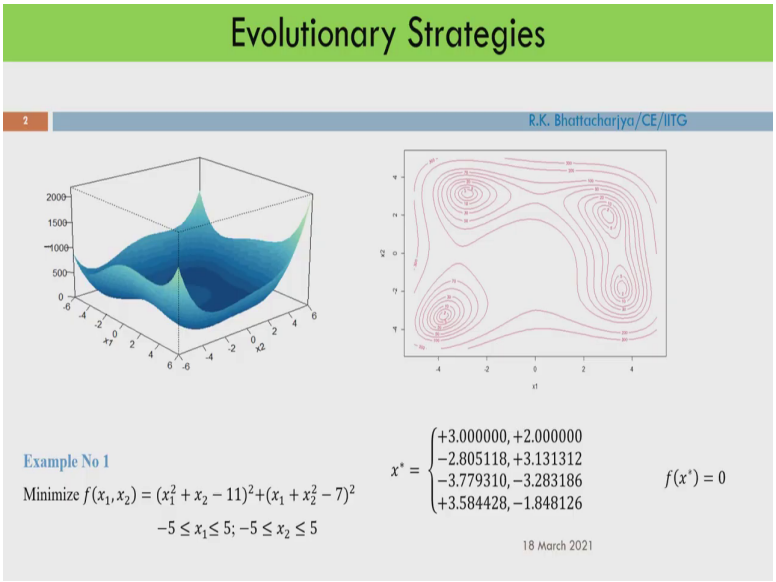
Optimization Methods for Civil Engineering
Dr. Rajib Kumar Bhattacharjya
Department of Civil Engineering
Indian Institute of Technology, Guwahati

Lecture - 26
Evolutionary Strategies

Hello student in this class I will solve two non-linear optimization problems. So, I will show you the function using R programming and I want to use lambda plus mu ES. Basically I would like to show you how this simple algorithm can solve a very complicated optimization problem. I will consider two function here.

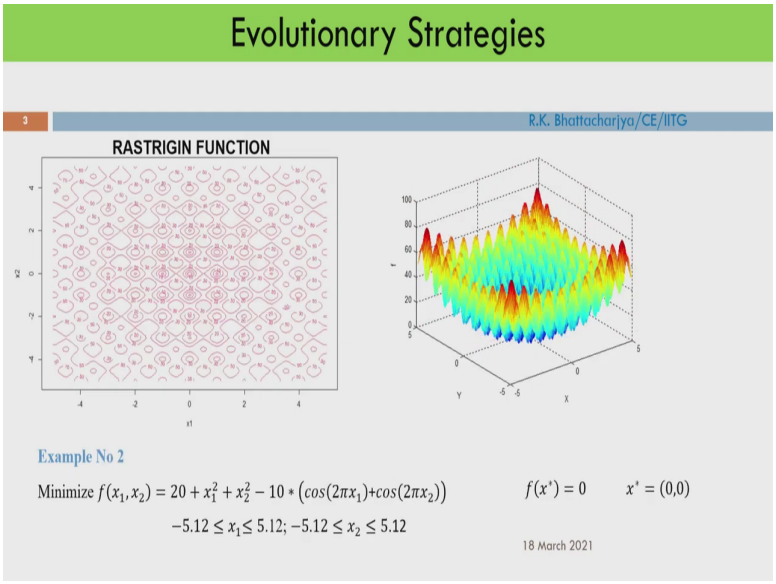
(Refer Slide Time: 00:55)



The first function as you have seen this, so this function already I think you have solved it. So, the function is x 1 square plus x 2 minus 11 whole square plus x 1 plus x 2 square minus 7 whole square and the range of x 1 and x 2 is minus 5 and plus 5.

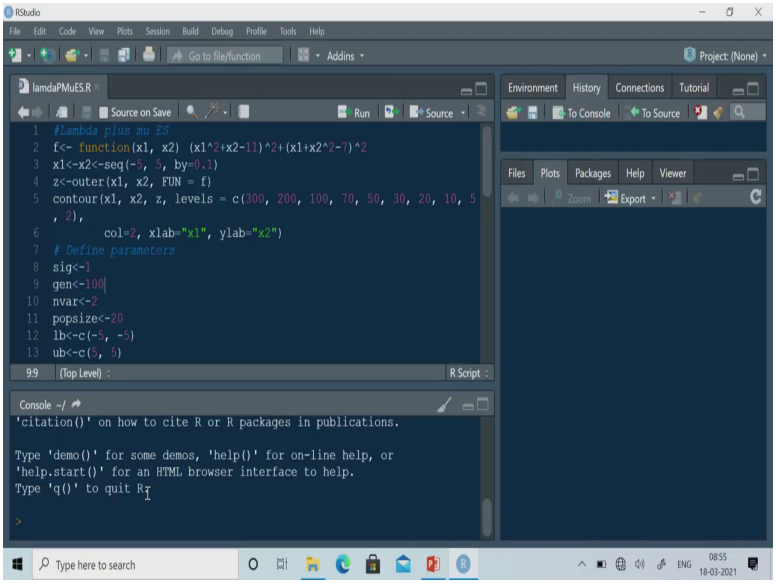
So, as you know that this function has four optimal solution and we will try to find out this optimal solution using lambda plus mu ES there are four optimal solution. So, one at the first quadrant, one at the second, one at the third and another one at the fourth quadrant and function value is 0; that means, these all are alternate optimal solution.

(Refer Slide Time: 01:42)



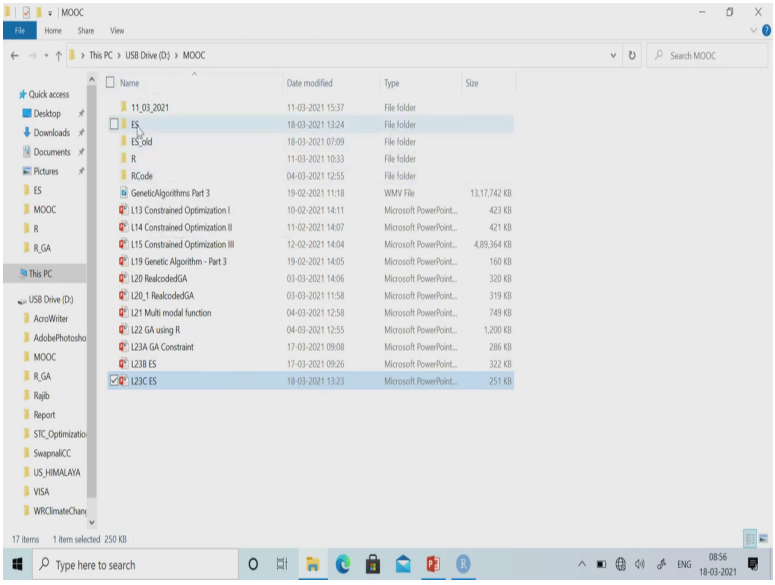
If you look at the second problem, so this is a very complicated function. So, as you have seen for this function several local optimal solutions. So, I have shown you the contour plot as well as the surface plot and you can see the global optimal solution is 0 0 and function value is 0 0 and it has several local optimal solution. So, we will try to find out the effectiveness of lambda plus mu ES in finding the optimal solution of these two functions. Now, let us open R studio.

(Refer Slide Time: 02:20)



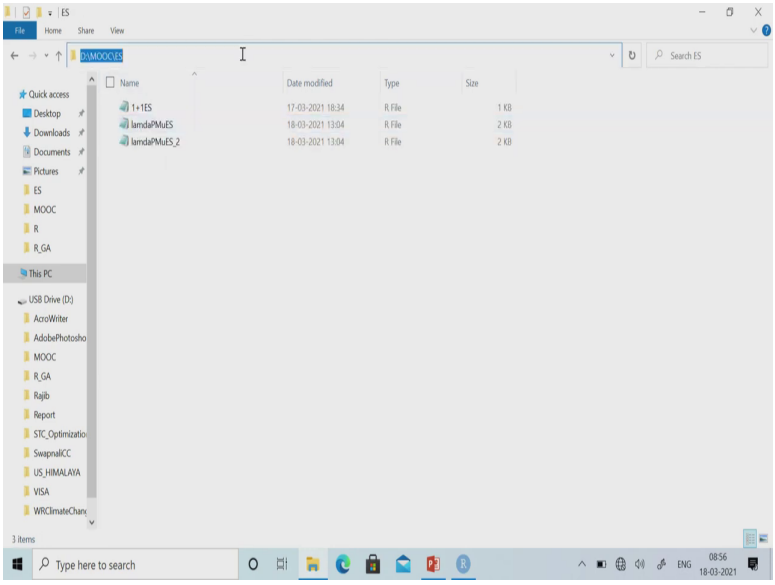
So, I would like to clean everything and I would like to open this function.

(Refer Slide Time: 02:27)

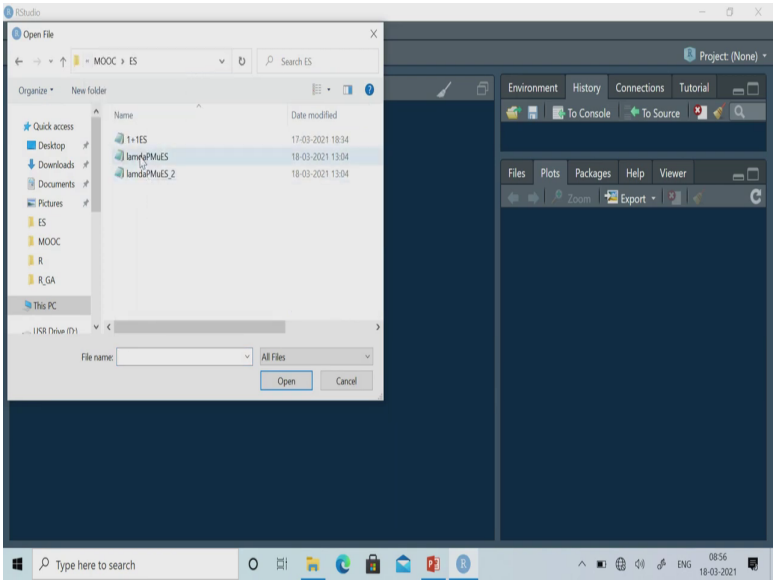


So, let us open the program, so I have stored this program somewhere here.

(Refer Slide Time: 02:31)

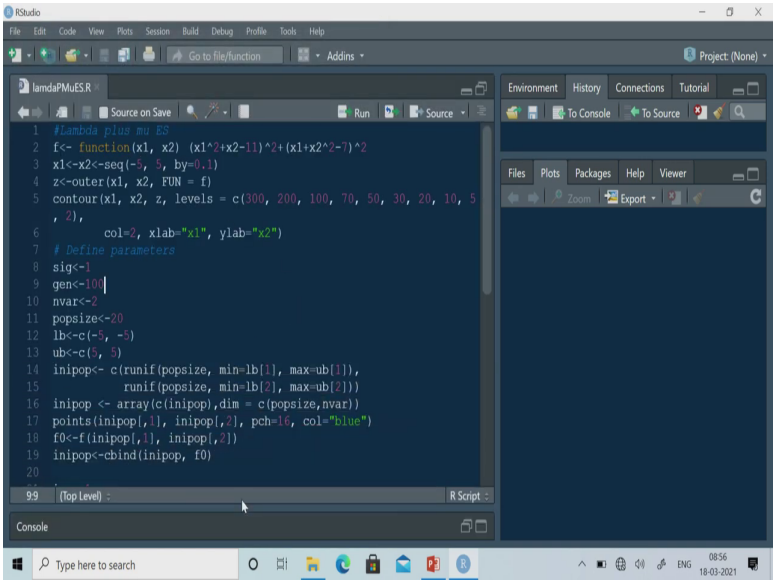


(Refer Slide Time: 02:40)



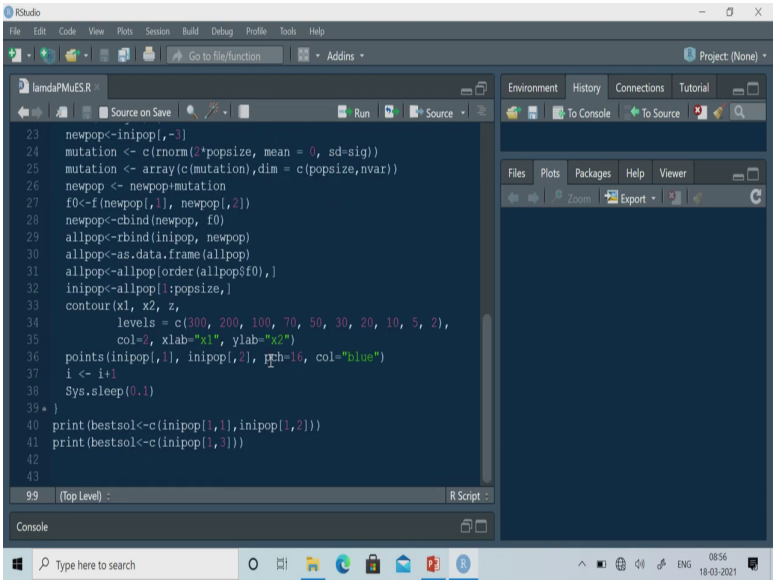
So, this is lambda plus mu ES.

(Refer Slide Time: 02:44)



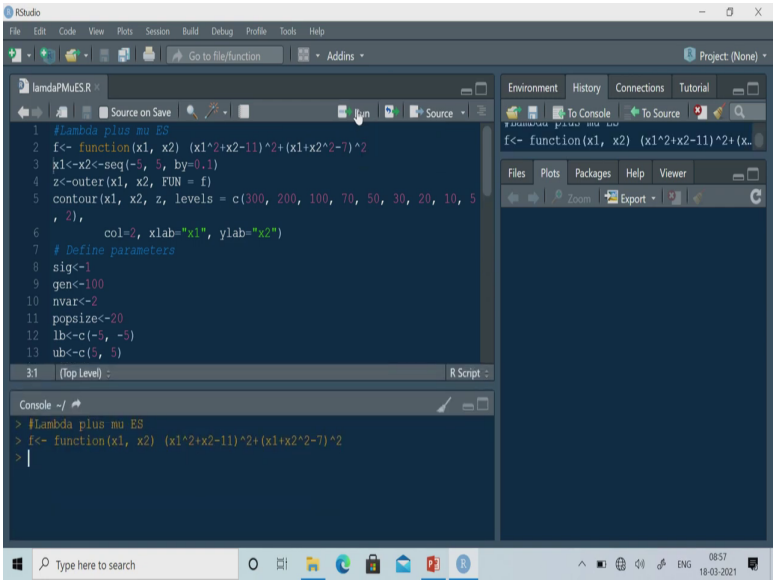
So, this is the program I have written to implement lambda plus ES algorithm.

(Refer Slide Time: 02:54)



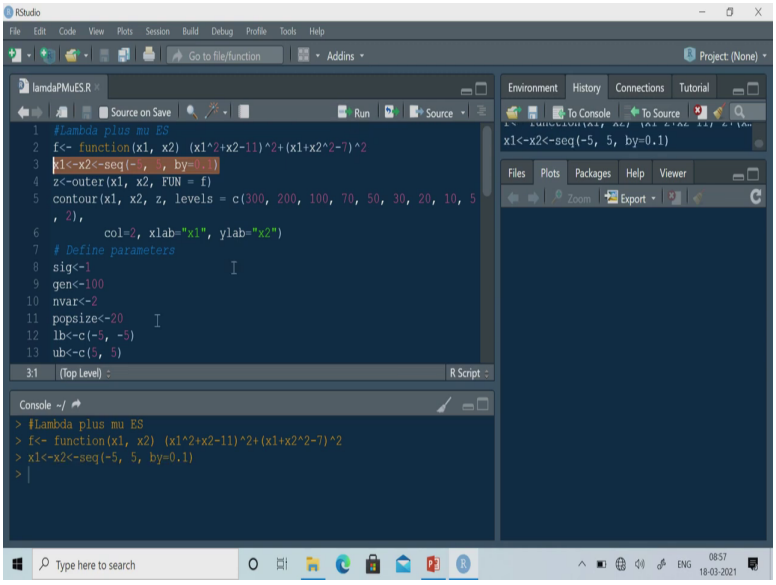
So, I will explain the lines one by one, so it has total 41 lines. So, as you have seen the first few lines are just to plot the function. So, here the function is x 1 square, so this is the x 1 square plus x 2 minus 11 whole square plus x 1 plus x 2 square minus 7 whole square, so this is the function I am using initially this is the problem 1 and it is between minus 5 and plus 5.

(Refer Slide Time: 03:21)



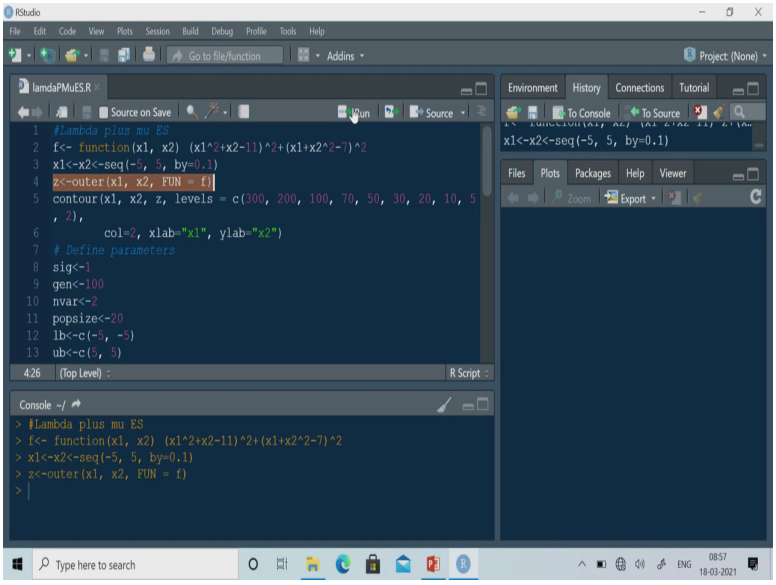
So, I am generating x 1 and x 2 using this function. So, already you know that one so using this line. So, I am generating the value of x 1 and x 2 using this line, so I have used sequence function here.

(Refer Slide Time: 03:40)



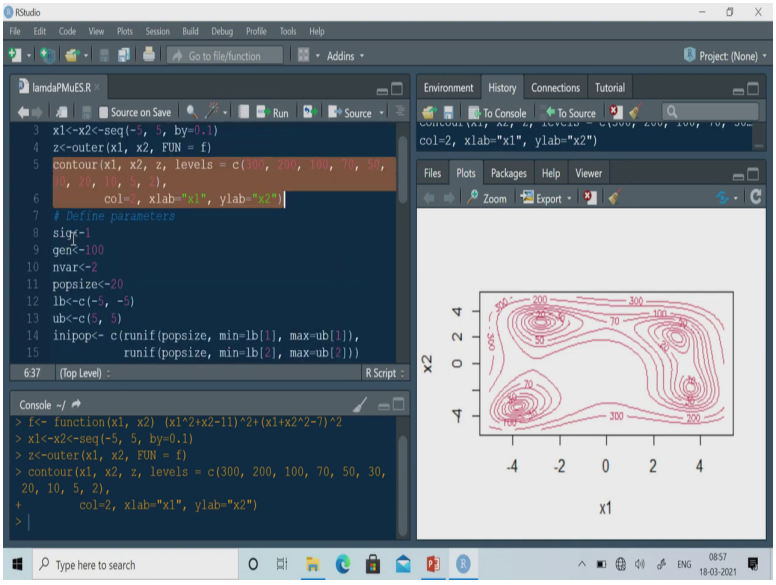
So now, I am getting x 1 and x 2 and now I am calculating the z value function value at its grid point using outer function.

(Refer Slide Time: 03:50)



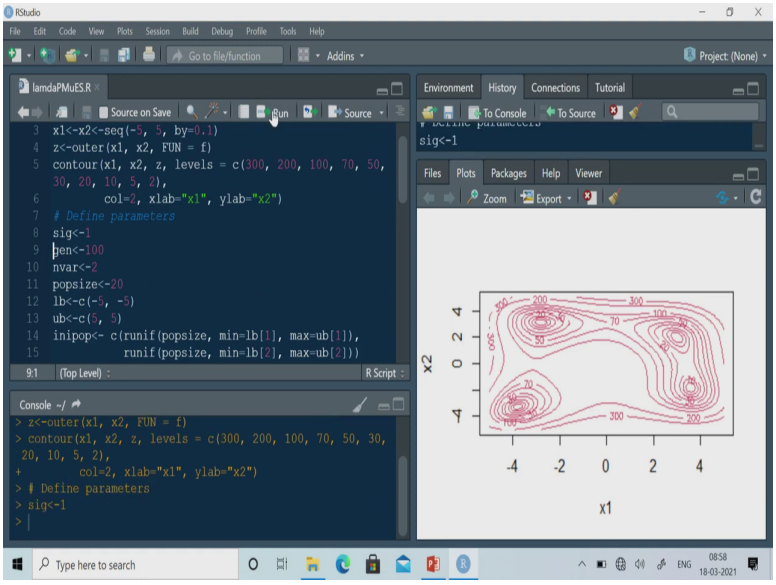
So, I think and then I can plot the contour using contour function.

(Refer Slide Time: 04:00)



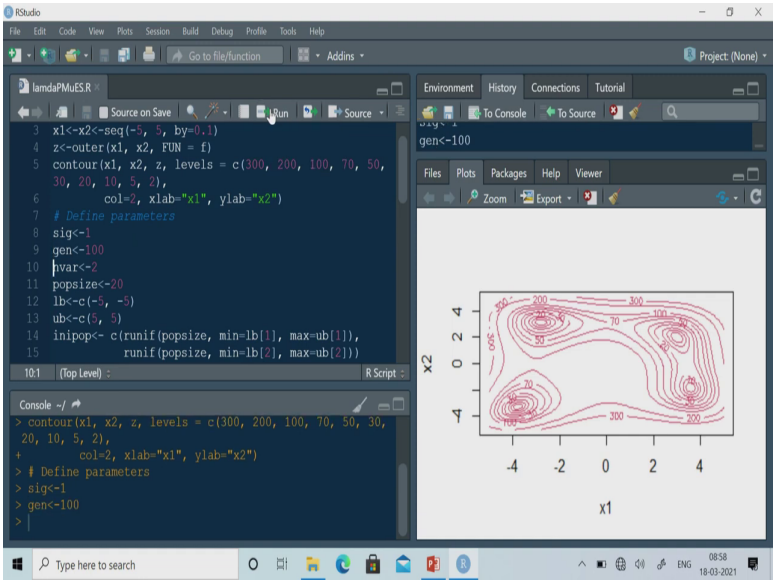
So, you are getting; you are getting the contour plot. So now, let us define the parameters of this algorithm. So, what we have to define? So, I am defining sigma equal to 1, so I am not changing the value of sigma.

