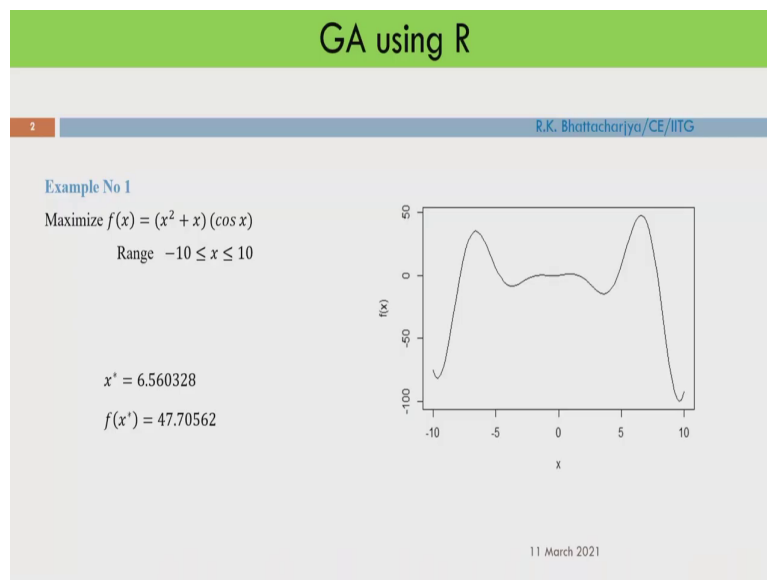


Optimization Methods for Civil Engineering
Prof. Rajib Kumar Bhattacharjya
Department of Civil Engineering
Indian Institute of Technology, Guwahati

Lecture - 23
GA Using R

Welcome back to the course on Optimization. So, today, we will solve some non-linear optimization problem using GA and we will use R platform. So, let us see the problem that will be discussed in this particular class. The first problem is a maximization problem. So, it is a single variable function.

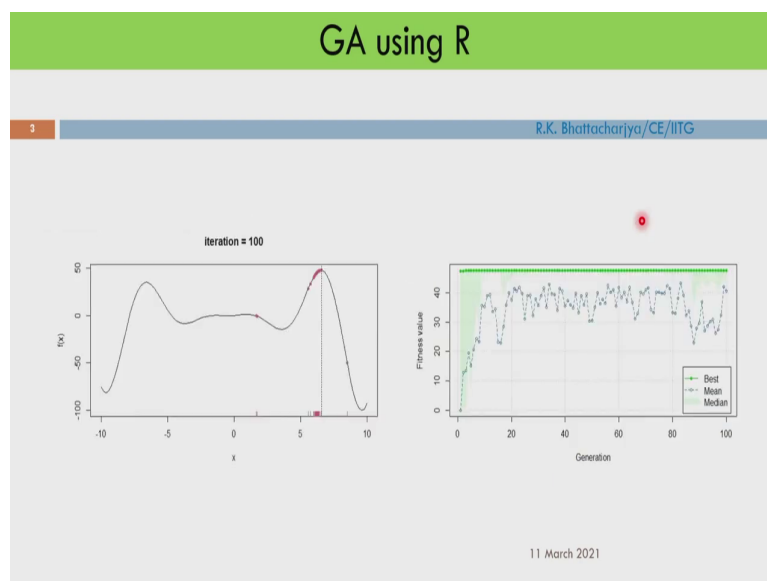
(Refer Slide Time: 00:56)



So, the function is $f(x)$ equal to $x^2 + x$ into $\cos x$. So, the range for x is between minus 10 and plus 10. So, there are two optimal solutions so as you have seen on the graph. So, in this particular function, there are two optimal solutions.

So, you can see that there are two maxima and we will try to find out the global maxima between minus 10 and plus 10 using genetic algorithm and the solution is x^* equal to 6.560328. So, that is the solution. So, this is the maxima of this particular function and the function value at maxima is 47.70562.

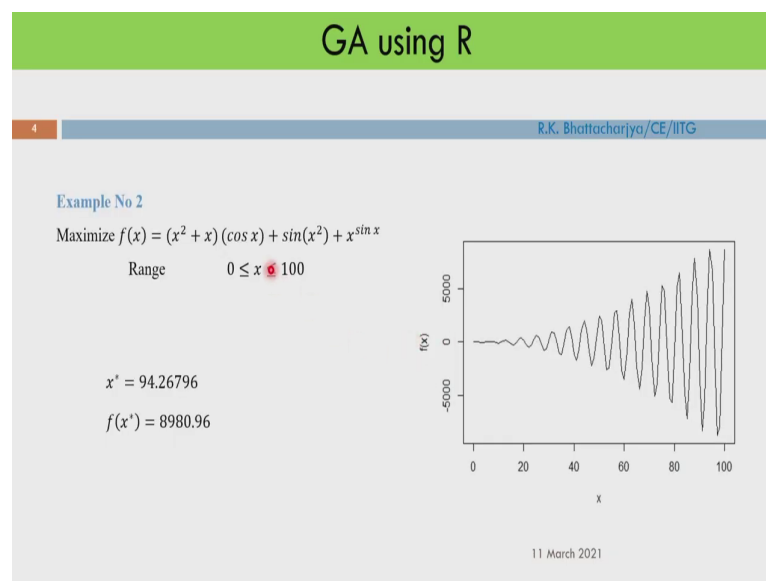
(Refer Slide Time: 01:49)



So, this is the first problem, we will solve using genetic algorithm. And so, this is the solution of this problem, when we applied genetic algorithm and you can see basically that all the population. So, it is at 100 iterations and we have given maximum iteration of 100 here and you can see that at 100 iteration, all the solutions are near the global maxima and this figure is showing fitness versus generation curve and you can see the base fitness, then the mean value as well as the median value.

And you can see the with the progress of generation, so with the generation, the mean value is increasing and base value anyway. So you got the base value probably at after few iteration, may be around 5-6 iteration. So, you have got the solution of this problem and then, mean value of all these populations are increasing with the generation.

(Refer Slide Time: 02:42)



The second problem is also a maximization problem. So, you can see. So, again, this is a single variable function. The function is $f(x) = x^2 + x \cos x + \sin x^2 + x^{\sin x}$ and the range for x is 0 and 100. So, x is between 0 and 100 and if we solve this problem, so it has several maxima and several minima and here, we are trying to find out the maxima.

You can see that there are several maxima of this particular function and one of them is the global maxima and genetic algorithm will give you the global maxima here and the solution is

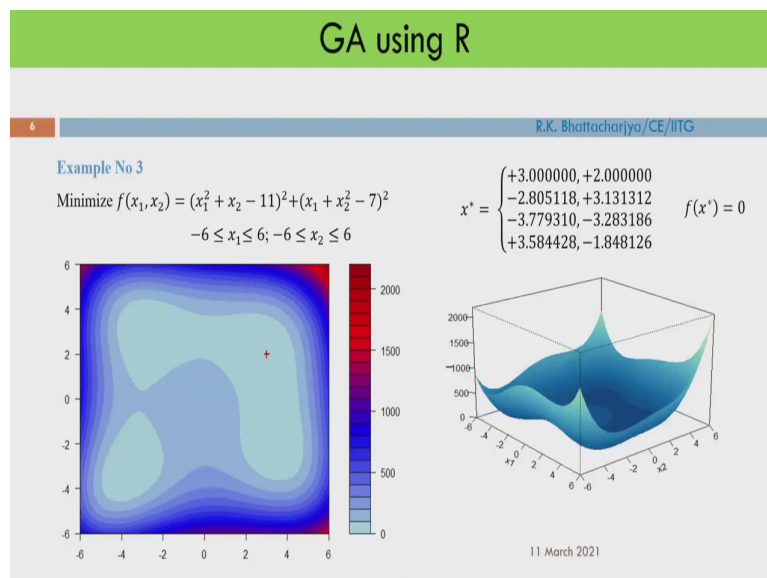
x^* equal to 94.26796. So, this is the solution and this point and the function value at that optimal global maxima is 8980.96. So, this is the solution of this problem.

So, we will solve this problem using genetic algorithm and you can see here that if we are applying, if we apply classical optimization method here. So it is quite difficult or it is very difficult to get the global maxima of this or global minima of this particular problem; but genetic algorithm will give you the global maxima of this problem.

So, we will solve this problem using genetic algorithm. Then, this is the solution of this problem. So, you can see that at 100 iterations, so most of the population are at global maxima. So, this is the global maxima somewhere here and you can also see that fitness the curve between fitness value and generation.

So, you are getting the optimal solution probably at around 50 iteration and this is also showing the mean value, how mean value is increasing and at the end, all the values are near the global maxima. So, this problem also we will solve using genetic algorithm and the third problem is a multi-variable problem. So, it is a two variable problem.

(Refer Slide Time: 04:52)



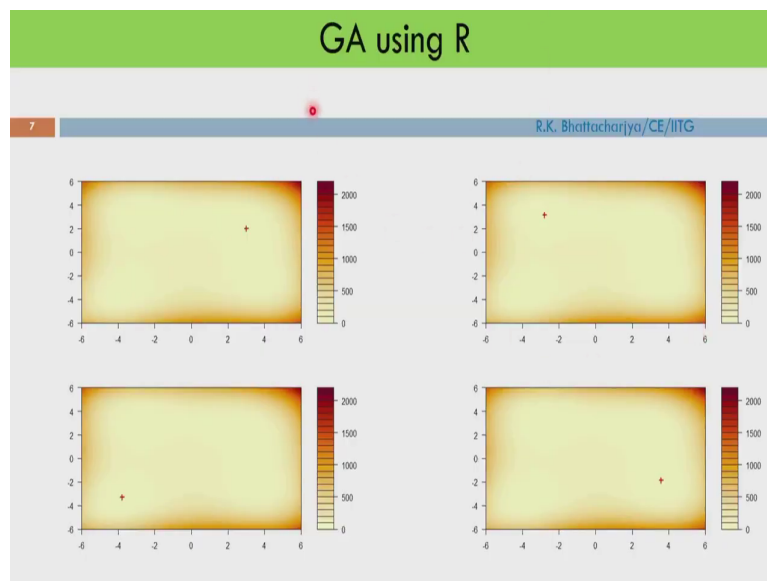
So, the function is given here. So, you can see the function is a minimization function or it is a minimization problem. It is a two variable function of x_1 and x_2 and f of x_1 and x_2 is $x_1^2 + x_2 - 11$ whole square plus $x_1 + x_2^2 - 7$ whole square. And range is minus 6 to plus 6 for x_1 and similarly, for x_2 also minus 6 to plus 6 and there are as you can see from this contour map, there are total four optimal solution and all of them are minima and these solutions are given here.

So, x^* equal to there are four optimal solution x^* is equal to 3 and 2, so this is at the first quadrant. And then, you have minus 2.805118 and 3.131312, so that is on the second quadrant. And then, there is another solution at the third quadrant that is minus 3.779310 and minus 3.283186; so, that is on the third quadrant.

And you have another optimal solution and that is on the fourth quadrant and the solution is 3.584428 and minus 1.848126. So, there are four optimal solution and function value at all these optimal points is 0. So, therefore, this particular function has alternate optimal solution.

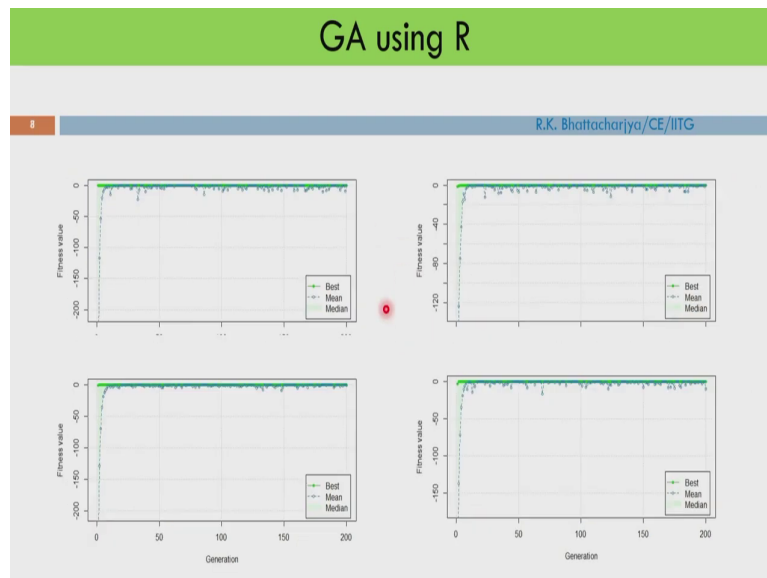
So, function value is same, but the solutions are different ok; all of them are minima and function value is 0 at all this optimal solution. So, I have also shown you the surface plot. So, you can see. So, there are total four optimal solution. So, this problem also we will try to solve using genetic algorithm. So, if you apply genetic algorithm, so you will get one of these four solutions.

(Refer Slide Time: 06:56)



So, I have shown here the four solution. So, you can see that one solution at the first quadrant, then there is another solution at second quadrant, another solution at third quadrant and another solution at fourth quadrant.

(Refer Slide Time: 07:13)

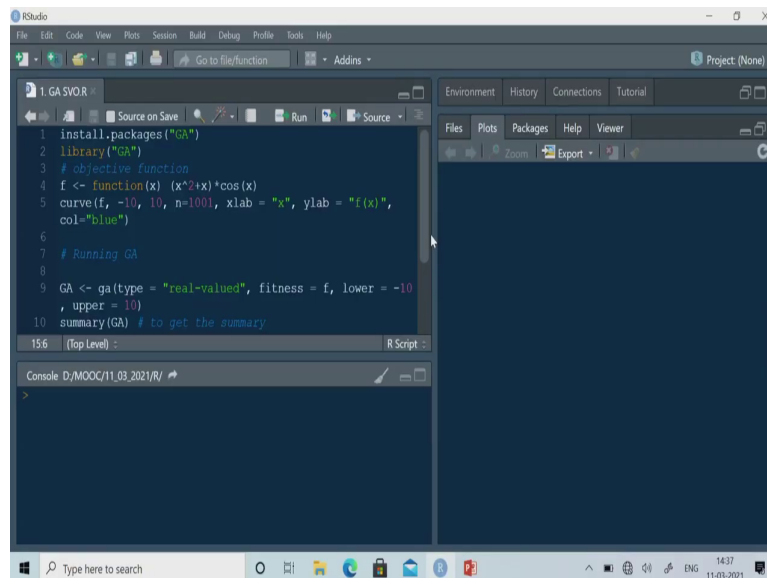


So, this is basically the solution obtained using genetic algorithm and I have also shown here, the fitness versus generation curve and you can see that genetic algorithm is so powerful that it can get the or it can reach the optimal solution probably after few iteration. So, here maybe after 10-15 iteration, so genetic algorithm is converging at one of the optimal solution.

So, now, we will try to solve this problem using R software. So, I will show you, already you have learned what is R and how to solve the classical problem using R. So, today, we will

learn how to solve genetic algorithm using R. So, today, we will solve some problem using genetic algorithm.

