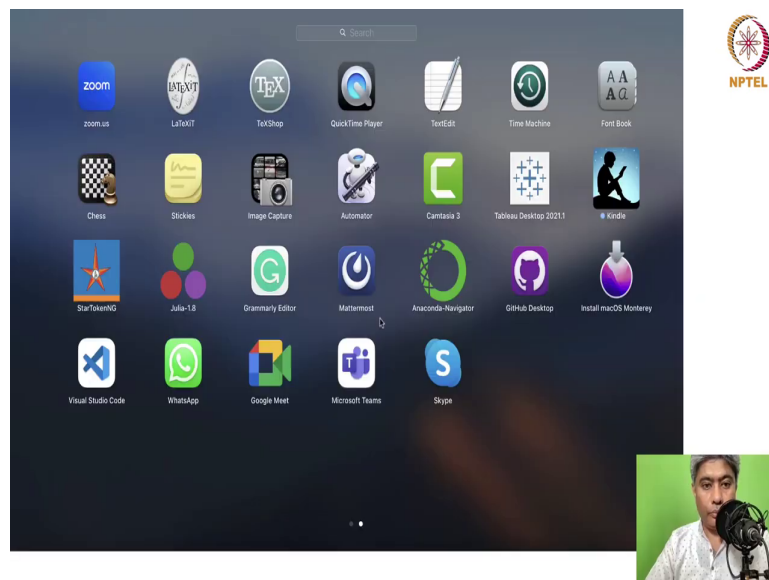


Predictive Analytics - Regression and Classification
Prof. Sourish Das
Department of Mathematics
Indian Institute of Technology, Madras

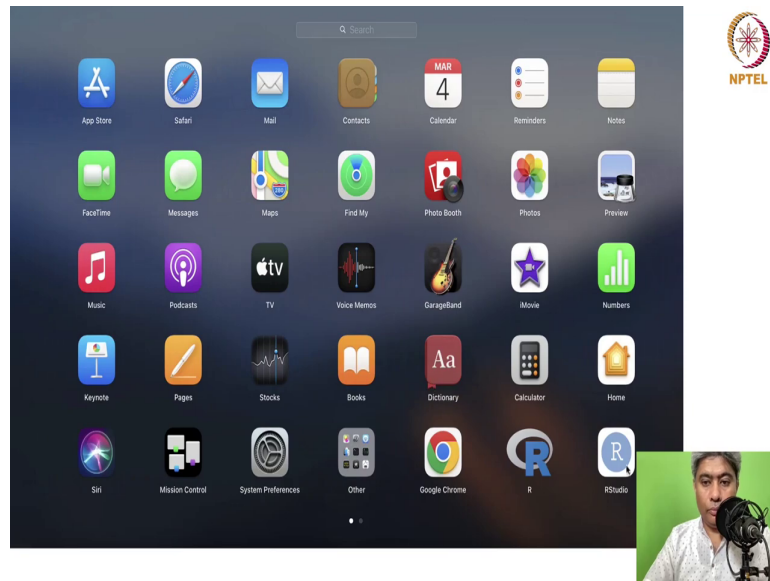
Lecture - 50
Hands on with R: Implement GP Regression from scratch

Hello all welcome back to part B of lecture 15. In this discussion, we are going to see a demo of how to Implement Gaussian Process Regression from scratch using R.

(Refer Slide Time: 00:36)

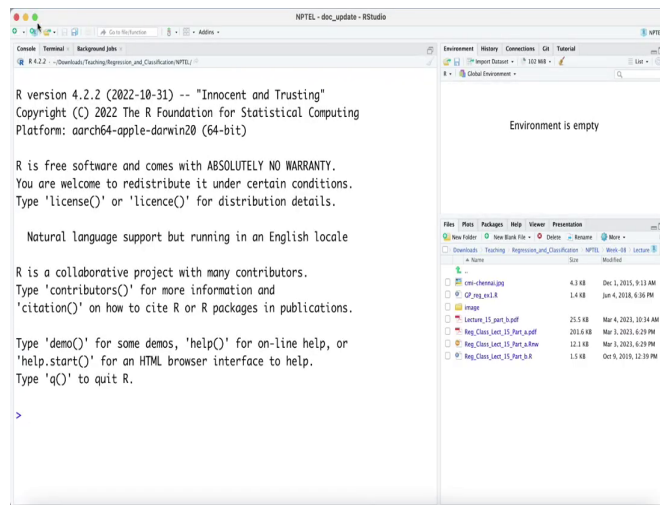


(Refer Slide Time: 00:37)



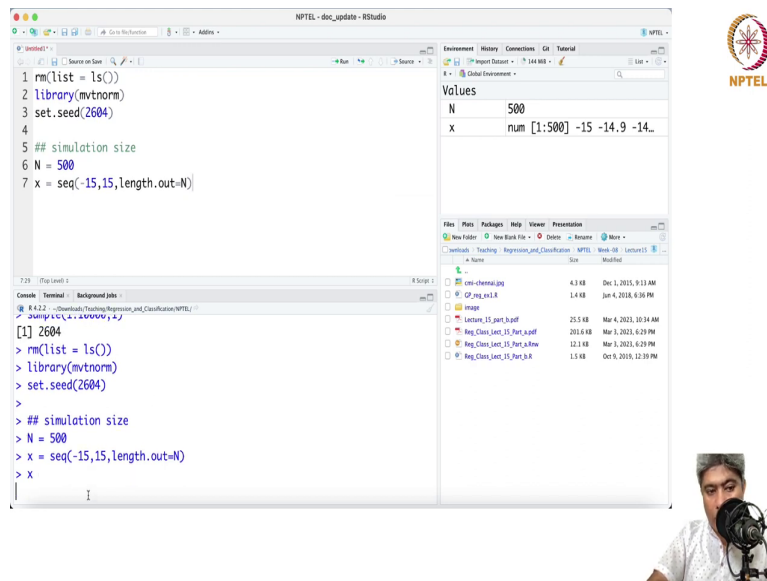
So, I am going to start my R.

(Refer Slide Time: 00:41)



I am going to open R and R script.

(Refer Slide Time: 00:49)



The screenshot displays an RStudio window titled "NPTEL - doc_update - RStudio". The script editor contains the following R code:

```
1 rm(list = ls())
2 library(mvtnorm)
3 set.seed(2604)
4
5 ## simulation size
6 N = 500
7 x = seq(-15,15,length.out=N)
```

The console shows the execution of these commands, resulting in the following output:

```
[1] 2604
> rm(list = ls())
> library(mvtnorm)
> set.seed(2604)
>
> ## simulation size
> N = 500
> x = seq(-15,15,length.out=N)
> x
```

The Environment pane on the right shows the values of the variables:

Variable	Value
N	500
x	num [1:500] -15 -14.9 -14...

The NPTEL logo is visible in the top right corner of the slide.

So, first thing I am going to say that ok, remove just this list anything if it is there ls. So, anything if there is any later if I you know have any just clean the environment essentially. I am going to say install a load mvtnorm package. Let me see, I hope it is there. Yeah, it is there. Then I am going to set up a seed.

So, let me just draw a random sample sample sorry, a sample between a number between 1 and say 10000 one sample is good enough 2604. So, I will just pick this random number and set dot seed as this. I am setting this number so that when you will use this number, you will get the exact same answer.

Then I will decide a simulation size simulation size of say around 500. I think ok, that many data points we will simulate, ok. So, first I am going to define a x variables sequence of

number between say minus 15 to 15 and length, length dot out equal to capital N. So, if you just run this. So, now, you can see if you just run x.

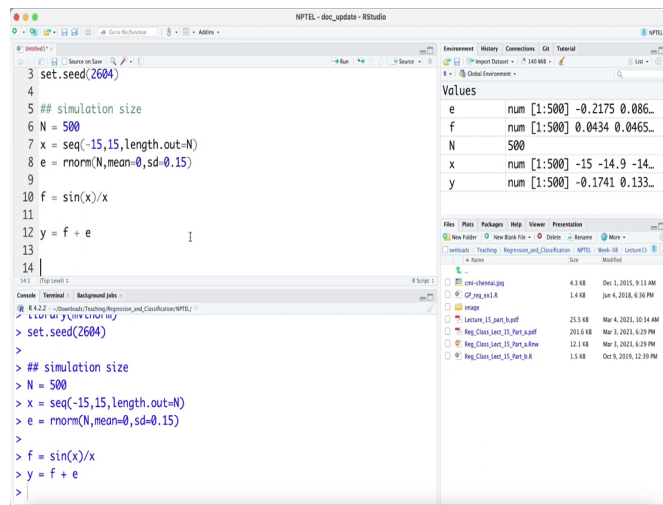
(Refer Slide Time: 02:54)

The screenshot displays the RStudio interface. The console window shows a matrix of 500 rows and 5 columns of random values. The 'Values' window shows the variable 'N' set to 500 and 'x' set to 'run(1:500)' with a range of [-15, -14.9, -14...]. The file explorer shows a project named 'NPTEL' with several files and folders.