(Refer Slide Time: 04:25)



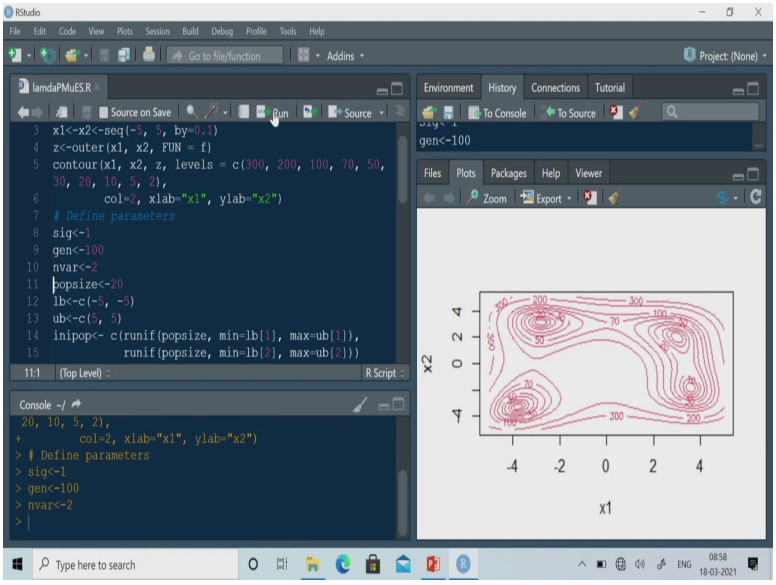
So, I am considering constant value of sigma that is 1, then generation I have defined 100.

(Refer Slide Time: 04:32)



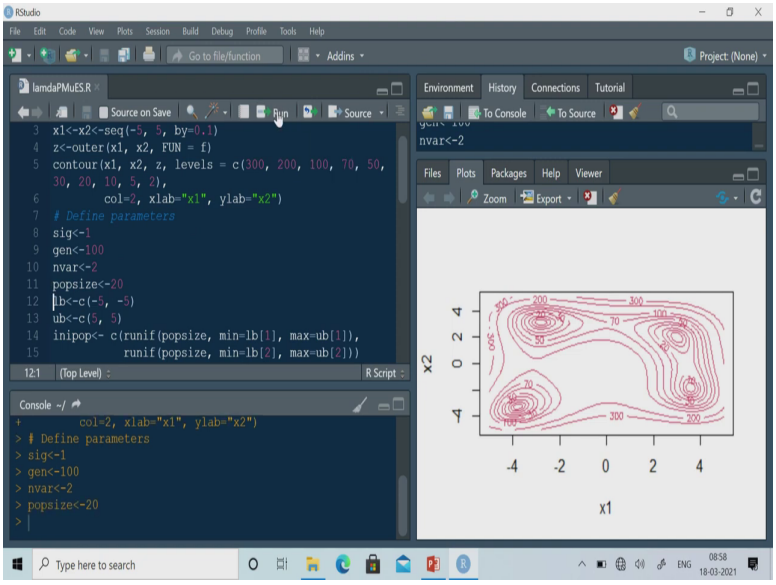
So, this is the so algorithm will run for 100 generation then number of variables 2.

(Refer Slide Time: 04:42)



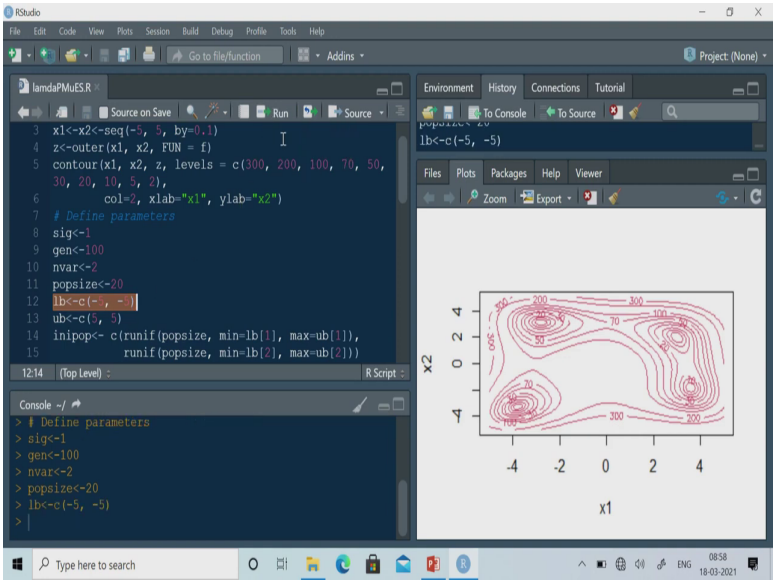
So, it is a 2 variable functions it is a 2 variable function.

(Refer Slide Time: 04:48)



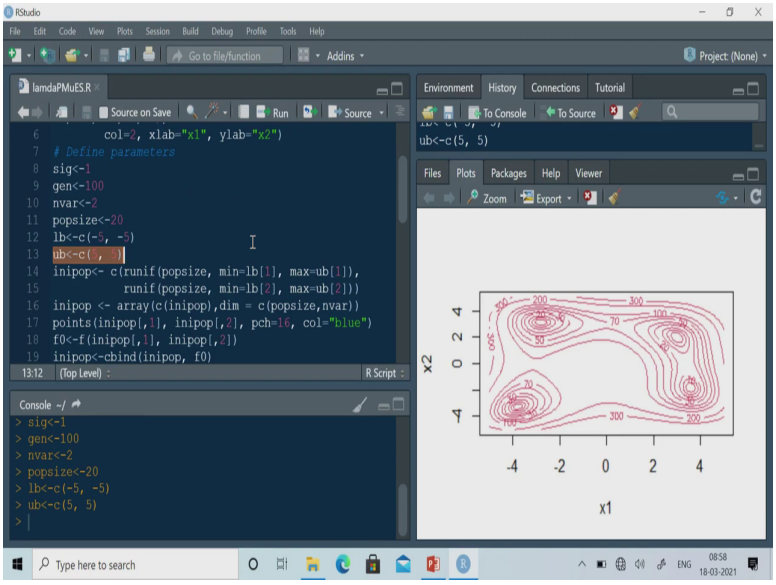
And then population size I have considered 20 initially and then I have defined lower bound and upper bound.

(Refer Slide Time: 04:59)



So, lower bound is minus 5 and 5, so lower bound of x 1 is minus 5 and x 2 is also minus 5.

(Refer Slide Time: 05:09)



Then I have defined upper bound of x 1 which is 5 and x 2 is also 5, so I am writing upper bound 5 5. So, now what I have to do I have to generate the initial population. So, for generating the initial population I have used run if function ok.

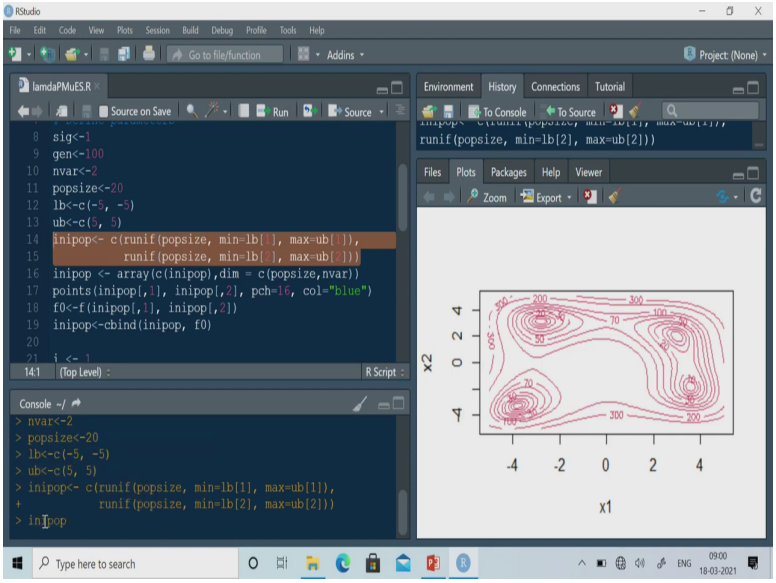
So, this is actually generating a random number between upper bound and lower bound. So, I have defined what is min between minimum and maxima. So, min is lower bound and maxima is upper bound. So, this particular line this particular line will generate the generate random population between lower bound and upper bound for variable 1.

So, in this case both are equal that variable x 1 is between minus 5 and plus 5 and similarly x 2 is also between minus 5 and plus 5, but if your variable range is different. So, you can define the other limits, but in this case both are same anyway. So, I have defined so how

many random value I am generating; I am generating random value equal to population size; that means, suppose population size is 20, so I will generate 20 solution.

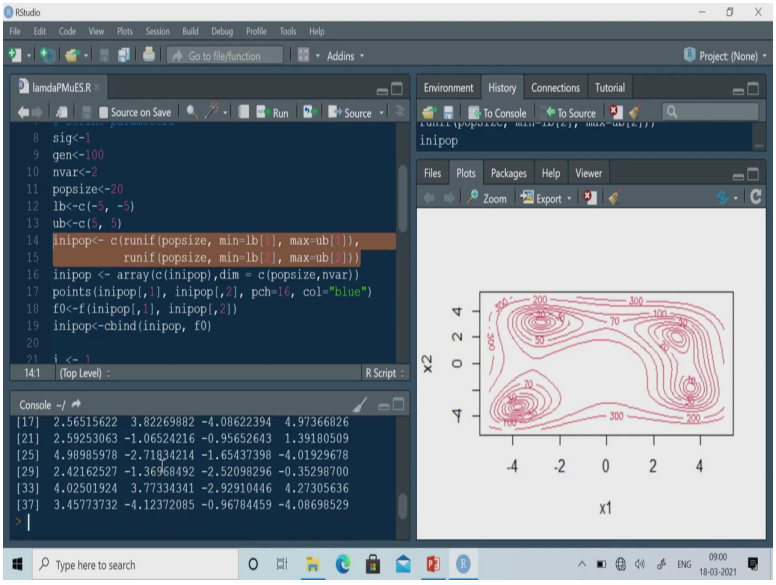
So, similarly for variable 2 also I have defined lower bound and upper bound of variable 2. So now, if we run this particular line so I will generate initial population like this ok.

(Refer Slide Time: 06:32)



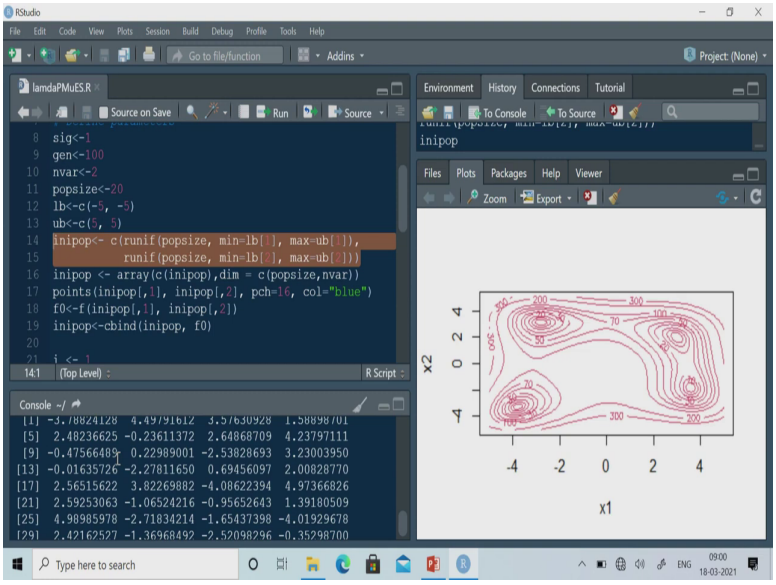
So, you can see this initial population.

(Refer Slide Time: 06:36)



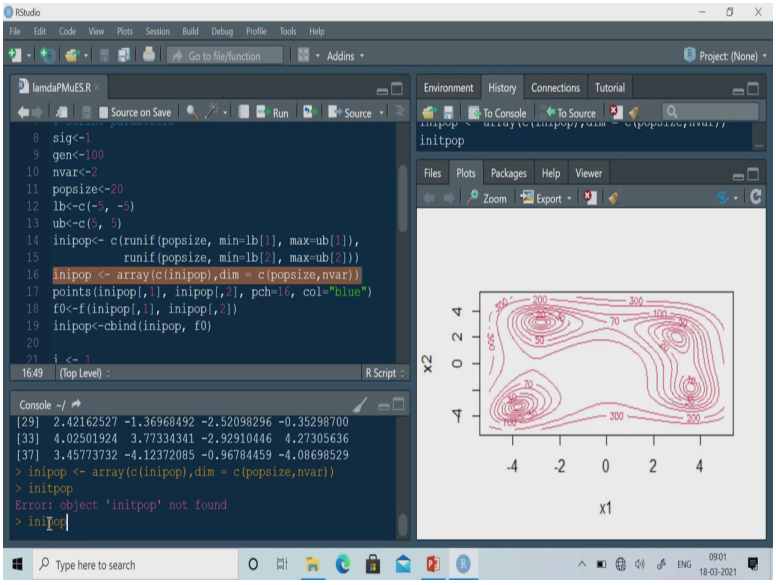
So, you can see that so we have total 20 values.

(Refer Slide Time: 06:40)



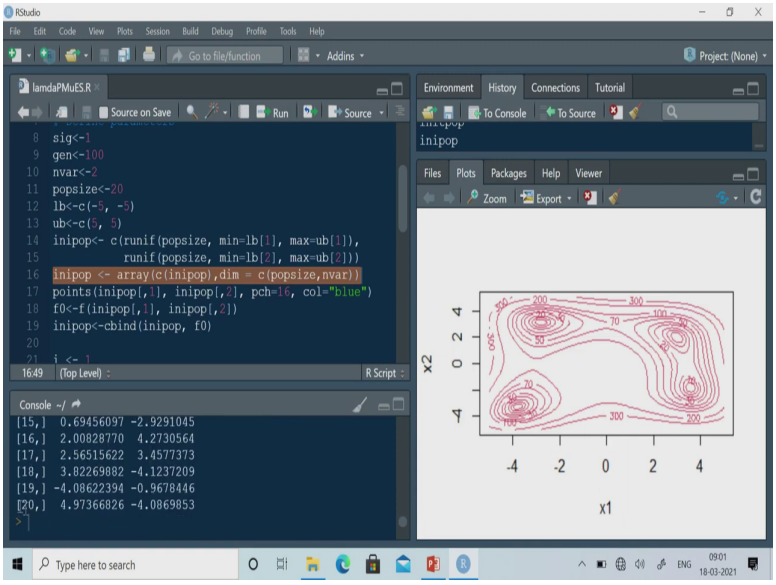
But, this is a one dimensional array and I am getting 20, so first 20 is x 1 and second 20 is your x 2. So, now I have to define it as a two dimensional array, so I have defined this is some you I have used I have used the array function just to define the two dimensional array.

(Refer Slide Time: 07:08)

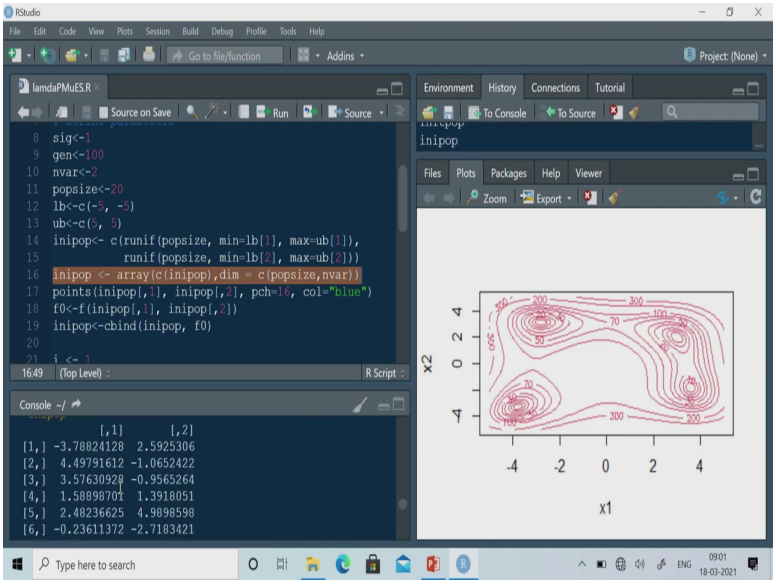


So, I if I run this particular line so you can see so I am getting initial population.

(Refer Slide Time: 07:23)



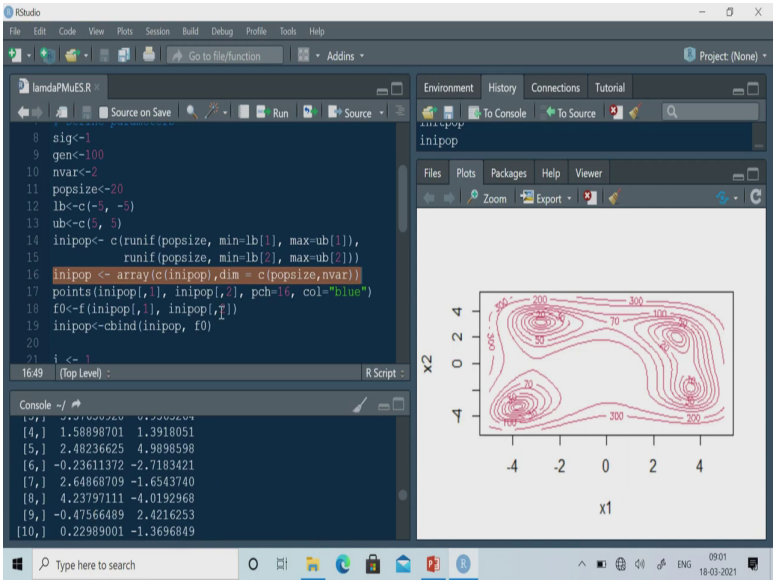
(Refer Slide Time: 07:25)



So, you can see. So now, you are getting this is your population 1, this is population 2 something like that. So, all together we have total 20 solution and this is variable 1. So, this is variable 1 and this is variable 2.

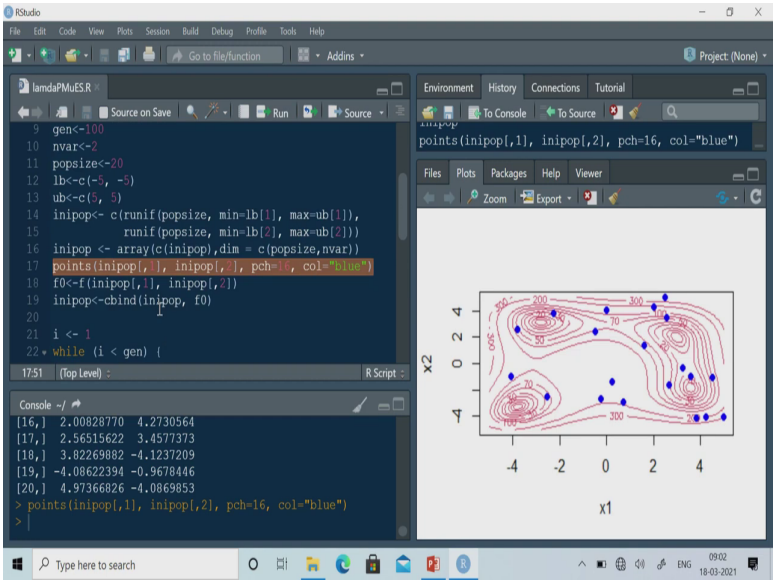
So, with this I have generated total 20 solution I have generated total 20 solution and I have the x 1 and x 2 and the solutions are between lower bound and upper bound.

(Refer Slide Time: 07:50)



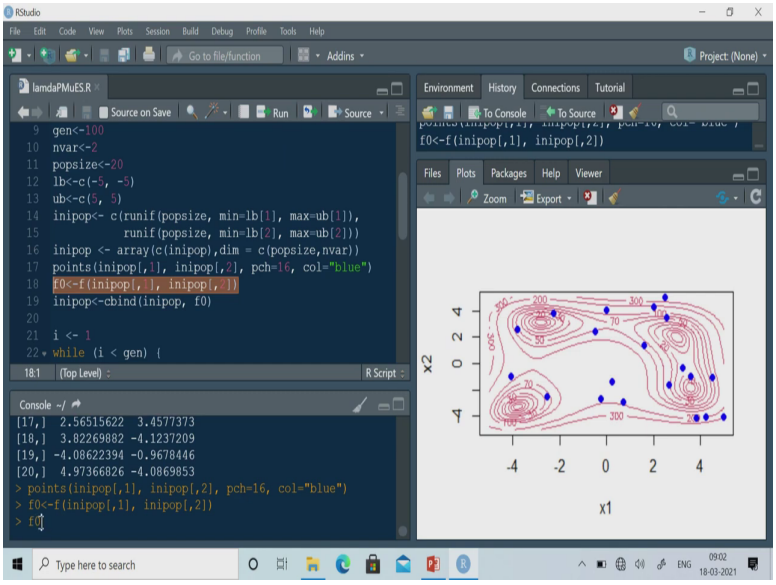
Now I would like to plot this point over this contour.

(Refer Slide Time: 08:00)

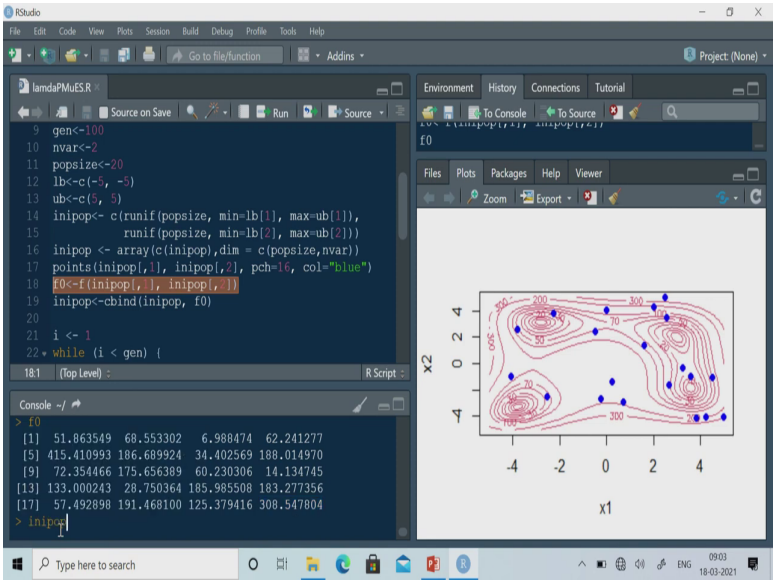


So, let us run this particular line, so you will get that the population over this contour. So, now let us calculate the function value. So, function value so what I will do? So, I have to use this function already I have defined this function and so only thing I have to call it. So, I am writing f naught. So, what is f naught? f naught is the function value of initial population. So, I have to pass two arguments that is your x 1 and x 2. So, initial pop this is variable 1 and this is variable 2 and if I run this particular line. So, I will calculate the function value of all this solution.