(Refer Slide Time: 07:58)



```
1 install.packages("GA")
2 library("GA")
3 # objective function
4 f <- function(x) (x^2+x)*cos(x)
5 curve(f, -10, 10, n=1001, xlab = "x", ylab = "f(x)",
6       col="blue")
7
8 # Running GA
9 GA <- ga(type = "real-valued", fitness = f, lower = -10
10         , upper = 10)
11 summary(GA) # to get the summary
```

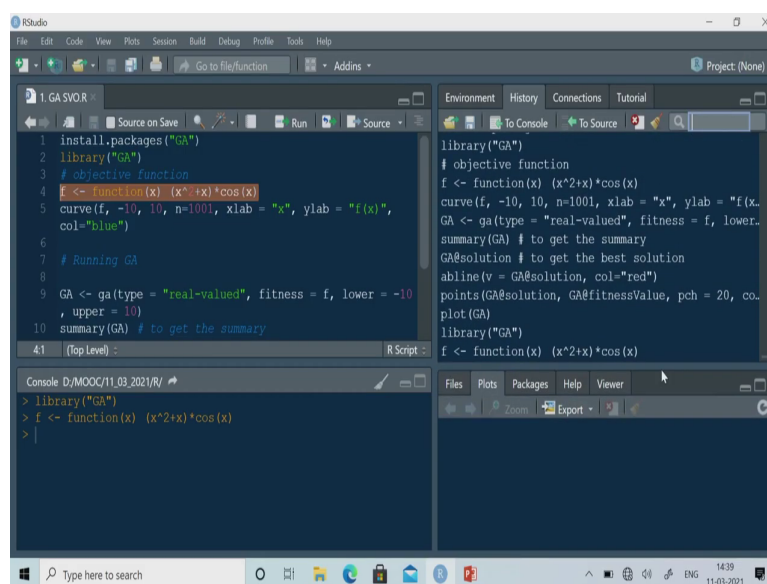
So, now let us solve this problem using R. So, I have given the code here. So, only there are 14 or 15 lines code. So, I will explain all these lines one by one. So, first one the first line is that I have to install the GA package ok. So, GA package is required and the name of the package is GA.

So, therefore, you have to install this package. So, as you know that you can install this package by writing install dot packages and then, "GA". So, GA package can be installed and I have already installed in my computer. So, I will not install it again, but if you have not installed, so you can execute this particular code and you can install GA on your computer.

So, next is I have to include this library ok. So, to include the library, so you will write library “GA”. So, you will execute this code and you can install the library. So, I will execute it by clicking this run button. So, I have selected this particular line and then, I am executing this or I am running this one.

So, library is included here now and next is I have to write the function. So, here the function is written f equal to it is a function of x, so one variable only and this is x square plus x into cos x. So, this is the function. So, I am executing this particular function. So, now, this function has been executed.

(Refer Slide Time: 09:36)



The screenshot shows the RStudio interface with the following content:

```
1. GA.SVO.R
1 install.packages("GA")
2 library("GA")
3 # objective function
4 f <- function(x) (x^2+x)*cos(x)
5 curve(f, -10, 10, n=1001, xlab = "x", ylab = "f(x)",
6     col="blue")
7 # Running GA
8
9 GA <- ga(type = "real-valued", fitness = f, lower = -10
10 , upper = 10)
11 summary(GA) # to get the summary
```

The Environment pane on the right shows the loaded package and the function f:

```
library("GA")
# objective function
f <- function(x) (x^2+x)*cos(x)
curve(f, -10, 10, n=1001, xlab = "x", ylab = "f(x).
GA <- ga(type = "real-valued", fitness = f, lower.
summary(GA) # to get the summary
GA@solution # to get the best solution
abline(v = GA@solution, col="red")
points(GA@solution, GA@fitnessValue, pch = 20, co.
plot(GA)
library("GA")
f <- function(x) (x^2+x)*cos(x)
```

The Console shows the execution of the library and function definition:

```
> library("GA")
> f <- function(x) (x^2+x)*cos(x)
>
```

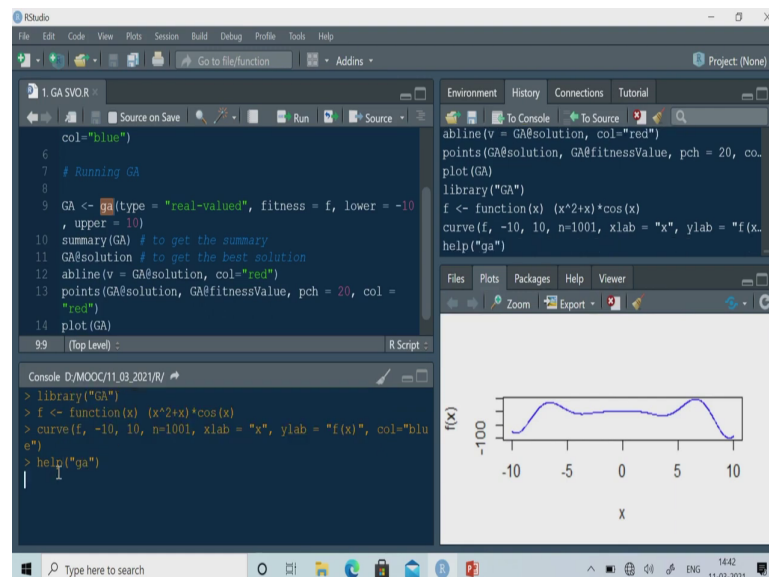
And whatever you have executed, so you can see actually value, so on the history here. So, the function is there, library has been included. So, next is just I would like to plot this

particular function between minus 1 and 10. So, for plotting this one single variable function, so I can use the curve function of R.

So, this is the curve function and so, you are using curve and the arguments are so f; f is the function. So, this function, I would like to plot and then, this is between minus 10 and plus 10. So, lower bound and upper bound, I am explaining here. And number of points you need basically. So, you can define or otherwise whatever by default value will be taken.

So, in this case, I am putting 1001. So, you can put some other value. So, if you are putting a large value, then your curve will be smooth; otherwise, it may not be smooth and then, I am putting x label and y label of this particular curve. So, I am putting x label is x and y label is your f of x and color I am defining blue basically. So, therefore, the curve will be of blue color. So, color you can define.

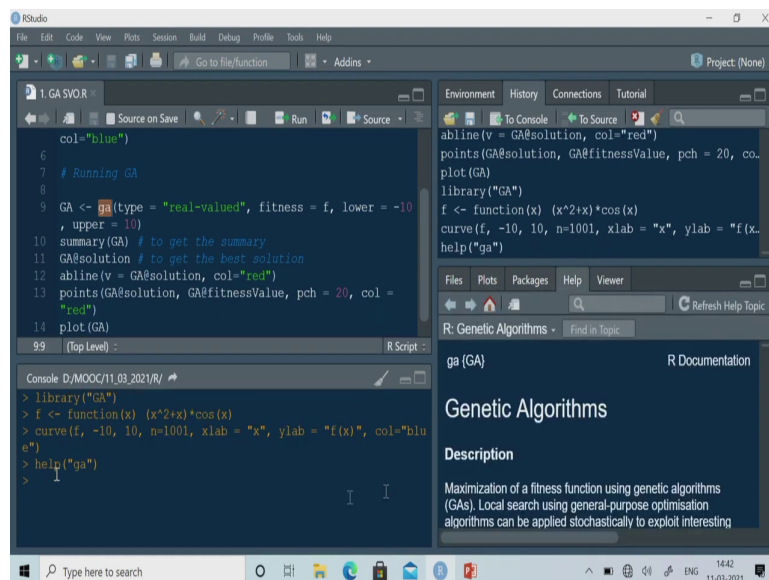
(Refer Slide Time: 10:58)



So, now if I run this particular line, so just select it and just run it, then you will get this curve. So, you can see, so this is the curve I am getting. So, x axis is your x and y axis is f x and there are two optimal solution. So, this particular few lines, so the this is I am just using to plot the function and then, I can apply; apply genetic algorithm.

So, genetic algorithm function is GA and then, I have to define some parameters here. So, if you want to see the parameters here. So, you can go to help and just see what are the arguments of genetic algorithm. So, I can write help and this is “ga”.

(Refer Slide Time: 11:51)



The screenshot displays the RStudio interface. The main editor window contains the following R code:

```
col="blue")
# Running GA
GA <- ga(type = "real-valued", fitness = f, lower = -10, upper = 10)
summary(GA) # to get the summary
GA@solution # to get the best solution
abline(v = GA@solution, col="red")
points(GA@solution, GA@fitnessValue, pch = 20, col = "red")
plot(GA)
```

The console window at the bottom shows the execution of the following commands:

```
> library("GA")
> f <- function(x) (x^2+x)*cos(x)
> curve(f, -10, 10, n=1001, xlab = "x", ylab = "f(x)", col="blue")
> help("ga")
>
```

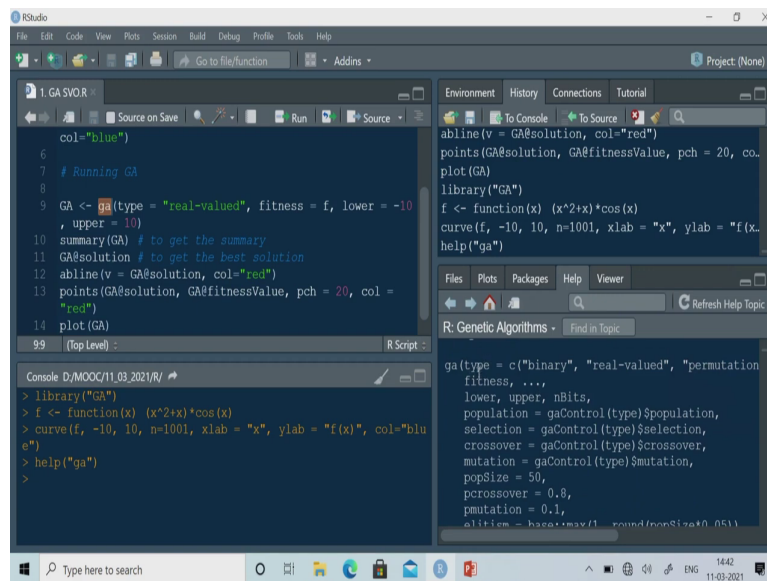
The right-hand pane shows the help documentation for the 'ga' function. The title is 'Genetic Algorithms'. The description states: 'Maximization of a fitness function using genetic algorithms (GAs). Local search using general-purpose optimisation algorithms can be applied stochastically to exploit interesting'.

(Refer Slide Time: 11:56)

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for running a Genetic Algorithm (GA). The code includes comments and function calls like `col="blue"`, `library("GA")`, `GA <- ga(type = "real-valued", fitness = f, lower = -10, upper = 10)`, `summary(GA)`, `abline(v = GA@solution, col="red")`, `points(GA@solution, GA@fitnessValue, pch = 20, col = "red")`, and `plot(GA)`.
- Console:** Shows the execution of the code, including `> library("GA")`, `> f <- function(x) (x^2+x)*cos(x)`, `> curve(f, -10, 10, n=1001, xlab = "x", ylab = "f(x)", col="blue")`, and `> help("ga")`.
- Environment/Help Panel:** Displays the help documentation for the `ga` function from the `Genetic Algorithms` package. It includes a description: "Maximization of a fitness function using genetic algorithms (GAs). Local search using general-purpose optimisation algorithms can be applied stochastically to exploit interesting regions. The algorithm can be run sequentially or in parallel using an explicit master-slave parallelisation." and a **Usage** section showing the function signature: `ga(type = c("binary", "real-valued", "permutation", "fitness", ...), lower, upper, nBite)`.

(Refer Slide Time: 12:01)

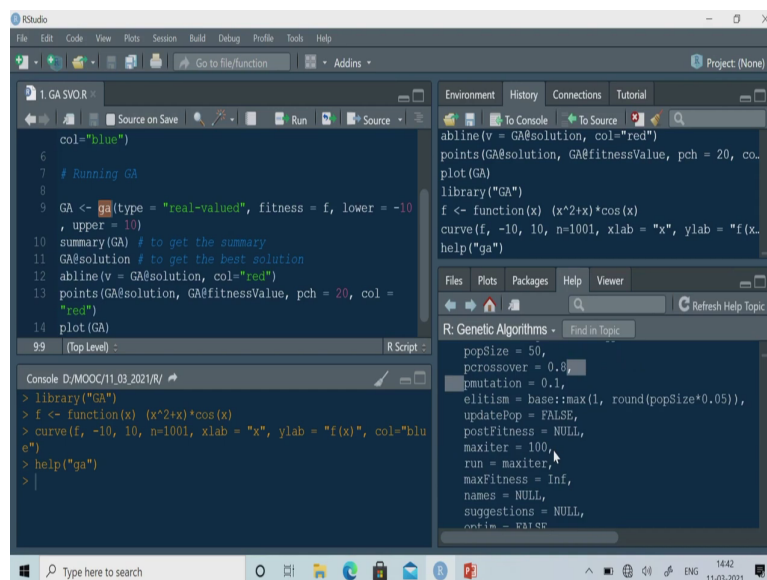


The screenshot displays the RStudio interface with the following components:

- Source Editor (Left):** Contains R code for a Genetic Algorithm. The code includes comments like "# Running GA" and "# to get the best solution". It defines a fitness function `f` and uses the `ga` function from the `GA` package.
- Console (Bottom Left):** Shows the execution of the code, including the command `library("GA")` and the definition of the fitness function `f`.
- Environment/History/Connections/Tutorial (Top Right):** The 'Environment' tab is active, showing the loaded package `GA` and the function `f`.
- Help Viewer (Bottom Right):** Displays the help documentation for the `ga` function, showing its arguments: `type`, `fitness`, `lower`, `upper`, `nBits`, `population`, `selection`, `crossover`, `mutation`, `popSize`, `pcrossover`, `pmutation`, and `elitism`.

So, you can see that right hand side, you will get the genetic algorithm function ok. So, this is the GA function. So, what you have to do? You have to define some parameters here something. So, you have to define whether you want to use binary coded, real coded, so that you can define.