Line	Col 1	Col 2	Col 3	Col 4	Col 5
[391]	8.44689379	8.50701403	8.56713427	8.62725451	8.68737475
[396]	8.74749499	8.80761523	8.86773547	8.92785571	8.98797595
[401]	9.04809619	9.10821643	9.16833667	9.22845691	9.28857715
[406]	9.34869739	9.40881764	9.46893788	9.52905812	9.58917836
[411]	9.64929860	9.70941884	9.76953908	9.82965932	9.88977956
[416]	9.94989980	10.01002004	10.07014028	10.13026052	10.19038076
[421]	10.25050100	10.31062124	10.37074148	10.43086172	10.49098196
[426]	10.55110220	10.61122244	10.67134269	10.73146293	10.79158317
[431]	10.85170341	10.91182365	10.97194389	11.03206413	11.09218437
[436]	11.15230461	11.21242485	11.27254509	11.33266533	11.39278557
[441]	11.45290581	11.51302605	11.57314629	11.63326653	11.69338677
[446]	11.75350701	11.81362725	11.87374749	11.93386774	11.99398798
[451]	12.05410822	12.11422846	12.17434870	12.23446894	12.29458918
[456]	12.35470942	12.41482966	12.47494990	12.53507014	12.59519038
[461]	12.65531062	12.71543086	12.77555110	12.83567134	12.89579158
[466]	12.95591182	13.01603206	13.07615230	13.13627255	13.19639279
[471]	13.25651303	13.31663327	13.37675351	13.43687375	13.49699399
[476]	13.55711423	13.61723447	13.67735471	13.73747495	13.79759519
[481]	13.85771543	13.91783567	13.97795591	14.03807615	14.09819639
[486]	14.15831663	14.21843687	14.27855711	14.33867735	14.39879760
[491]	14.45891784	14.51903808	14.57915832	14.63927856	14.69939880
[496]	14.75951904	14.81963928	14.87975952	14.93987976	15.00000000

If you can see there are 500 samples 500 values are being created between minus 15 and 15 each are of equals with equal you know interval, ok. Now, I am going to draw a random samples bunch of random samples say e equals to rnorm.

(Refer Slide Time: 03:32)



```
3 set.seed(2604)
4
5 ## simulation size
6 N = 500
7 x = seq(-15,15,length.out=N)
8 e = rnorm(N,mean=0,sd=0.15)
9
10 f = sin(x)/x
11
12 y = f + e
13
14
```

Console

```
> set.seed(2604)
>
> ## simulation size
> N = 500
> x = seq(-15,15,length.out=N)
> e = rnorm(N,mean=0,sd=0.15)
>
> f = sin(x)/x
> y = f + e
>
```

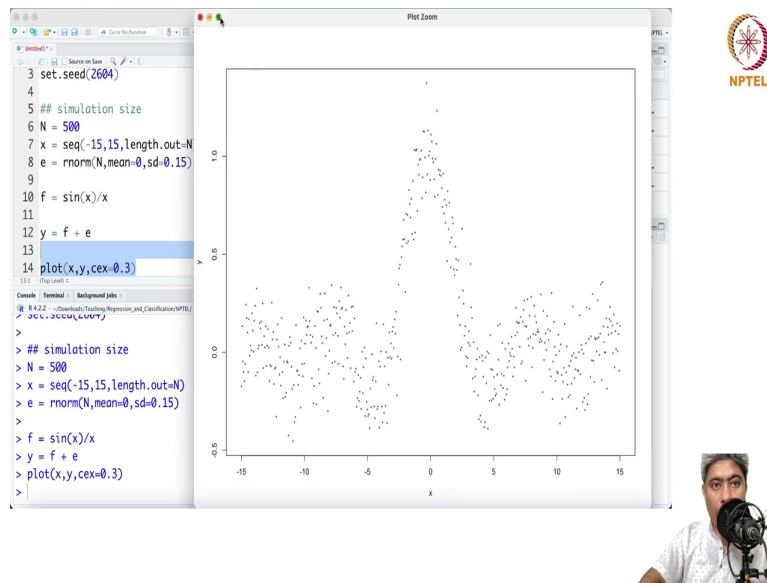
Values

e	num [1:500]	-0.2175	0.086...
f	num [1:500]	0.0434	0.0465...
N		500	
x	num [1:500]	-15	-14.9 -14...
y	num [1:500]	-0.1741	0.133...



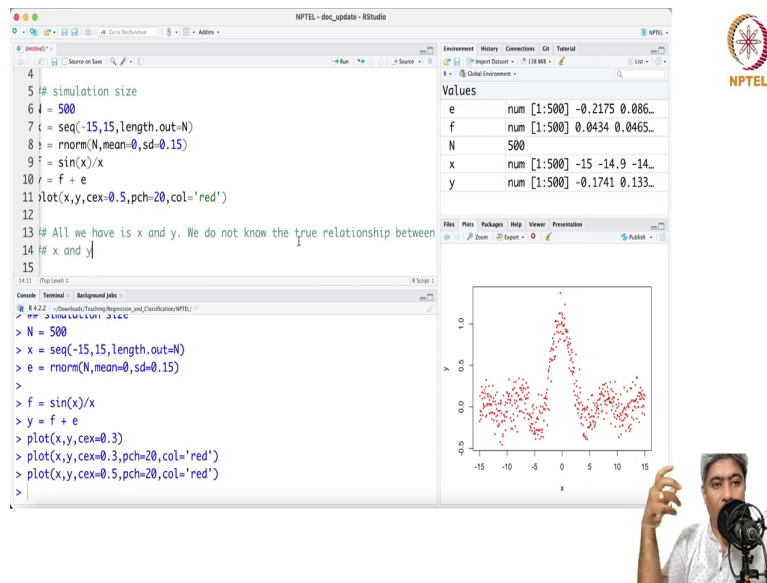
How many N many with mean equal to 0 and sd equal to 0.15, ok. And then I am going to run right, f of x. If is sin of x by x and then I just run this and then now y is equal to y values are my whatever f plus e, f of x plus e, correct.

(Refer Slide Time: 04:23)



Now, if I plot x, y cex equal to 0.3. So, I can let me zoom this, ok. Let me just zoom it. So, you can see all the points are around this, ok. And in fact, you can make it pch equal to 20 maybe color equal to red.

(Refer Slide Time: 05:01)



The screenshot shows the RStudio interface with the following R code in the editor:

```
4  
5 # simulation size  
6 i = 500  
7 i = seq(-15,15,length.out=N)  
8 i = rnorm(N,mean=0,sd=0.15)  
9 f = sin(x)/x  
10 r = f + e  
11 plot(x,y,cex=0.5,pch=20,col='red')  
12  
13 # All we have is x and y. We do not know the true relationship between  
14 # x and y  
15
```

The console shows the execution of the code:

```
> N = 500  
> x = seq(-15,15,length.out=N)  
> e = rnorm(N,mean=0,sd=0.15)  
>  
> f = sin(x)/x  
> y = f + e  
> plot(x,y,cex=0.3)  
> plot(x,y,cex=0.3,pch=20,col='red')  
> plot(x,y,cex=0.5,pch=20,col='red')  
>
```

The 'Values' pane displays the following summary statistics:

Variable	Summary
e	num [1:500] -0.2175 0.086...
f	num [1:500] 0.0434 0.0465...
N	500
x	num [1:500] -15 -14.9 -14...
y	num [1:500] -0.1741 0.133...

The plot shows a scatter plot of red points with a cex of 0.5, forming a bell-shaped curve centered at x=0. The x-axis ranges from -15 to 15, and the y-axis ranges from -0.5 to 1.0.

The NPTEL logo is visible in the top right corner of the slide.

So, this is this will be little brighter or we can just increase the value little bit cex. Yeah, it is slightly better, ok; looks much better, ok.

(Refer Slide Time: 07:05)

The screenshot shows an RStudio interface with the following code in the script editor:

```
10 y = f + e
11 plot(x,y,cex=0.5,pch=20,col='red')
12
13 ## All we have is x and y. We do not know the true relationship between
14 ## x and y
15
16
17 data = cbind.data.frame(x,y)
18
19 ## Write negative log-likelihood function of GP Regression
20
21 neg_log_like = function(theta,D)
```



The console shows the execution of the following commands:

```
> e = rnorm(N,mean=0,sd=0.15)
> f = sin(x)/x
> y = f + e
> plot(x,y,cex=0.3)
> plot(x,y,cex=0.3,pch=20,col='red')
> plot(x,y,cex=0.5,pch=20,col='red')
> data = cbind.data.frame(x,y)
> View(data)
```

The environment pane shows the following data:

Variable	Class	Summary
e	num	[1:500] -0.2175 0.086...
f	num	[1:500] 0.0434 0.0465...
N	num	500
x	num	[1:500] -15 -14.9 -14...

The plot shows a scatter of red points with a distinct peak at x=0. The x-axis ranges from -15 to 15, and the y-axis ranges from -0.5 to 1.0.



So, now this is my plot. This is my data. Now, we are going to pretend that all we have is x and y . We are going to pretend all we have is x and y , ok. All we have is x and y and we do not know the true relationship between x and y . So, given the data, can we pick up the this completely. There are some there is no trend actually there is bunch of seasonality and most of the values are hovering around 0.

And then there is at 0, there is a signal boom. There is a burst of a signal and then it came down. And then again, it is sort of a hovering around 0. So, can we capture this complete non-linear behavior using GP regression? Ok. So, this is our idea. So, now what I am going to do, we are going to create a data set. First thing what we will do, we will just say cbind data dot frame x comma y . So, here is my data x and y 's and these are the. So, now we have a data which has one x and one y and we are trying to fit a model out of it.

So, the first thing is we will fit a Gaussian process regression and we will model it from scratch. There are some built-in packages are there, but I want to understand if you use built-in package, we will not really understand how GP regression really works, ok. So, what I am going to do? I am going to teach you how the GP regression works.

So, best way to learn is do programming from scratch, alright. So, first thing I will do, write GP regression negative log likelihood of GP regression, Write negative log-likelihood function of GP Regression, ok. So, neg log like equal to function y comma x, ok; and then no, other way we will do is theta and D. So, D for data, theta is all the parameters.