(Refer Slide Time: 08:41)

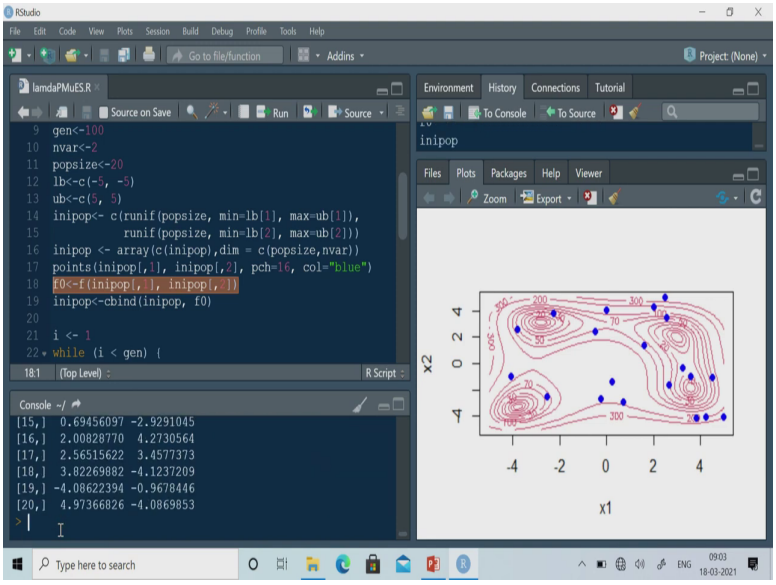


(Refer Slide Time: 08:45)



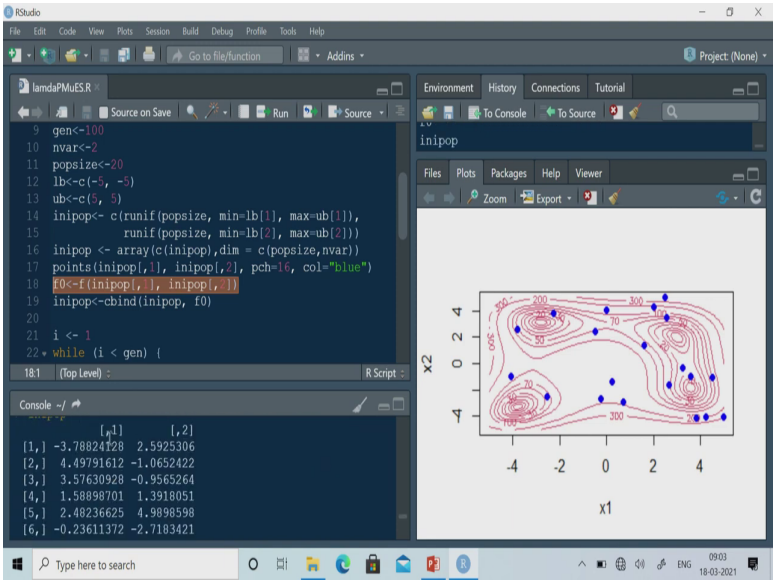
So, you can see here that f naught you can see that I am getting the function value of all the solutions. So, function value of 1st solution is 51 and similarly 20th solution is 308 ok. So now, what I am doing; I am putting this function value on the third column of initial population.

(Refer Slide Time: 09:10)



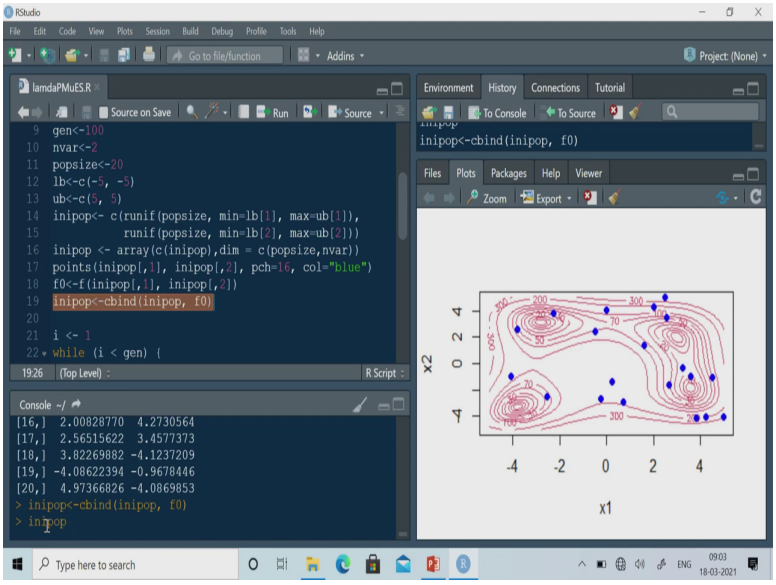
So, you can see that initial population if you look at.

(Refer Slide Time: 09:12)



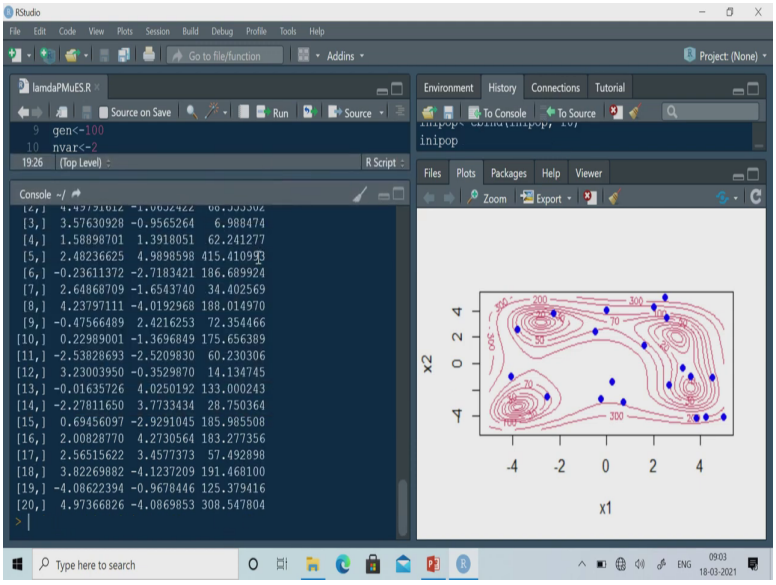
So, it has two column that is your x 1 and x 2 and I would like to make another column and where I would like to keep the function value. So, what I can do I can use c bind and I can add another column and this column will contain f naught.

(Refer Slide Time: 09:29)



So, you can see that one, so now if you look at initial population. So, is the c so you are getting another column here, so this is the initial population.

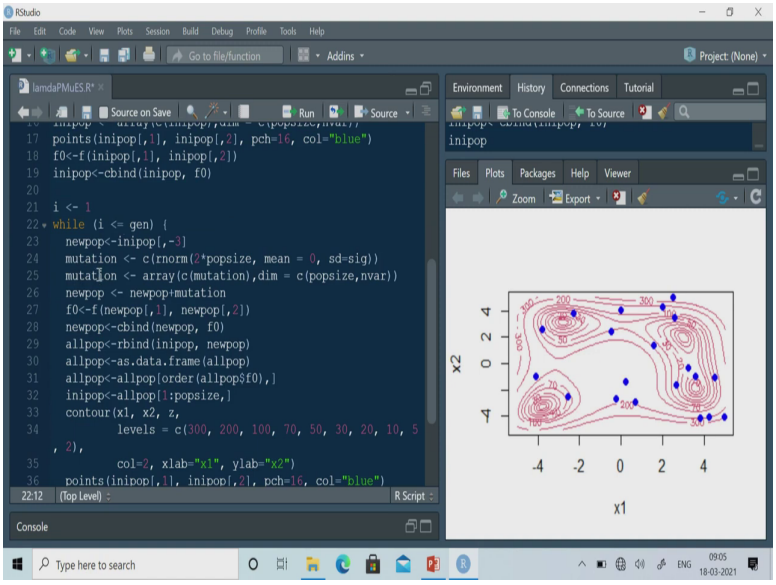
(Refer Slide Time: 09:34)



So, you can see this is the value of x 1, this is the value of x 2 and this is the function value ok objective function value.

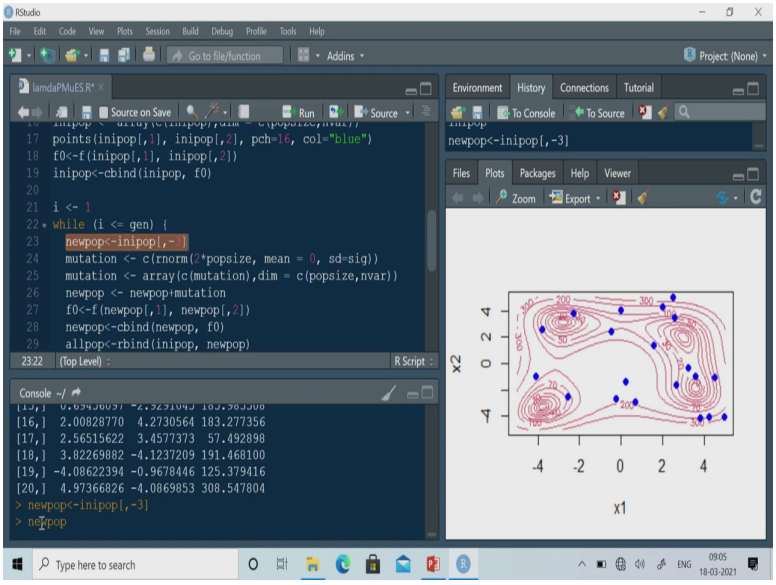
So, I am getting for all population that 20 population I am getting x 1, x 2 and function value. So, I am keeping everything in my initial population that is the solution as well as the function value. Now, I will start so now I will start the iteration process. So, I have used while function here; that means, while i is less than gen. So, I am putting i equal to 1 and then I am putting while function. So, this loop will continue unless i is less than generation; generation I have put 100 here, so it will go up to 100, 100 means it will go up to 99.

(Refer Slide Time: 10:34)



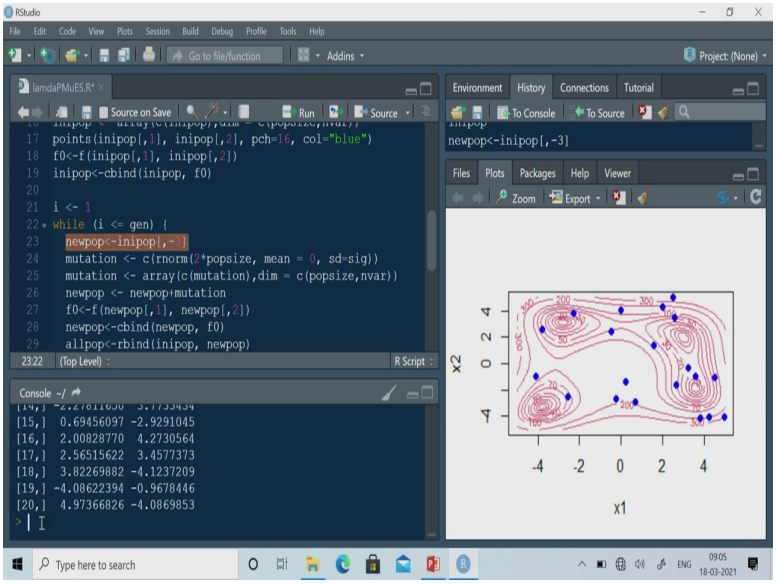
Or if I put this is equal to gen then in that case it will go from 1 to 100. So, now what I have to do the first step is I have to create new population. So, new population what I am doing from this new population I am creating lambda population using a Gaussian distribution ok. So, what I am doing first I am copying this initial population to new population ok.

(Refer Slide Time: 11:04)

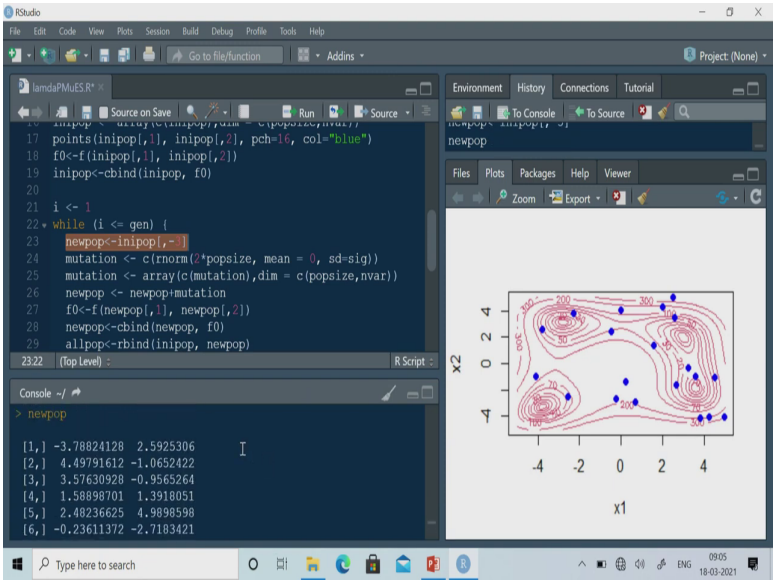


So, here I have used minus 3, so minus 3 means that function value I do not need, I need the x 1 and x 2 value.

(Refer Slide Time: 11:22)

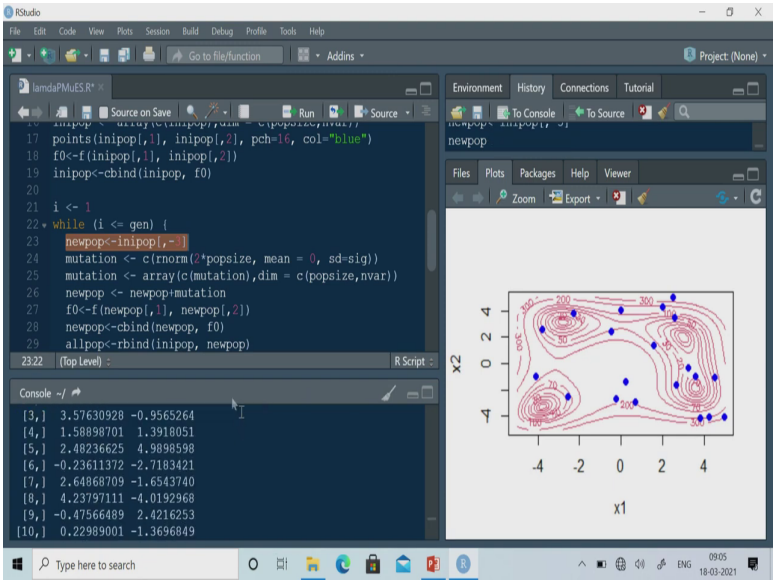


(Refer Slide Time: 11:23)



So, you can see the new population will not have the third column.

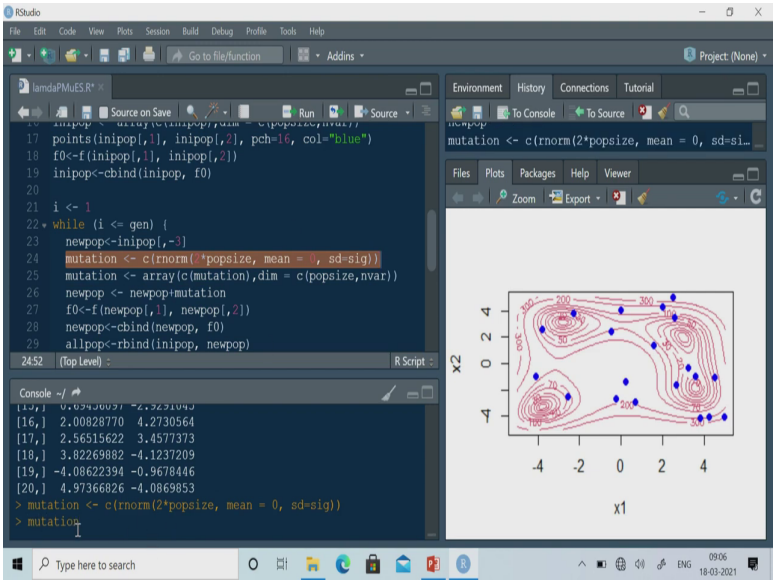
(Refer Slide Time: 11:24)



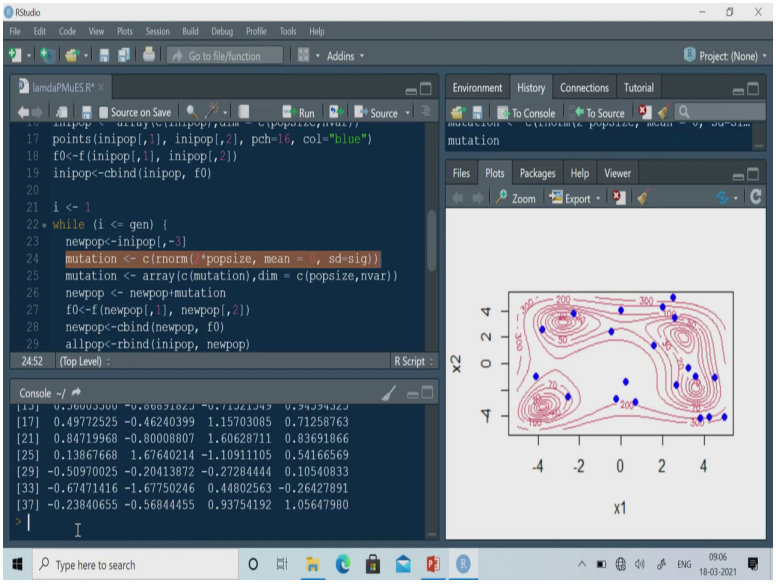
So, you can see that new population will not have the third column, so because I have subtracted minus 3. So, minus 3 is not there minus 3 is the third column so you have only x 1 and x 2.

So, now I have to I have to add the mutated value. So, here I have written mutation and what I am using, so I want in order to create a random number with mean 0 and sigma. So, I have used r norm function, so in this case I am calculating for twice of populations size; that means, you have 20 so I need 40 value. So therefore, I am writing 2 into pop size population size and this random number will be generated with mean 0 and sigma equal to 6. So, I have defined s i g sig equal to 1 ok. So, this line will generate the mutation value ok.

(Refer Slide Time: 12:22)

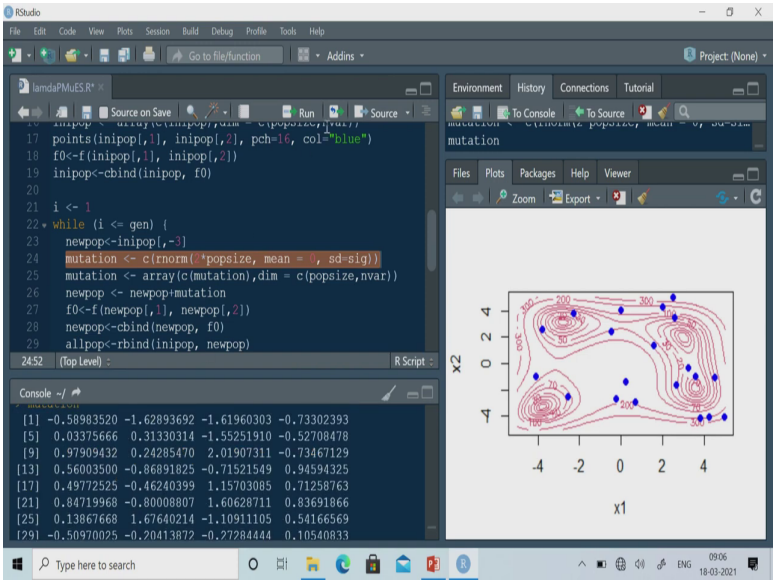


(Refer Slide Time: 12:26)



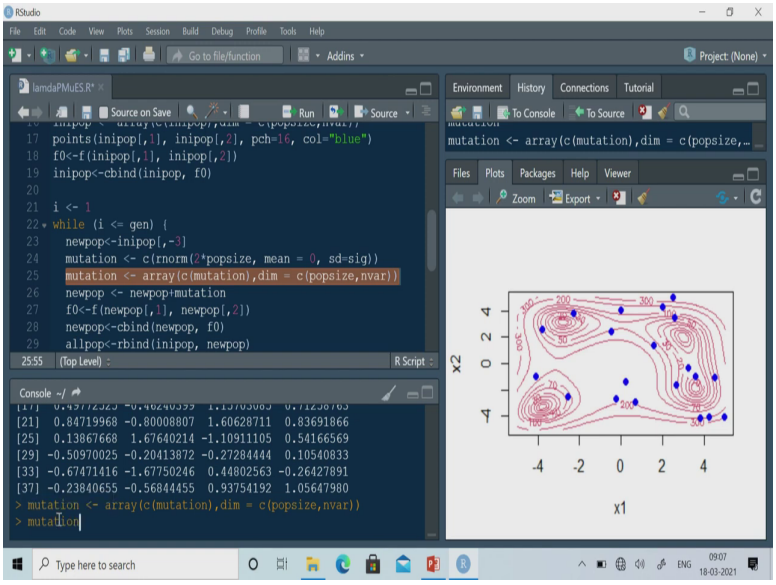
So, that is so you can see. So, you can see that one.

(Refer Slide Time: 12:29)

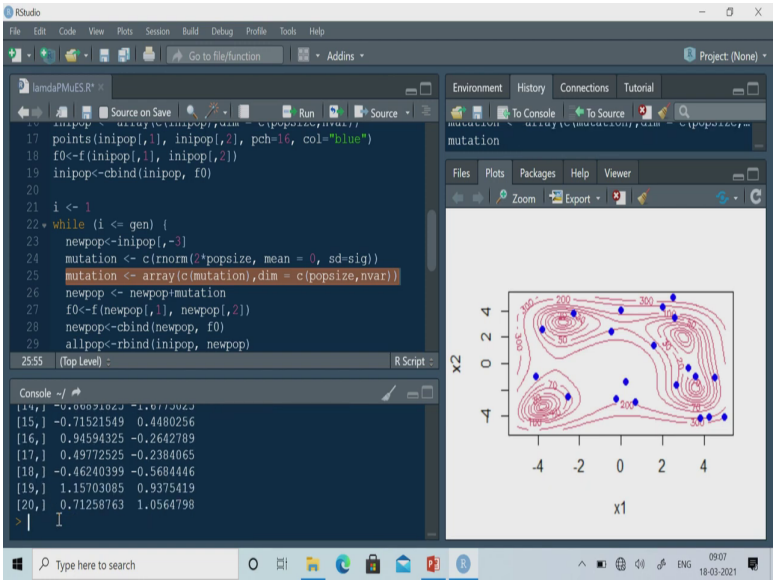


So, this is the number it has generated with mean 0 and sigma, so I will add this with the initial population. Now, it is a one dimensional array so I have to put in a two dimensional array so I am using array function and dimension is population size, the population size is a row and it is number of variable is the column ok. So, I am running this particular line.