(Refer Slide Time: 12:34)



The screenshot shows the RStudio interface with the following content:

- Source Editor:** Contains R code for a Genetic Algorithm (GA). The code includes comments like "# Running GA" and "# to get the summary". It defines a fitness function `f` and uses the `GA` package to solve the problem. The code is as follows:

```
col="blue")
# Running GA
GA <- ga(type = "real-valued", fitness = f, lower = -10, upper = 10)
summary(GA) # to get the summary
GA@solution # to get the best solution
abline(v = GA@solution, col="red")
points(GA@solution, GA@fitnessValue, pch = 20, col = "red")
plot(GA)
```
- Console:** Shows the execution of the code, including the command `library("GA")` and the definition of the fitness function `f`.

```
> library("GA")
> f <- function(x) (x^2+x)*cos(x)
> curve(f, -10, 10, n=1001, xlab = "x", ylab = "f(x)", col="blue")
> help("ga")
```
- Environment/Help:** Displays the parameters of the `GA` package, including `popSize`, `pcrossover`, `mutation`, `elitism`, `updatePop`, `postFitness`, `maxiter`, `run`, `maxFitness`, `names`, `suggestions`, and `optim`.

```
popSize = 50,
pcrossover = 0.6,
mutation = 0.1,
elitism = base::max(1, round(popSize*0.05)),
updatePop = FALSE,
postFitness = NULL,
maxiter = 100,
run = maxiter,
maxFitness = Inf,
names = NULL,
suggestions = NULL,
optim = FALSE
```

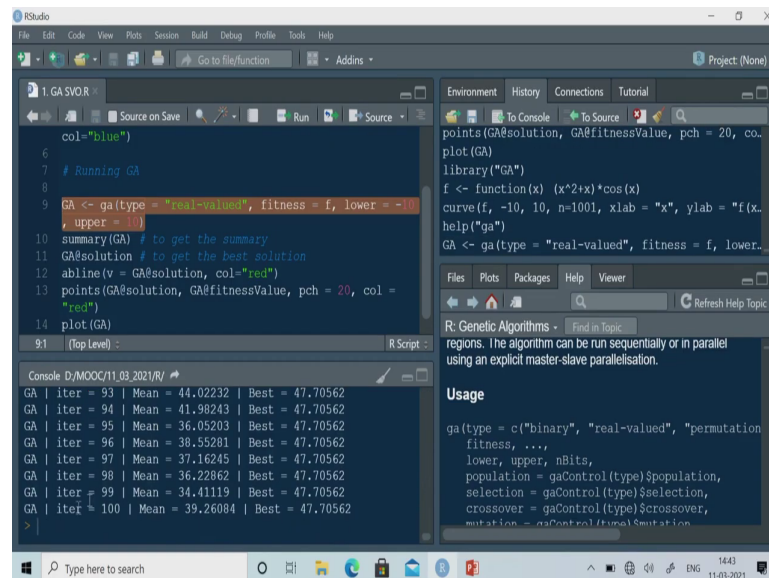
And similarly, some control parameter so you have to define. Then, you have to define the fitness value. So, in this case fitness value is my `f`. So, I will define that one and then, you can actually define the other parameters; population size, crossover probability, mutation probability, elitism so that you can define.

So, anyway, so if you are not defining, then the default value will be taken. Here, I have defined that I am using real value GA ok. So, I am not using binary valued, but I am using real valued and my fitness function is `f`. So, this whatever function value so that is actually I am defining as my fitness value. I am also providing the lower bound and upper bound.

So, lower bound is 10 and upper bound is 10 here and other values I am not defining here. So, whatever default value is assigned, so that has been considered here. So, now, if I execute this

particular line, so if I run this particular line, so GA we will solve this particular problem and we will try to find out the global maxima of this function. So, you can see. So, let us run it.

(Refer Slide Time: 13:25)



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for running a Genetic Algorithm (GA). The code includes setting a color, running the GA, summarizing the results, and plotting the solution.
- Environment:** Shows the loaded packages: 'GA' and 'graphics'.
- Console:** Displays the output of the GA execution, showing the mean and best fitness values for iterations 93 to 100.
- Help:** Shows the 'Usage' section for the 'ga' function, detailing its arguments and control parameters.

```
col="blue")
# Running GA
GA <- ga(type = "real-valued", fitness = f, lower = -10, upper = 10)
summary(GA) # to get the summary
GA@solution # to get the best solution
abline(v = GA@solution, col="red")
points(GA@solution, GA@fitnessValue, pch = 20, col = "red")
plot(GA)
```

Console Output:

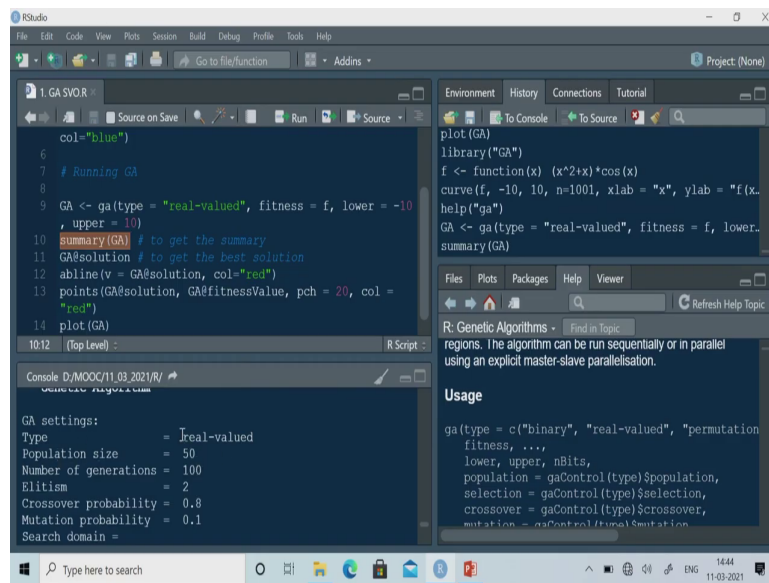
GA	iter	Mean	Best
GA	93	44.02232	47.70562
GA	94	41.98243	47.70562
GA	95	36.05203	47.70562
GA	96	38.55281	47.70562
GA	97	37.16245	47.70562
GA	98	36.22862	47.70562
GA	99	34.41119	47.70562
GA	100	39.26084	47.70562

Usage:

```
ga(type = c("binary", "real-valued", "permutation", "fitness", ...),
    lower, upper, nBits,
    population = gaControl(type)$population,
    selection = gaControl(type)$selection,
    crossover = gaControl(type)$crossover,
    mutation = gaControl(type)$mutation)
```

So, there are 100 iteration and that is the by default value. So, if you are not giving, so iteration will be 100 and you can see that GA has been executed and finally, it reach the 100 iteration. And now, I can see the results; so results, I can see from summaries here. So, you can see. So, if I execute this one, then I can see the results ok.

(Refer Slide Time: 13:55)



```
1: GAVO.R
2: col="blue")
3:
4: # Running GA
5:
6: GA <- ga(type = "real-valued", fitness = f, lower = -10, upper = 10)
7: summary(GA) # to get the summary
8: GAsolution # to get the best solution
9: abline(v = GAsolution, col="red")
10: points(GAsolution, GAFitnessValue, pch = 20, col = "red")
11: plot(GA)
```

Environment

plot(GA)

library("GA")

f <- function(x) (x^2+x)*cos(x)

curve(f, -10, 10, n=1001, xlab = "x", ylab = "f(x)")

help("ga")

GA <- ga(type = "real-valued", fitness = f, lower = -10, upper = 10)

summary(GA)

R: Genetic Algorithms

regions. The algorithm can be run sequentially or in parallel using an explicit master-slave parallelisation.

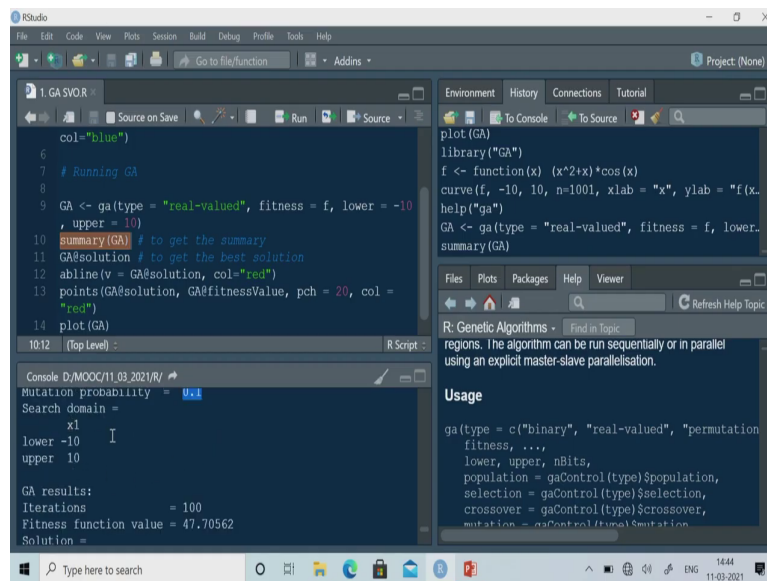
Usage

ga(type = c("binary", "real-valued", "permutation", "fitness", ...), lower, upper, nBits, population = gaControl(type)\$population, selection = gaControl(type)\$selection, crossover = gaControl(type)\$crossover, mutation = gaControl(type)\$mutation)

GA settings:

Parameter	Value
Type	real-valued
Population size	50
Number of generations	100
Elitism	2
Crossover probability	0.8
Mutation probability	0.1
Search domain	

(Refer Slide Time: 14:05)



The screenshot shows the RStudio interface with an R script file named '1. GASVOR.R'. The script defines a fitness function `f` and runs a Genetic Algorithm (`ga`) with the following parameters: `type = "real-valued"`, `fitness = f`, `lower = -10`, and `upper = 10`. The script also includes a `summary` call, a `GA@solution` extraction, a `abline` plot, and a `plot` of the solution. The console output displays the GA results, including the search domain, iterations (100), fitness function value (47.70562), and the final solution.

```
col="blue")
# Running GA
GA <- ga(type = "real-valued", fitness = f, lower = -10, upper = 10)
summary(GA) # to get the summary
GA@solution # to get the best solution
abline(v = GA@solution, col="red")
points(GA@solution, GA@fitnessValue, pch = 20, col = "red")
plot(GA)
```

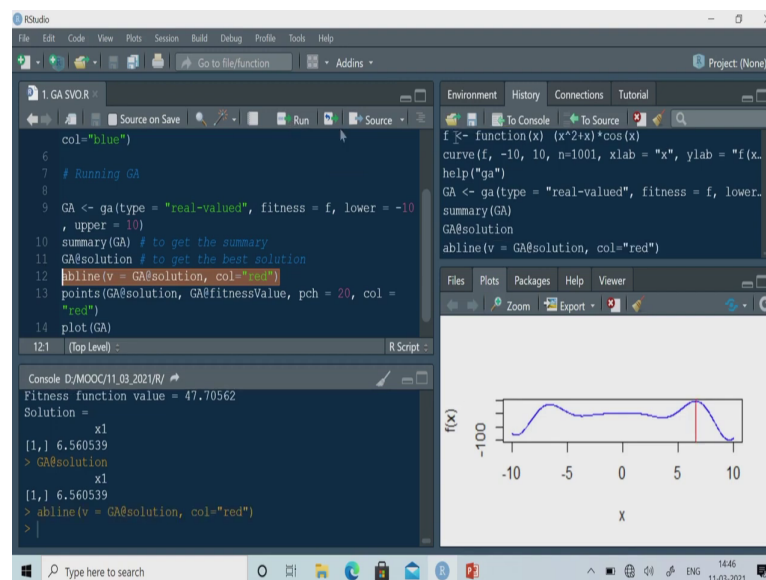
Console Output:

```
10:12 (Top Level) : R Script --
Mutation probability = 0.1
Search domain =
  x1
lower -10
upper 10

GA results:
Iterations      = 100
Fitness function value = 47.70562
Solution =
```

So, here you can see the GA setting, so it is a real-valued; then population size, I did not change. So, 50 is the by default value; the number of generation again I did not change, so that is 100, the default value; elitism is 2; crossover probability is 0.8; mutation probability is 0.1.

(Refer Slide Time: 14:18)



And then, lower bound, upper bound, I have considered minus 10 and plus 10 and finally, after 100 iteration, so I am getting the fitness function value is 47.70562 and the optimal value that is the global maxima is 6.560539. So, that is the your solution obtained that is the base solution obtained by genetic algorithm.

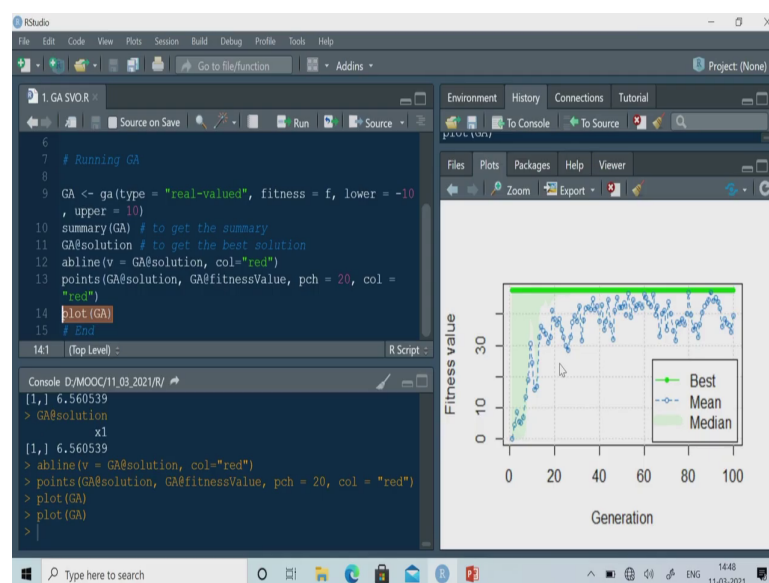
So, I am getting this solution. And I can also see the solution, the final solution using GA at the range solution. So I can if I execute this particular line, so I will get the solution. You can also see that one. So, this is the solution that x_1 equal to 6.560539 that is the solution. Now, I can actually write a ab line. So, ab line is a function. So, that will draw a line on the available plot basically.

So, it will draw a line. So, I can write this thing. So, what I would like to draw a line at optimal solution; that means, at 6.560539? So, I am just defining this is a vertical line, I

would like to plot and this is equal to GA solution; color I would like to give red. So, if I execute this particular line, so I am getting the solution somewhere here and that solution is 6.560539.

So, I can also put the point optimal point. So, I can execute by putting points functions. So, I would like to put a point here and this is x axis is your GA dot solution and y axis is the base fitness value. So, therefore, I am writing this is the x value and this is the y value and then, I am putting a round here. So, circular dot basically. So, pch equal to 20, I am putting. So, if you are giving some other value, so some other shape will come and the color, I have used red.

(Refer Slide Time: 16:38)



So, if I execute this particular line, then you can see that one this is the optimal so point actually. So, I am just putting an optimal point in this particular function. So, I am getting that

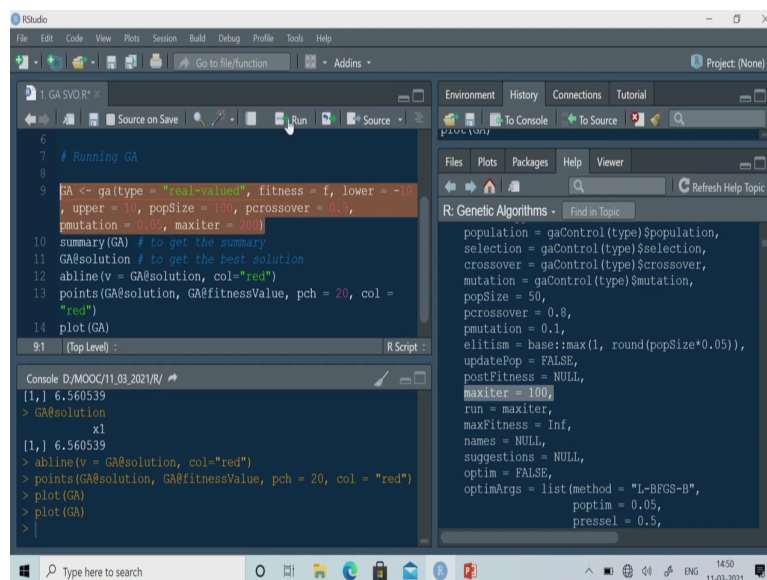
one and then, then I can plot GA. So, I will get the fitness versus generation curve. So, if I execute this particular line, so I will be getting fitness versus generation. So, you can see. So, it is showing the base fitness and then, mean value as well as the median value and you can see that with generation the mean value of the that means, all the populations are actually converging towards the global optimal solution ok.