(Refer Slide Time: 08:57)

The screenshot shows the RStudio interface with the following code in the editor:

```
18  
19 ## Write negative log-likelihood function of GP Regression  
20  
21 neg_log_like = function(theta,D){  
22   y=D[, "y"]  
23   x=D[, "x"]  
24   print(sum(y))  
25   print(sum(x))  
26 }  
27 neg_log_like(c(0,1),D=data)  
28 sum(data$x) }  
29 sum  
30
```

The console output shows the execution of the function and the sum of the data:

```
> neg_log_like(c(0,1),D=data)  
[1] 57.73925  
[1] -7.531753e-13  
> sum(data$x)  
[1] -7.531753e-13  
>
```

The 'Values' pane displays the following data:

	num	[1:500]	
e	num	[1:500]	-0.2175 0.086...
f	num	[1:500]	0.0434 0.0465...
N			500
x	num	[1:500]	-15 -14.9 -14...
y	num	[1:500]	-0.1741 0.133...

The 'Functions' pane shows the function definition. A scatter plot of the data points is visible in the bottom right, showing a distribution of points with x-axis ranging from -15 to 15 and y-axis ranging from -0.5 to 1.0.

So, what we will do is y equal to D comma y and x equal to D comma x, ok. And let me just write; let me see if it is actually working, ok. So, the best way of doing it is just write sum y print, ok print sum of y and print sum of x, ok. So, let me just run only this part and negative

log likelihood and say 0 comma 1 that will be for theta and D equal to data. Yeah, 57.73 and some very close to 0. So, if I take data, just check data dollar x. If I just take sum of that. And they are exactly matching and then if I just take sum of data dollar y 57.73925.

(Refer Slide Time: 10:50)

The screenshot shows the RStudio interface with the following R code in the editor:

```
19 ## Write negative log-likelihood function of regression
20 ## We will use exponential covariance function (RBF)
21
22 neg_log_lik = function(theta,D){
23   ## Read the data
24   y=D[,"y"]
25   x=D[,"x"]
26   n=nrow(D)
27
28   I
29
30 }
```

The console shows the execution of the function and subsequent data checks:

```
> neg_log_lik(c(0,1),D=data)
[1] 57.73925
[1] -7.531753e-13
> sum(data$x)
[1] -7.531753e-13
> sum(data$y)
[1] 57.73925
> |
```

The Environment pane shows the following values:

Variable	Value
e	num [1:500] -0.2175 0.086...
f	num [1:500] 0.0434 0.0465...
N	500
x	num [1:500] -15 -14.9 -14...
y	num [1:500] -0.1741 0.133...

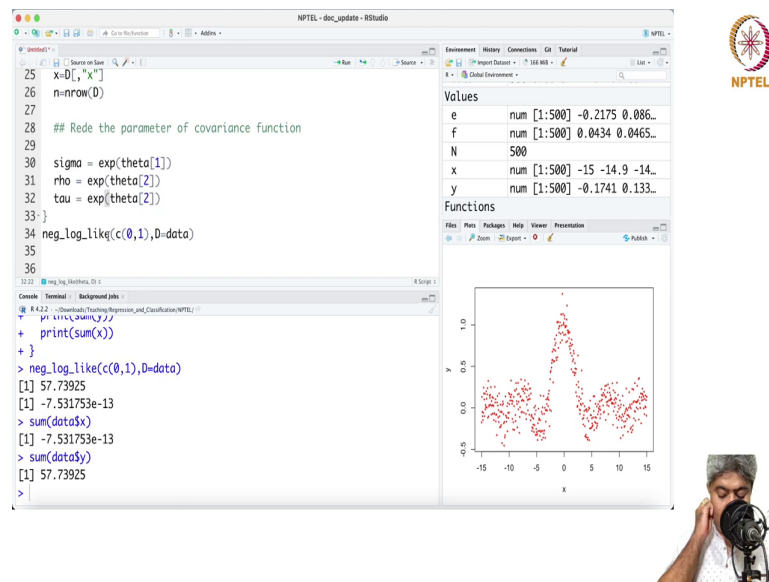
The Functions pane shows the function definition. A scatter plot of y vs x is visible, showing a noisy distribution. The NPTEL logo is in the top right, and a small video inset of a person speaking is in the bottom right.

So, they are exactly matching. So, that means, this part is happening absolutely correctly. So, I can delete this part and I can delete the print also, ok. So, now we have (Refer Time: 11:09) that means, this function reading the data correctly, ok. So, next thing we will do n equal to nrow D, ok. And I think that will be not a problem.

And so read the data. This is first. These are the three things. I think first is Read the data. And in this GP regression, we are going to use exponential covariance function. So, let me write it down explicitly We will use exponential covariance function also sometimes known

as RBF Radial Basis Function, ok. So, now and that will have three parameters in that covariance function.

(Refer Slide Time: 12:37)



So, the read the parameters of the covariance functions. Read the parameters of covariance function, ok. And what are those? First is sigma, ok. Sigma equal to e to the power theta 1. And why I am taking e to the power theta 1? What happens eventually I want to put it through an optimization subroutine.

I am going to call the optim function in this case. And in the optim function, the theta will be expected to be varying from minus infinity to infinity. But I want my sigma obviously, on the 0 to infinity range. So, as soon as it the I get the theta, I will transform it and then I will use it.

So, we will get we will effectively optimize a transform variable or transform parameter. Then next is rho. Rho is e to the power theta 2 and tau. Tau is equal to also exponent theta 2, ok. Now, what I am going to do is I am going to calculate the distance matrix.

(Refer Slide Time: 14:38)

The screenshot shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The editor contains the following R code:

```

34 ## Distance matrix
35
36 Dist = absps.matrix(dist(x,method = "euclidean",diag = T,upper = T))
37
38 ## Compute covariance matrix Sigma
39
40 Sigma = sigma*exp(-rho*abs(Dist))
41
42 }
43 neg_log_lik(c(0,1),D=data)
44

```

The console shows the execution of the code:

```

> neg_log_lik(c(0,1),D=data)
[1] 57.73925
[1] -7.531753e-13
> sum(data$x)
[1] -7.531753e-13
> sum(data$y)
[1] 57.73925
>

```

The environment pane on the right shows the following values:

Variable	Value
e	num [1:500] -0.2175 0.086...
f	num [1:500] 0.0434 0.0465...
N	500
x	num [1:500] -15 -14.9 -14...
y	num [1:500] -0.1741 0.133...

The plot pane shows a scatter plot of data points (x vs y) with a red density gradient. The x-axis ranges from -15 to 15, and the y-axis ranges from -0.5 to 1.0. The data points are clustered in a shape resembling a stylized 'A' or a similar character.

Let us first calculate the distance matrix. Distance matrix so, Dist where effectively what I am going to do call the dist function available in the R. So, Dist x method equals to euclidean diagonal equal to diagonal equals to true and also I want the upper part of the distance matrix as well. And I want to store it as a matrix so that I can do operation matrix operation on it later when we will do the optimization, ok.

So, this is the distance matrix and then I am going to compute the covariance matrix sigma. Let us compute. Compute covariance matrix sigma, ok. What is covariance matrix sigma? So, sigma is essentially the sigma parameter we have just right here sigma parameter times e to

the power e to the power minus ρ times absolute value of the distance absolute value of the distance. So, in fact, what I can do? I can take the absolute value here. So, that you will or I can just write it. I do not want to make it too big. Distance I can equal to abs distance.

(Refer Slide Time: 17:01)

The screenshot shows an RStudio window titled 'NPTEL - doc_update - RStudio'. The editor contains the following R code:

```

36 Dist = as.matrix(dist(x,method = "euclidean",diag = T,upper = T))
37
38
39 ## Compute covariance matrix Sigma
40
41 Sigma = sigma*exp(-rho*abs(Dist)+diag(tau,nrow=n))
42 mu = rep(mean(y),n)
43
44 nll = -dmvnorm(y,mean=mu, sigma=Sigma,log=TRUE)
45 return(nll)
46
47 }

```

The console shows the following output:

```

> neg_log_likelihood(D=data)
[1] 57.73925
[1] -7.531753e-13
> sum(data$x)
[1] -7.531753e-13
> sum(data$y)
[1] 57.73925
>

```

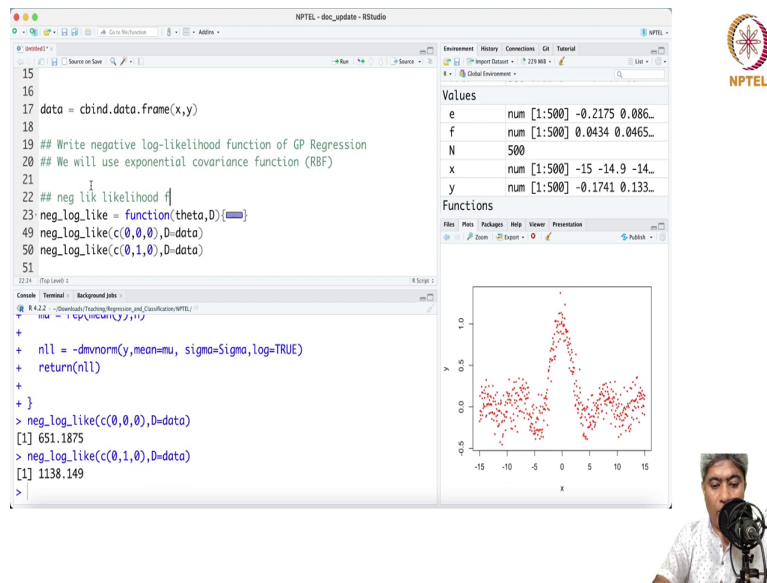
The environment pane shows the following values:

Variable	Value
e	num [1:500] -0.2175 0.086...
f	num [1:500] 0.0434 0.0465...
N	500
x	num [1:500] -15 -14.9 -14...
y	num [1:500] -0.1741 0.133...