(Refer Slide Time: 12:55)

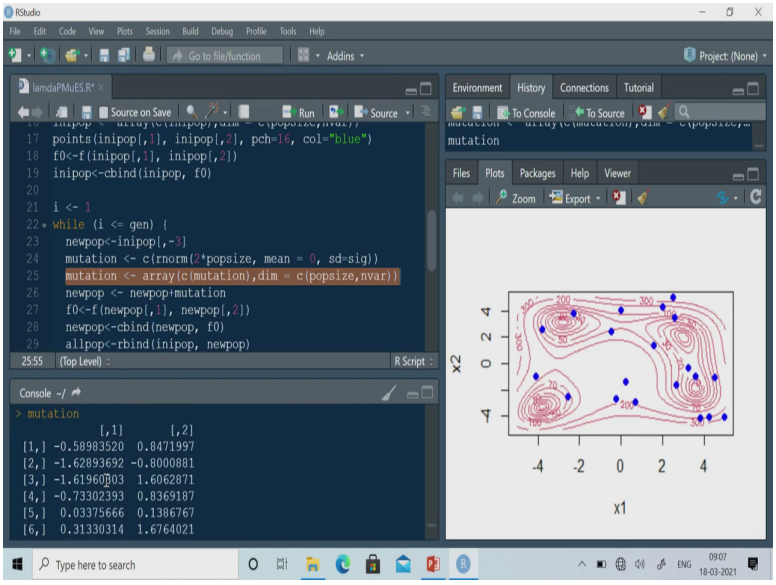


(Refer Slide Time: 13:02)



So, I will get this mutated value, so you can see now ok.

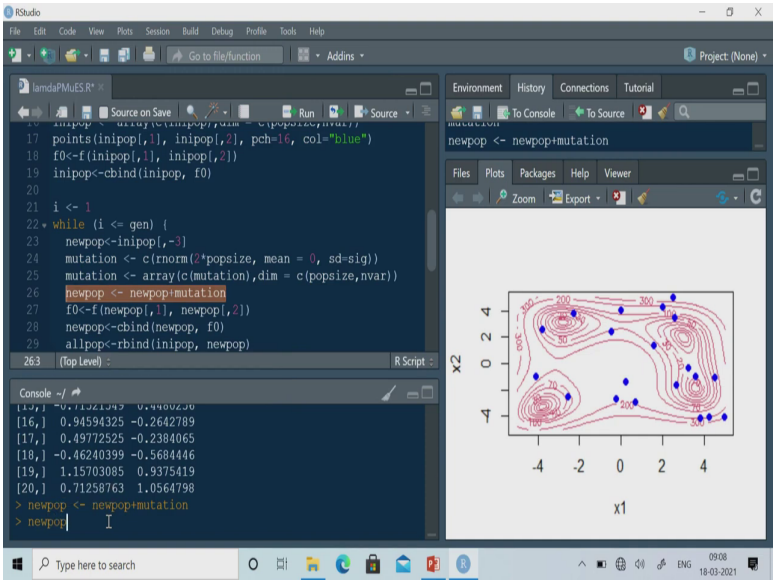
(Refer Slide Time: 13:03)



You can see now so I am getting this value and this is for variable 1 and this is for variable 2 and this is for that 1st population and 2nd population and you are getting up to the last population 20th population.

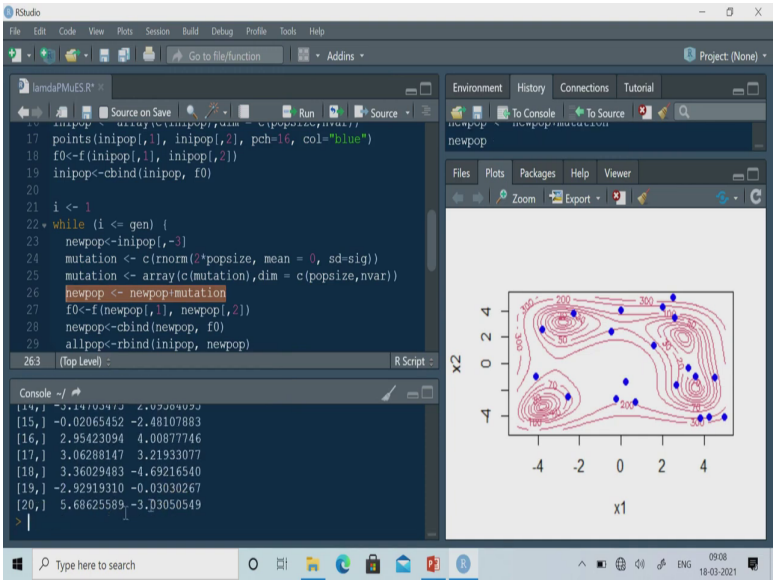
So, now how we are creating new population? So, new population equal to this is the new population; that means, the whole population what we have plus mutation. So, I am adding this mutated value ok.

(Refer Slide Time: 13:30)



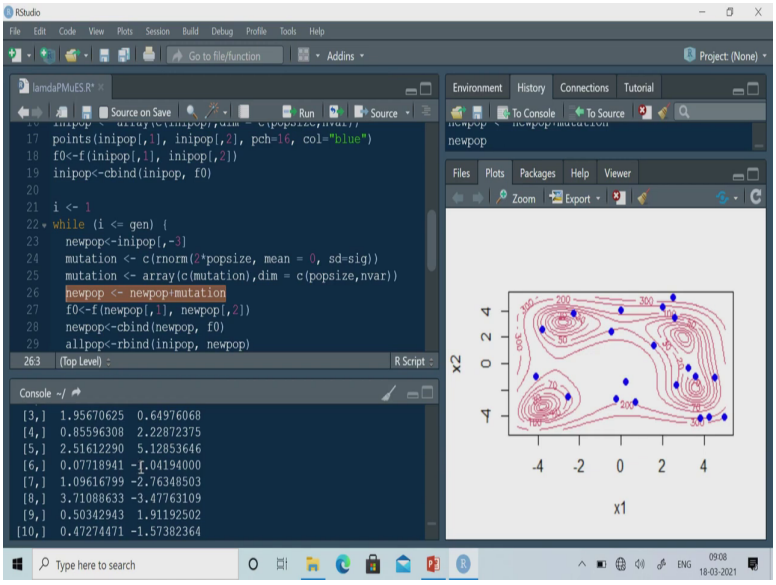
So, please run this particular line, so you can see whatever new population you are getting.

(Refer Slide Time: 13:36)



New population so now this is the difference.

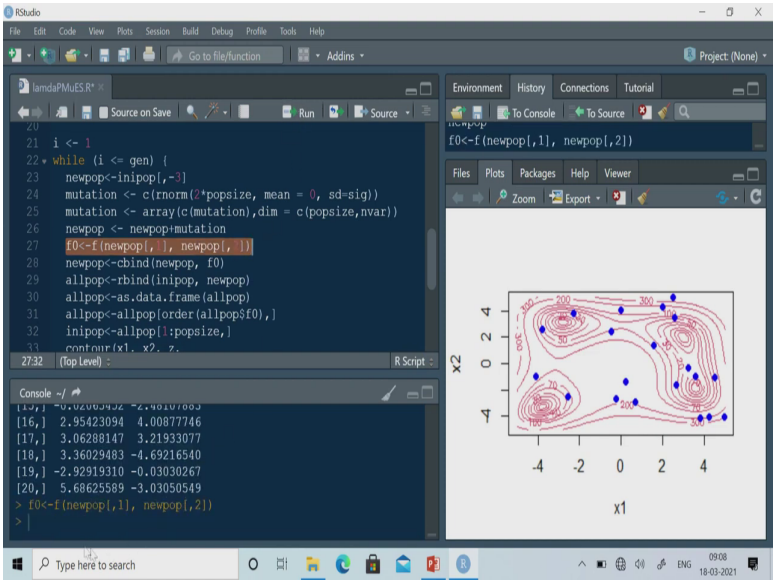
(Refer Slide Time: 13:39)



So, I have added the random number generated with mean 0 and sigma ok.

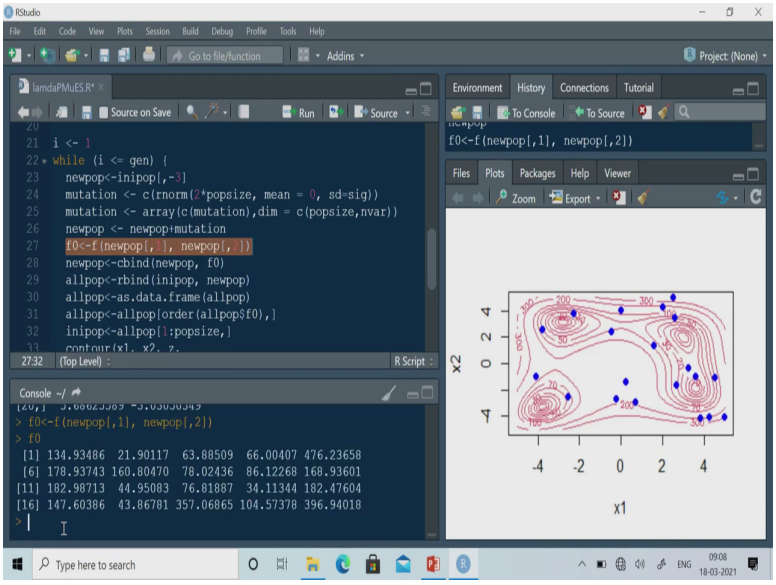
So, now what I will do, I will calculate the function value ok. So, I am writing f 0 again. So, you can write anything any variable name you can give, so but I have given f 0 or you can say this is your f 1 also you can put.

(Refer Slide Time: 14:07)



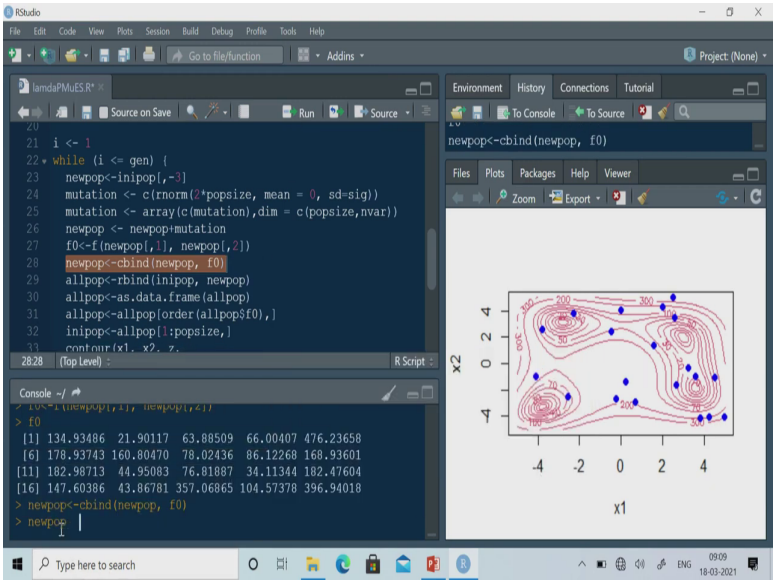
But, anyway so I am putting f 0 here, so if I run this particular line. So, I am calculating the function value.

(Refer Slide Time: 14:10)



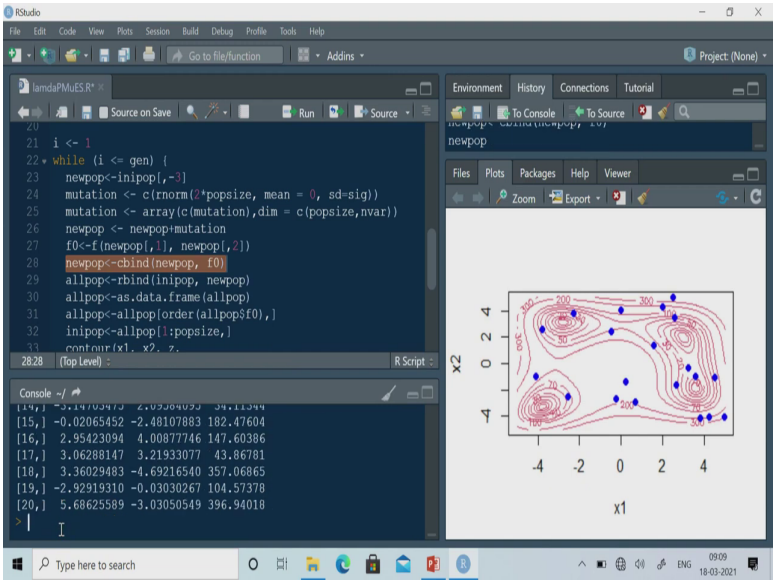
So, this is your f 0, so you can see that you are getting the function value for all these population that 20th population. So, again I am adding as a third column with the new population. So, I have used the c bind comma c bind function I have used the c bind function and I am adding this function value with the new population.

(Refer Slide Time: 14:35)

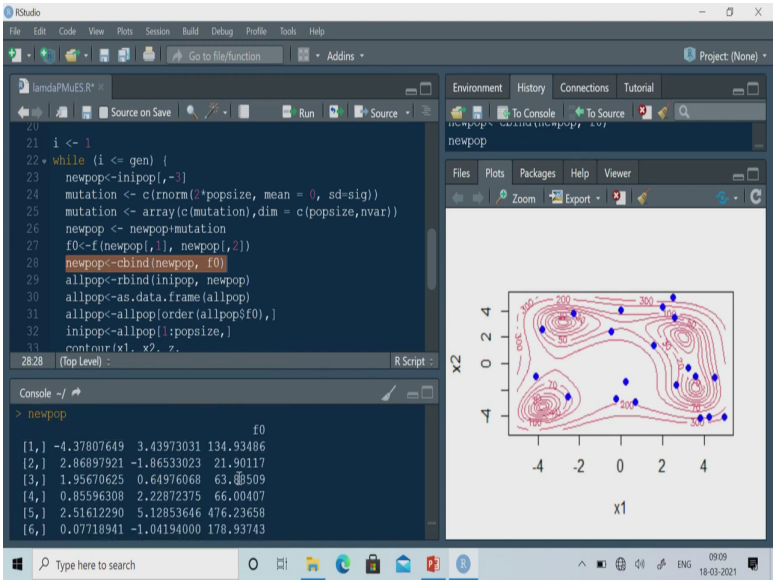


So, if you run it. So now you can see that new population ok.

(Refer Slide Time: 14:41)



(Refer Slide Time: 14:44)

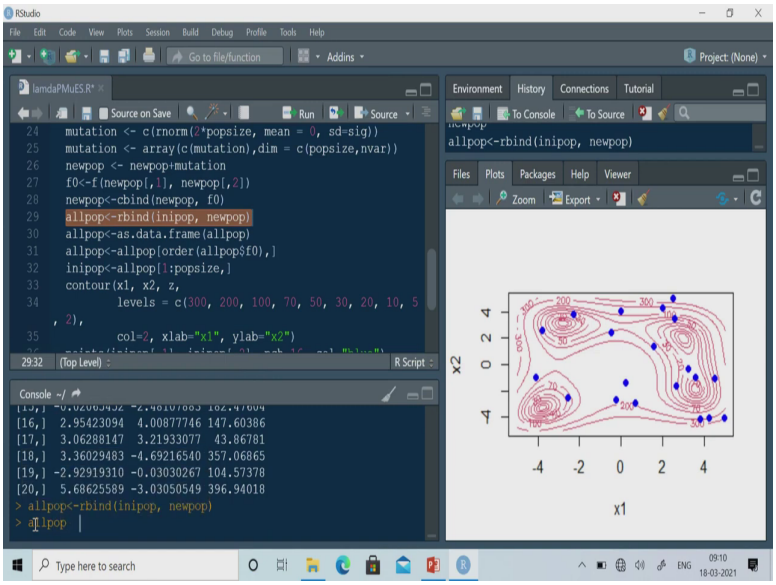


So, new population has also the objective function value. So, objective function value for the first solution is first new solution this is not the old solution that is 135; 134.93486. Now, so I have generated a new solution, so next step is that you combine this solution and you are selecting mu based solution ok.

So, you can implement this line in a different your using different algorithm, suppose you can go for selection operator, suppose I can use roulette wheel selection, proportionate selection I can use. But, here I am not using that one, so what I am doing; I am combining these two population; that means, initial population and new population. So, initial population we have 20 solution, then new population we have another 20 solution; that means, total we have 40 solution and out of 40 I am selecting the best 20 solution.

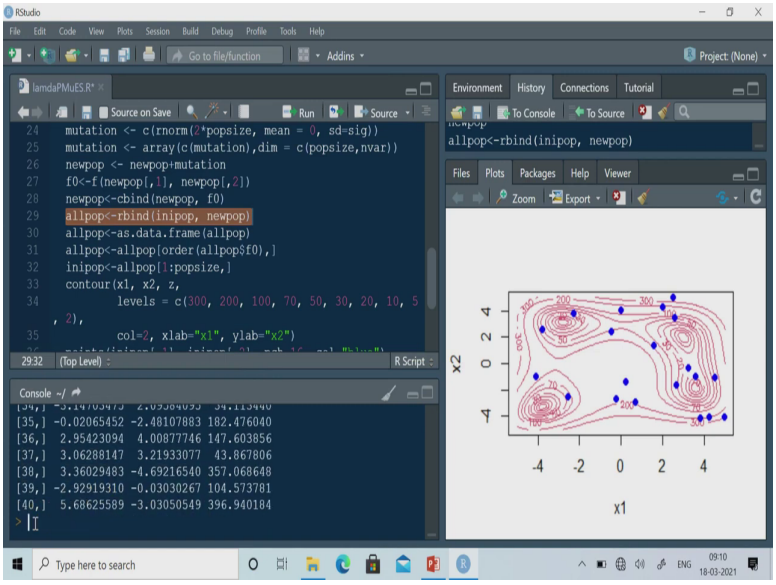
So, what I am doing I am combining these two population; that means, initial population and new population. So, with so I am writing all pop, so here I am using rbind. So, rbind initial pop new pop. So, initially we will have so in this array, so initial 20 solution will be the initial solution and the next 20 will be the new population.

(Refer Slide Time: 16:09)



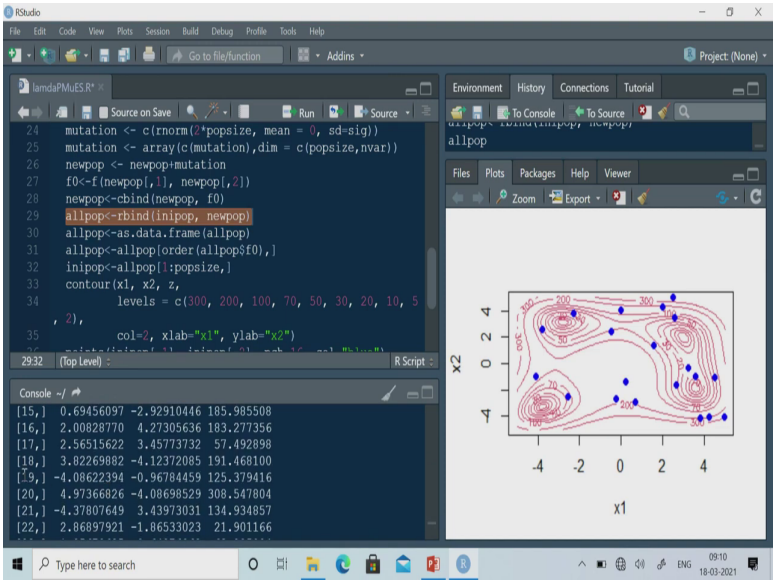
So, if I execute this particular line so I am getting all pop.

(Refer Slide Time: 16:14)



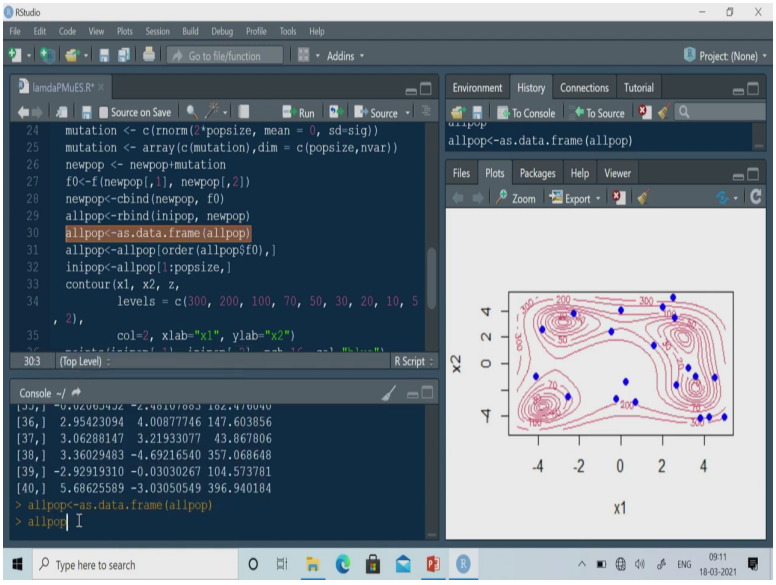
So, you can see so all pop means total 40 solution.

(Refer Slide Time: 16:17)



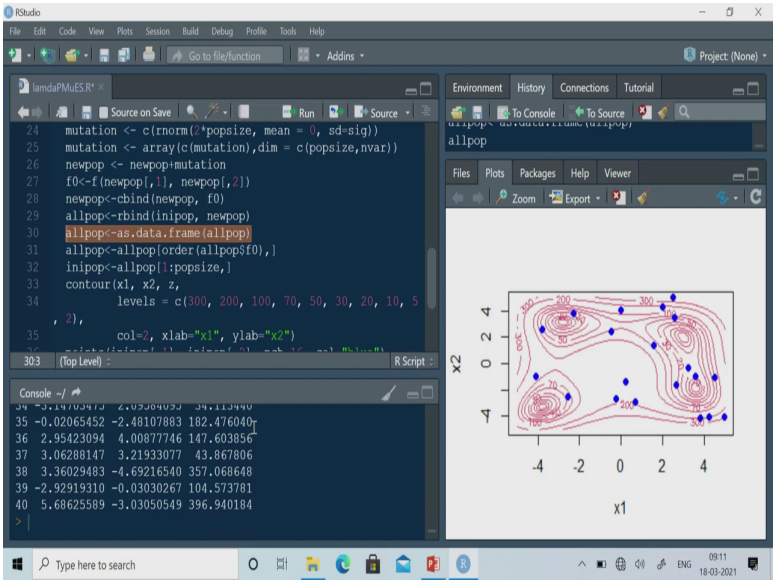
So, first 20 is the initial solution up to here and from 21 to 40, so this is the from 21 to 40, so that is the new population. So, then I have defined it as a data frame because I have to order them in order to use the order function. So, I have defined all pop as a data frame. So, I can write as dot data dot frame all pop.

