if you

want to do some more manipulation now you know the theory and you can obtain these values separately, right.

(Refer Slide Time: 36:16)



Using the "boot" package: Example

```
> summary(corrboot)
      Length Class      Mode
t0         1 -none-    numeric
t          500 -none-    numeric
R           1 -none-    numeric
data        3 data.frame list
seed       626 -none-    numeric
statistic   1 -none-    function
sim         1 -none-    character
call        4 -none-    call
stype       1 -none-    character
strata      20 -none-    numeric
weights     20 -none-    numeric

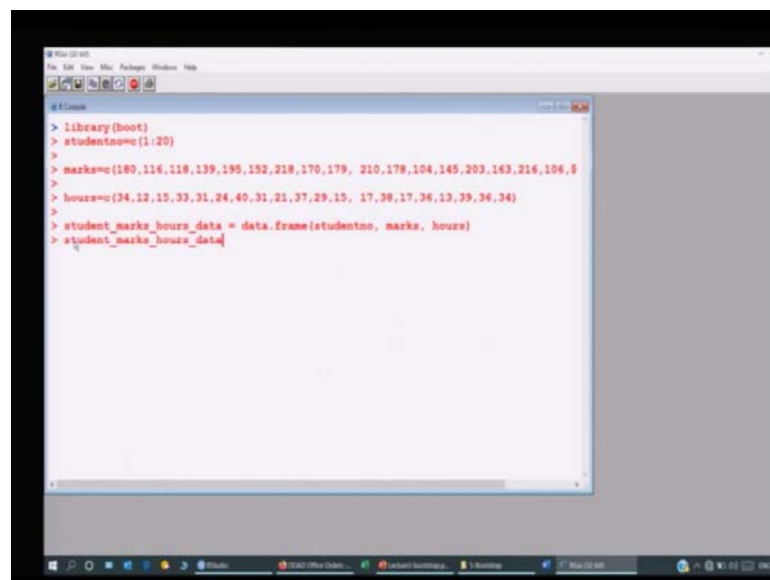
> mean(corrboot$t)
[1] 0.8680327

> mean(corrboot$t) - corrboot$t0
[1] -0.009871274

> sd(corrboot$t)
[1] 0.07328005
>
```

And here is the screenshot of whatever I have shown you on this say screen, right.

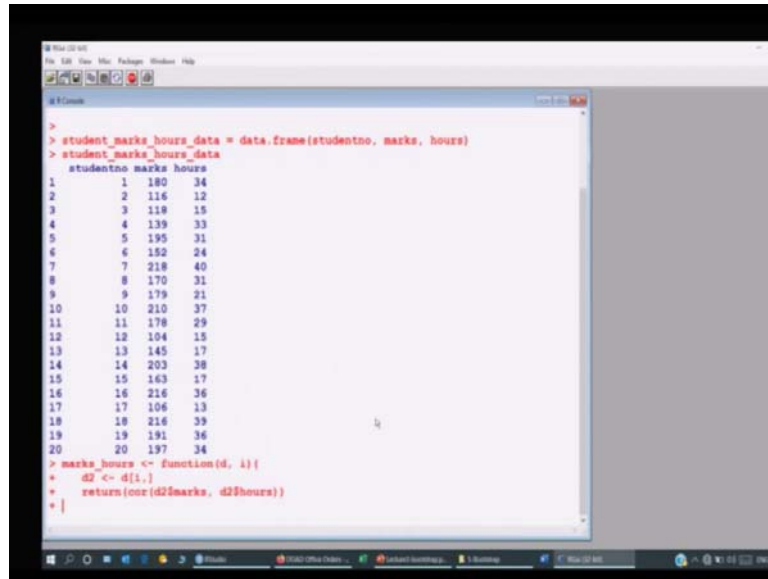
(Refer Slide Time: 36:27)



```
> library(boot)
> studentno=(1:20)
>
> marks=(180,116,118,139,195,152,218,170,179, 210,178,104,145,203,163,216,106,8)
>
> hours=(24,12,15,33,31,24,40,31,21,37,29,15, 17,38,17,36,13,39,36,34)
>
> student_marks_hours_data = data.frame(studentno, marks, hours)
> student_marks_hours_data
```

So, after this I come to R console. So, let me first load the library boot. I already had have installed this package on my laptop and so, I come here and I try to create this data frame and you can see here.