So, this is the solution of this problem using genetic algorithm. So, as you have seen, so we have used GA function here. So, now, I can show you, so I did not define many parameters here. So, I have only defined what type of GA I am using. I have used real code real value and fitness function anyway you have to define and I am using just upper limit and lower limit.

But if you if you see that there are some other parameter suppose I would like to use or I would like to change the population size. So what I can do that population size by default value is 50, but I would like to make it 100. So, I can make it 100 by just you increase the pop size suppose this is 100.

And then, I would like to change the crossover probability. So, you just change the crossover probability, you define what is the crossover probability. Suppose I would like to give 0.9 and mutation probability, if I want to change this suppose by default value is 0.1 and I would like to put a mutation probability of 0.05.

(Refer Slide Time: 18:31)



```
GA <- ga(type = "real-valued", fitness = f, lower = -10, upper = 10, popSize = 100, pcrossover = 0.9, pmutation = 0.05, maxiter = 200)
summary(GA) # to get the summary
GA@solution # to get the best solution
abline(v = GA@solution, col="red")
points(GA@solution, GA@fitnessValue, pch = 20, col = "red")
plot(GA)
```

```
[1,] 6.560539
> GA@solution
      x1
[1,] 6.560539
> abline(v = GA@solution, col="red")
> points(GA@solution, GA@fitnessValue, pch = 20, col = "red")
> plot(GA)
```

```
population = gaControl(type)$population,
selection = gaControl(type)$selection,
crossover = gaControl(type)$crossover,
mutation = gaControl(type)$mutation,
popSize = 50,
pcrossover = 0.8,
pmutation = 0.1,
elitism = base::max(1, round(popSize*0.05)),
updatePop = FALSE,
postFitness = NULL,
maxiter = 100,
run = maxiter,
maxFitness = Inf,
names = NULL,
suggestions = NULL,
optim = FALSE,
optimArgs = list(method = "L-BFGS-B",
                  poptim = 0.05,
                  pressel = 0.5,
```

So, this is the mutation probability, I am defining and maximum iteration that is 100 by default value. So, but if you want to change it, you can change it. So, I can define that maximum iteration is suppose 200 now. You can also define the other parameters. So, let us run it.

So, now, the parameters are population size of 100, then crossover probabilities 0.9 now, mutation probability is 0.05 and maximum iteration is 200. So, I am just running it again. So, you can see that now iteration is 200 ok. So, you can see from the summary GA, so if I execute this one, then you can see.

So, now, again real-valued I have used. Now, population size is 100, then number of generation is 200. So, that is the generation. Elitism is 5 and the crossover probability is 0.9

and mutation probability is 0.05. Anyway, so you got the solution. The solution is I think we are getting the same solution that is 6.56054 and the optimal solution is 47.70562.

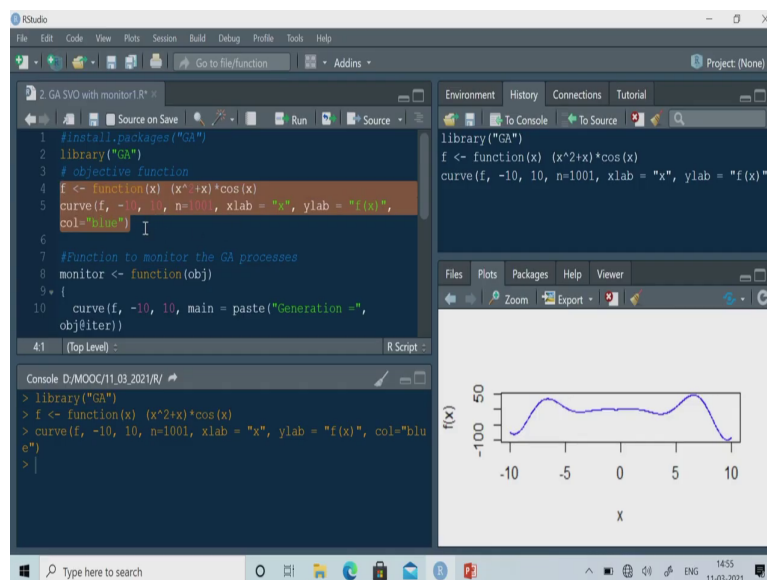
So, with this, I can actually change the GA parameter. So, depending upon what type of problem you are solving, if you want to change this parameter. So, you I can define these parameters here. So, I hope this is clear to you. So, we can apply genetic algorithm. So, this is just a one line your code. So, only what you have to do? You have to define the function, then you use the GA function available in GA package.

So, this is just a two line and other things are we are just plotting the function, we are trying to look at the optimal solution something like that. So, other your some function we have used; but you only need two lines that is the function you have to define and then, you have to run or you have to execute the GA functions. This is quite simple to solve and non-linear optimization problem using GA and you can also define the GA parameters.

So, within that function, so if you are not defining, the default value will be taken; but if you want to define, so you can change those parameter like crossover probability, mutation probability, generation, maximum iteration ok and then, population size; so all these parameters you can change. Now, for the same function, let us see how we can monitor the progress during GA iteration.

So, we can also monitor how GA is converging towards the global optimal solution or how GA is converging at global optimal solution, we can monitor at one and that I would like to show using this particular code. So, only thing is that, so you have to use a monitor function here.

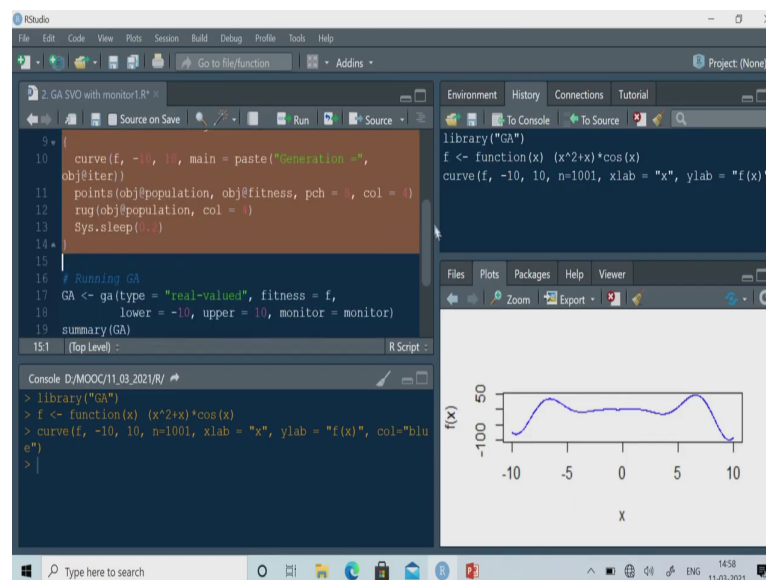
(Refer Slide Time: 21:32)



So, I am explaining that one. So, I am not installing GA package here because it is already installed. So, I will just make a comment. So, I will not install that one. So, I will use the library. So, you execute the GA library. And then, you have to write the function. So, already I have written this function. So, function; so, I will execute this one. So, this is the function I have written already and then, I have to execute this particular line to draw the curve. So, I am just drawing this curve. So, using curve function.

So, again it is x is between minus 10 and plus 10 and then, I am putting label x label and y label and color, I am putting giving blue. So, I am getting this particular function. So, I think up to this up to this line already we have discussed and I think now you are familiar with these two lines.

(Refer Slide Time: 22:26)



Now, next is that I would like to monitor the progress ok. So, monitor the progress of GA or monitor the GA processes ok. So, GA processes, I would like to monitor. So, now, I would like to monitor the GA process and for that, I have to write these monitor function ok. So, I am explaining.

So, here you are defining it as a function. So, monitor is a function and this is a function of objective and here. So, first line is to plot the curve ok. So, this is the plot, I would like to plot. So, this is the first line and this is minus between minus 10 and 10. So, curve is already you have used. So, if you execute this particular line, this curve will be drawn and then, I would like to put the points and I would like to put the population.

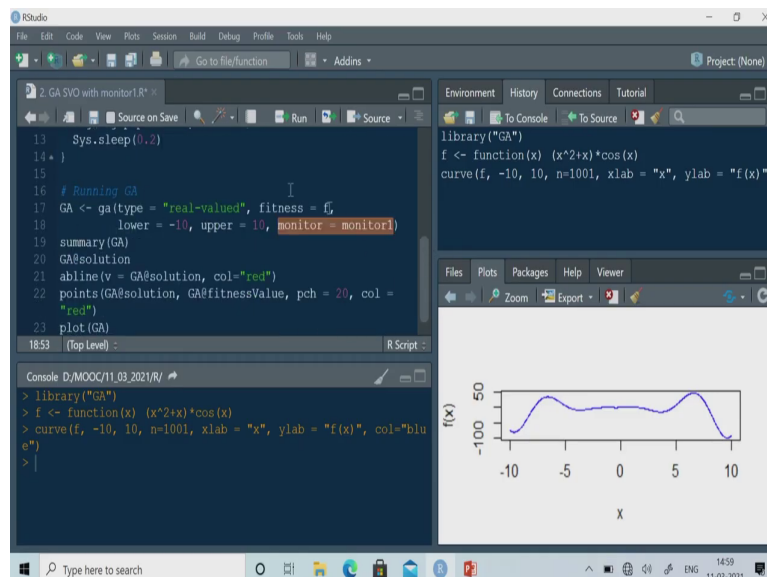
So, therefore, I am using the point function. So, this point function, we will put the points over this particular curve. So, here the population is given by objective population. So, within

GA, GA at the rate, so we will get the population. So, objective population will give you the population.

This is the x axis value and y axis, this is objective fitness value. So, here in x axis, this is the objective population and in y axis, this is the fitness value. So, I am putting that one and then, I am putting 0.8, pch 0.8. So, it will give a shape, so you can see that one and color, I am using four and then, this rug, so it will give you the location of this particular population on the x axis.

So, it will also show that one and then, when I am executing that one because I would like to see the progress, so system will sleep for 0.2; otherwise, you will not be able to see that one. So, that is why I am just putting this one. So, what is happening? So, when you are running the GA function, this monitor function will be executed in every iteration.

(Refer Slide Time: 24:34)



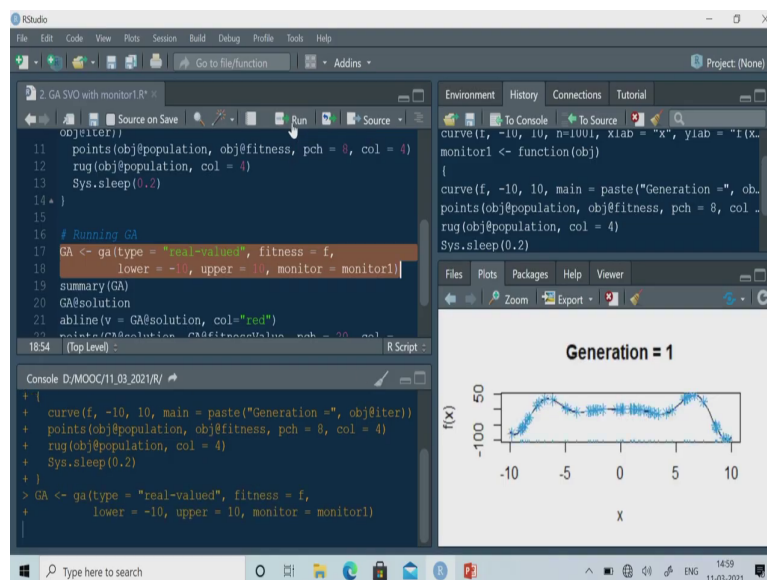
So, you can see that one. So, now, I have written the GA function. So, now, what is the difference? So, here I am using real-valued again; the fitness value is f , lower bound is minus 10, upper bound is 10; only differences that now I would like to monitor the processes. So, therefore, I am writing monitor equal to monitor.

So, here this is monitors function. So, whatever function we have given. Suppose, if I want that I am changing this monitor to monitor 1, then in that case you have to write 1 here ok. So, this is basically using this one. So, I can monitor the progress. In the previous case, I did not use that one. So, I have just solved the problem and I did not monitored.

Now if I execute this line. So, it will also monitor the processes ok. So, I can see that one. So, let us execute this particular function ok, that monitor function. So, I am selecting this one and I am executing this one. So, you can see on the right hand side, so you can see that monitor function is executed and it is stored as a monitor 1 ok.

So, now, I will execute this particular line, so I will run this particular line. So, then, the GA will run and along with that, I can see the processes ok. So, how GA population is converging towards the optimal solution. So, there I can do, just see.

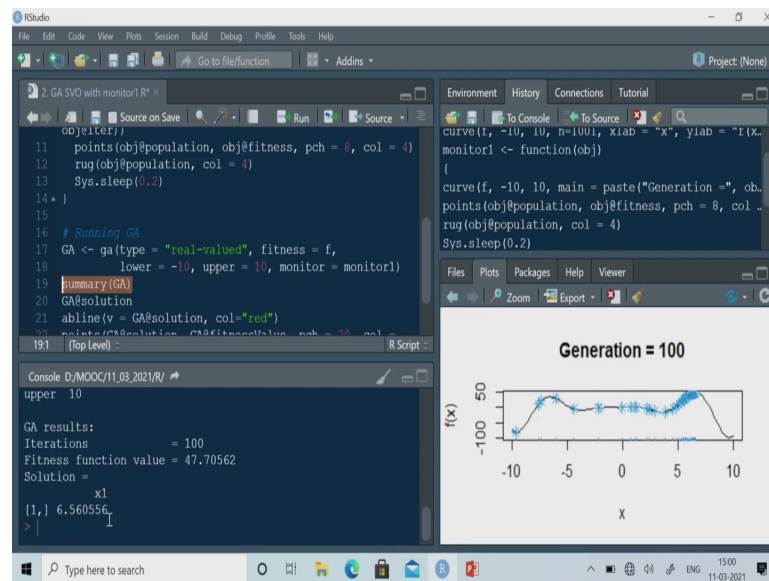
(Refer Slide Time: 26:05)



So, you can see that one. So, initially the population is between upper bound and lower bound, it was uniformly distributed and with the iteration, you can see that population is converging at global optimal solution. It is showing the generation, now it is 46, 47 like that.

So, you can see, this is the global optimal solution and all these populations are trying to converge at global optimal solution. So, it will run up to 100th generation because we have defined maximum iteration equal to 100.

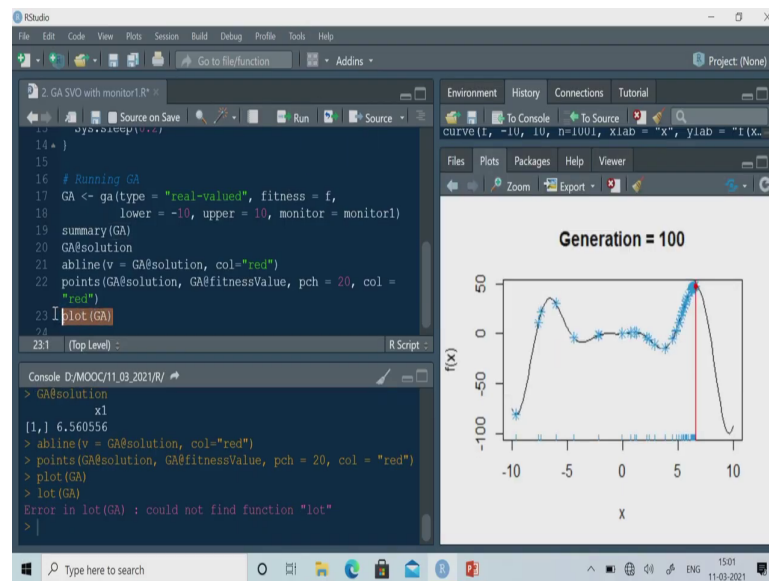
(Refer Slide Time: 26:39)



So, now you just see. So, this is the solution we got and you can see whether we got the solution, just I can see that summary. So, this is the solution you got 6.560556 and I can see the GA solution. So, summary, you can also see the other things. So, we have used real-valued, then population size 50, number of generation 100.