(Refer Slide Time: 16:48)

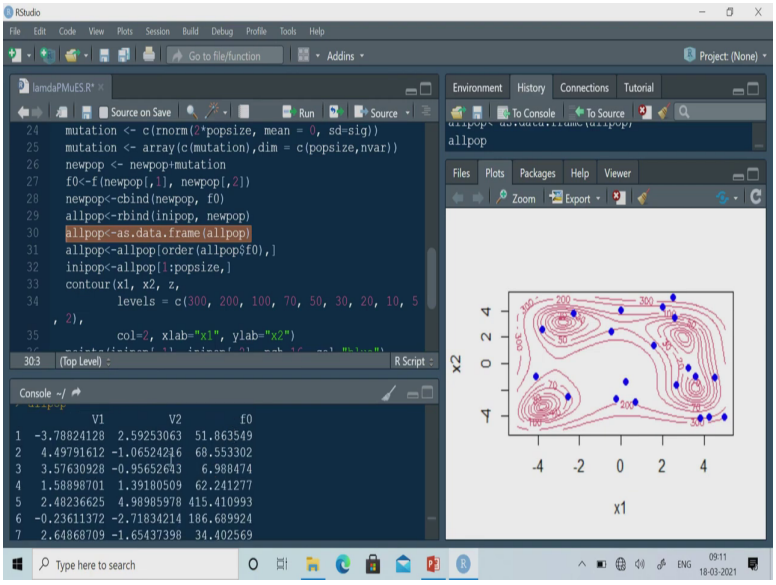


So, if I execute this one so I am getting all pop as a data frame.

(Refer Slide Time: 16:56)



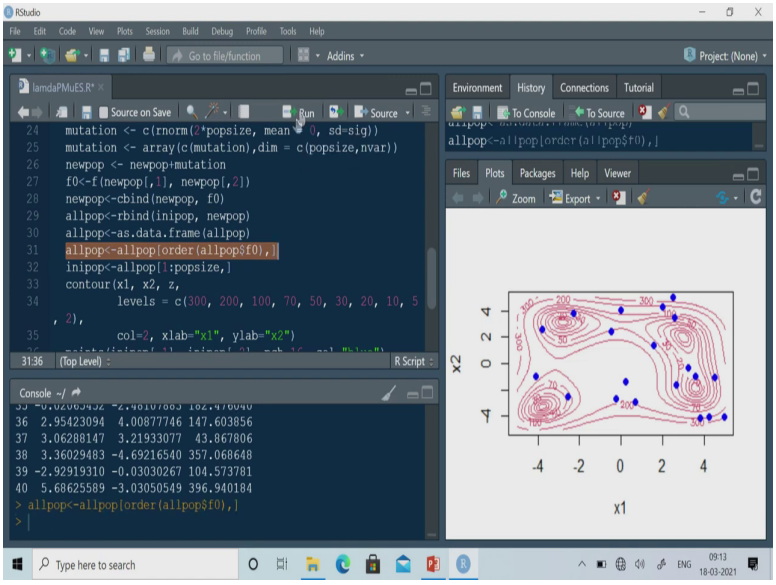
(Refer Slide Time: 16:58)



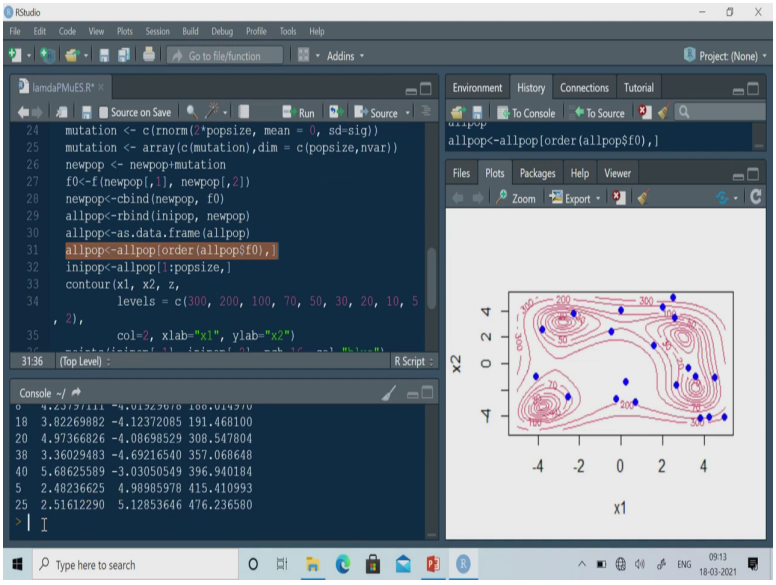
So, you can see now, so it is now data frame and here this is V 1 means this is the first variable x 1, this is second variable and third one is the objective function value and we have total 40 solution.

The next step is to order them, that means I would like to sort it. So, how I am sorting; I am sorting in terms of f naught; that means, large variable that is the function value. So, to do that so I can use the order function and how I am sorting I am sorting as per the function value. So, I am writing this if I execute this so I will get the order of the sort data.

(Refer Slide Time: 17:37)

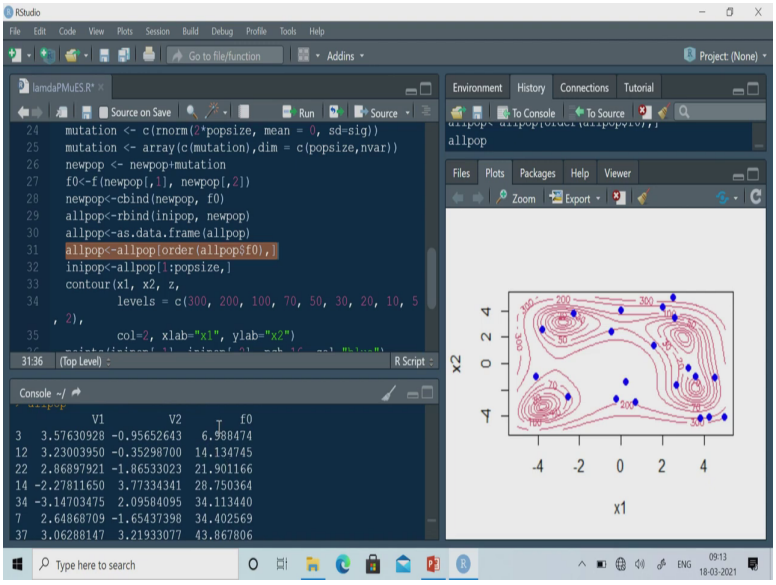


(Refer Slide Time: 17:40)



So, you can see, so now it is in terms of order.

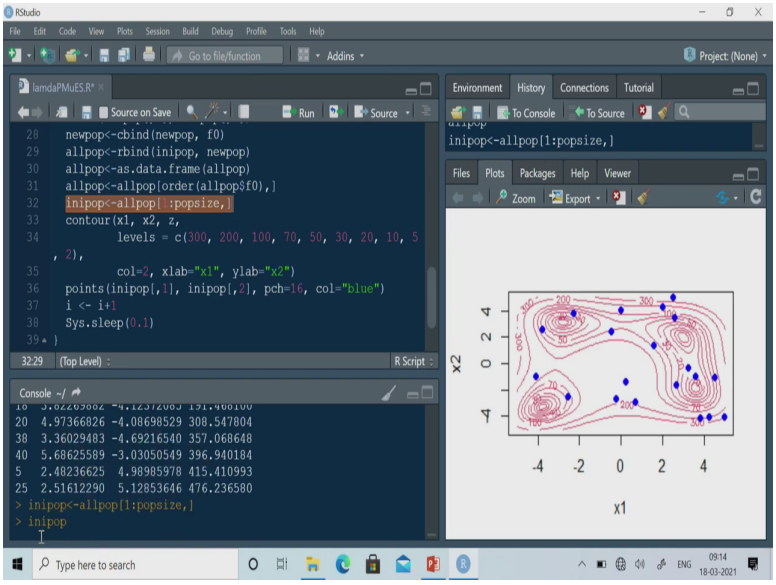
(Refer Slide Time: 17:44)



You can see the minimum function value is 6 that is the best solution, so which is at the top and then it is in increasing order. So, increasing function value. So, the worst solution is at the bottom and the function value is 476.236580 and you can see that. So, it is in terms of increasing order.

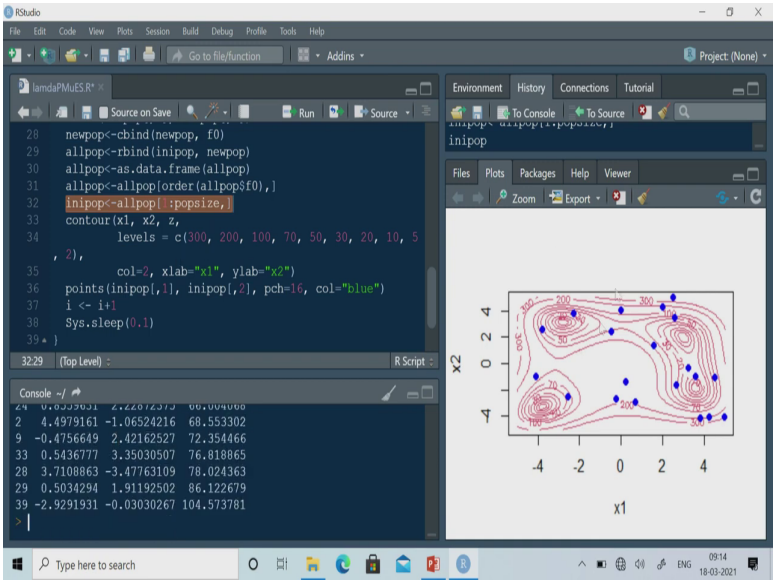
So therefore, what we have to do? So, we have to select the 20 best solution, so I will select the 20 best solution and that is that means I will select from 1st solution to up to 20 ok. So, I am doing that so I am using that I am storing at initials population now. So, from all pop so I am taking 1 to pop size ok, so 1 to pop size.

(Refer Slide Time: 18:39)



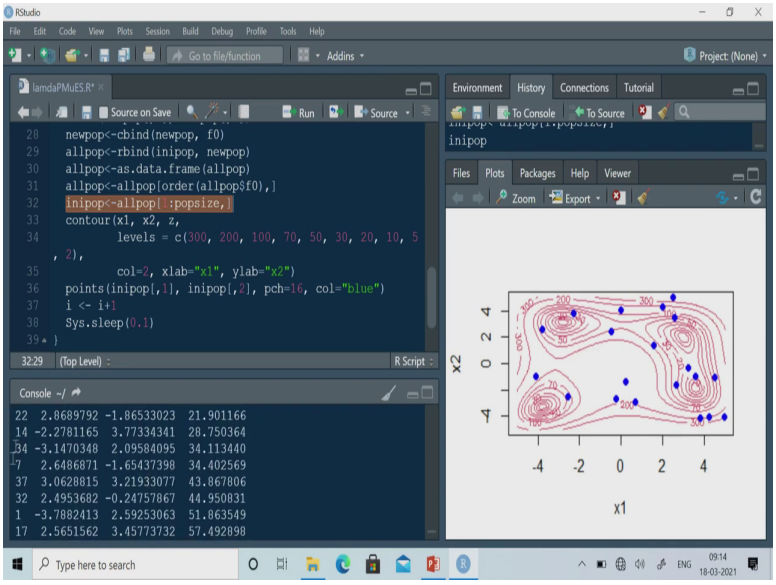
So, if I execute this so I will get the new initial population and that is basically I am storing at initial population. So, you can see that so this initial population.

(Refer Slide Time: 18:46)



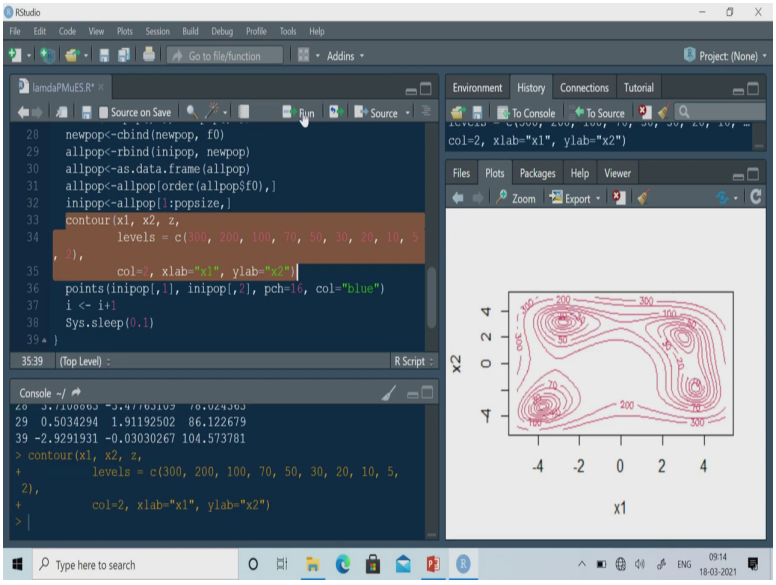
Now, will have total 20 solution.

(Refer Slide Time: 18:51)



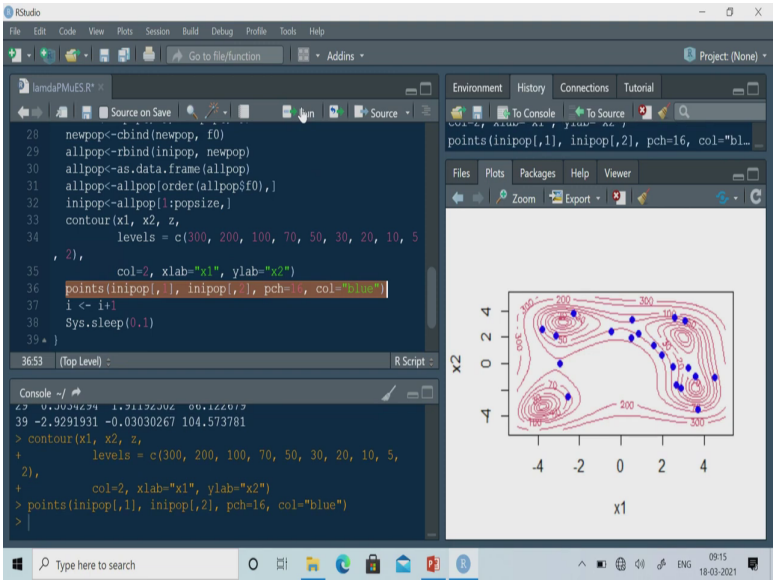
Anyway, so these are the order you are getting but you have total 20 solution. So, next line is so this is the initial population. So, whatever you have seen over the contour. So, I would like to have a different contour map and I would like to erase that one, so therefore I am executing this line again.

(Refer Slide Time: 19:07)



So, I am getting a different contour then I am putting the new population of what is contour ok.

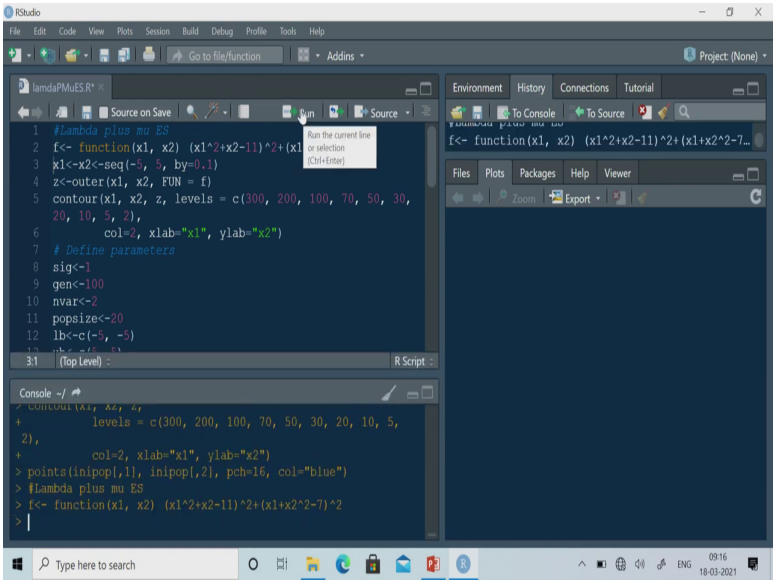
(Refer Slide Time: 19:17)



So, I am putting this one, so you can see if you are executing that one. So, this is the new population you are getting from the initial population. So, I have completed one iteration, then it will go to the next iteration and then system will sleep in order to see the animation. So, it will sleep for 0.1 and this will continue and finally I can means I can display the best solution and basically I also I can also display the function value of the best solution ok.

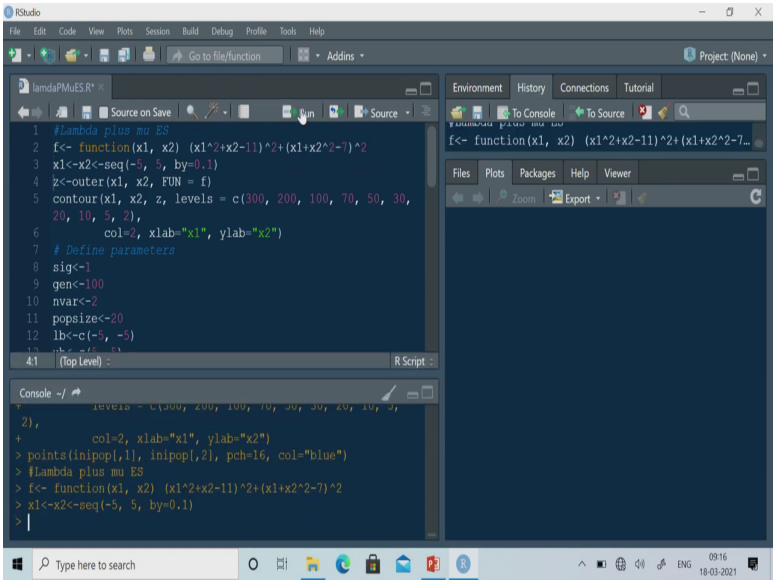
So, this is the program, so you just see it is not a very long program. So, only few lines program, so we have here around 41 line and as you know that some of the lines are just to plot this function, but if you are not plotting that excel algorithm will be hardly 15 to 20 lines ok So, now let us execute this particular program. So, from the beginning so I am cleaning everything now let us execute.

(Refer Slide Time: 20:20)



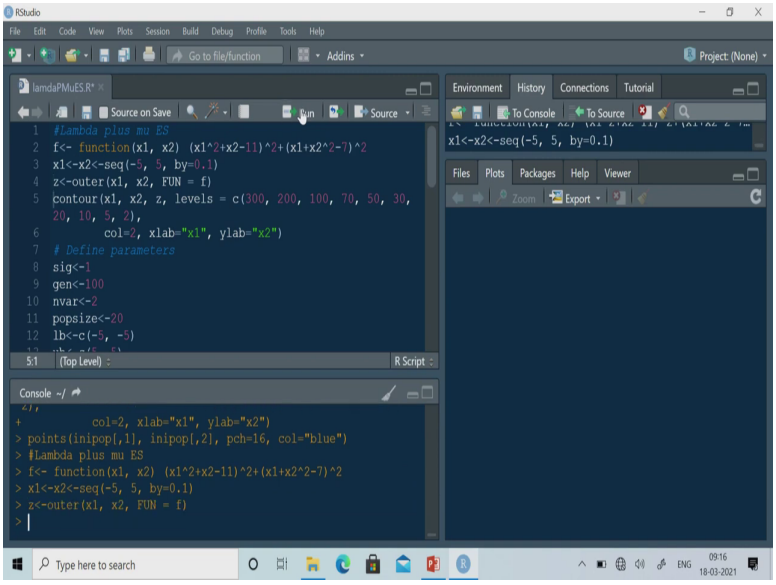
So, the first one is objective function.

(Refer Slide Time: 20:22)



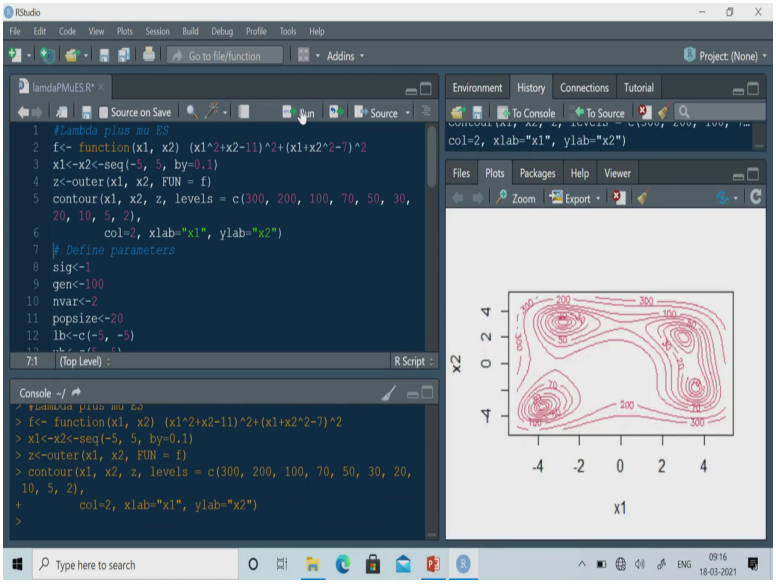
Then I am defining x 1 x 2.

(Refer Slide Time: 20:25)



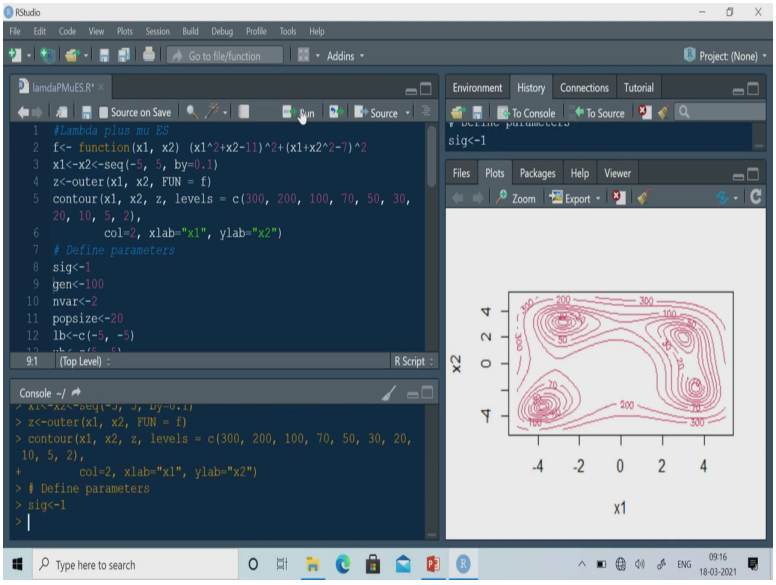
Then I am calculating z value.