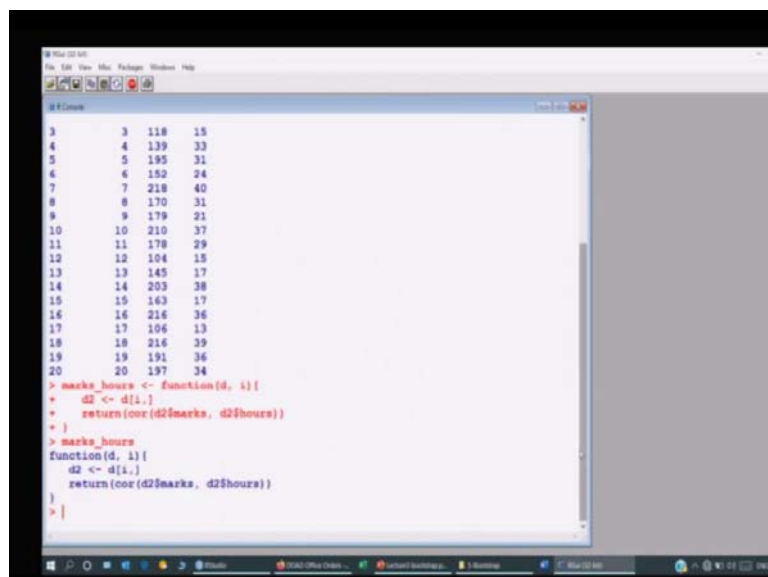
(Refer Slide Time: 36:52)



```
> student_marks_hours_data = data.frame(studentno, marks, hours)
> student_marks_hours_data
  studentno marks hours
1          1  180   34
2          2  116   12
3          3  118   15
4          4  139   33
5          5  195   31
6          6  152   24
7          7  218   40
8          8  170   31
9          9  179   21
10         10  210   37
11         11  178   29
12         12  104   15
13         13  145   17
14         14  203   38
15         15  143   17
16         16  216   36
17         17  106   13
18         18  216   39
19         19  191   36
20         20  197   34
> marks_hours <- function(d, i){
+   d2 <- d[i,]
+   return(corr(d2$marks, d2$hours))
+ }
```

So, now, you can see here this is the same data frame which I had created here. This is the student number and marks of the students and the number of hours of study. And after that, I try to come here and first I try to write down here this function. So, you need to first type this function and this function has to be present in the same directory, so, I try to you can see here.

(Refer Slide Time: 37:25)

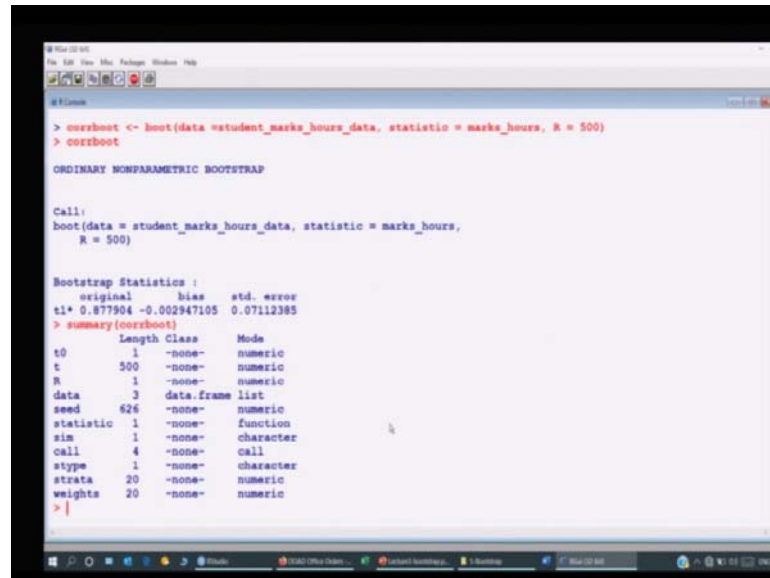


```
> marks_hours
  studentno marks hours
3          3  118   15
4          4  139   33
5          5  195   31
6          6  152   24
7          7  218   40
8          8  170   31
9          9  179   21
10         10  210   37
11         11  178   29
12         12  104   15
13         13  145   17
14         14  203   38
15         15  143   17
16         16  216   36
17         17  106   13
18         18  216   39
19         19  191   36
20         20  197   34
> marks_hours <- function(d, i){
+   d2 <- d[i,]
+   return(corr(d2$marks, d2$hours))
+ }
> marks_hours
function(d, i){
  d2 <- d[i,]
  return(corr(d2$marks, d2$hours))
}
>
```



So, this is here the function you can see that the structure marks hours, right. So, this is essentially trying to compute the statistics coordination coefficient which I want to bootstrap, and then I try to give here the outcome and I try to use the function here ok.