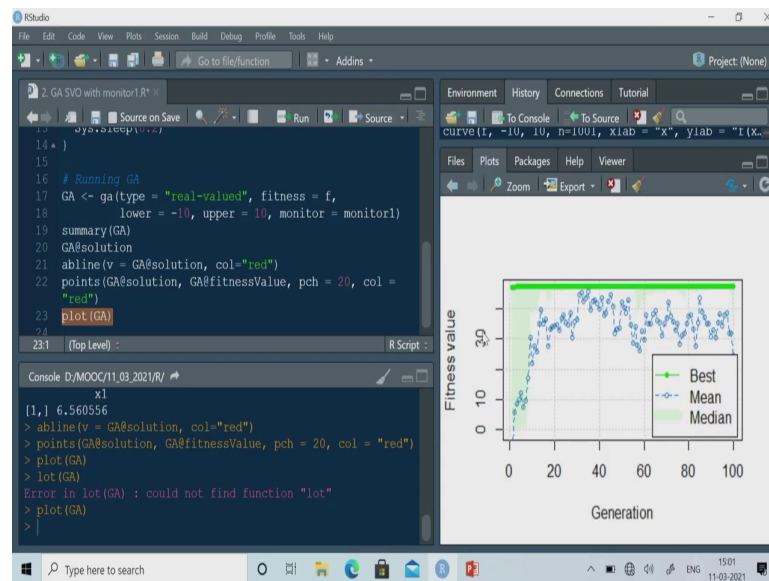
So, crossover probability, mutation probability we did not change. So, default value has been taken and finally, you got the solution. And then, GA solution you can see. What is the base solution you got? And solution is 6.560556 and I can write a ab line vertical line, so using this particular line.

(Refer Slide Time: 27:24)



So, this is the solution here and I can also look at the point; using this, so this is the optimal solution obtained by GA that is the base solution of is 10 by GA. And I can also execute this particular line to see the generation versus fitness curve.

(Refer Slide Time: 27:50)

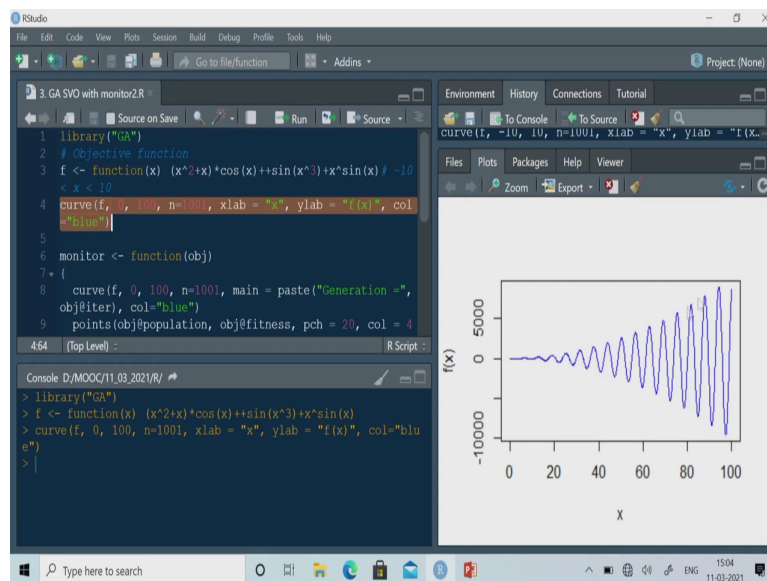


So, if you execute this particular line, if you plot GA, so you can see fitness generation versus fitness value ok. So, that has already been defined. So, now, by using this monitor function, you can monitor the movement of the population. So, how it is moving towards the global optimal solution? So, that you can monitor using the monitor function; so that I have shown here. And so, but if you want to just solve the problem and then, you only need to execute the GA function ok. As I said this is only a two lines code, so you can just write the function and then, use the GA function to get the solution.

The other lines are basically to see or to plot the function, then to see the optimal solution, to monitor the optimal solution something like that. But that is not required, if you want to get the solution just you run your objective function, write your objective function and then, you run the GA function ok.

Now, I would like to solve the second problem. So, second problem as you have seen, so that is also a maximization problem and we have lot of local optimal solution. So, several local optimal solutions are there and we will use GA to get the global optimal solution of this particular problem ok.

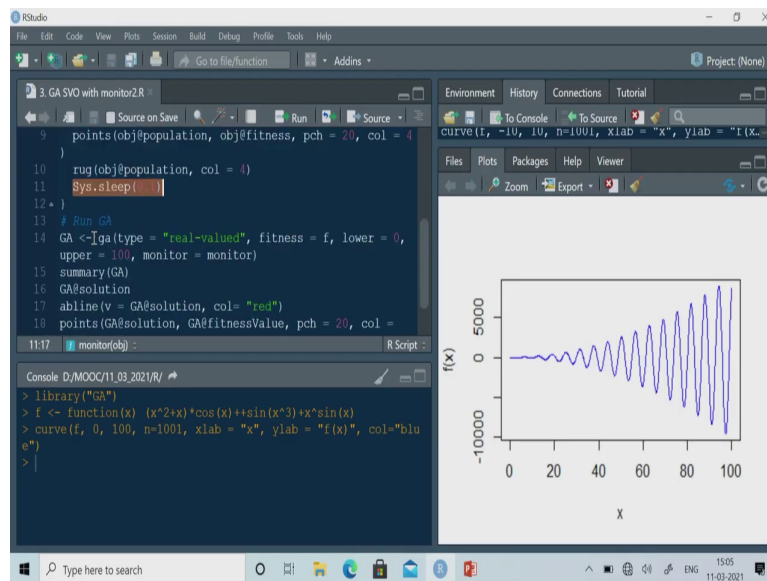
(Refer Slide Time: 29:10)



So, this is the code I have shown here. So, here you have to include the library. So, already GA has been installed. So, just include the library, “GA” library and then write the function ok. So, let us execute this one. The first lines, I have included the library here and then, you run the second lines that is the function actually and then, I can plot the function using curve function. So, I am just plotting it.

So, you can see that there are several optimal solution ok, several optimal solution; both minima and maxima. So, these are all maximum points and one of them is the global optimal solution.

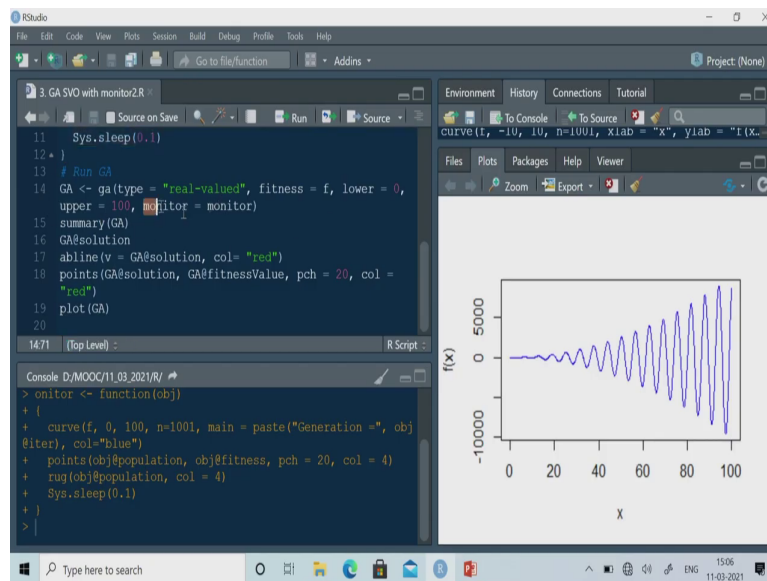
(Refer Slide Time: 29:52)



So, we will apply GA here and here, I would like to monitor the convergence of the population, how population is moving in different generation. So, I can monitor that one. So, I have written the monitor function here. So, I just put the curve function, just to draw this one.

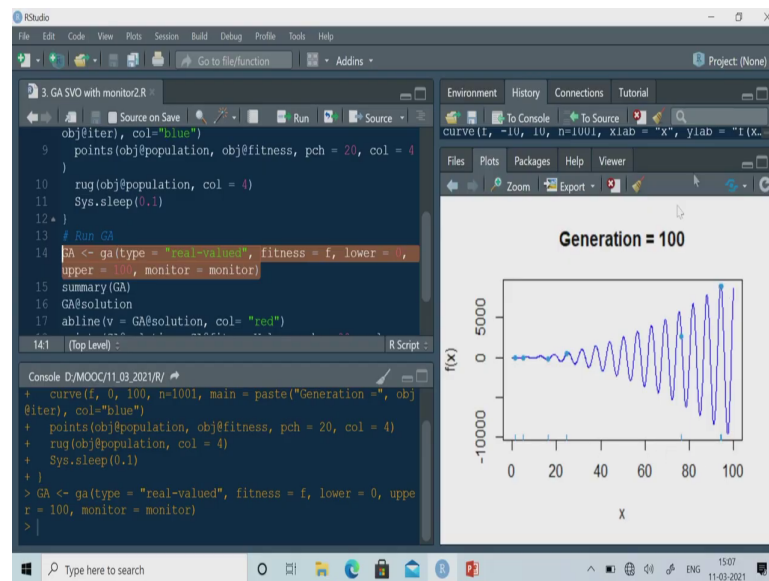
Then, I am putting the points, so this is the population I am putting and then, I am also looking at the value at x axis. So, I am putting rug function and then, system will sleep for 0.1. So, this is the monitor function. So, let us execute this one ok.

(Refer Slide Time: 30:34)



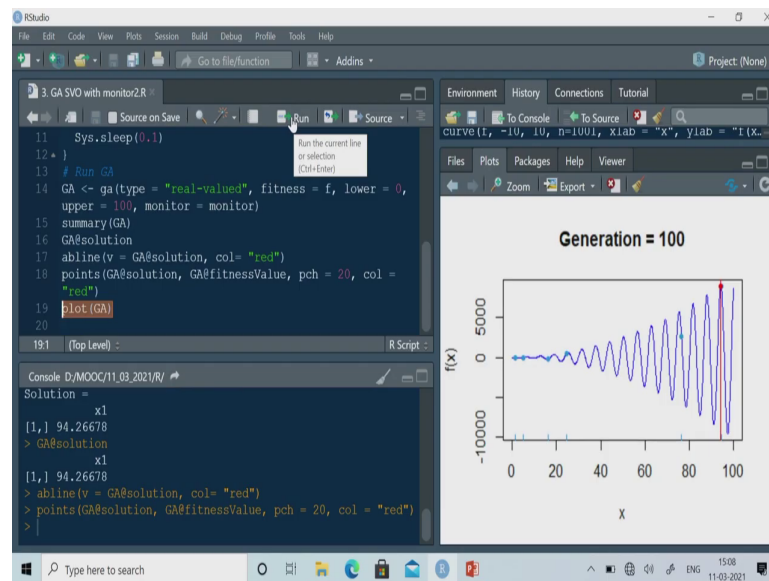
So, I have executed this monitor function and then, now you execute the GA function. So, I have just used so, this is the same function; here, I have used lower bound is 0 for this particular problem and upper bound is 100 and I would like to monitor the processes ok, monitor the progress. So, let us run this particular line.

(Refer Slide Time: 30:57)



So, you can see, you can see that initially populations were between upper bound and lower bound and with the progress of iteration ok, so with the generation and all these solutions are converging at global maxima. So, you can see the global maxima is somewhere here and this is the solution I got and finally, you can see that GA summary.

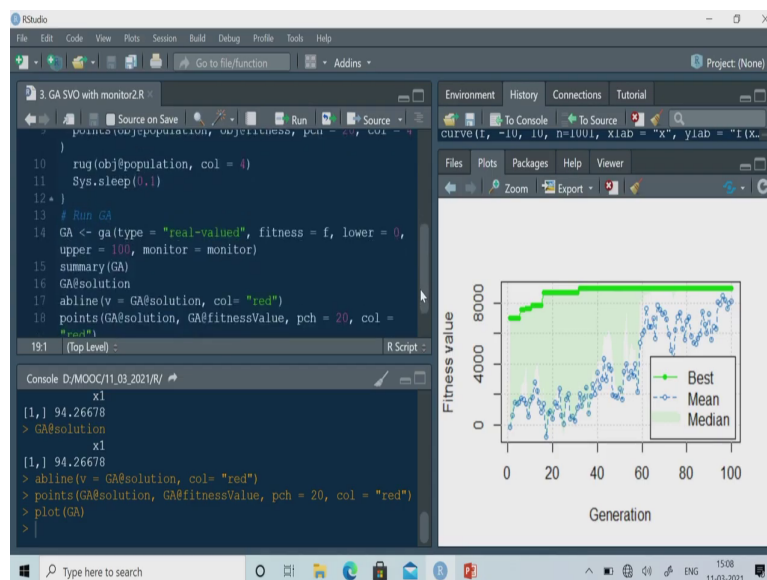
(Refer Slide Time: 31:29)



And so, solution is 94.26678. So, this is the point actually 94.26678. And I can write a I can see the GA solution. So, this is the solution 94.26678 and I can draw a vertical line using ab line command. So, this is the vertical line.

So, this is the solution; final solution I am getting and I would like to locate the optimal point and this is the optimal point somewhere here. So, this is the optimal point and similarly, I can also see; similarly, I can also see the generation versus fitness ok; fitness versus generation curve.

(Refer Slide Time: 32:13)



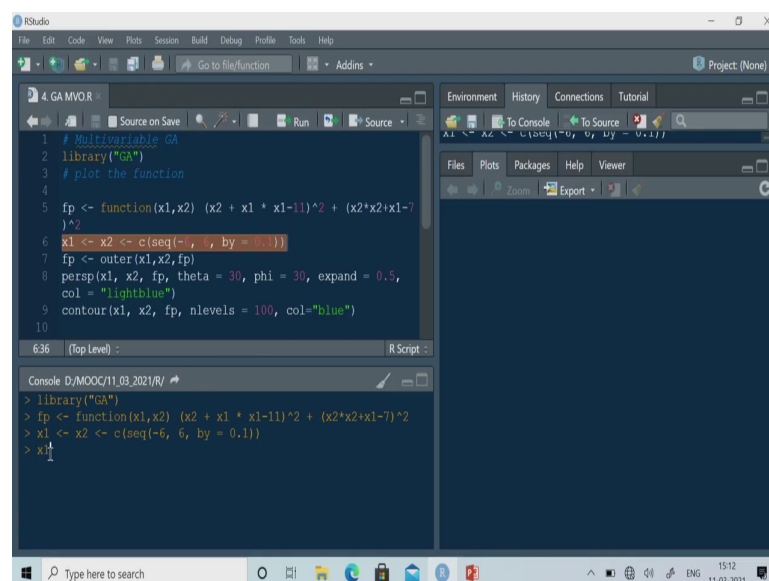
And here, you can see that GA could converge a optimal solution probably at around 35, after 35 iteration or something like that. So, you got the optimal solution and after some iteration, actually all the populations are near the global optimal solution. So, using the monitor function, so we can monitor the population, how these populations are moving towards the optimal solution. So, that we can monitor using monitor function. So, you can see that that how GA is converging at global optimal solution.