(Refer Slide Time: 20:27).



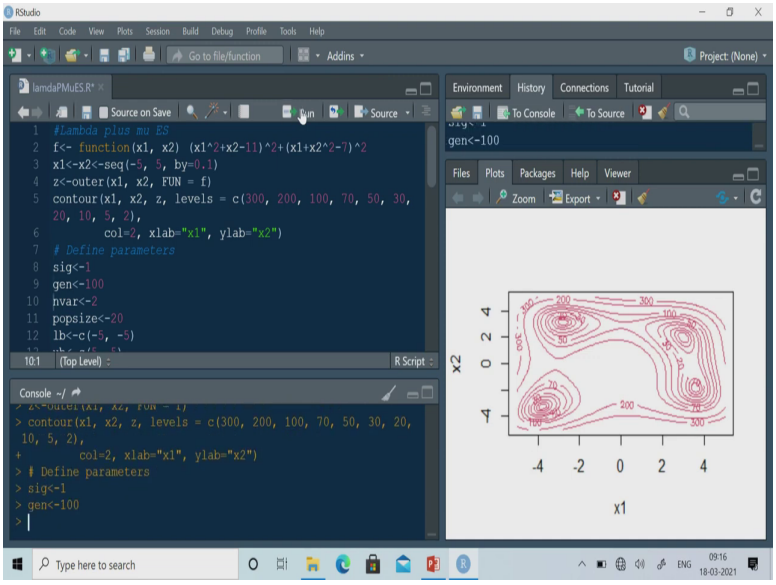
Then contour map.

(Refer Slide Time: 20:29)

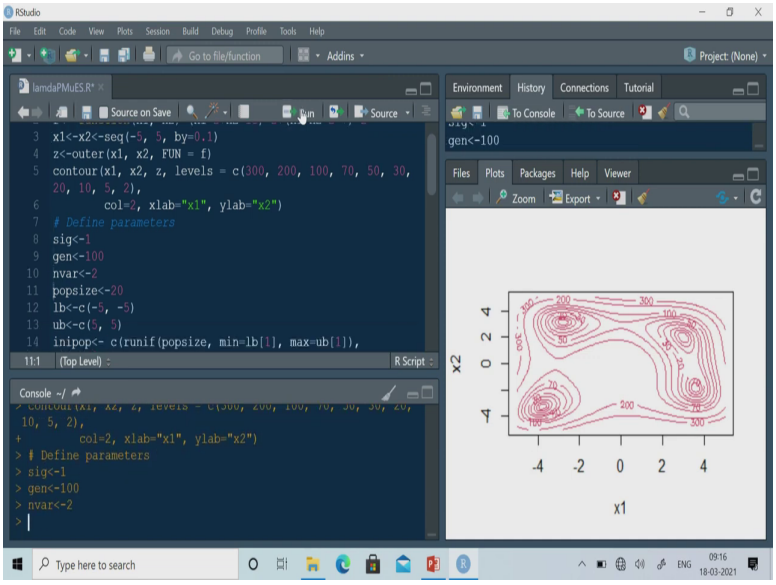


Then define sigma generation.

(Refer Slide Time: 20:31)

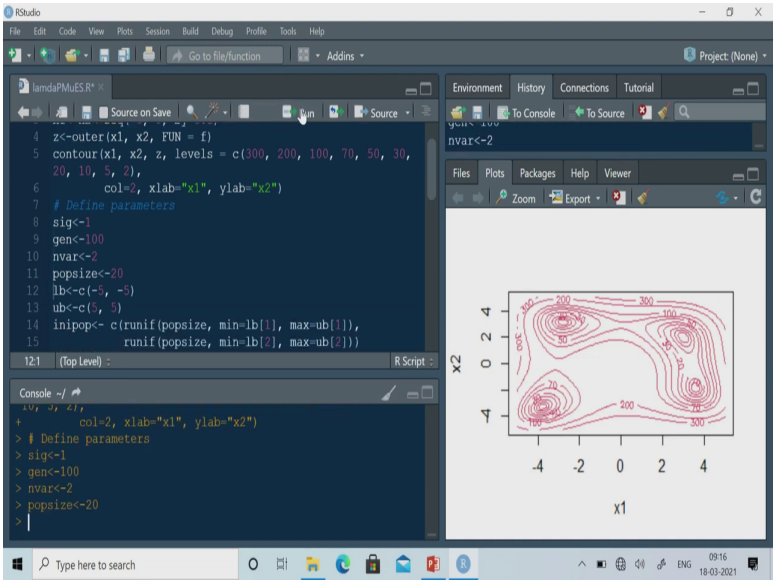


(Refer Slide Time: 20:32)



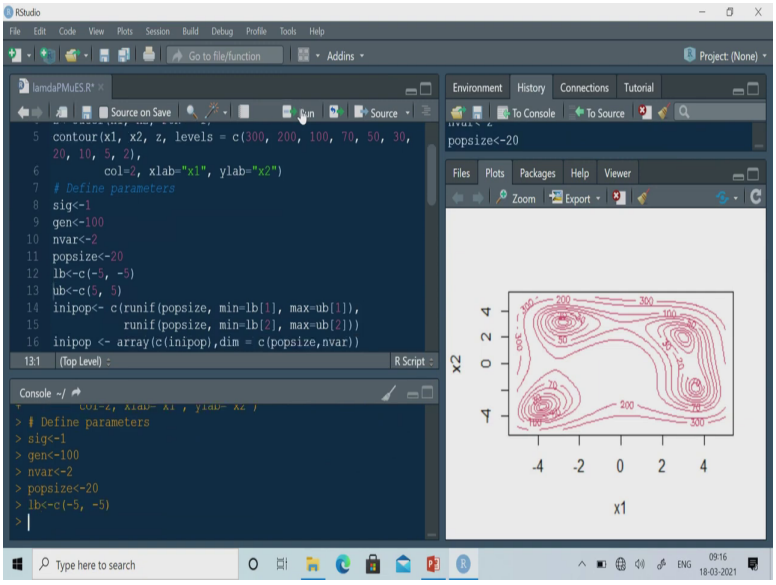
Number of variable.

(Refer Slide Time: 20:33)



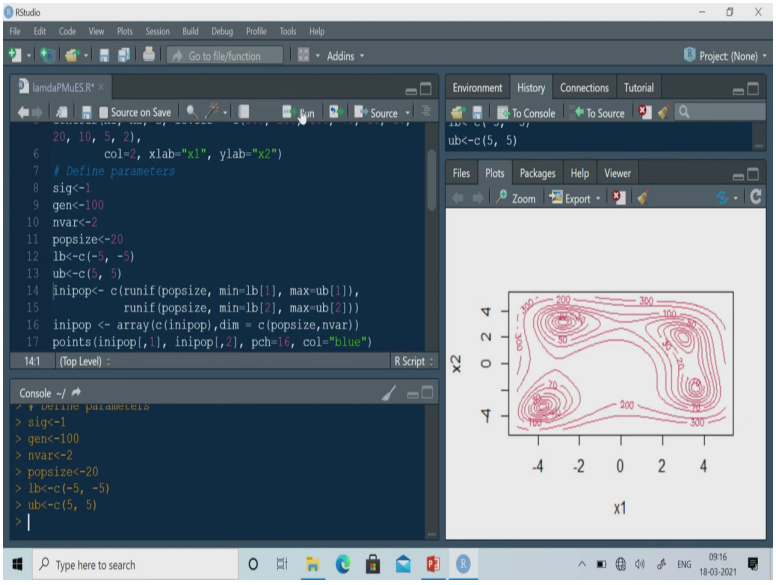
Population size.

(Refer Slide Time: 20:34)



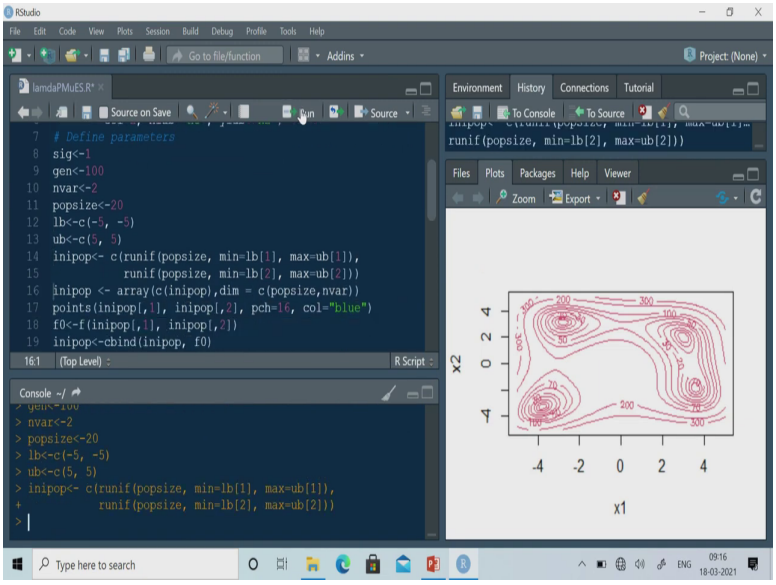
Lower bound.

(Refer Slide Time: 20:36)



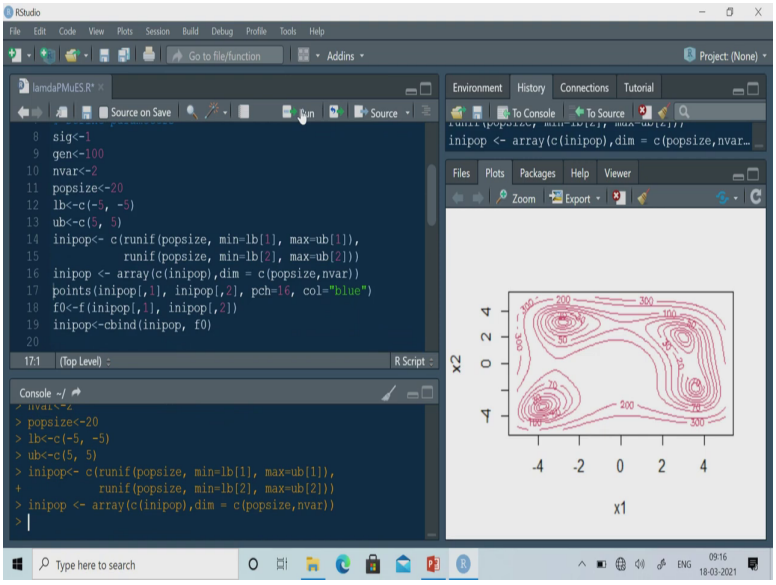
Upper bound.

(Refer Slide Time: 20:37)



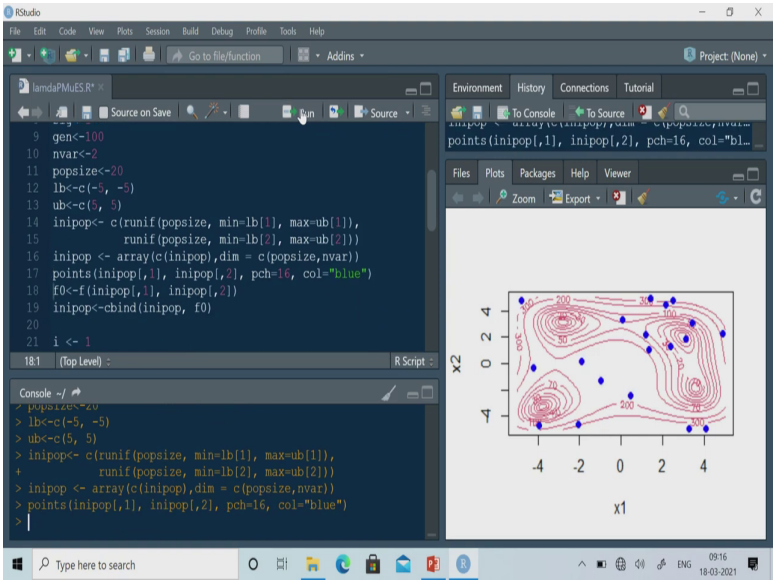
Initial population.

(Refer Slide Time: 20:38)



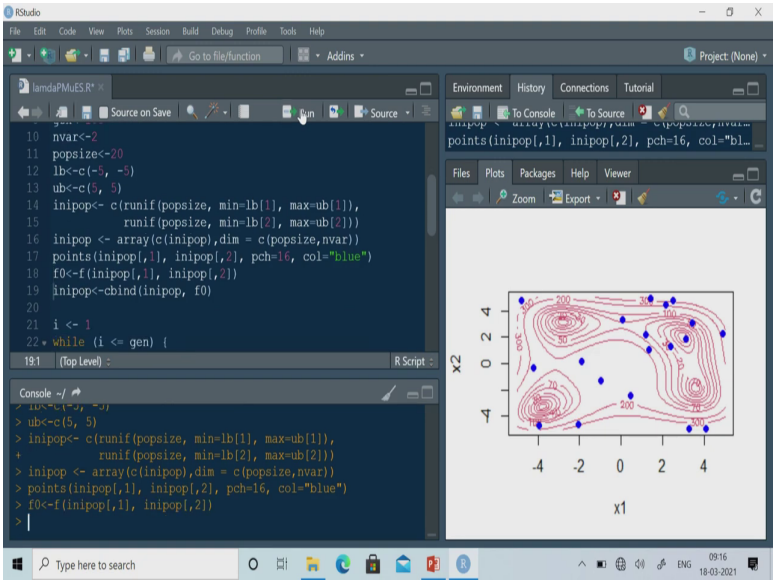
Making an array.

(Refer Slide Time: 20:40)



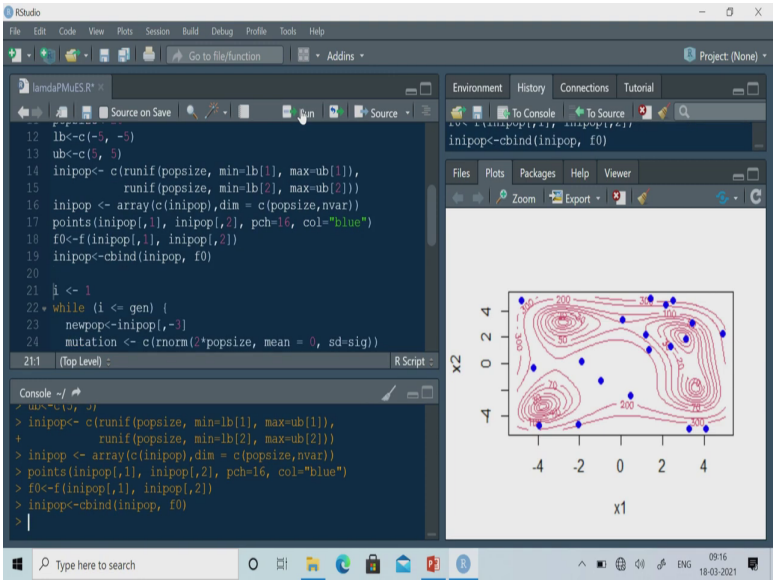
Then I am plotting this initial population over the contour map.

(Refer Slide Time: 20:45)



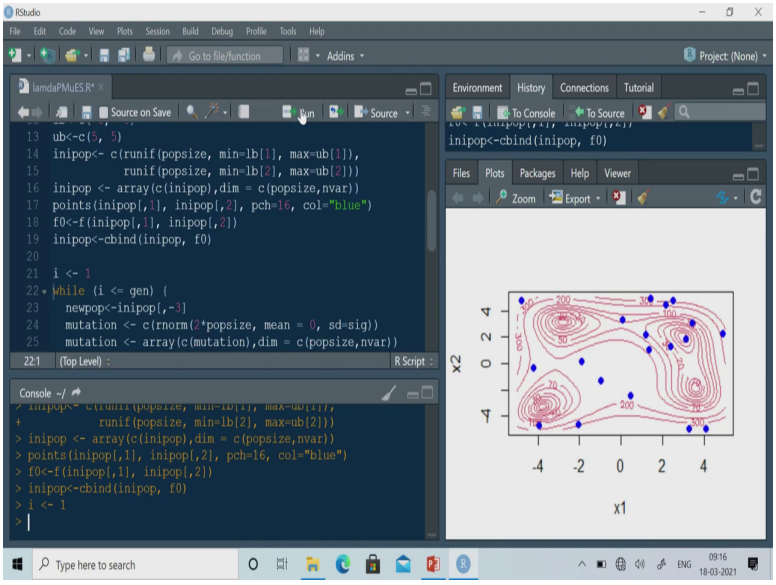
Then I am calculating the function value and that function value I am storing in initial pop.

(Refer Slide Time: 20:51)



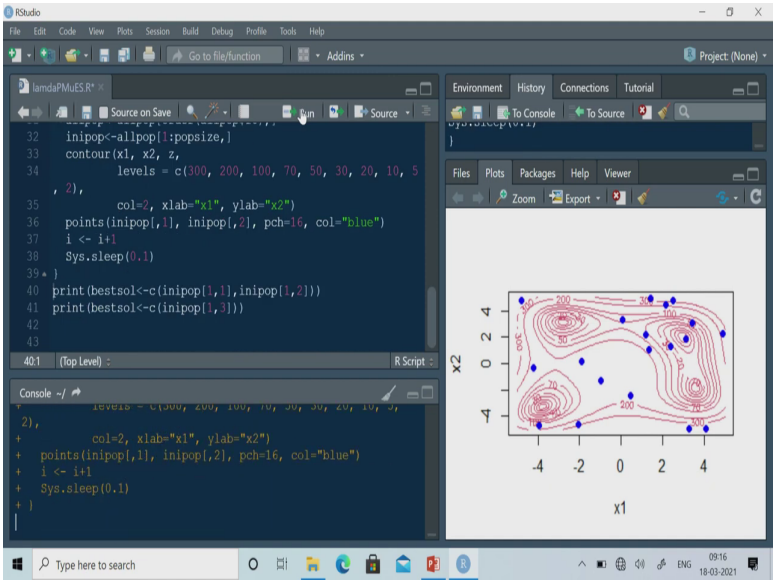
I am creating another column and then I am running this for while loop.

(Refer Slide Time: 20:58)

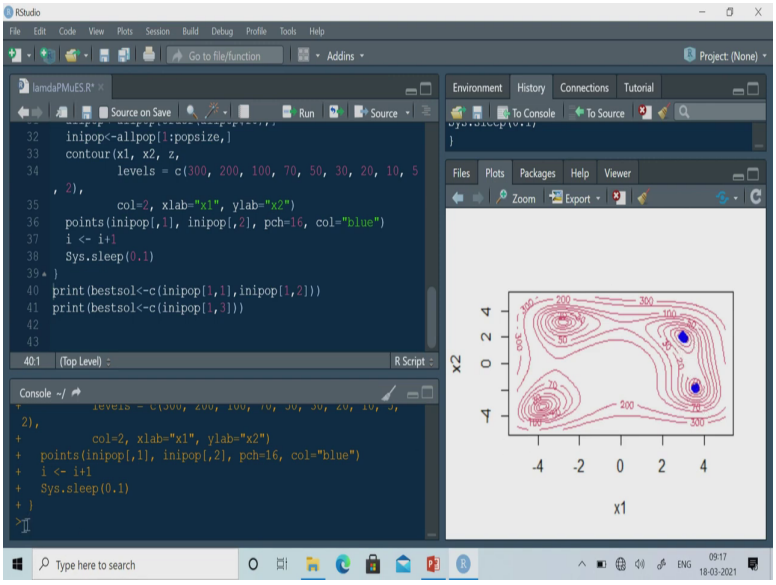


Then I am running this while loop.

(Refer Slide Time: 20:59)



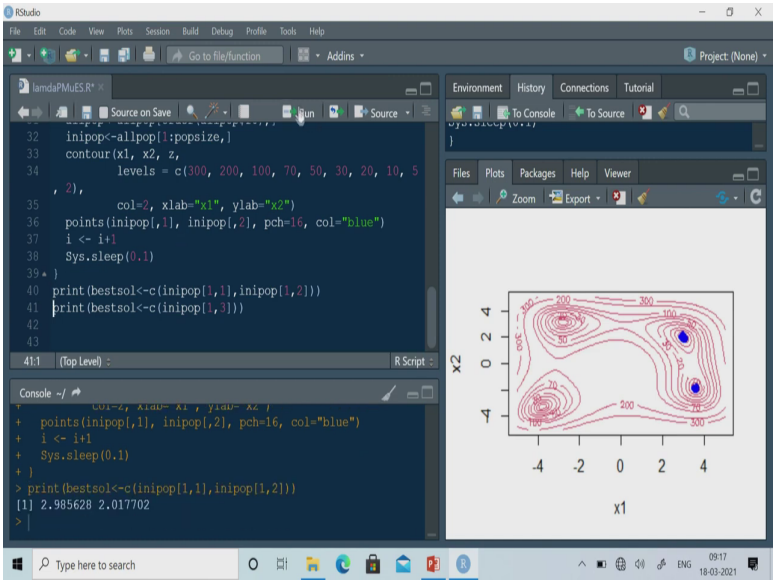
(Refer Slide Time: 21:03)



So, now you can see that it is trying to converge at one of the optimal solution and here what is happening because all these solutions the function value is same; that means, function value is 0, so therefore when you are comparing two solutions, so function value will be 0. So therefore, some of the population will be maybe at this solution and some of them will be at this solution. So, you are getting both the solution where function value is 0.

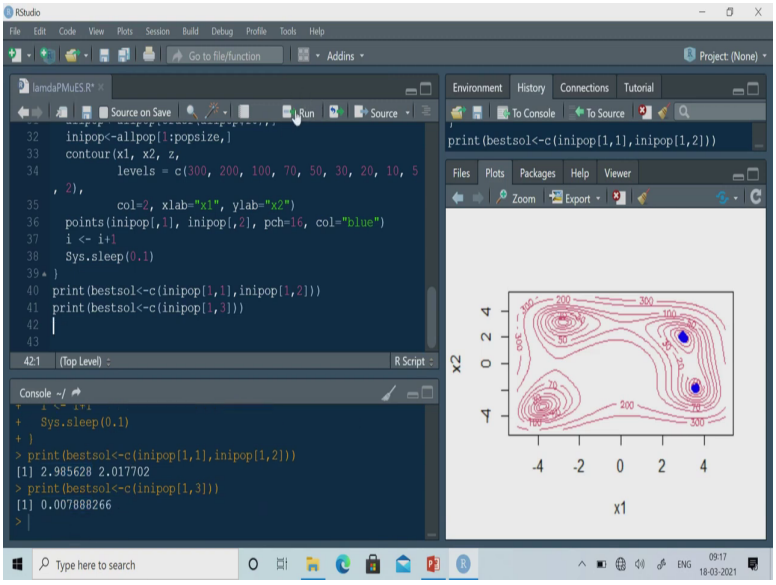
So, you can see that so all the solutions are converging at these two optimal solution or two optimal these two points.