The plot pane shows a scatter plot of data points (x vs y) with a red density contour overlaid, indicating a bimodal distribution along the x-axis.

Or it does not matter actually we can just have it here. I think this is fine and diagonal of τ diagonal of τ where n row equal to n , ok. Now, μ I am going to say that replicate mean of y as a function of n , ok. And then I am going to write negative log likelihood, evaluate negative log likelihood as function of negative value of first dmvnorm y as a mean equal to μ σ equal to this capital σ and \log equal to true. And I have to have a negative sign in the beginning and return, return negative log likelihood, ok.

(Refer Slide Time: 18:33)





The screenshot shows an RStudio window with the following code in the editor:

```
15
16
17 data = cbind.data.frame(x,y)
18
19 ## Write negative log-likelihood function of GP Regression
20 ## We will use exponential covariance function (RBF)
21
22 ## neg lik likelihood f
23 neg_log_like = function(theta,D){
49 neg_log_like(c(0,0,0),D=data)
50 neg_log_like(c(0,1,0),D=data)
51
22:24 (Rip Level)
Console Terminal Background Jobs
R 4.2.2 - Downloads/Teaching/Regression_and_Classification/NPTEL/
> nll = -dmvnorm(y,mean=mu, sigma=Sigma,log=TRUE)
+ return(nll)
+ }
> neg_log_Like(c(0,0,0),D=data)
[1] 651.1875
> neg_log_Like(c(0,1,0),D=data)
[1] 1138.149
>
```

The console output shows the results of the function calls. The 'Values' pane displays the following data:

	num	[1:500]	-0.2175	0.086...
e	num	[1:500]	0.0434	0.0465...
f	num	[1:500]	-15	-14.9
N	num	[1:500]	-14	-14...
x	num	[1:500]	-0.1741	0.133...
y	num	[1:500]		

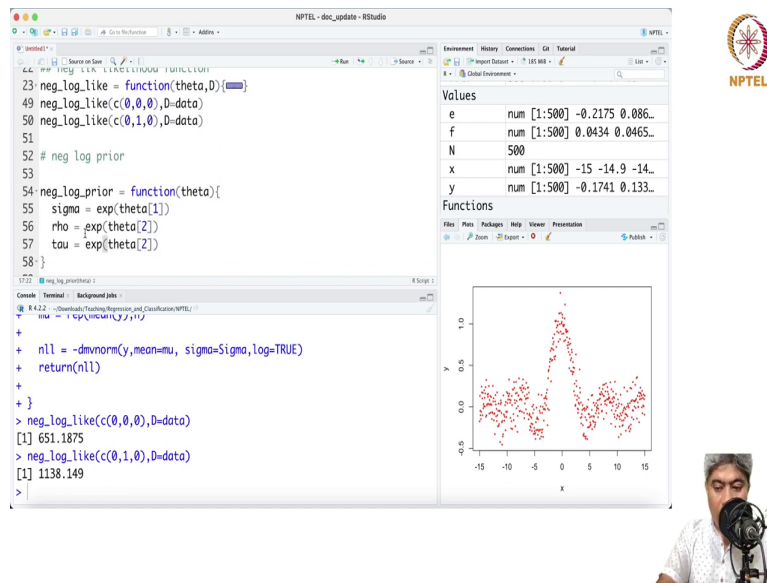
The 'Functions' pane shows the function definition. The plot pane displays a scatter plot of data points (x vs y) with a red density contour overlaid, showing a distribution centered around x=0 and y=0.



So, this is the function. And let me just call this function, ok. And try to evaluate this function at some initial value. So, this gives me one value, this could be another value, ok 1138. So, at some initial values it runs and it gives you some good reason things, ok. Now, next what I am going to do. I am going to write optimize say.

Now, I am going to write some optimization technique optimization. Actually, we have to put some prior on the top of that otherwise, optimization might throw some error. So, let us define. So, this is my negative log likelihood. So, write. Let us write some this was negative log likelihood function.

(Refer Slide Time: 20:01)



The screenshot shows an RStudio window with the following code in the editor:

```
23 neg_log_likelike = function(theta,D){  
49 neg_log_likelike(c(0,0,0),D=data)  
50 neg_log_likelike(c(0,1,0),D=data)  
51  
52 # neg log prior  
53  
54 neg_log_prior = function(theta){  
55   sigma = exp(theta[1])  
56   rho = exp(theta[2])  
57   tau = exp(theta[2])  
58 }  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500
```



The console shows the following output:

```
+ nll = -dmvnorm(y,mean=mu, sigma=Sigma,log=TRUE)  
+ return(nll)  
+ }  
> neg_log_likelike(c(0,0,0),D=data)  
[1] 651.1875  
> neg_log_likelike(c(0,1,0),D=data)  
[1] 1138.149  
>
```

The environment pane shows the following values:

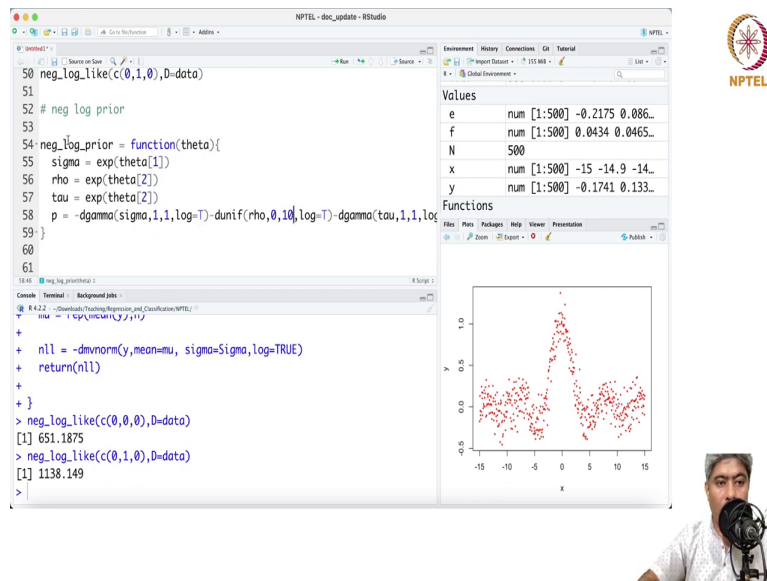
Variable	Value
e	num [1:500] -0.2175 0.086...
f	num [1:500] 0.0434 0.0465...
N	500
x	num [1:500] -15 -14.9 -14...
y	num [1:500] -0.1741 0.133...

The plot pane shows a scatter plot of data points (x, y) with x ranging from -15 to 15 and y ranging from -0.5 to 1.0. The data points are scattered around the origin, with a slight concentration near the top.



And then negative log prior. We have to write. So, we will do negative log prior function we will define as function of theta only, ok. It will not be function of data. It will be only function of theta. And what I am going to do is just copy these few lines at the beginning from the log likelihood. And paste it here. And then, p will be say dgamma.

(Refer Slide Time: 20:53)



The screenshot shows an RStudio interface with the following code in the editor:

```
50 neg_log_like(c(0,1,0),D=data)
51
52 # neg log prior
53
54 neg_log_prior = function(theta){
55   sigma = exp(theta[1])
56   rho = exp(theta[2])
57   tau = exp(theta[2])
58   p = -dgamma(sigma,1,1,log=T) - dunif(rho,0,10,log=T) - dgamma(tau,1,1,log=T)
59 }
60
61
```

The console output shows:

```
> neg_log_like(c(0,0,0),D=data)
[1] 651.1875
> neg_log_like(c(0,1,0),D=data)
[1] 1138.149
>
```

The 'Values' pane shows the following data:

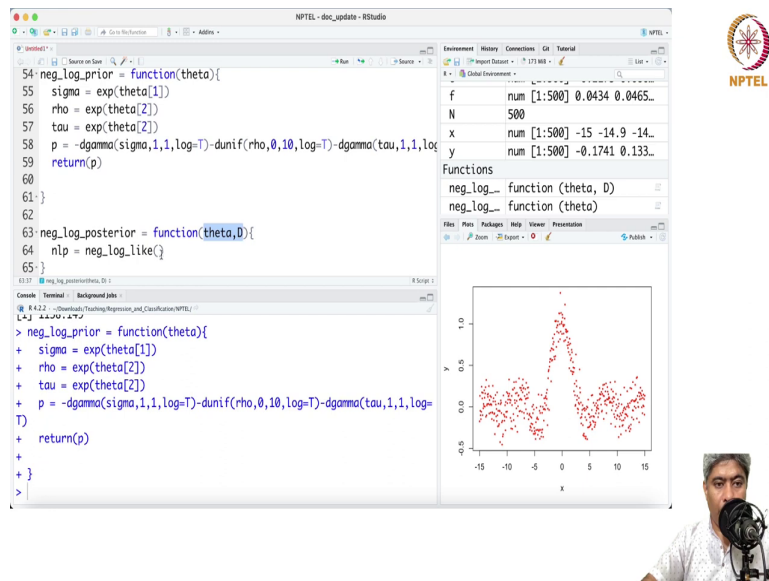
	num	[1:500]	-0.2175	0.086...	
e	num	[1:500]	0.0434	0.0465...	
f	num	[1:500]			
N			500		
x	num	[1:500]	-15	-14.9	-14...
y	num	[1:500]	-0.1741	0.133...	