For the second problem, there are several local optimal solution, but GA could identify the global optimal solution. Just see if you are applying the classical optimization method, so classical optimization method will give you only a one local optimal solution; but GA is capable of finding the global optimal solution of this problem.

So, now, let us see the third problem. So, third problem as I have shown you already. So, it is a two variable problem and we will try to find out the optimal solution of this particular problem and here, all of them are local optimal solution means function value is same for all of them.

So, function value is 0, but we call it alternate optimal solution. There is no global optimal solution because the function value for all of them is 0 ok. So, function value is 0, so therefore, it is not that one solution is better, other one is inferior. All of them have same quality; that means, same solution. Objective function will be same for all of them. So, therefore, if you are applying GA, so you will get one of them. So, you will get one of these solutions ok. So, let us see let us try to solve this problem using GA.

(Refer Slide Time: 34:12)



```
4. GA MVO.R
1 # Multivariable GA
2 library("GA")
3 # plot the function
4
5 fp <- function(x1,x2) (x2 + x1 * x1-11)^2 + (x2*x2+x1-7)^2
6 x1 <- x2 <- c(seq(-6, 6, by = 0.1))
7 fp <- outer(x1,x2,fp)
8 persp(x1, x2, fp, theta = 30, phi = 30, expand = 0.5,
9       col = "lightblue")
9 contour(x1, x2, fp, nlevels = 100, col="blue")
10

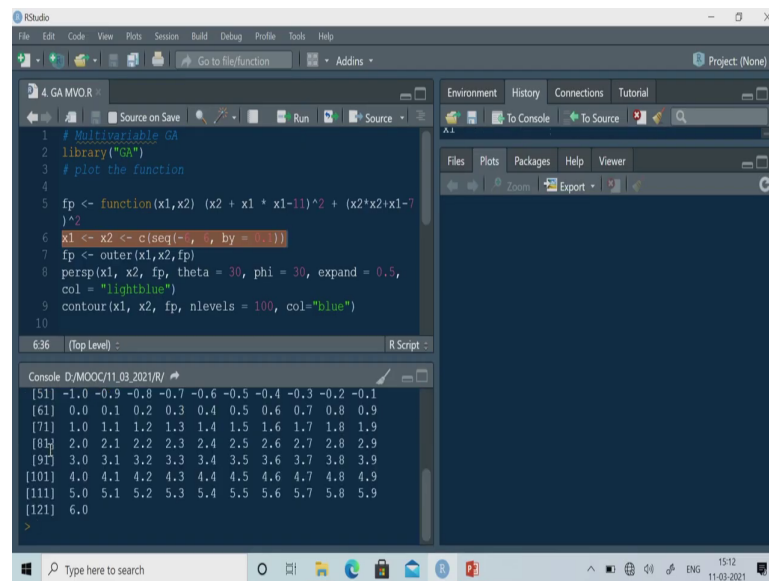
6:36 (Top Level) : R Script
> library("GA")
> fp <- function(x1,x2) (x2 + x1 * x1-11)^2 + (x2*x2+x1-7)^2
> x1 <- x2 <- c(seq(-6, 6, by = 0.1))
> x1
```

So, here, this is the multivariable function. So, I am writing multivariable GA. So, you have to include the GA library. So, just include it and then, first I would like to plot the function. So, these few lines are just to plot the function. So, here I am just writing the function. So, that is f_p . So, the function to plot and it is a two-variable function.

So, therefore, x_1 and x_2 and the function is x_2 plus x_1 square minus 11 whole square, then plus x_2 square plus x_1 minus 7 whole square. So, this is the function. So, let us execute this one. So, I have executed this particular function. And then, I am defining what is x and y . So, x and y is between minus 6 and plus 6. So, I can use the sequence `seq`, the sequence this is minus 6 to plus 6 and then, the points will be created at an interval of 0.01.

So, this is for true for both the cases; that means, x_1 is varying between minus 6 and plus 6 and similarly, x_2 is also varying between minus 6 to plus 6. So, let us execute this particular line. So, I can see basically. So, this has been executed.

(Refer Slide Time: 35:34)



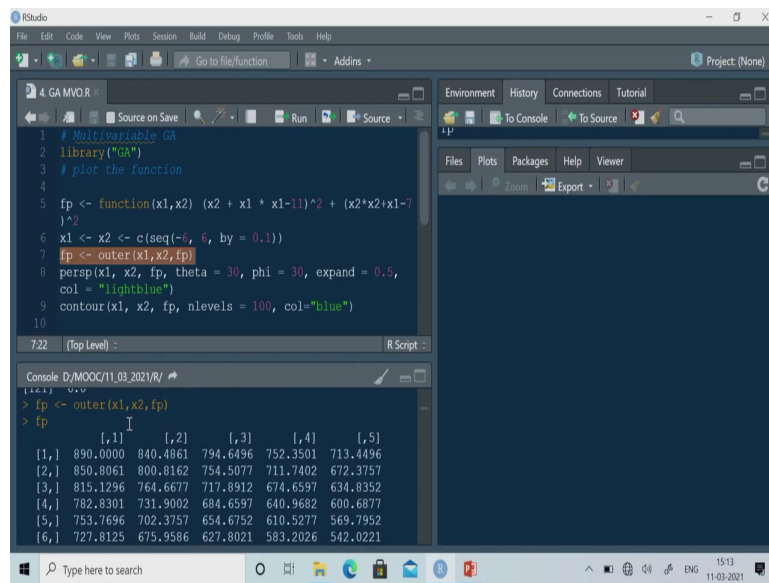
```
1 # Multivariable GA
2 library("GA")
3 # plot the function
4
5 fp <- function(x1,x2) (x2 + x1 * x1-11)^2 + (x2*x2+x1-7
6 )^2
7 x1 <- x2 <- c(seq(-, , by = .1))
8 fp <- outer(x1,x2,fp)
9 persp(x1, x2, fp, theta = 30, phi = 30, expand = 0.5,
10 col = "lightblue")
11 contour(x1, x2, fp, nlevels = 100, col="blue")
12
```

Console D:\MOOC\11_03_2021\R\ >

```
[51] -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1
[61] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
[71] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9
[81] 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9
[91] 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9
[101] 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9
[111] 5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9
[121] 6.0
>
```

So, now if want to see what is x 1? So, this is the x 1 values ok. So, you can see that minus 6 to plus 6 and at an interval of 0.1. Similarly, x 2 also I can see. So, this is the x 2 value. So, this is minus 6 to plus 6. So, now, I would like to find out the function value at each grid point. So, for that, I am using outer function. So, this is the outer that x 1 and x 2 and f p function is taken and it will calculate the function value at each grid point ok.

(Refer Slide Time: 36:07)



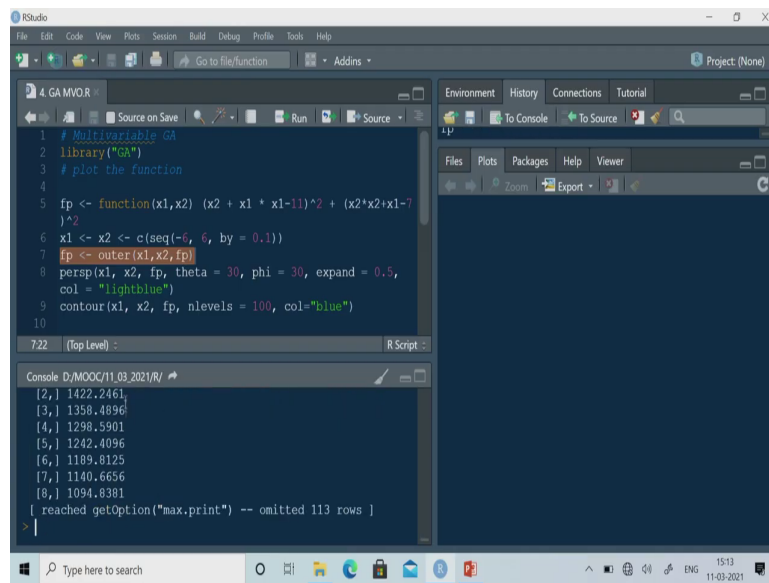
The image shows the RStudio interface with a script editor and a console. The script defines a function `fp` and uses `outer` to calculate values for a 6x5 grid of `x1` and `x2` values. The console displays the resulting matrix.

```
1 # Multivariable GA
2 library("GA")
3 # plot the function
4
5 fp <- function(x1,x2) (x2 + x1 * x1-11)^2 + (x2*x2+x1-7
6 )^2
7 x1 <- x2 <- c(seq(-6, 6, by = 0.1))
8 fp <- outer(x1,x2,fp)
9 persp(x1, x2, fp, theta = 30, phi = 30, expand = 0.5,
10 col = "lightblue")
11 contour(x1, x2, fp, nlevels = 100, col="blue")
```

Console output:

```
> fp <- outer(x1,x2,fp)
> fp
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 890.0000 840.4861 794.6496 752.3501 713.4496
[2,] 850.8061 800.8162 754.5077 711.7402 672.3757
[3,] 815.1296 764.6677 717.8912 674.6597 634.8352
[4,] 782.8301 731.9002 684.6597 640.9682 600.6877
[5,] 753.7696 702.3757 654.6752 610.5277 569.7952
[6,] 727.8125 675.9586 627.8021 583.2026 542.0221
```

(Refer Slide Time: 36:12)



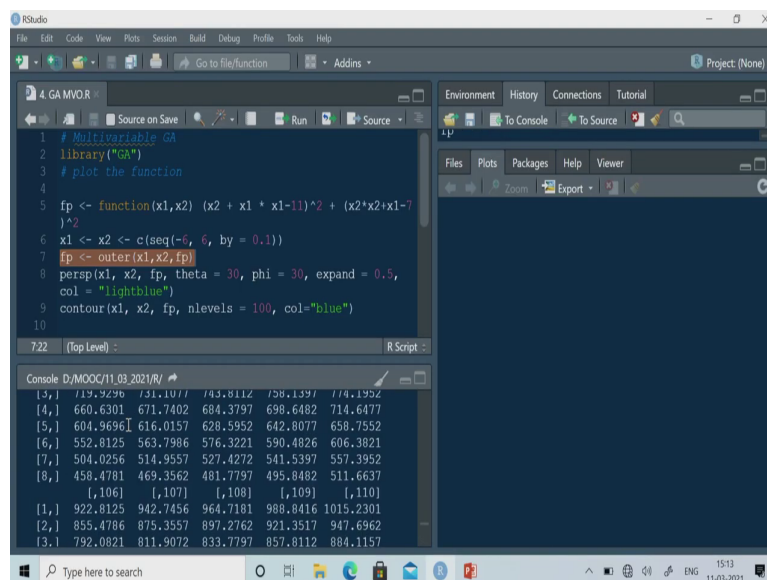
The image shows the RStudio interface with a script editor on the left and a console on the bottom. The script defines a function `fp` and uses `outer` to calculate values for a 3D surface plot and a contour plot.

```
1 # Multivariable GA
2 library("GA")
3 # plot the function
4
5 fp <- function(x1,x2) (x2 + x1 * x1-11)^2 + (x2*x2+x1-7
6 )^2
7 x1 <- x2 <- c(seq(-6, 6, by = 0.1))
8 fp <- outer(x1,x2,fp)
9 persp(x1, x2, fp, theta = 30, phi = 30, expand = 0.5,
10 col = "lightblue")
11 contour(x1, x2, fp, nlevels = 100, col="blue")
12
```

The console output shows the results of the `outer` function for the first 8 rows of the grid, with a message indicating that 113 rows were omitted due to the `getOption("max.print")` limit.

```
722 (Top Level) : R Script
Console D:/MOOC/11_03_2021/R/
[2,] 1422.2461
[3,] 1358.4896
[4,] 1298.5901
[5,] 1242.4096
[6,] 1189.8125
[7,] 1140.6656
[8,] 1094.8381
[ reached getOption("max.print") -- omitted 113 rows ]
>
```

(Refer Slide Time: 36:14)



```
1 # Multivariable GA
2 library("GA")
3 # plot the function
4
5 fp <- function(x1,x2) (x2 + x1 * x1-11)^2 + (x2*x2+x1-7
6 )^2
7 x1 <- x2 <- c(seq(-6, 6, by = 0.1))
8 fp <- outer(x1,x2,fp)
9 persp(x1, x2, fp, theta = 30, phi = 30, expand = 0.5,
10 col = "lightblue")
11 contour(x1, x2, fp, nlevels = 100, col="blue")
```

Console D:\MOOC\11_03_2021\R\

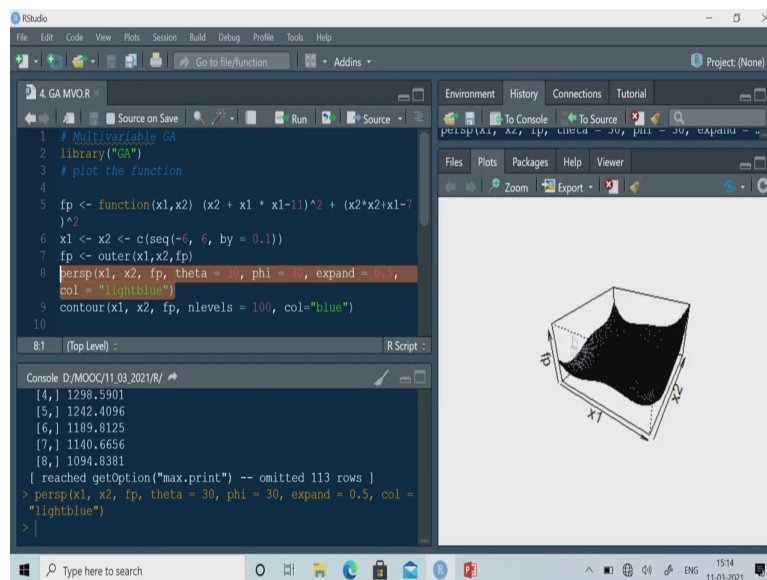
[3,]	719.9296	731.1077	743.8112	758.1397	774.1952
[4,]	660.6301	671.7402	684.3797	698.6482	714.6477
[5,]	604.9696	616.0157	628.5952	642.8077	658.7552
[6,]	552.8125	563.7986	576.3221	590.4826	606.3821
[7,]	504.0256	514.9557	527.4272	541.5397	557.3952
[8,]	458.4781	469.3562	481.7797	495.8482	511.6637
	[,106]	[,107]	[,108]	[,109]	[,110]
[1,]	922.8125	942.7456	964.7181	988.8416	1015.2301
[2,]	855.4786	875.3557	897.2762	921.3517	947.6962
[3,]	792.0821	811.9072	833.7797	857.8112	884.1157

So, now let me execute this one. So, you can see. So, this is the f_p result ok. So, here, so it is not showing all these points; but you are getting the function value at each grid point. Suppose, this is 1 1, then 2 2, then 3 3 like that 1 1, 1 2, 1 3 something like that. So, you are getting the function value at its grid point and then, to plot this function, so I am using this particular function which is available in R. So, pr persp ok. So, this is the function, I am using to plot this.