(Refer Slide Time: 21:43)



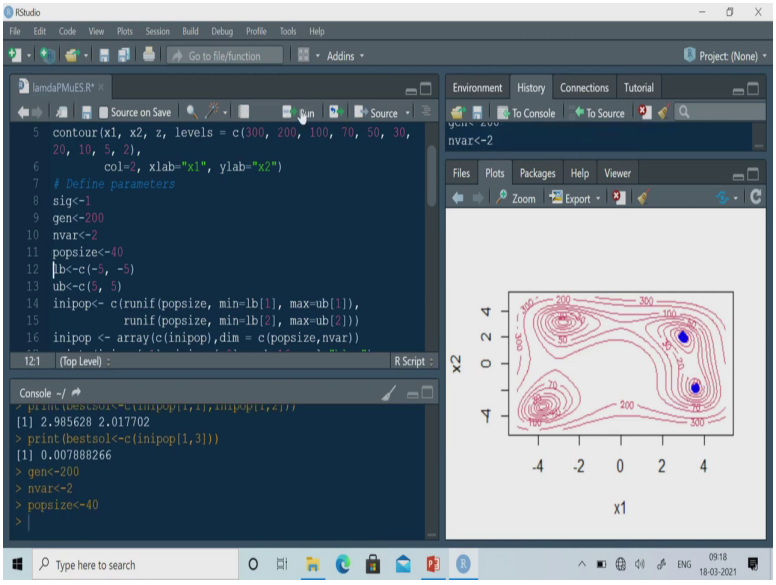
Now, I can see actually what is the best solution that is the 1st solution best solution is 2.98 and 2.01. So, I am not getting the exact solution, but this is the near optimal solution exact solution is 3 and 2, but I am getting 2.98 and 2.01 and the function value is 0.007.

(Refer Slide Time: 21:59)



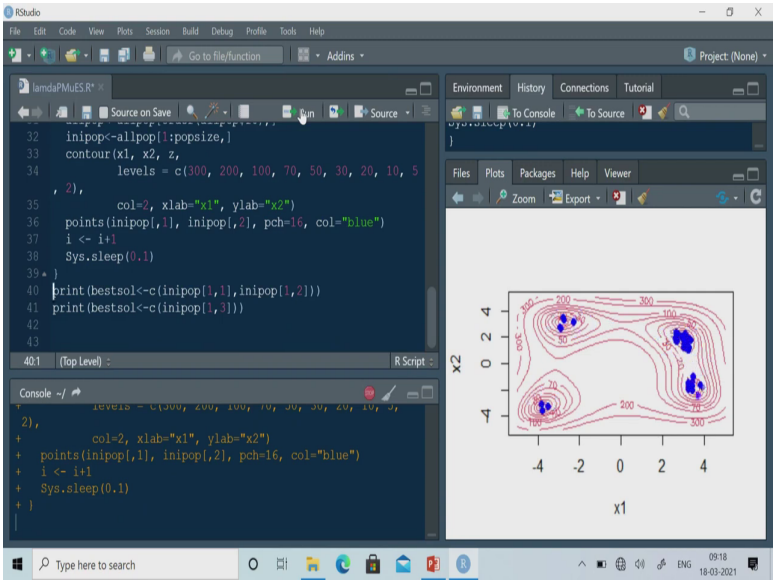
So, what I can do; I can increase this population size just see.

(Refer Slide Time: 22:11)



I would like to use 40 population suppose and maybe 200 iteration. Now, let us execute this.

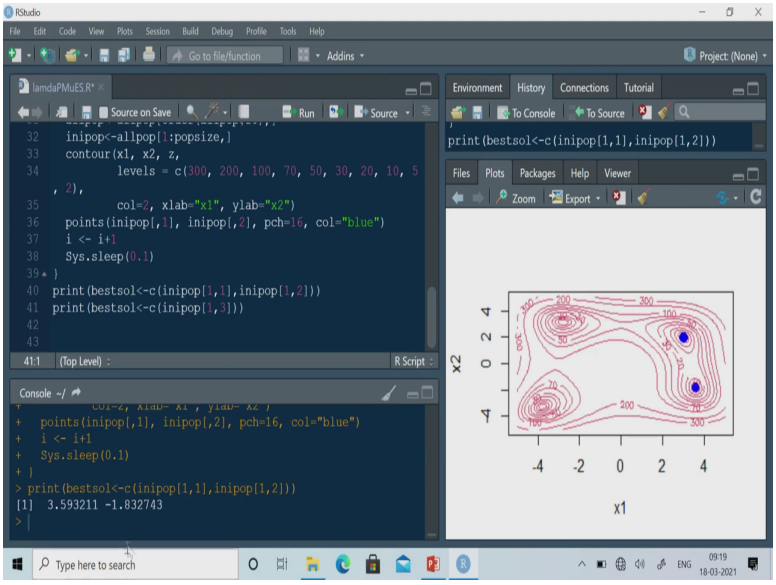
(Refer Slide Time: 22:27)



So, now, we have more population and just see whether we are getting the exact optimal solution or not. Now, you just see we are trying to get all these three anyway. So, when we are comparing this solution is vanished so again we are getting these two solutions.

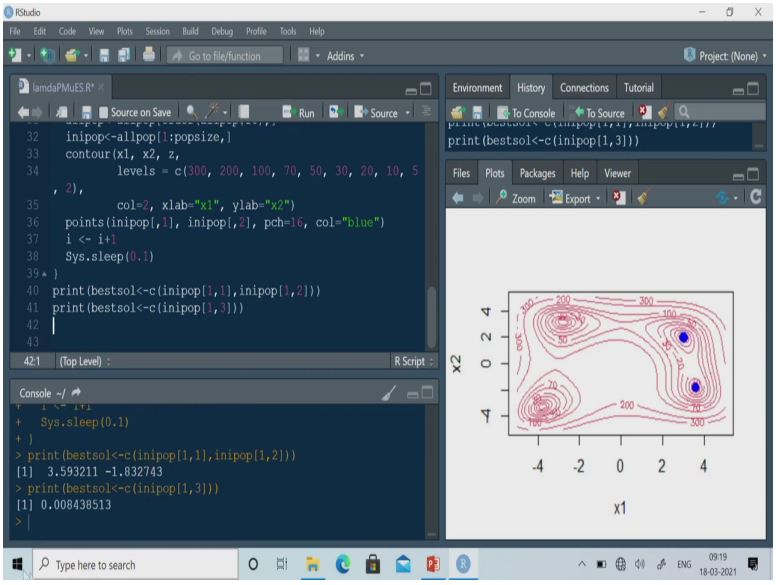
Let us see this yeah it is still running, because now we will go up to 200 iteration and population size is also more it is still running yeah. It has completed now so let us see the solution.

(Refer Slide Time: 23:02)



Now, the solution is 1 is 3.5932; that means, this on the 4th quadrant and this is minus 1.832743.

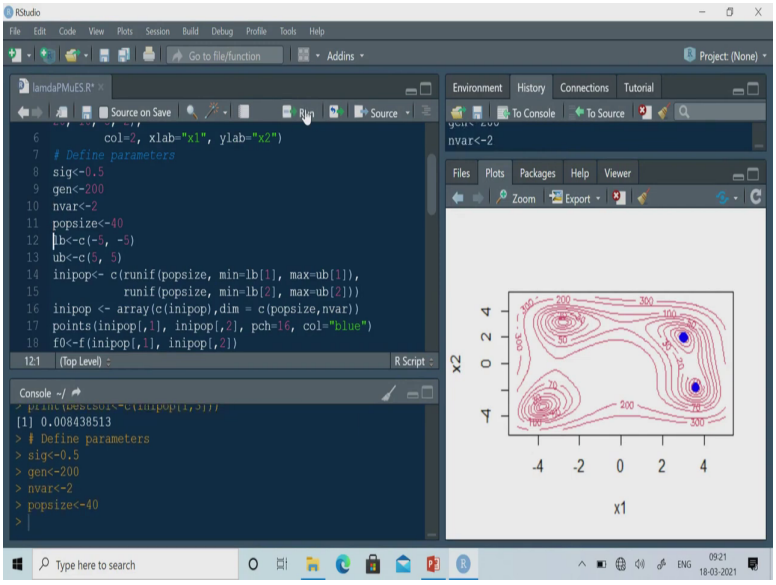
(Refer Slide Time: 23:16)



And just see what is the function value, so function value is 0.0084.

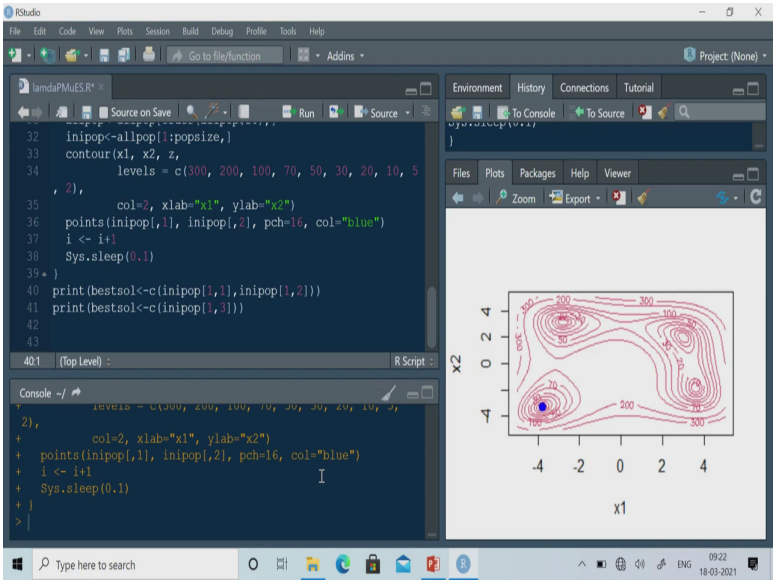
So, here we are not getting the exact optimal solution, because I have used sigma equal to 1, but if you are reducing the sigma value with iteration then you will get the exact you may get the exact optimal solution or maybe you will get a better solution than this.

(Refer Slide Time: 23:41)



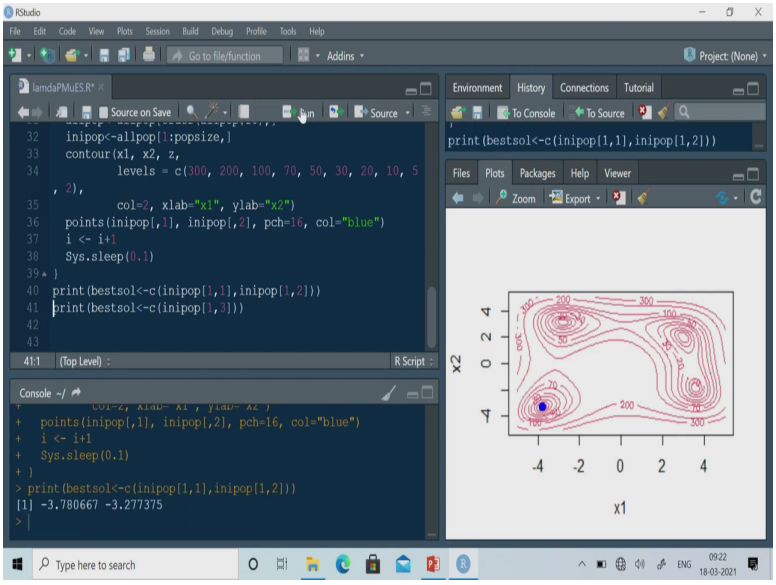
So, let us try with sigma of 0.5.

(Refer Slide Time: 23:48)



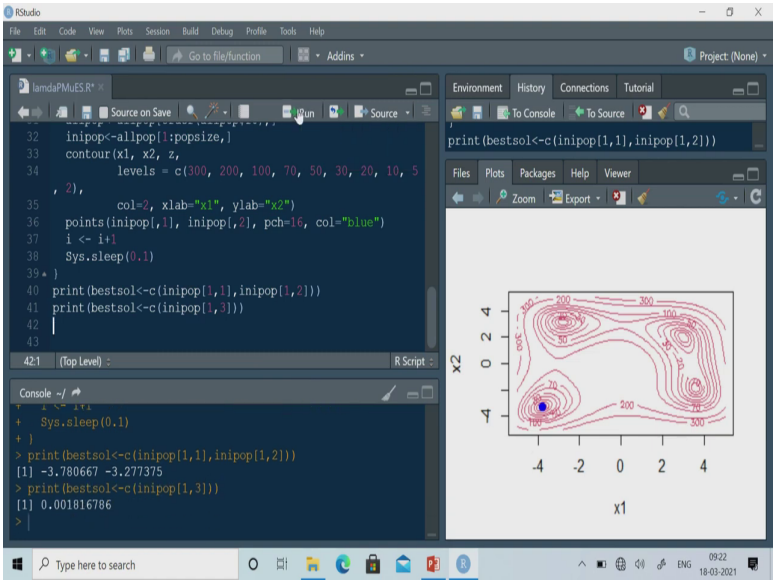
Now, let us see so it is still running. So, I am getting only one optimal solution. So, all the population are at this particular solution that is on the 3rd quadrant. So, it is still running. So, this time we may get a better solution because I have used lesser value of sigma, but the idea is that you start with a higher value of sigma and then you should reduce the value of sigma with your iteration. So, when you are approaching the optimal solution, so you should have a smaller value of sigma.

(Refer Slide Time: 24:40)



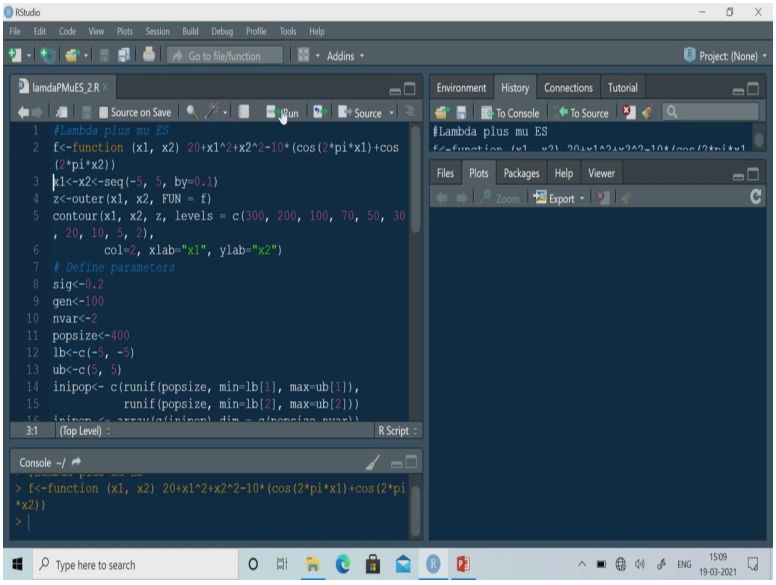
So, I got the solution just see yeah this time I am getting the solution at the 3rd quadrant, so that is minus 3.780667 and this is minus 3.277375.

(Refer Slide Time: 24:56)



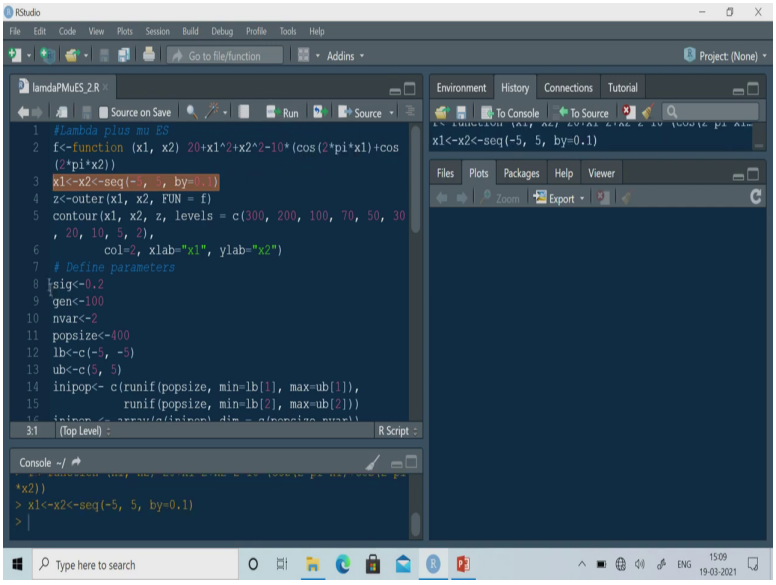
So, let us see so you just see now solution that function value is 0.0018 so slightly it is better solution. So, anyway so you are getting so even if you reduce your sigma value. So, maybe you may get even a better than this solution. Now, let us solve the second function using lambda plus mu ES.

(Refer Slide Time: 25:23)

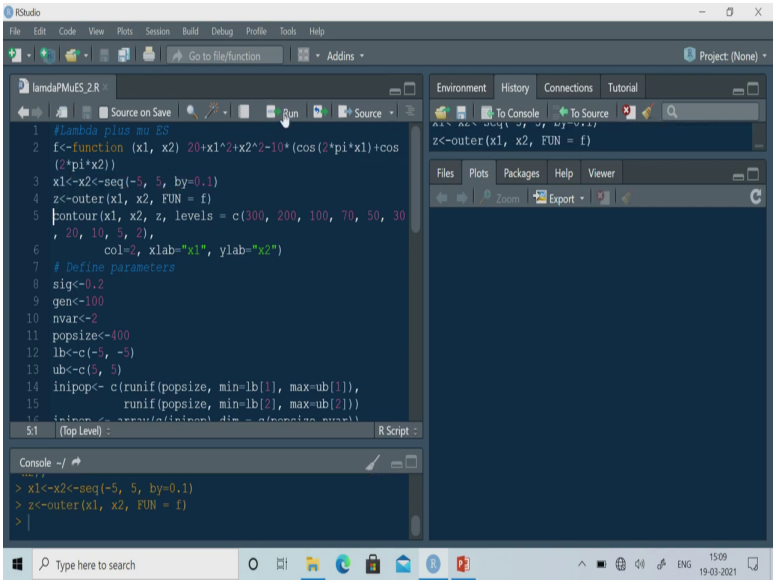


So, this function let us see so let us plot this function. So, this function is 20 plus x 1 square plus x 2 square minus 10 within bracket cos twice pi x 1 plus twice cos plus cos twice pi x 2. So, let us run this particular line and then find out then calculate the value of x 1 and x 2.

(Refer Slide Time: 25:49)

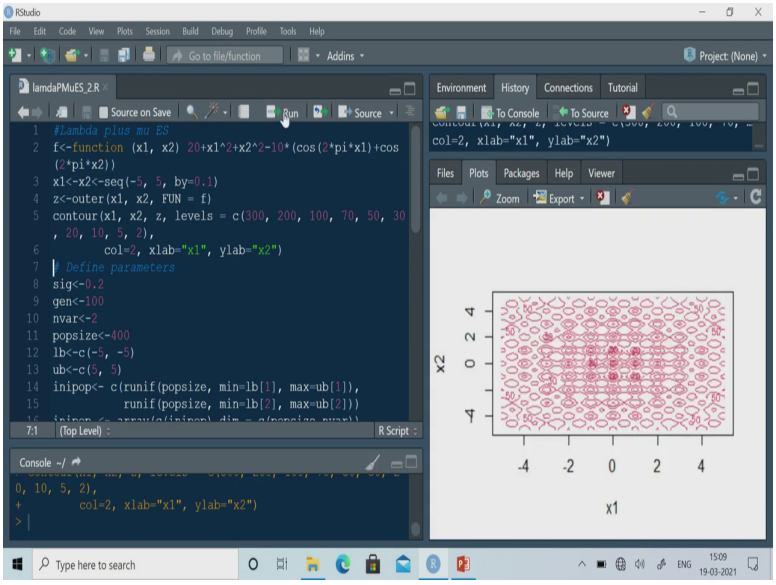


(Refer Slide Time: 25:54)

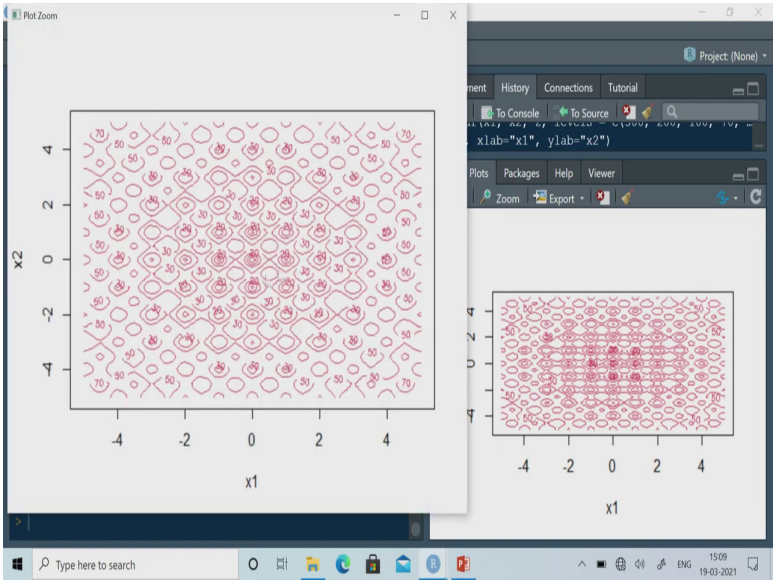


And then calculate the value of z and then plot it.