The 'Functions' pane shows a scatter plot of data points (x vs y) with a red density contour overlaid. The x-axis ranges from -15 to 15, and the y-axis ranges from -0.5 to 1.0.

Let us suppose this is these are all dgamma sigma equal to comma 1,1, log equal to true. Then minus all I have to just do rho and then so, actually what I can do instead of saying that. Yeah, I can just take since I am giving the same prior. Yeah, let me just write it down in any way because later if you want to change anything, you can change by yourself.

Because suppose you decide that ok, on rho, I do not want to put a gamma, I want to put say uniform between 0 and 5 or something like that, that you can do. So, if I want you can give a say uniform between say 0 and 15, ok. So, you can do that or 0 and 10 maybe. Yeah.

(Refer Slide Time: 22:32)



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

```
54: neg_log_prior = function(theta){  
55:   sigma = exp(theta[1])  
56:   rho = exp(theta[2])  
57:   tau = exp(theta[2])  
58:   p = -dgamma(sigma,1,1,log=T)-dunif(rho,0,10,log=T)-dgamma(tau,1,1,log=T)  
59:   return(p)  
60: }  
61: }  
62: }  
63: neg_log_posterior = function(theta,D){  
64:   nlp = neg_log_like(theta,D)  
65: }  
66: }
```

The console window shows the execution of the code:

```
> neg_log_prior = function(theta){  
+   sigma = exp(theta[1])  
+   rho = exp(theta[2])  
+   tau = exp(theta[2])  
+   p = -dgamma(sigma,1,1,log=T)-dunif(rho,0,10,log=T)-dgamma(tau,1,1,log=T)  
+   return(p)  
+ }  
>
```



The Environment pane on the right shows the following variables:

Object	Class	Value
f	num	[1:500] 0.0434 0.0465...
N	num	500
x	num	[1:500] -15 -14.9 -14.9...
y	num	[1:500] -0.1741 0.133...

The Functions pane lists the defined functions:

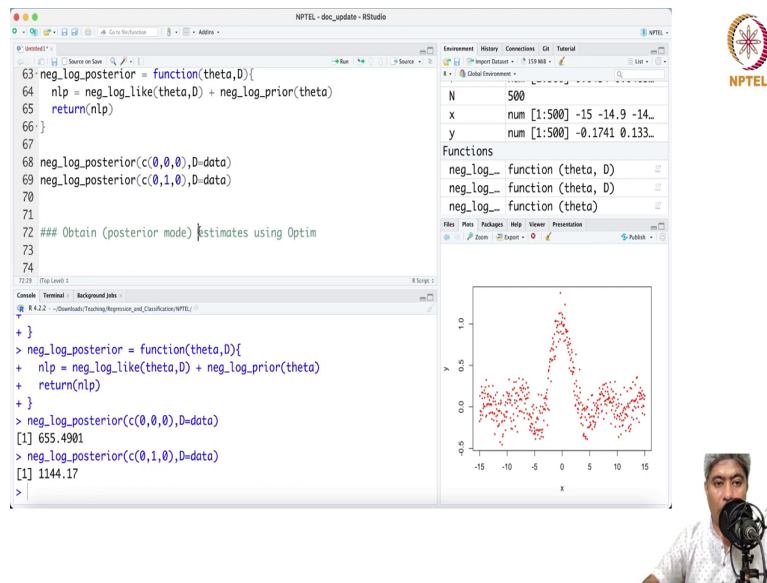
- neg_log_prior function(theta, D)
- neg_log_posterior function(theta, D)
- neg_log_like function(theta, D)

The plot window shows a scatter plot of the data points (x vs y). The x-axis ranges from -15 to 15, and the y-axis ranges from -0.5 to 1.0. The data points are scattered around the origin, with a slight upward trend.



So, you can do this kind of thing. And then eventually return. So, you can assign a choice prior of your choice. So, return p. So, this is my negative log prior, ok. And then finally, negative log posterior which will be simply negative log posterior function theta comma D. And nlp equal to first, what I have to do is basically negative log likelihood comma theta comma D plus negative log prior comma theta.

(Refer Slide Time: 23:45)



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

```
63 neg_log_posterior = function(theta,D){
64   nlp = neg_log_like(theta,D) + neg_log_prior(theta)
65   return(nlp)
66 }
67
68 neg_log_posterior(c(0,0,0),D=data)
69 neg_log_posterior(c(0,1,0),D=data)
70
71
72 ## Obtain (posterior mode) estimates using Optim
73
74
```

The console window shows the execution of the code:

```
> neg_log_posterior = function(theta,D){
+   nlp = neg_log_like(theta,D) + neg_log_prior(theta)
+   return(nlp)
+ }
> neg_log_posterior(c(0,0,0),D=data)
[1] 655.4901
> neg_log_posterior(c(0,1,0),D=data)
[1] 1144.17
>
```

The Environment pane on the right shows the following data:

N	num	500	
x	num	[1:500]	-15 -14.9 -14.9...
y	num	[1:500]	-0.1741 0.133...

The Functions pane lists the following functions:

- neg_log_... function (theta, D)
- neg_log_... function (theta, D)
- neg_log_... function (theta)

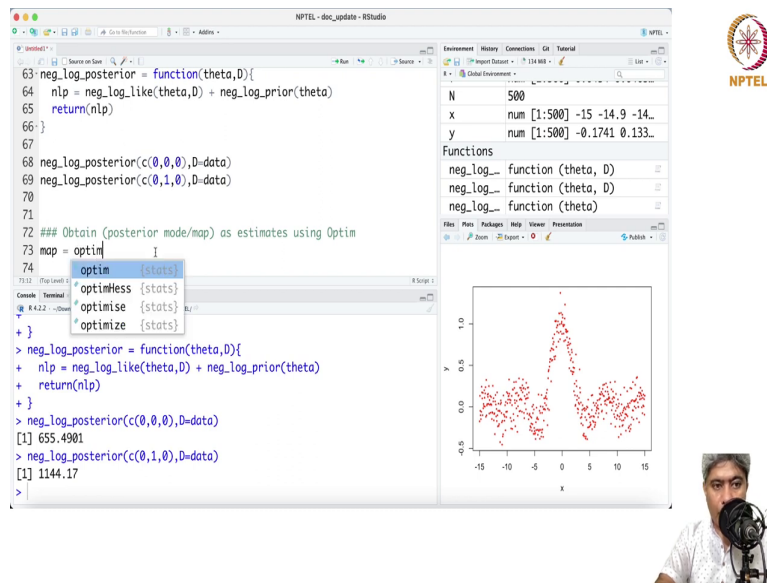
A scatter plot of the data points is shown in the bottom right corner, with x-axis ranging from -15 to 15 and y-axis ranging from -0.5 to 1.0. The plot shows a distribution of points with a peak around x=0 and y=1.0.

The NPTEL logo is visible in the top right corner of the slide.

And return nlp not natural language processing it is negative log posterior, ok. And if we run this negative log posterior. Let us check if it is working perfectly fine for at least these values. The initial values, ok. Yeah, it is working reasonably, ok. We can try one. And I think this is working perfectly fine, alright. Next, what I am thinking is that essentially what we need to do? We need to find.

So, what we will get that posterior mode Obtain the more posterior mode or Obtain estimates parameter estimates using Optim. These are typically posterior mode Optim Estimate using Optim or posterior mode. These are all posterior mode as estimate, ok; posterior mode as estimate.

(Refer Slide Time: 25:24)



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

```
63 neg_log_posterior = function(theta,D){
64   nlp = neg_log_like(theta,D) + neg_log_prior(theta)
65   return(nlp)
66 }
67
68 neg_log_posterior(c(0,0,0),D=data)
69 neg_log_posterior(c(0,1,0),D=data)
70
71
72 ## Obtain (posterior mode/map) as estimates using Optim
73 map = optim
74
75 optim {stats}
76 optimHess {stats}
77 optimise {stats}
78 optimize {stats}
```

The console window shows the execution of the code:

```
> neg_log_posterior = function(theta,D){
+   nlp = neg_log_like(theta,D) + neg_log_prior(theta)
+   return(nlp)
+ }
> neg_log_posterior(c(0,0,0),D=data)
[1] 655.4901
> neg_log_posterior(c(0,1,0),D=data)
[1] 1144.17
>
```



The Environment pane on the right shows the following data:

N	num	500
x	num	[1:500] -15 -14.9 -14...
y	num	[1:500] -0.1741 0.133...

The Functions pane lists the following functions:

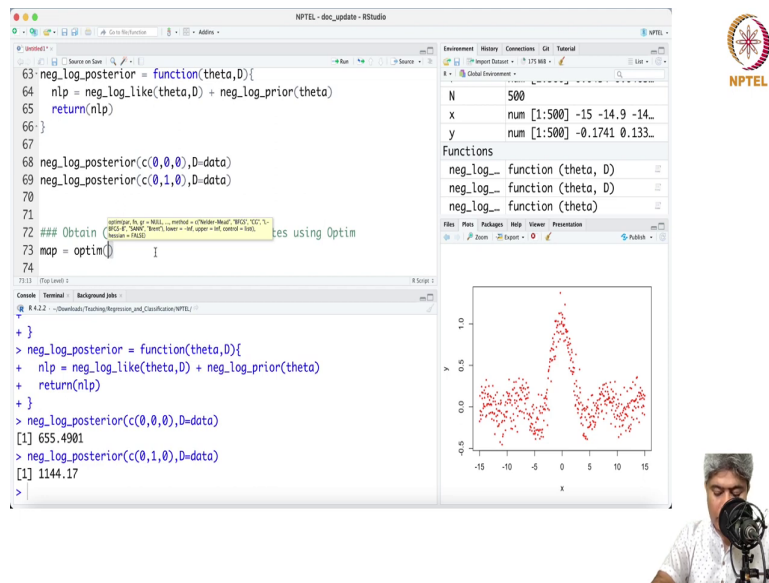
- neg_log_... function (theta, D)
- neg_log_... function (theta, D)
- neg_log_... function (theta)

The plot window shows a scatter plot of data points (x vs y) with a red density contour overlaid, indicating the posterior mode. The x-axis ranges from -15 to 15, and the y-axis ranges from -0.5 to 1.0. The density is highest around x=0 and y=0.5.