So, this will give you a 3D plot of this particular function. So, it is x_1 and x_2 we have to define, we have to give. So, this is x x is x_1 and this is x_2 and this is the f_p value; f_p value of this particular function and then, this θ and ϕ will give you horizontal angle and vertical angle. Suppose, in this case, I am putting 30 and 30. So, if you want to have a

different view, so you can change this angle. I will also show you. Then, expand, I am using 0.5 and color I am using “lightblue” ok.

(Refer Slide Time: 37:24)

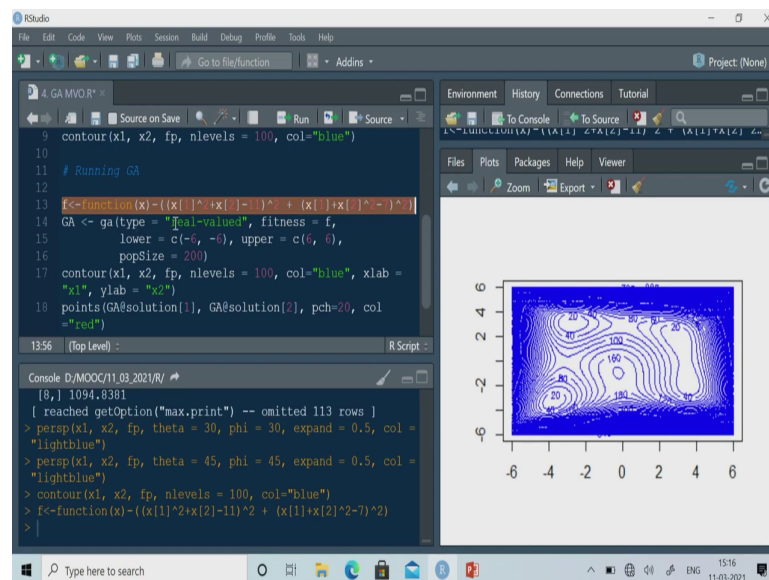


So, therefore, if you are executing this particular line, so you can plot this function ok. So, I have plot these particular function. So, this is the 3D plot and if you want the change your view, suppose I want to use 45 degrees and 45 degree. So, horizontal and vertical angle, I can change it ok. So, this is this I can change, then maybe I will get a different view. I am getting a different view.

So, this is the 3D view of this particular function; but I would like to see the contours ok. So, for that, I can use the contour function. So, contour function is the arguments are x_1 , x_2 ; then the function value you have to put and number of levels, so I am defining 100. So, if you

are not defining, the default value will be taken and here also I am here also I would like to give blue color ok. So, therefore, if I run this particular line, so I will get the contour map.

(Refer Slide Time: 38:27)



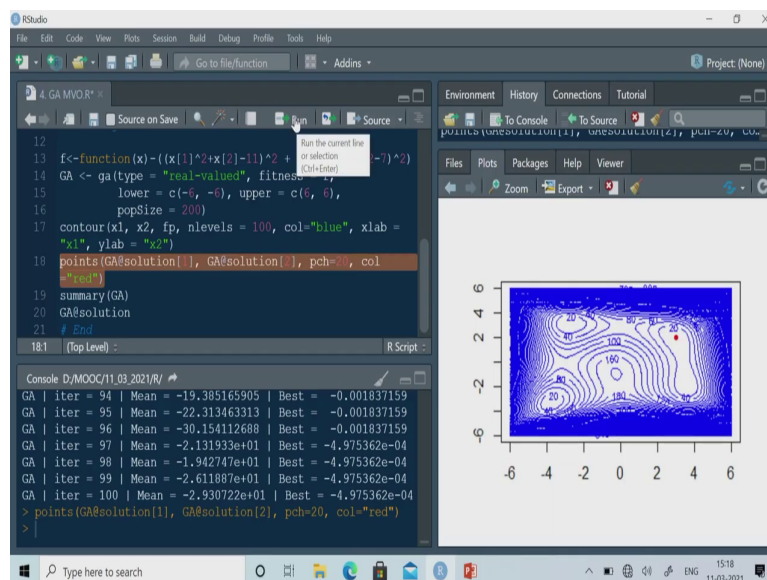
So, you can see from this contour map. So, there are total four optimal solution in each of the quadrant that is first quadrant, second, third and fourth quadrant and once you are applying GA, so you will get one of them. So, this particular lines so far whatever I have executed, so these lines are just to plot this particular function.

But now, I will execute the GA. So, for applying GA, so you have to write a function in terms of x. So, I cannot define x 1 and x 2 here. So, x is an array and between, we have x 1 and x 2. So, therefore, this is x 1 and this is x 2. So, I have to write this function in this in this way to implement that one. So, I cannot write the function like these x 1 and x 2. So, only thing, I can I have to put x ok.

So, therefore, I am writing this function. So, this is the function of x_1 and x_2 . So, this is $x_1^2 + x_2^2 - 11x_1 + 7x_2$, then so this is the function f . So, let us execute this particular line. So, I am getting this particular function. Now, I am running the GA function, so I am not putting the other parameters of GA. So, what I am doing? So, I am using the real value GA and function is f . So, this is a minimization function and therefore, I am putting minus sign here ok; GA is solving a maximization problem.

So, therefore, as this is a minimization problem. So, I am putting minus; otherwise, if it is a maximization problem, you need not put minus sign here and so, I am getting this one. Then, the fitness value is f . So, this is a maximization problem now and then, you have to define lower bound and upper bound. So, lower bound is minus 6 and minus 6; that means, for x_1 it is minus 6, for x_2 also it is minus 6 and upper bound is plus 6 and plus 6 and population size, I have defined 100. So, if I do not define, then default value will be taken ok.

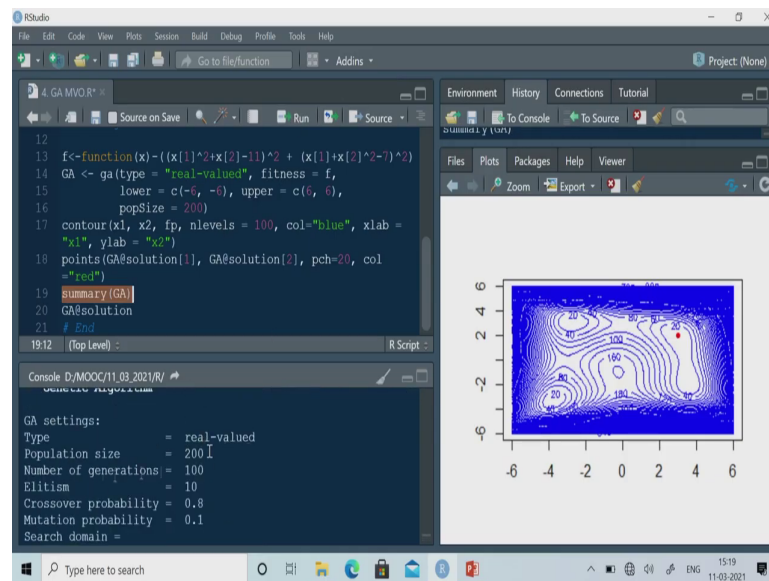
(Refer Slide Time: 40:41)



So, therefore, let us run this particular line. So, it is iteration up to 100. So, this is the default iteration value given. So, 100 and population size taken is 200. So, I did not change the other parameters like crossover probability, mutation probability, elitism, those I did not changed; I have taken the default value. And after that, I would like to look at the point. So, anyway, so I can run this particular line. So, but this is already plotted. So, therefore, I try to put the solution. So, I am using again point function.

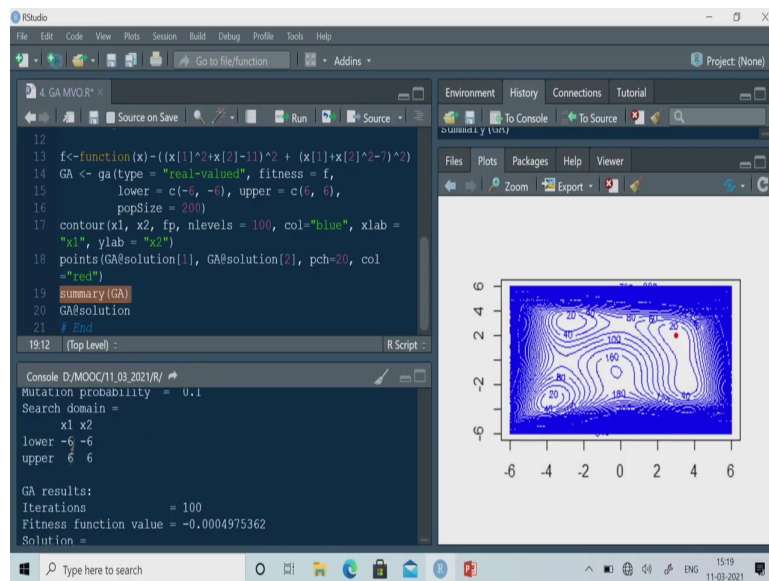
So, now, it is a two variables. So, therefore, this is the solution x 1 and this is the x 2 solution and then, pch I am taking 20; that means, I put a circle and the color, I am putting red. So, therefore, if I execute this particular line, so I am getting this particular. So, GA is giving this solution ok.

(Refer Slide Time: 41:51)

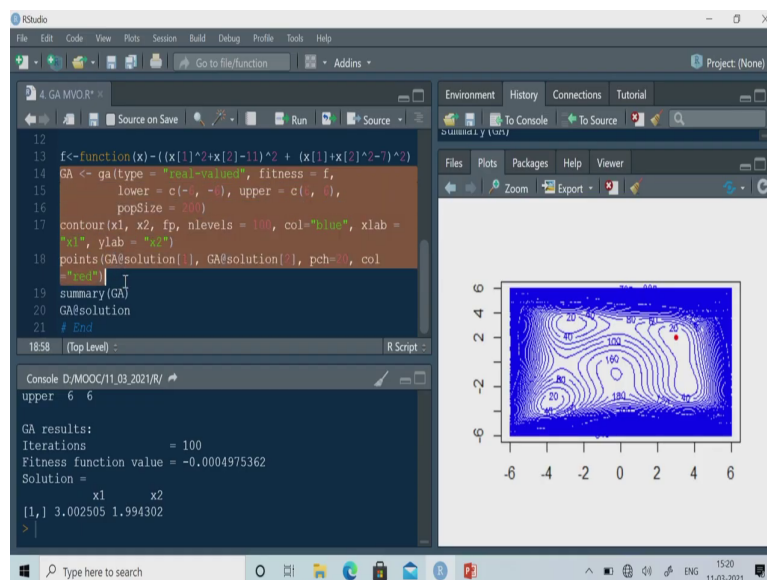


So, then, I can see the summary GA and so, GA solution. So, this is the summary and you can see that I have a real-valued, population size is 200, iteration is 100, number of generation is 100, elitism is 10, crossover probability I did not change.

(Refer Slide Time: 42:10)



(Refer Slide Time: 42:13)



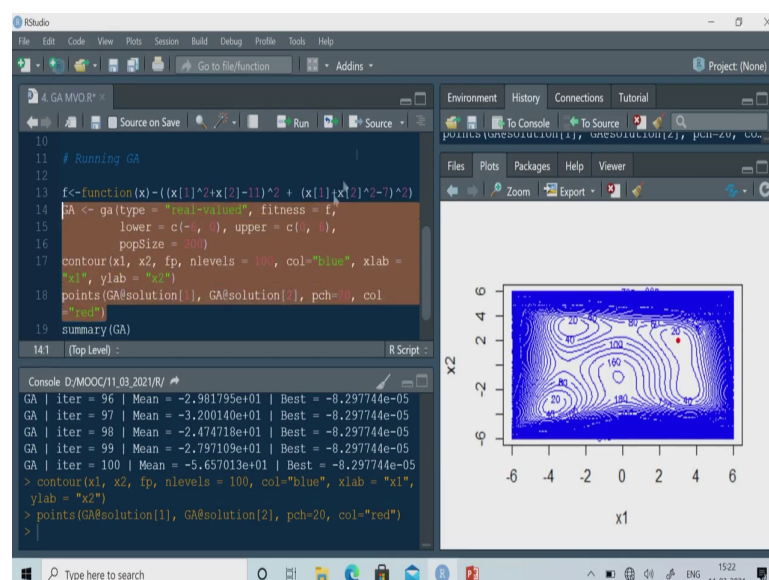
So, this is 0.1 0.8 and 0.1 and the solution there so this is upper bound, lower bound of x_1 and x_2 minus 6 and 6, minus 6 and 6 for x_1 and x_2 and you are getting the solution. So, it is not exactly 0, but it is around 10 to power minus 3. So, you can also define this precision; but whatever solution you are getting 3.00 and 1.99, but exact solution is 3 and 2.

So, you are not getting the exact solution, but you are getting close to the optimal solution that is 3.002. So, this is you are getting and this is you are getting 1.99. So, we are getting one solution. Now, if I execute this one for a second time, maybe I may get this solution or I may also get the other solution.

So, let us see that if I execute this particular lines, so second time. So, let us see whether I am getting the same solution or I am getting the different. I am getting a different solution. So, I am getting a different solution. This time you see that I got a solution in the second quadrant.

So, let us run it again, yeah, I am getting on the first quadrant, first quadrant, again first quadrant, again first quadrant. So, I am getting the solution on the first quadrant; but so you may get the solution in the other coordinate also. So, what I can do basically? I can also get the solution by changing this upper bound and lower bound. Suppose, I would like to get the solution of the second quadrant; then what I will do?

(Refer Slide Time: 44:06)

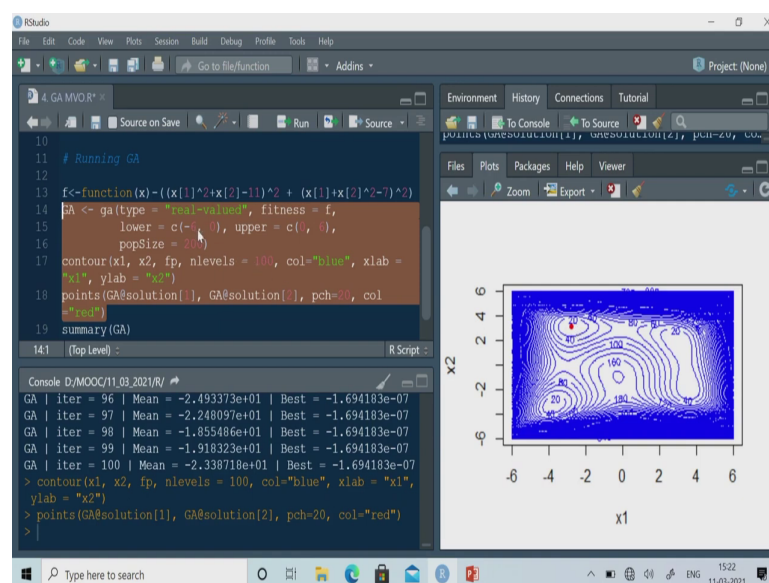


So, for lower bound, this is minus 6 and upper bound of x 2, I am putting 0 here; sorry a lower bound, I am putting 0 and upper bound of x 1, I am putting 0 and lower bound of that I

am putting 6. So, what will happen? Now, I have used the upper bound and lower bound of the second quadrant ok.