(Refer Slide Time: 25:55)

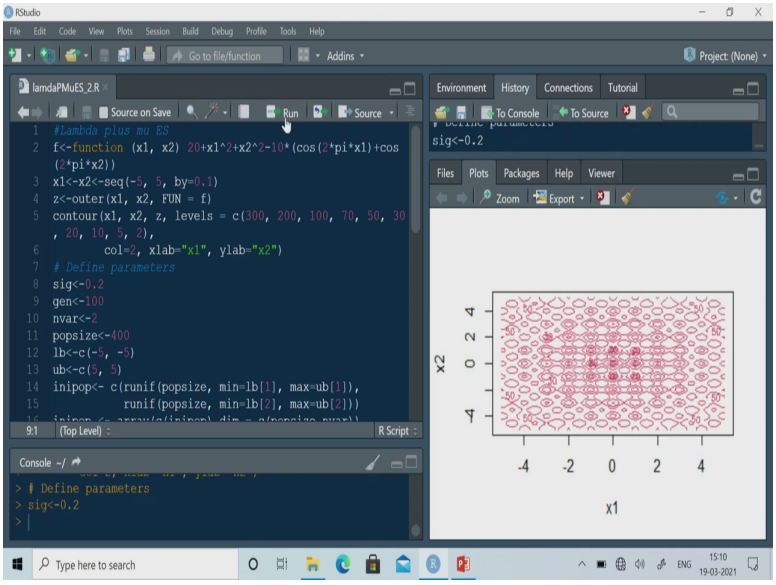


(Refer Slide Time: 25:57)



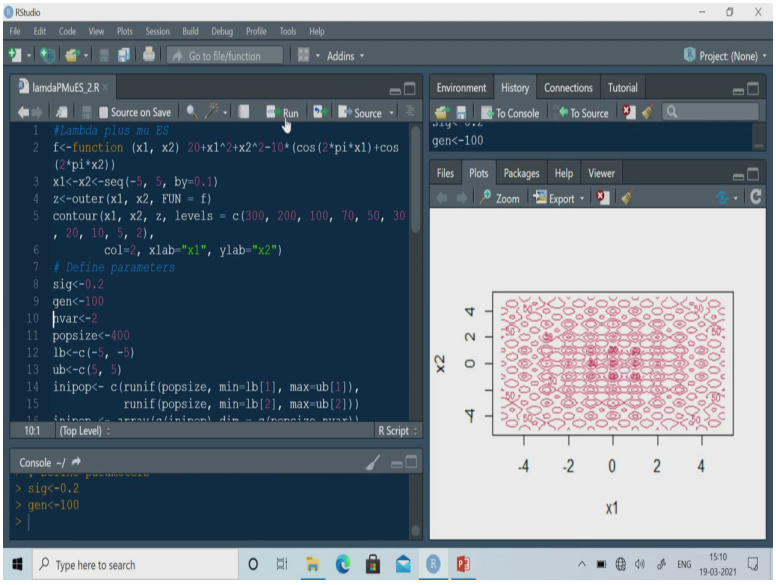
So, I am getting this plot so as you have seen. So, there are several local optimal solution and one of them is global optimal solution and global optimal solution is 0 0. So, somewhere here so this is the solution and we will try to find out the solution using lambda plus mu ES.

(Refer Slide Time: 26:18)



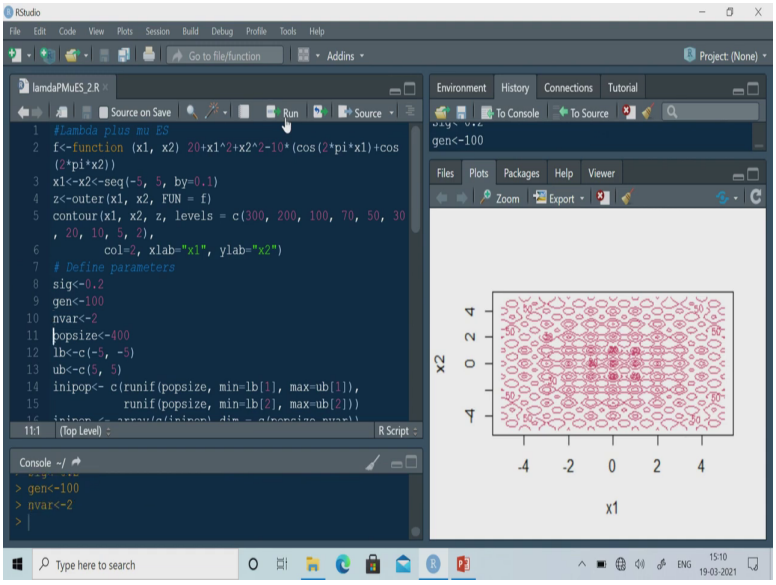
So, what we have to do we have to define the sigma value.

(Refer Slide Time: 26:21)



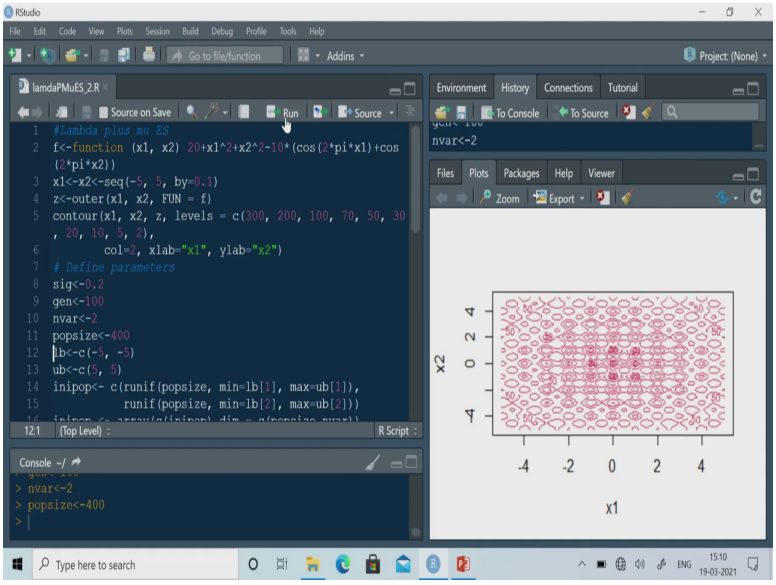
Then generation I am using 100 initially.

(Refer Slide Time: 26:24)



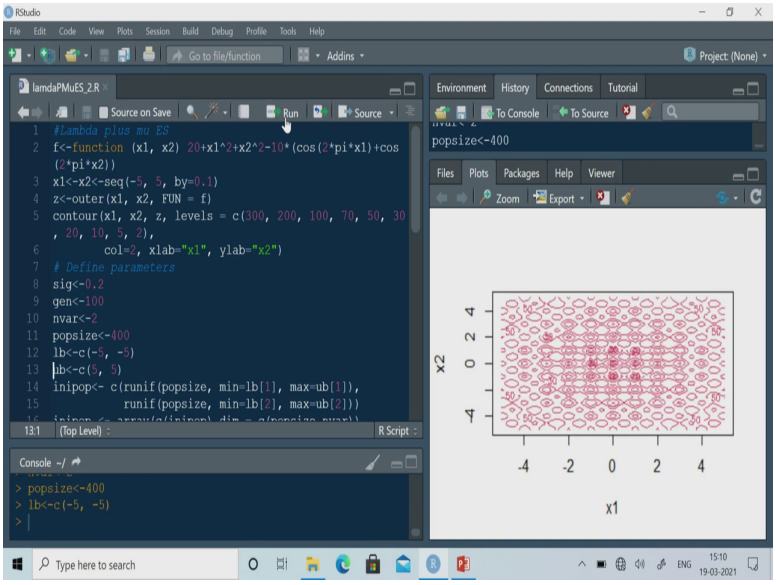
Then number of variable is 2 here.

(Refer Slide Time: 26:26)



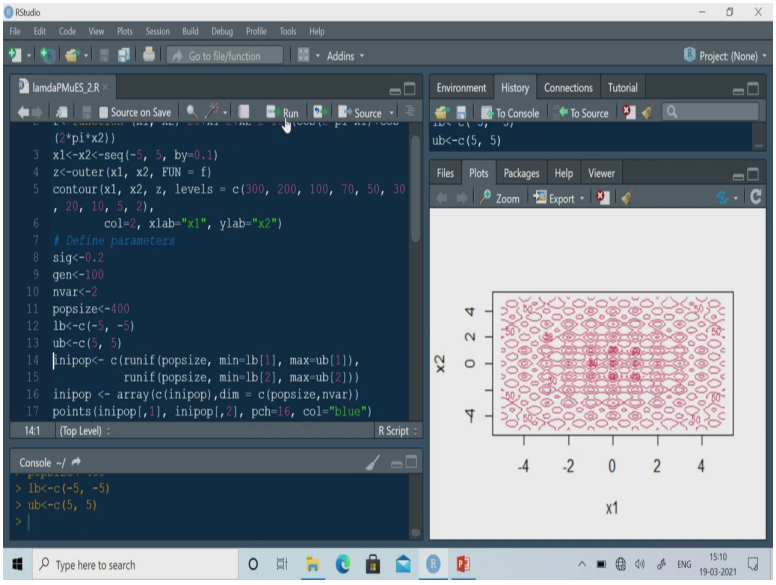
Population size 400.

(Refer Slide Time: 26:29)

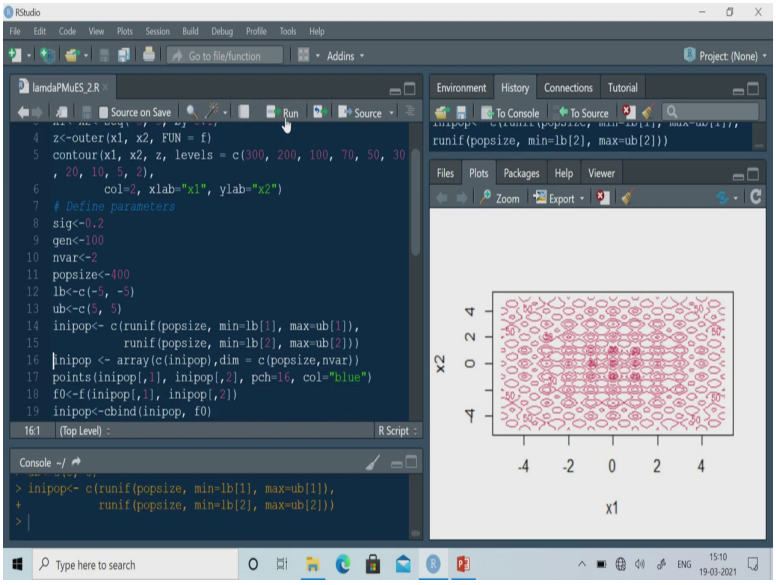


Lower bound minus 5 and minus 5 upper bound plus 5 and plus 5.

(Refer Slide Time: 26:31)

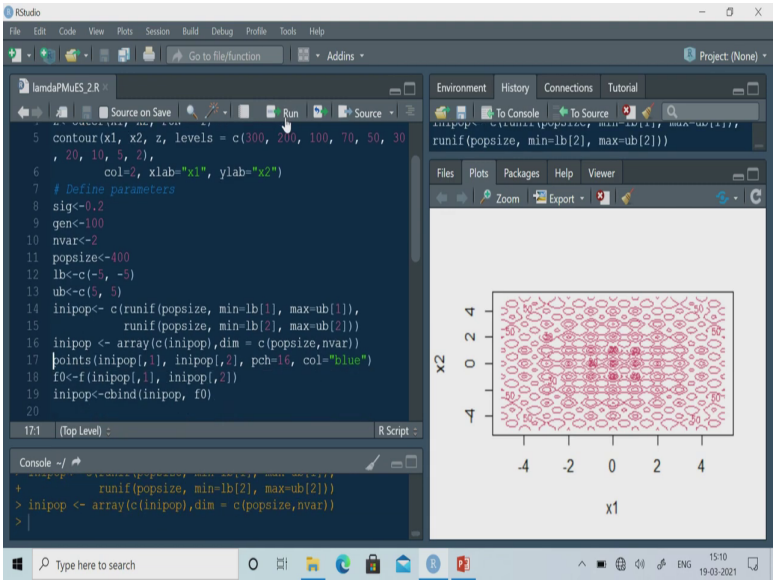


(Refer Slide Time: 26:34)



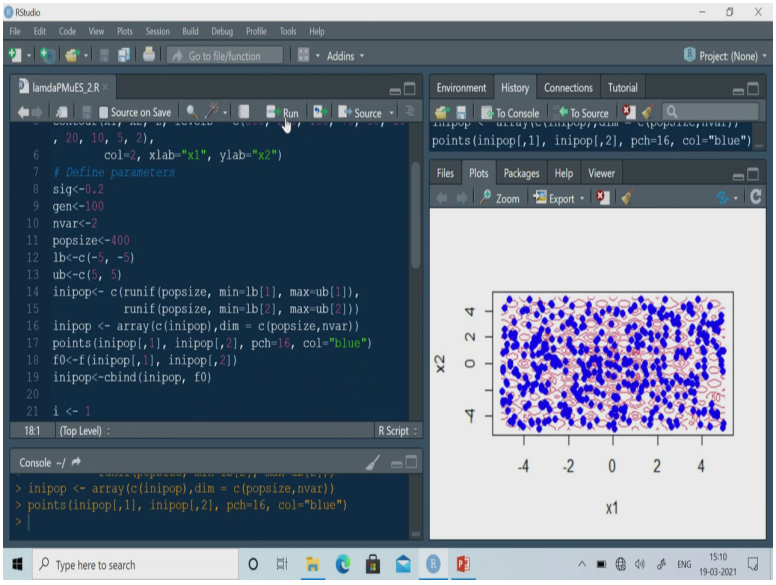
And then we are generating initial population.

(Refer Slide Time: 26:36)



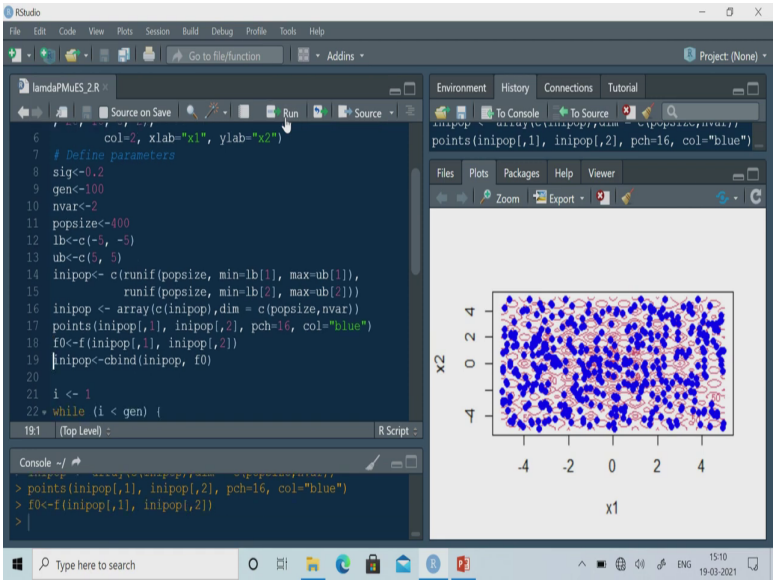
Then I am putting in an array.

(Refer Slide Time: 26:37)



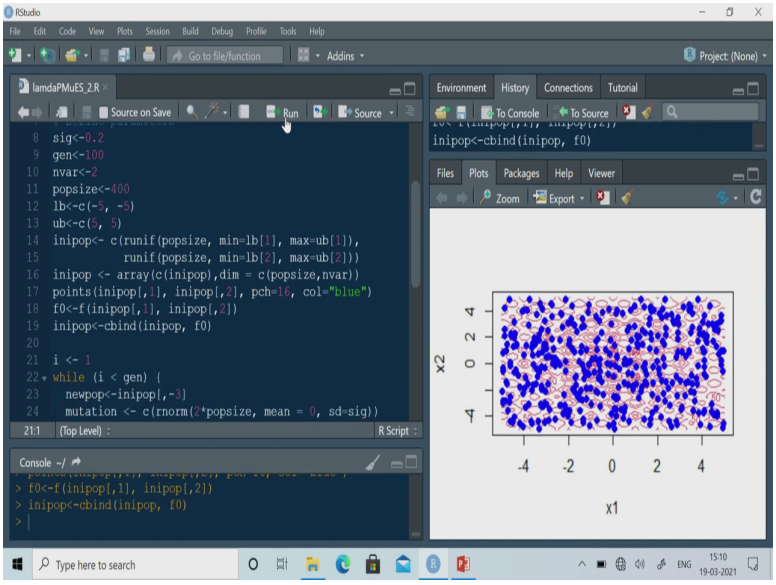
And then these are the initial population and then calculate the function value for this initial population.

(Refer Slide Time: 26:48)



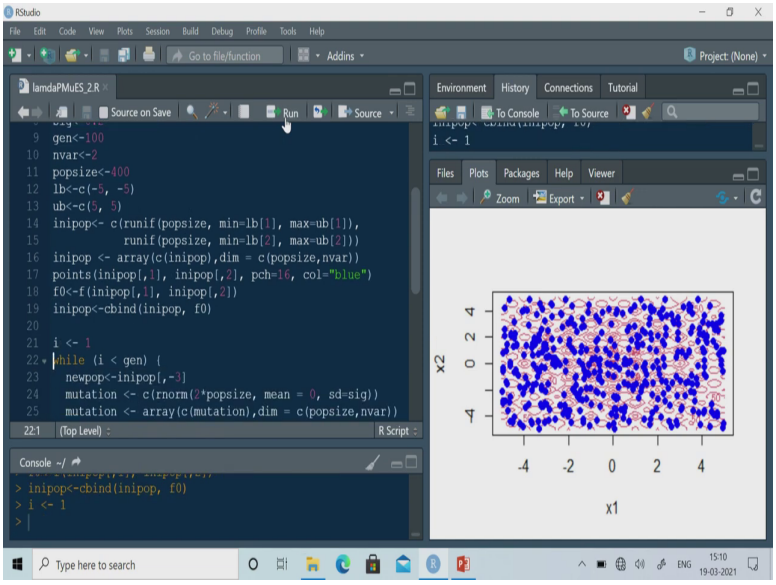
So, I am putting this function value that is I am putting this function value in initial population itself in the 3rd column.

(Refer Slide Time: 26:58)

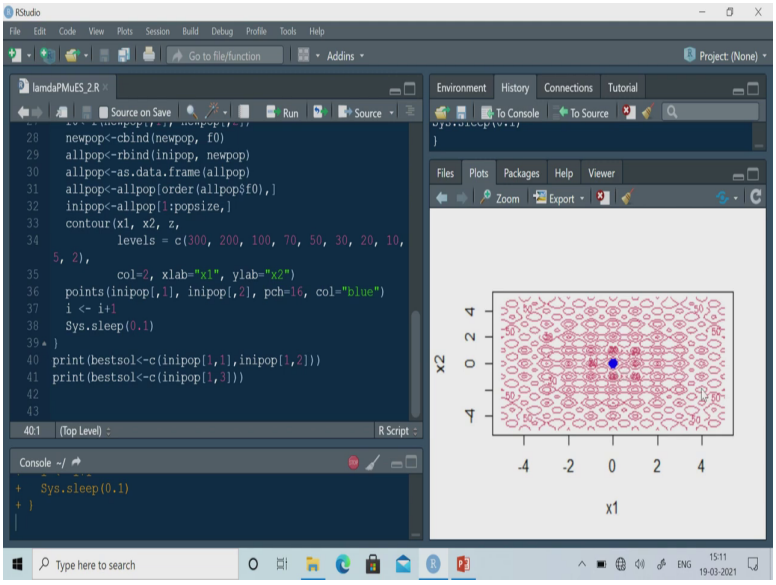


And then I am executing this while loop.

(Refer Slide Time: 27:02)



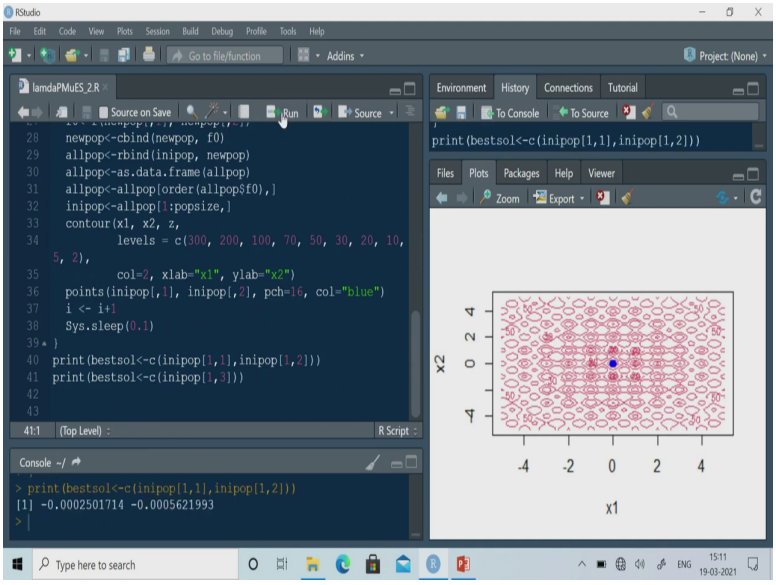
(Refer Slide Time: 27:04)



So, you can see that so with iteration this function this population is trying to converge at global optimal solution, you can see initially it was all over the surface. But, with iteration it is trying to converge at the global optimal solution.

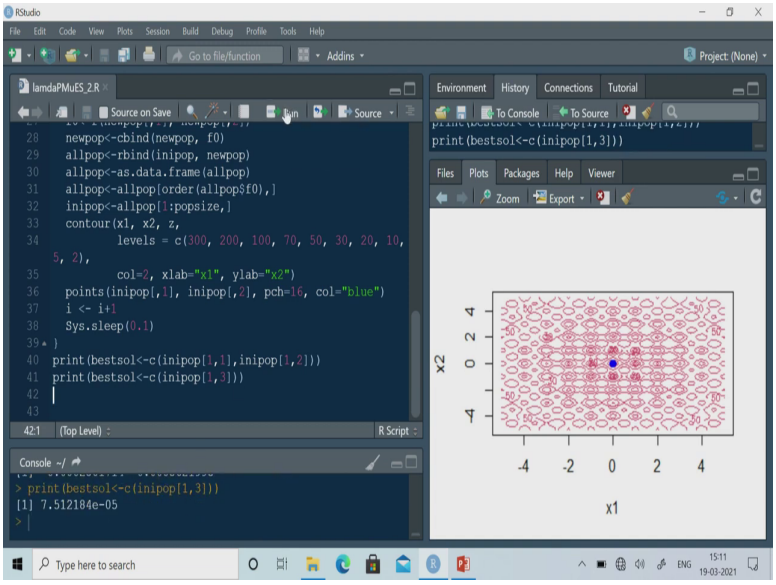
So, global optimal solution is somewhere at 0 0 and let us see whether we can or this we can get the global optimal solution or not. So, still running let us see still running. So, we have given generation of 100, so it will go up to 100. So, let us see whether we are getting the solution yeah.

(Refer Slide Time: 27:42)



So, we are getting the solution; solution is 0 0, but it is not exactly 0. But, it is near to 0 that is 0.00025 and here it is 0.00056.

(Refer Slide Time: 27:57)



So, best solution is 0, but we are getting 7.512 10 to the power minus 5. So, I can improve this solution by reducing the sigma value, but what we can do basically I can use this ES algorithm to find out a solution near the global optimal solution and after that we can use the hybrid optimization technique.

So, in the next class I will show you how we can apply hybrid optimization technique; that means, what we can do so we can initially we can apply ES and after that we can apply a classical optimization method and in that case you will get the exact optimal solution of this problem ok.

So, I can use any method any meta heuristic method to get a near optimal near global optimal solution and after that I can use a classical optimization method. So, then in that case we are

calling it hybrid technique; that means, we are using meta heuristic optimization algorithm as well as classical optimization algorithm. So, that I will show you in the next class.

Thank you.