It is not mle. If you hand to mle, you have to optimize negative log likelihood function. But we are going to optimize the negative log posterior function. So, we will get posterior mode. We will obtain try to obtain both. And we will try to see what are the differences. So, first sometimes posterior mode also called map estimate. Maximum (Refer Time: 25:48) posterior estimate. So, map equal to optim, ok.

(Refer Slide Time: 25:58)



The screenshot displays the RStudio interface for a project named 'NPTEL - doc_update - RStudio'. The editor window contains the following R code:

```
63 neg_log_posterior = function(theta,D){
64   nlp = neg_log_like(theta,D) + neg_log_prior(theta)
65   return(nlp)
66 }
67
68 neg_log_posterior(c(0,0,0),D=data)
69 neg_log_posterior(c(0,1,0),D=data)
70
71
72 ## Obtain  $\theta$  using Optim
73 map = optim()
74
```

The console window shows the execution of the code:

```
> neg_log_posterior = function(theta,D){
+   nlp = neg_log_like(theta,D) + neg_log_prior(theta)
+   return(nlp)
+ }
> neg_log_posterior(c(0,0,0),D=data)
[1] 655.4901
> neg_log_posterior(c(0,1,0),D=data)
[1] 1144.17
>
```



The Environment pane on the right shows the following data:

N	500	
x	num [1:500]	-15 -14.9 -14.9
y	num [1:500]	-0.1741 0.133...

The Functions pane lists the defined functions:

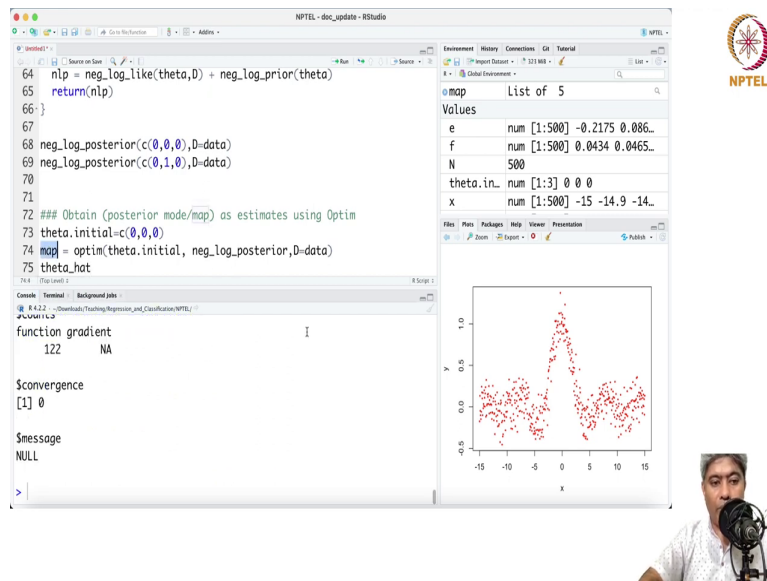
- neg_log_... function (theta, D)
- neg_log_... function (theta, D)
- neg_log_... function (theta)

A scatter plot of the data points is shown in the bottom right, with x-axis ranging from -15 to 15 and y-axis from -0.5 to 1.0. The data points are clustered around x=0 and y=0, with a notable peak at x=0, y=1.0.



And so, we have to have some initial value of theta.

(Refer Slide Time: 26:00)



The screenshot displays an RStudio interface with the following components:

- Source Editor:** Contains R code for a Bayesian model:

```
64 nlp = neg_log_like(theta,D) + neg_log_prior(theta)
65 return(nlp)
66 }
67
68 neg_log_posterior(c(0,0,0),D=data)
69 neg_log_posterior(c(0,1,0),D=data)
70
71
72 ## Obtain (posterior mode/map) as estimates using Optim
73 theta.initial=c(0,0,0)
74 map = optim(theta.initial, neg_log_posterior,D=data)
75 theta_hat
```
- Environment:** Shows a list of 5 objects with values:

Object	Class	Value
e	num	[1:500] -0.2175 0.086...
f	num	[1:500] 0.0434 0.0465...
N	num	500
theta.in_	num	[1:3] 0 0 0
x	num	[1:500] -15 -14.9 -14...
- Console:** Shows the output of the `optim` function:

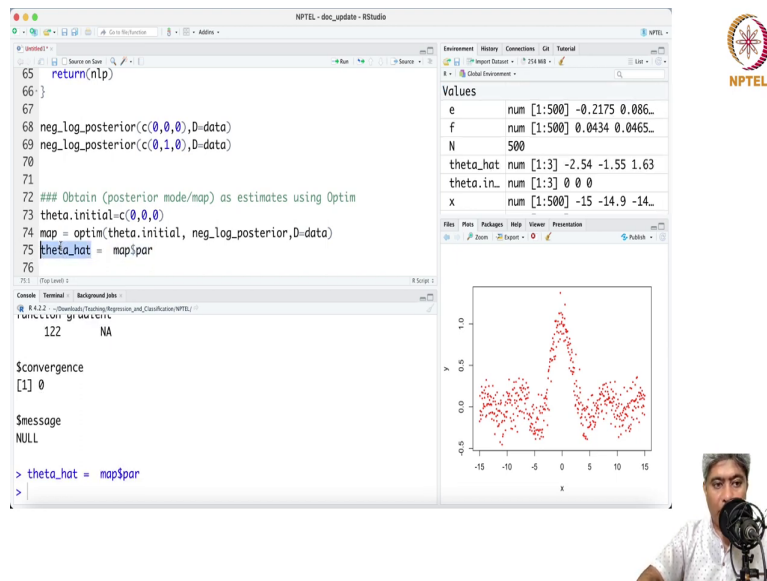
```
function gradient
 122 NA

$convergence
[1] 0

$message
NULL
```
- Plot:** A scatter plot of data points (x vs y) showing a distribution centered around x=0 and y=0. The x-axis ranges from -15 to 15, and the y-axis ranges from -0.5 to 1.0.

Let us take this initial value theta dot initial, ok. Initial value of theta. So, let me provide first the initial value of theta and then negative log posterior and then what we have to give D equal to data, ok. So, it takes few minutes maybe a few seconds, alright. And then from there theta hat we have to. So, if I just run the map.

(Refer Slide Time: 26:51)



The screenshot displays the RStudio interface. The script editor on the left contains the following R code:

```
65 return(nlp)
66 }
67
68 neg_log_posterior(c(0,0,0),D=data)
69 neg_log_posterior(c(0,1,0),D=data)
70
71
72 ## Obtain (posterior mode/map) as estimates using Optim
73 theta.initial=c(0,0,0)
74 map = optim(theta.initial, neg_log_posterior,D=data)
75 theta_hat = map$par
76
```



The console on the bottom left shows the output of the code:

```
122 NA
$convergence
[1] 0
$message
NULL
> theta_hat = map$par
>
```

The Environment pane on the right shows the following values:

Variable	Class	Value
e	num [1:500]	-0.2175 0.086...
f	num [1:500]	0.0434 0.0465...
N	num	500
theta_hat	num [1:3]	-2.54 -1.55 1.63
theta.in_	num [1:3]	0 0 0
x	num [1:500]	-15 -14.9 -14...

The plot on the right shows a scatter plot of data points (x vs y) with a red line representing the estimated path or boundary.



So, you can see. That it gives me three estimates. This is the value the at optimal value the negative log posterior in these estimates. And convergence equal to 0 means there is the convergence did happen and it has not find any problem with the convergence. The optimization did not had any problem. So, theta hat equal to from the map. Now, what I will do? I will just extract the parameter optimized parameters, ok. And then my theta hat these are the values and then what I am going to do sigma hat equal to e to the power theta hat 1.

