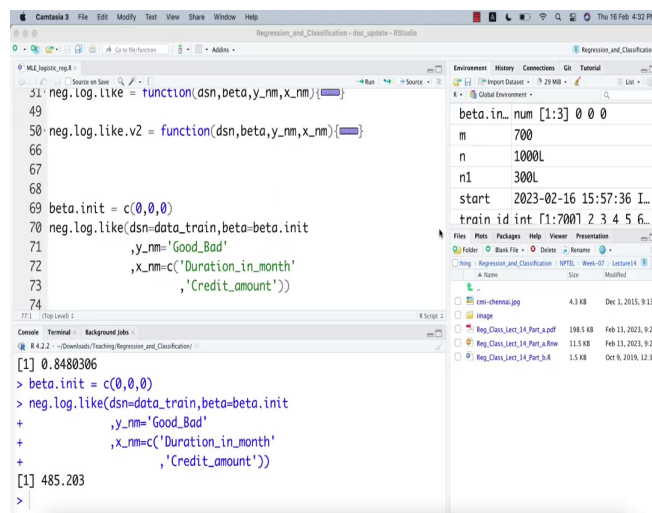


Predictive Analytics - Regression and Classification
Prof. Sourish Das
Department of Mathematics
Chennai Mathematical Institute

Lecture - 39
Hands on with R: Measure Time performance of R code

Hello all, welcome back to the part D of the lecture 11. I am going to do a small demo on time complexity of R and of the code that we wrote last in last video.

(Refer Slide Time: 00:32)



```
51 neg.log.like = function(dsn, beta, y_nm, x_nm){
49
50 neg.log.like.v2 = function(dsn, beta, y_nm, x_nm){
66
67
68
69 beta.init = c(0,0,0)
70 neg.log.like(dsn=data_train, beta=beta.init
71             , y_nm='Good_Bad'
72             , x_nm=c('Duration_in_month'
73                   , 'Credit_amount'))
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
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
Environment: Global Environment
beta.in_ num [1:3] 0 0 0
m          700
n          1000L
n1         300L
start      2023-02-16 15:57:36 I...
train id int [1:700] 2 3 4 5 6 ...
```

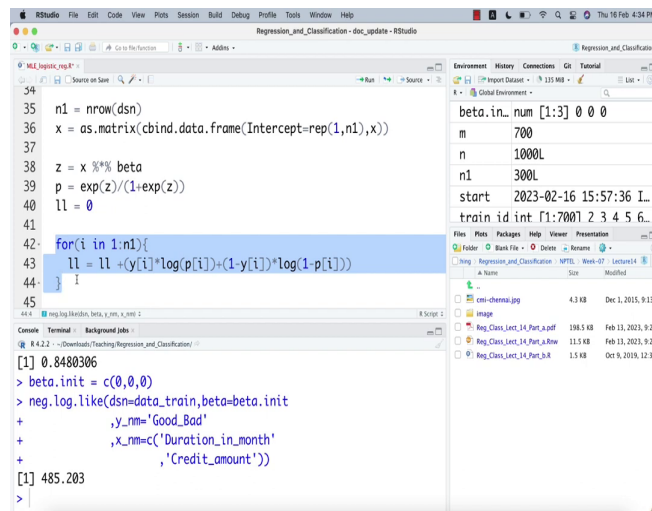
```
Files: Packages Help Viewer Presentation
Folder Blank File Delete Rename
~/reg Regression_and_Classification NPTEL Week-07 Lecture14
  A Name Size Modified
csi-chemical.jpg 4.3 kB Dec 1, 2015, 9:13 A
image
Reg_Class_Lect_14_Part_a.pdf 198.5 kB Feb 13, 2023, 9:26
Reg_Class_Lect_14_Part_a_Row 11.5 kB Feb 13, 2023, 9:26
Reg_Class_Lect_14_Part_b.R 1.5 kB Oct 9, 2019, 12:39 I
```

```
Console Terminal Background jobs
R 4.2.2 - (Downloads/Teaching/Regression_and_Classification)
[1] 0.8480306
> beta.init = c(0,0,0)
> neg.log.like(dsn=data_train, beta=beta.init
+             , y_nm='Good_Bad'
+             , x_nm=c('Duration_in_month'
+                   , 'Credit_amount'))
[1] 485.203
>
```



So, this is if you are from Computer Science or Engineering background and if you have taken a course on algorithms, then you understand this issue of time complexity that if you write a code how much time it takes to complete the code. That is the typical time complexity. And if you see I want to show you a demo here.