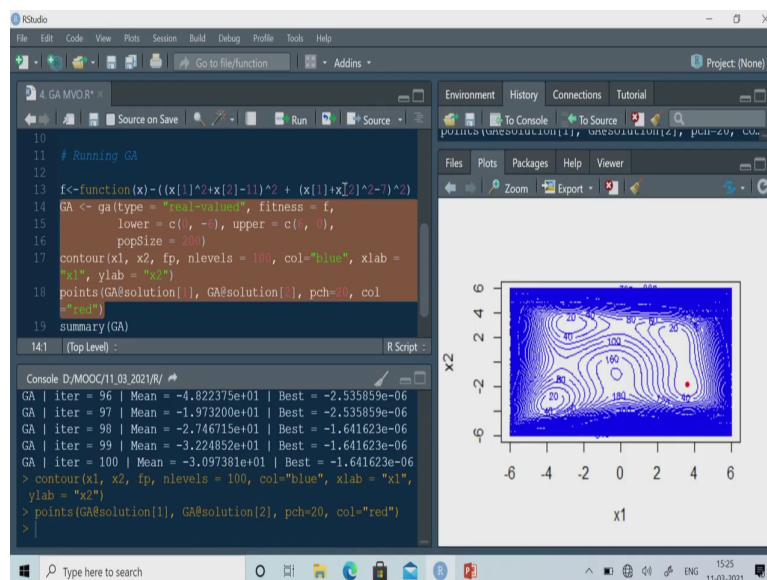
So, minus 6 and 0 and this is 0 and 6 ok. So, in that case, what I should get the solution in the second quadrant. So, you can see that one whether I am getting the solution on the second quadrant. So, let us see. Yeah, I am getting the solution on the second quadrant.

(Refer Slide Time: 44:49)



So, similarly you can also change this thing. So, in order to get the solution at the third quadrant, so let us put that x 1 is between minus 6 and 0 and x 2 is also between minus 6 and 0 and if I execute this particular lines, then I should get the solution at the third quadrant.

(Refer Slide Time: 45:17)



So, similarly, I can also get the solution at the fourth quadrant. So, fourth quadrant this is x_1 is between 0 and 6, 0 and 6 and x_2 is between minus 6 and 0. So, let us execute this particular lines. So, I am getting the solution at the fourth quadrant. So, therefore, by changing these upper bound and lower bound, so I can actually get the solution at different quadrant.

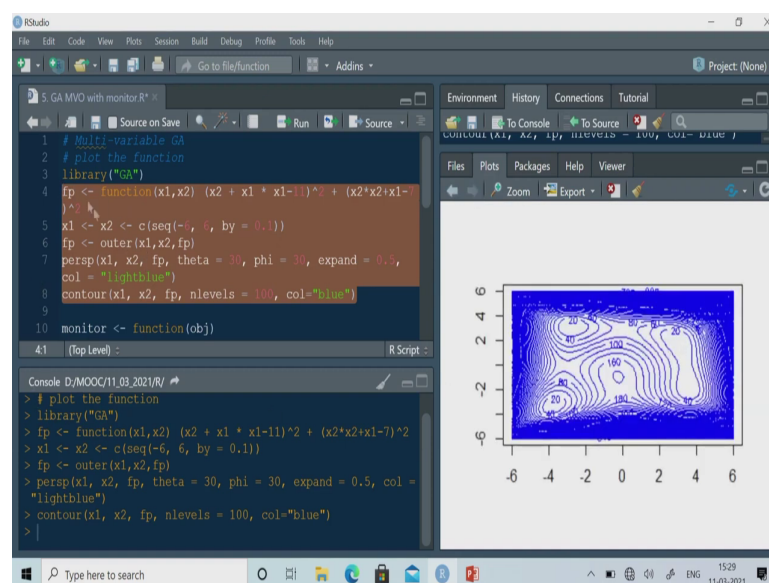
But if you are giving between if you are giving this range between minus 6 and plus 6, you will get one of the solution. Like the single variable function, I can also monitor the solution in case of multi-variable function, let us see how we can monitor the GA iteration.

So, here for this particular function again that is x_2 plus x_1 square minus 11, then plus x_2 square plus x_1 minus 7 whole square. So, for this particular function, so I would like to put

the monitor function ok. So, that I will do. So, as you have seen, so first part is just to plot this particular function. So, I am not explaining again.

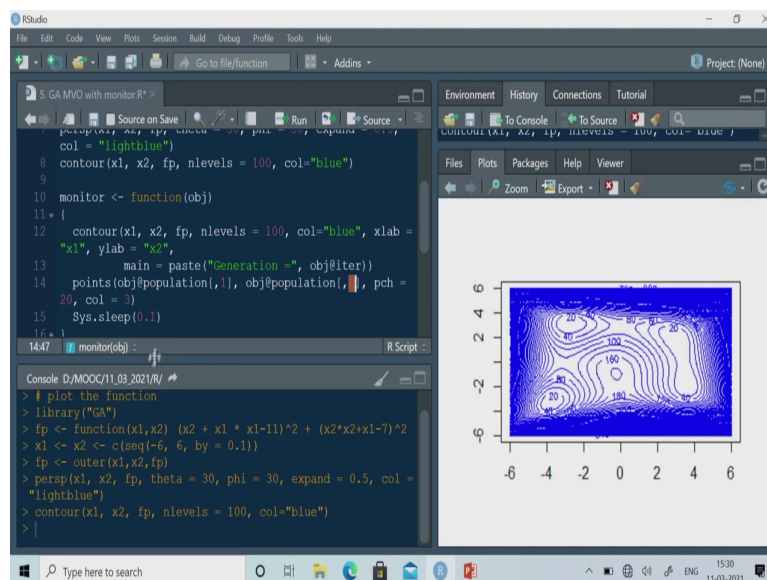
So, I have written the function, then I am defining x_1 and x_2 , then I am calculating the function value at is grid point using outer function and then, I am plotting it; the 3D plot using persp function and I can I can also plot the contour lines using contour function.

(Refer Slide Time: 46:54)



So, you have to use library ok. So, library is “GA”. Now, let us execute this few lines. So, just to plot that one, plot the function. So, I am getting the 3D view as well as the contour plots. So, I am just plotting here. So, we are and then, I have written this monitor function.

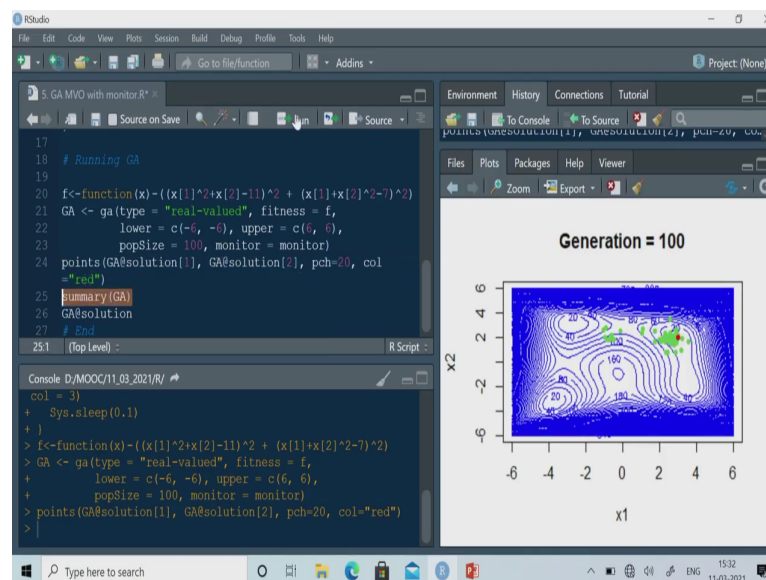
(Refer Slide Time: 47:31)



So, in this case, so earlier monitor function, so we have used curve because that was a single variable function; but in this case, I am I have used contour. So, here, we have two variables; that is x 1 and x 2. This is the function value, the number of number of levels, contour levels I have used 100; then, color I am using blue and then, I am also defining what is x level and y level.

And then, in the main, I am writing or I would like to write the generation. So, that I am doing here and then, I am putting the population that using points functions. So, in this case, there are two variables. So, therefore, first one is give you the first variable; this is x 1 and I am writing comma 1; that means, first variable for all rows and similarly, for second variable for all rows ok and then, pch you are using 20.

(Refer Slide Time: 48:27)



So, it will give you circular shape and then color, I am using 3 and then system will go for sleep for 0.1. So, this is the monitor function. So, let us execute this. So, this is the monitor function I have; the monitor function I have executed and now, I have written the function the fitness value and then, I have used the GA.

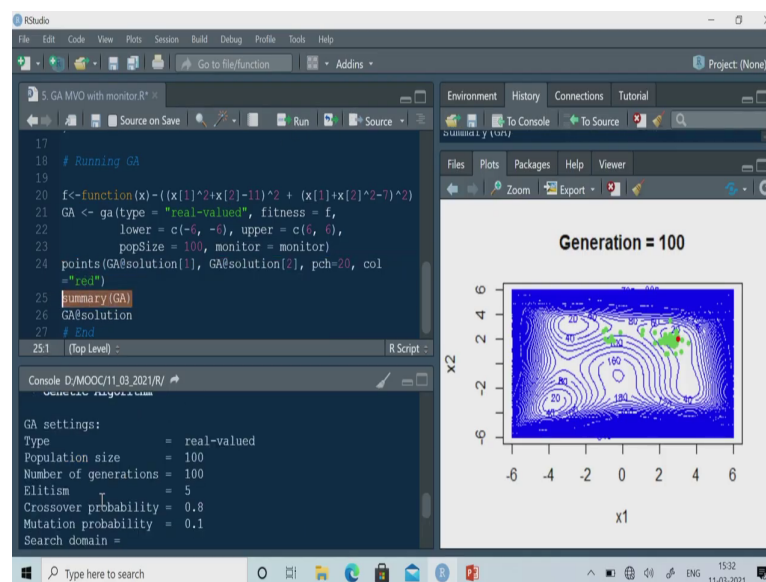
So, in this case, so I am just using monitor equal to monitor, then other part is similar ok. So, you can see that it is a real-valued, fitness function is f, then lower bound and upper bound I have defined, population size I am using 100, but you can increase change it and now, I am putting monitor. So, I would like to monitor the progress ok; monitor the iteration.

So, let us execute this one. So, you can see the population all over the source space and with the progress of iteration, you can see that it will converge at one of the optimal solution. So, I

think in this case, it will converge at the first quadrant solution. So, just see. So, it is converging the solution at the first quadrant.

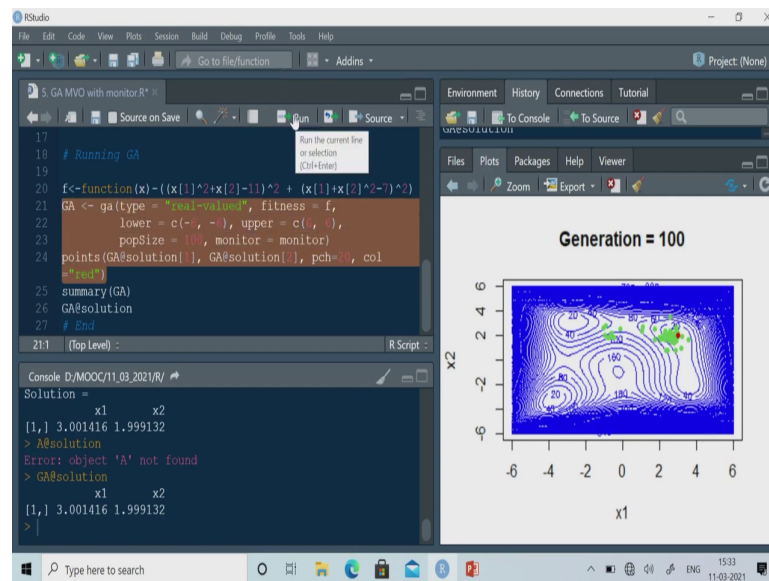
So, you can see that initially all these populations were uniformly distributed all over the source space that is between minus 6 and plus 6 and finally, it is converging at one of the solution. So, you can see, so it will go up to 100 iteration and so, we got the solution; solution is somewhere here and just to plot the solution, so I have used the point function. So, I would like to plot the final solution. So, you can see that this is the solution, final solution we got.

(Refer Slide Time: 50:51)



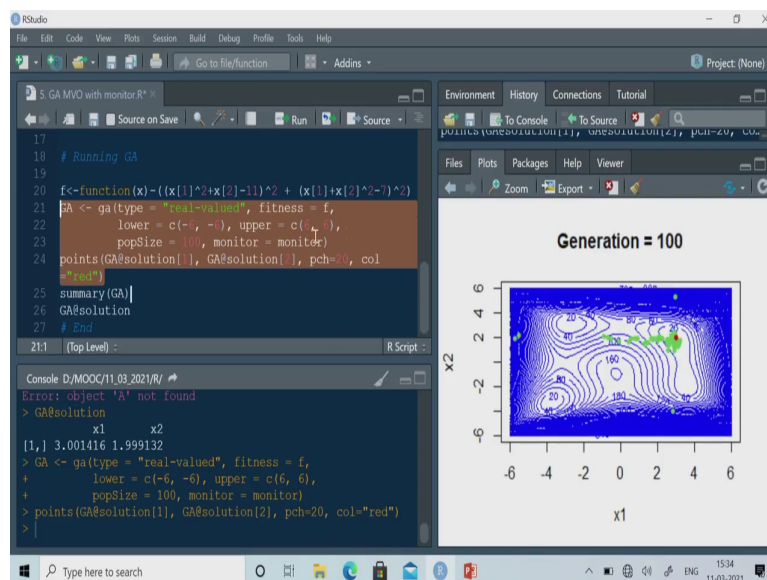
And if you want to see the value, so I can use GA summary. And GA summary will give you everything that is what type of setting you have used and then, if I run this particular GA solution.

(Refer Slide Time: 51:07)



So, I can see the solution. So, the solution is 3.00 and 1.99. So, as you know that you will get the a near optimal solution, exact solution you may not get. But I hope this is acceptable to us. So, we got this solution and maybe if you are running it for the second time, so we may get other solution; but we may also get this particular solution. So, let us see.

(Refer Slide Time: 51:36)



So, just see whether we are getting the same solution or we may get some other solution. So, probably, we will get the solution and the first quadrant itself and you can see that with population, initially the solutions were uniformly distributed; but over the iteration it is converging one of one of these four solution and now, converging it converging at the solution on the first quadrant. So, but you may get the solution at the other quadrant also. So, these are all alternate optimal solution. So, therefore, GA will give one of them.

So, in this case also, so we will get the solution at the first quadrant. Yeah. So, you can see that we got the solution at the first quadrant itself ok. So, we have applied GA in three problems. So, two single variable problems and one multivariable problems. We tried to solve this problem using R.

So, I will also give you the R code, so you try yourself to get the solution. So, in the next class, so we will solve problem with constraint. So, I will explain how GA can handle in non-linear problem with constraint and GA is very efficient in handling constraints, so that I will explain in the next class.