(Refer Slide Time: 27:54)

The screenshot displays an RStudio interface with the following components:

- Source Editor:** Contains R code for a Bayesian model:

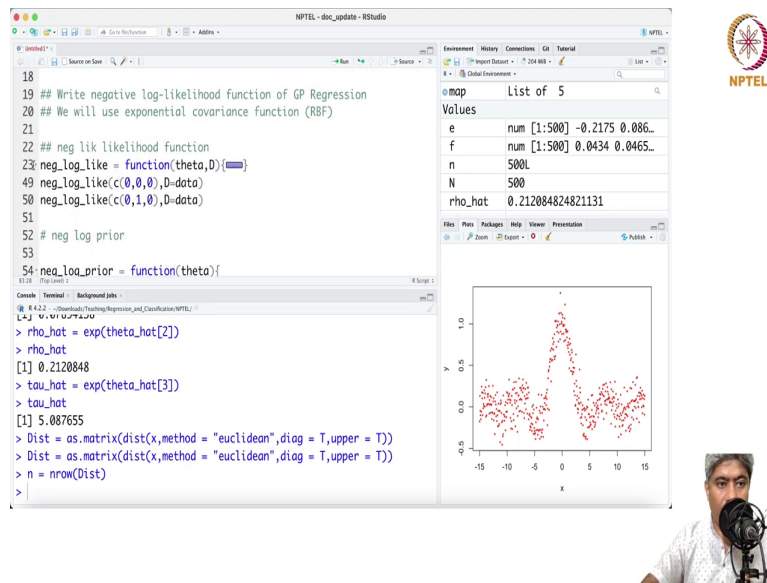
```
54: neg_log_prior = function(theta){  
55:   sigma = exp(theta[1])  
56:   rho = exp(theta[2])  
57:   tau = exp(theta[3])  
58:   p = -dgamma(sigma,1,1,log=T) - duni(rho,0,10,log=T) - dgamma(tau,1,1,log=T)  
59:   return(p)  
60: }  
61:  
62:  
63: neg_log_posterior = function(theta,D){  
64:   nlp = neg_log_like(theta,D) + neg_log_prior(theta)  
65:   return(nlp)  
66: }
```
- Environment:** Shows the following variables:

N	500
rho_hat	0.212084824821131
sigma_hat	0.0789413822427583
tau_hat	5.08765475422802
theta_hat	num [1:3] -2.54 -1.55 1.63
theta.in_	num [1:3] 0 0 0
x	num [1:500] -15 -14.9 -14.9
- Console:** Shows the execution of the following commands and their output:

```
> sigma_hat = exp(theta_hat[1])  
> sigma_hat  
[1] 0.07894138  
> rho_hat = exp(theta_hat[2])  
> rho_hat  
[1] 0.2120848  
> tau_hat = exp(theta_hat[3])  
> tau_hat  
[1] 5.087655  
>
```
- Plots:** A scatter plot of data points (x vs y) showing a distribution centered around x=0 and y=0. The x-axis ranges from -15 to 15, and the y-axis ranges from -0.5 to 1.0.

That is my sigma hat and then rho hat rho hat equal to e to the power theta hat 2. What is rho hat? Let me see so, 0.21 and then tau hat equal to e to the power theta hat 3. So, tau hat, ok. So, we got the estimates. Now, we have to have the distance. Remember that we calculated this distance matrix out of the box, ok. Out in the function we have not done it in the out in the environment. So, we have to just calculate that. So, now we have the distance.

(Refer Slide Time: 29:10)



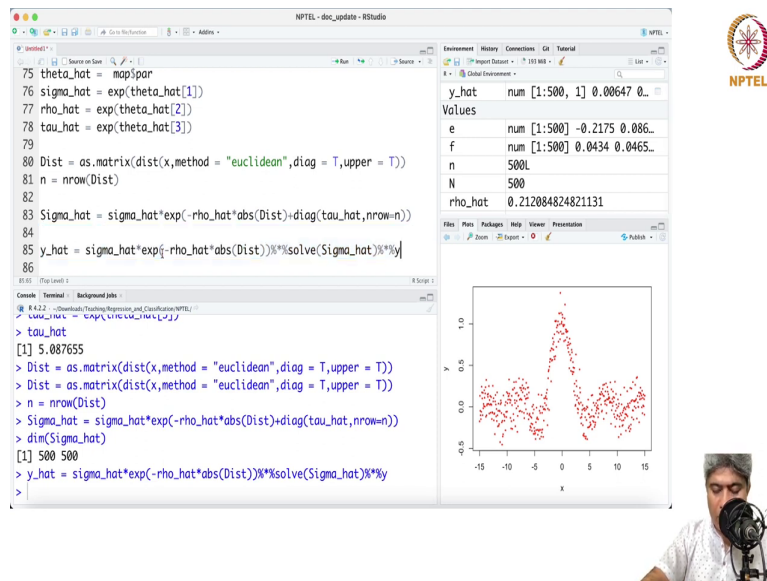
The screenshot displays an RStudio interface with the following components:

- Source Editor:** Contains R code for a Gaussian Process regression model. Lines 18-21 are comments. Lines 22-50 define the negative log-likelihood function `neg_log_lik`. Lines 51-53 are comments. Line 54 defines the negative log-prior function `neg_log_prior`.
- Environment:** Shows a list of 5 objects: `e`, `f`, `n`, `N`, and `rho_hat`.

Object	Class	Value
e	num	[1:500] -0.2175 0.086...
f	num	[1:500] 0.0434 0.0465...
n		500L
N		500
rho_hat		0.212084824821131
- Console:** Shows the execution of the R code. The output for `rho_hat` is `[1] 0.2120848` and for `tau_hat` is `[1] 5.087655`. It also shows the calculation of the distance matrix `Dist` and the number of rows `n`.
- Plots:** A scatter plot of data points (red dots) in a 2D space with axes labeled 'x' and 'y'. The x-axis ranges from -15 to 15, and the y-axis ranges from -0.5 to 1.0. The points form a shape resembling a stylized 'A' or a similar character.

n equal to nrow of the Distance and now first we have to calculate sigma, ok. Capital S Sigma hat equal to first sigma hat times exp. So, remember that we want this function to be here.

(Refer Slide Time: 30:00)



The screenshot displays the RStudio interface. The script editor on the left contains the following R code:

```
75 theta_hat = mapspar
76 sigma_hat = exp(theta_hat[1])
77 rho_hat = exp(theta_hat[2])
78 tau_hat = exp(theta_hat[3])
79
80 Dist = as.matrix(dist(x,method = "euclidean",diag = T,upper = T))
81 n = nrow(Dist)
82
83 Sigma_hat = sigma_hat*exp(-rho_hat*abs(Dist))+diag(tau_hat,nrow=n))
84
85 y_hat = sigma_hat*exp(-rho_hat*abs(Dist))%*%solve(Sigma_hat)%*%y
86
```

The console on the bottom left shows the execution of the code:

```
> tau_hat
[1] 5.087655
> Dist = as.matrix(dist(x,method = "euclidean",diag = T,upper = T))
> Dist = as.matrix(dist(x,method = "euclidean",diag = T,upper = T))
> n = nrow(Dist)
> Sigma_hat = sigma_hat*exp(-rho_hat*abs(Dist))+diag(tau_hat,nrow=n))
> dim(Sigma_hat)
[1] 500 500
> y_hat = sigma_hat*exp(-rho_hat*abs(Dist))%*%solve(Sigma_hat)%*%y
>
```

The Environment pane on the right shows the following values:

Variable	Value
y_hat	num [1:500, 1] 0.00647 0...
e	num [1:500] -0.2175 0.086...
f	num [1:500] 0.0434 0.0465...
n	500L
N	500
rho_hat	0.212084824821131

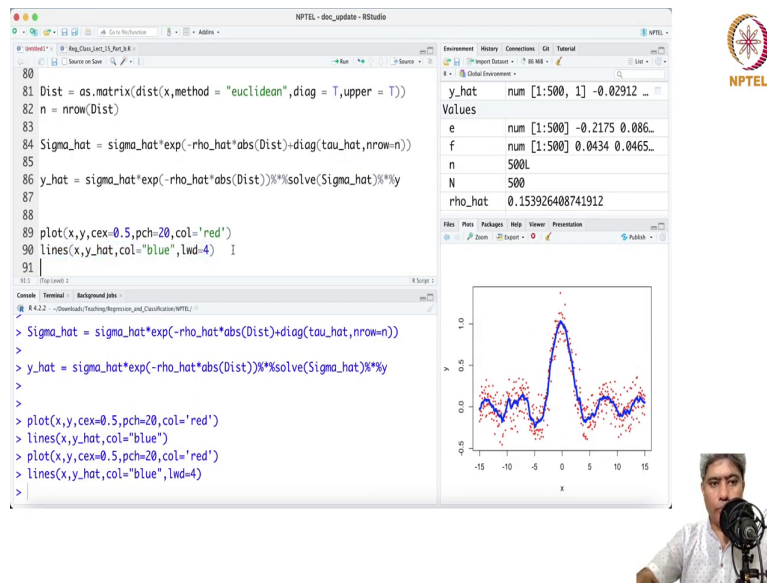
The plot on the bottom right shows a scatter plot of data points (red dots) in a 2D space with x and y axes ranging from -15 to 15. The points form a distinct shape, possibly a letter 'A' or a similar pattern.

The NPTEL logo is visible in the top right corner of the RStudio window.

So, Sigma will be replaced by the estimated Sigma hat rho will be rho hat and tau will be by tau hat. Now that gave us Sigma hat, ok. So, remember Sigma is a very large matrix going to be. This is a very large matrix of if you just say dimension of Sigma of Sigma hat is 500 by 500. Remember that.

And then y hat then will be y hat will be sigma hat. Times exp minus rho hat times distribution, ok. Percentage star percentage solve Sigma hat percentage star percentage y. So, if I just run this, this will be my y hat. And now what I am going to do on the plot. Let me just bring that plot that we have done here. Let me just bring that plot here.