(Refer Slide Time: 37:45)



```
> corboot <- boot(data = student_marks_hours_data, statistic = marks_hours, R = 500)
> corboot

ORDINARY NONPARAMETRIC BOOTSTRAP

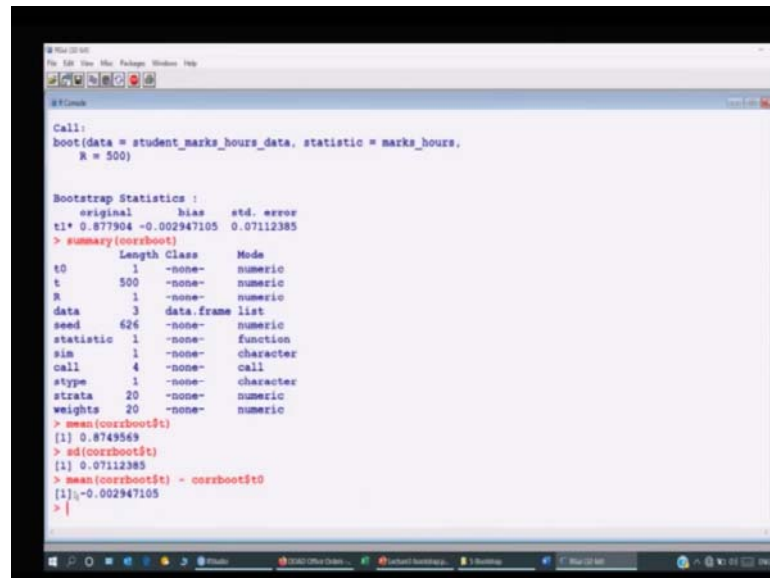
Call:
boot(data = student_marks_hours_data, statistic = marks_hours,
      R = 500)

Bootstrap Statistics :
  original    bias    std. error
t1* 0.877904 -0.002947105  0.07112385
> summary(corboot)
  Length Class      Mode
t0      1 -none-    numeric
t       500 -none-    numeric
R        1 -none-    numeric
data     3 data.frame list
seed    626 -none-    numeric
statistic 1 -none-    function
sim      1 -none-    character
call     4 -none-    call
stype    1 -none-    character
strata   20 -none-    numeric
weights  20 -none-    numeric
> |
```

I clear the screen. So, you can see here this is how I have executed the bootstrapping function and you can see here the outcome looks like this, corboot like this ok. So, you can see here this is the original value, bias, standard error etc.. And you can see this is different than what I had obtained in my slides, right, you can see here that it is different, right.

So, once I have obtained this thing then I would try to obtain the summary and mean of these things. So, if you go for see here summary of the outcome which is stored in corboot you can see here this is this gives you the details of the inputs which have been given to the boot function.

(Refer Slide Time: 38:58)



```
Call:
boot(data = student_marks_hours_data, statistic = marks_hours,
      R = 500)

Bootstrap Statistics :
      original      bias  std. error
t1* 0.877904 -0.002947105  0.07112385
> summary(corrboot$t)
      Length Class      Mode
t0         1 -none-    numeric
t          500 -none-    numeric
R           1 -none-    numeric
data       3 data.frame list
seed      626 -none-    numeric
statistic  1 -none-    function
sim        1 -none-    character
call       4 -none-    call
stype      1 -none-    character
strata     20 -none-    numeric
weights    20 -none-    numeric
> mean(corrboot$t)
[1] 0.8749549
> sd(corrboot$t)
[1] 0.07112385
> mean(corrboot$t) - corrboot$t0
[1] -0.002947105
> |
```

And then if I try to find out here the mean this is here the mean, and if you want to find out here the standard deviation you see the command here is standard deviation that is sd of the same thing. So, you can see here the standard deviation which you had obtained here and the standard deviation which you obtained here they are the same thing.

So, your outcome is going to give you the same thing. Similarly, if you compute the bootstrap bias also manually just for your assurance that the software is doing the same thing which you wanted you can see here the bias which you have obtained from this results and the bias which has been given by this bootstrap they are the same thing, right.

So, now you see we have completed this is this, these are the same slides which I have shown you here. So, now we come to the finishing of this lecture, but now you have seen the bootstrap which used to be which appears to be a difficult topic up to now, this is very simple extremely simple and using it in r that is rather the most simplest thing.

So, now, I do not think there should be any fear, lack of confidence inside you in using the bootstrap and now you can think that what will be the applications in decision sciences, whatever you want to compute now theory and finite sample properties are not the barrier you can just obtain whatever you want.

Yes, I agree those things may not be 100 percent same as the exact values, but they are pretty close and they are not really bad they are good values. It is something like this, if

you ask my height or my age then possibly some person will say ok, the age is 40 years and you believe it they are that is a good value, but the exact age might be 40 years few months few days, but do you really bother for those things?

Yes, there yes, I agree there can be some situations where this marginal error even may be very important to take care, but in general that will not make much impact on the final outcome. So, I would say now, you try to take some artificial data sets or you take any data set from the book, try to take any statistics.

For example, if I say why do not you try to compute the bootstrap bias standard error of the square root of correlation coefficient or log of correlation coefficient. But, you have to be careful that correlation coefficient can be negative, so, you have to take the log of absolute value of  $r$ , right.

You can consider absolute deviation, you can consider the division around median, division around mode, division around geometric mean although you can also compute the sample geometric mean sample harmonic mean whatever you want. So, you practice and in the next then I will try to explain you how to compute the confidence interval.

So, first I will try to give you the details about the possible confidence interval which can be computed in the R software using the bootstrap methodology and then I will try to show you how they can be computed in the R software. So, you practice and I will see you in the next lecture, till then good bye.