(Refer Slide Time: 31:45)



The screenshot shows the RStudio interface. The script editor contains the following R code:

```
80
81 Dist = as.matrix(dist(x,method = "euclidean",diag = T,upper = T))
82 n = nrow(Dist)
83
84 Sigma_hat = sigma_hat*exp(-rho_hat*abs(Dist))+diag(tau_hat,nrow=n)
85
86 y_hat = sigma_hat*exp(-rho_hat*abs(Dist))%solve(Sigma_hat)%y
87
88
89 plot(x,y,cex=0.5,pch=20,col="red")
90 lines(x,y_hat,col="blue",lwd=4)
91
```



The console shows the execution of the code:

```
> Sigma_hat = sigma_hat*exp(-rho_hat*abs(Dist))+diag(tau_hat,nrow=n)
>
> y_hat = sigma_hat*exp(-rho_hat*abs(Dist))%solve(Sigma_hat)%y
>
> plot(x,y,cex=0.5,pch=20,col="red")
> lines(x,y_hat,col="blue")
> plot(x,y,cex=0.5,pch=20,col="red")
> lines(x,y_hat,col="blue",lwd=4)
>
```

The Environment pane shows the following variables:

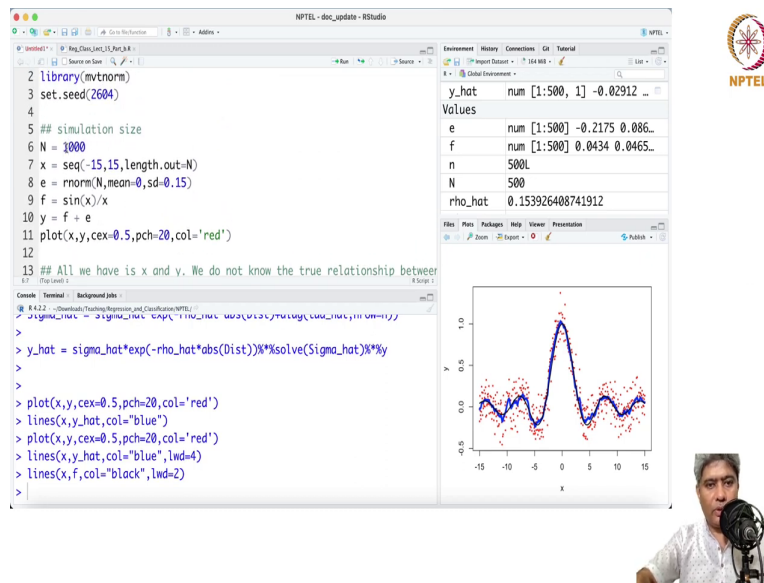
Variable	Class	Value
y_hat	num	[1:500, 1] -0.02912 ...
Values		
e	num	[1:500] -0.2175 0.086...
f	num	[1:500] 0.0434 0.0465...
n		500L
N		500
rho_hat		0.153926408741912

The plot shows a scatter plot of red points with a blue line representing the fitted curve. The x-axis ranges from -15 to 15, and the y-axis ranges from -0.5 to 1.0. The blue line is significantly thicker than the default.





Now, I am going to write lines x comma y, x comma y hat. Now, I am going to my x color y hat color equal to blue maybe and l width equal to be 4, ok. So, if you just run it, yeah. So, you can see, let me just. So, you can see that it is bit jittery, but it is actually picking up to an extent. Let me plot the original curve here. Let me plot the original curve here, ok. Say lines equals to x comma y f; f is the original curve, I think.

(Refer Slide Time: 32:54)




```
2 library(mvtnorm)
3 set.seed(2604)
4
5 ## simulation size
6 N = 3000
7 x = seq(-15,15,length.out=N)
8 e = rnorm(N,mean=0,sd=0.15)
9 f = sin(x)/x
10 y = f + e
11 plot(x,y,cex=0.5,pch=20,col='red')
12
13 ## All we have is x and y. We do not know the true relationship between
14
15 > y_hat = sigma_hat*exp(-rho_hat*abs(Dist))%*%solve(Sigma_hat)%*%y
16
17 > plot(x,y,cex=0.5,pch=20,col='red')
18 > lines(x,y_hat,col='blue')
19 > plot(x,y,cex=0.5,pch=20,col='red')
20 > lines(x,y_hat,col='blue',lwd=4)
21 > lines(x,f,col='black',lwd=2)
22 >
```

Variable	Value
y_hat	num [1:500, 1] -0.02912 ...
e	num [1:500] -0.2175 0.086...
f	num [1:500] 0.0434 0.0465...
n	500L
N	500
rho_hat	0.153926408741912



And we can take some other color. Color equal to maybe some other color may be black, ok. And line width equal to 2. So, if you now, let me just you can see this black color is the true curve. And this, this blue color is the estimated \hat{f} and that \hat{f} is essentially very close. Now, what I am going to do, we will see that as the sample simulation size increases. So, instead of 5000, if I just increase the simulation size to 1000, that will make the curve much more. We will see how the curve will behave.

(Refer Slide Time: 34:06)



The screenshot displays the RStudio interface. The script editor on the left contains the following R code:

```
2 library(mvtnorm)
3 set.seed(2604)
4
5 ## simulation size
6 N = 1000
7 x = seq(-15,15,length.out=N)
8 e = rnorm(N,mean=0,sd=0.15)
9 f = sin(x)/x
10 y = f + e
11 plot(x,y,cex=0.5,pch=20,col='red')
12
13 ## All we have is x and y. We do not know the true relationship between
```

The console on the bottom left shows the execution of the following commands and their outputs:

```
> neg_log_posterior(c(0,0,0),D=data)
[1] 1279.225
> neg_log_posterior(c(0,1,0),D=data)
[1] 1319.844
>
> ## Obtain (posterior mode/map) as estimates using Optim
> theta.initial=c(0,0,0)
> map = optim(theta.initial, neg_log_posterior,D=data)
```

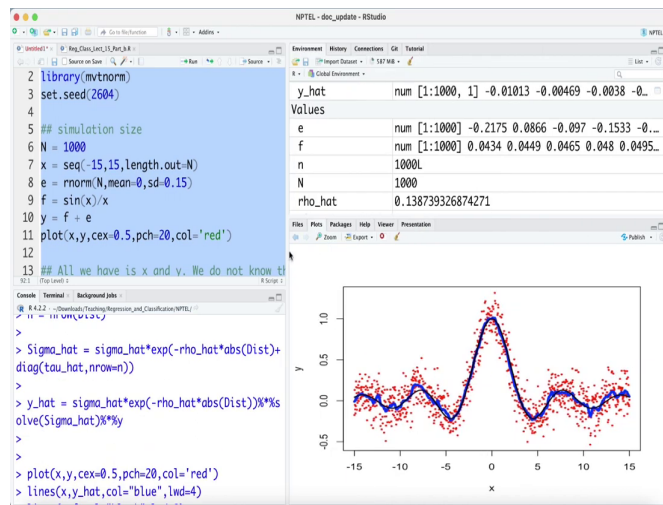
The environment pane on the right shows the following values:

Variable	Value
y_hat	num [1:500, 1] -0.02912 ...
e	num [1:500] -0.2175 0.086...
f	num [1:500] 0.0434 0.0465...
n	500L
N	500
rho_hat	0.153926408741912

The plot window on the right shows a scatter plot of red points (pch=20) with x-axis ranging from -15 to 15 and y-axis ranging from -0.5 to 1.0. The points form a noisy pattern that roughly follows a sine wave. The NPTEL logo is visible in the top right corner, and a small video feed of the presenter is in the bottom right corner.

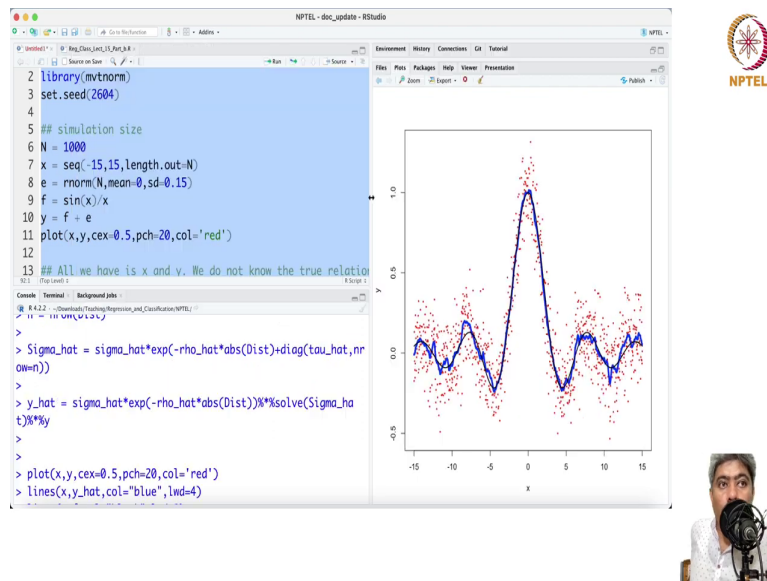
If we have 5000 samples so it will take a little bit time because we have more samples now. So, it might take about a minute so, if you.

(Refer Slide Time: 34:48)



Let me just increase it. So, this is with the 1000 samples we can see. And if this is with the 10000 samples, 500 samples, this is with the 500 samples. And this curve is with the 1000 samples.

(Refer Slide Time: 35:14)



So, almost the 1000 sample, probably a little bit closer. So, the GP regression has no clue what is the true original relationship between x and y . And yet the GP regression actually picking up the actual relationship between x and y . So, this is a very strong and very useful methodology if you want to do non-linear regression.

Now, you can ask me. If you have only one x and then we know how to do that, can I do it for more than one x , if I have, you know, multiple x . The answer is yes, you can do exactly the same way. Only thing is you have to calculate the distance, but we know how to calculate the distance.

And here we have used Euclidean distance because it was a simple toy problem. Sometimes you may have to be bit careful about what kind of distance function you want to compute. For example, if you are using a spatial data, you may have to use earth distance. But if you are

using say a categorical data, then you may have to use appropriate distance like Gower distance, which do calculate the distance between the categorical variables.

So, you have to be bit careful about what kind of distance functions that you want to use. But if you are sure this is the correct distance function that you want to use. Then perhaps Gaussian process method or Gaussian process regression is one of the best methodology that you can think of. So, with that, I will stop here in this video lecture. Let us see you in the next video.