(Refer Slide Time: 01:12)



```
34
35 n1 = nrow(dsn)
36 x = as.matrix(cbind.data.frame(Intercept=rep(1,n1),x))
37
38 z = x %*% beta
39 p = exp(z)/(1+exp(z))
40 ll = 0
41
42 for(i in 1:n1){
43   ll = ll +(y[i]*log(p[i])+(1-y[i])*log(1-p[i]))
44 }
45
```

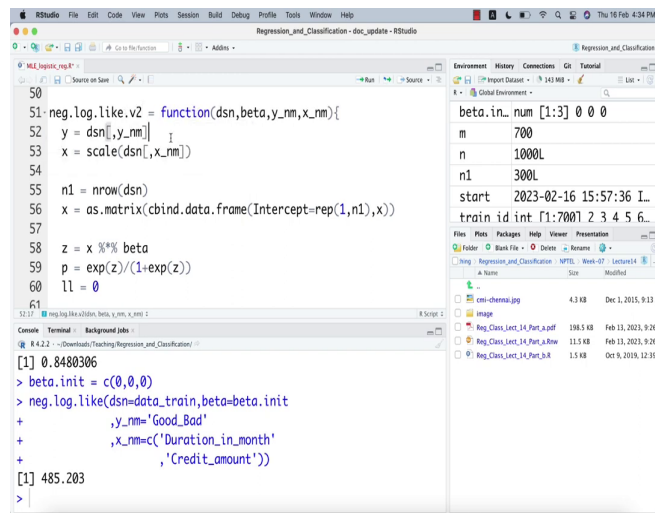
```
[1] 0.8480306
> beta.init = c(0,0,0)
> neg.log.like(dsn=data_train,beta=beta.init
+             ,y_nm='Good_Bad'
+             ,x_nm=c('Duration_in_month'
+                   , 'Credit_amount'))
[1] 485.203
>
```



So, you see if you have this is the negative log likelihood, I function that I wrote and here I wrote a for loop to compute the log likelihood value. Now, what happens is when in my previous video you have heard that I am saying that if you write for loop in R then it might be not efficient way to write it, instead we should write it in a vector form. What happens is R is a high level language and its compiler is in C and the compiler creates the machine language.

So, the whole thing gets in computed in C language, but as a result what happens if you running. So, it is R is often called R both R in Python are often called interpretable language. Means, if you write a for loop every time the loop is running it is going back to C doing the computation coming back. So, that is where the for loop is. Now, if you say run let me run copy this function and I will show you this could be actually a painful it could slow your computation to good extent.

(Refer Slide Time: 02:30)

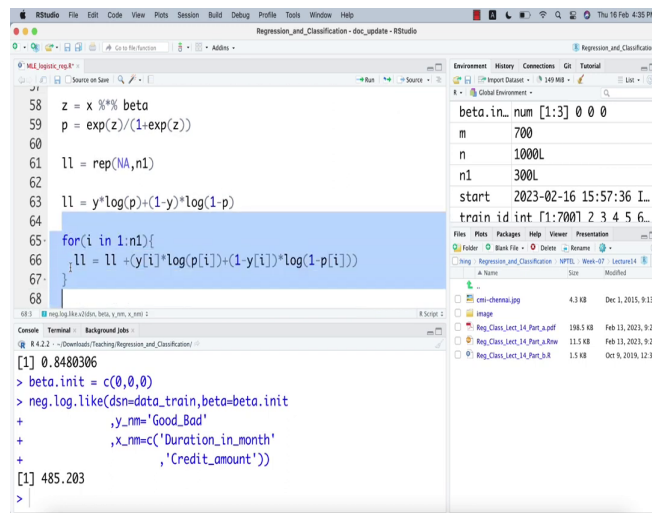


The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for a function `neg.log.like.v2` and its execution. The code defines a function that takes `dsn`, `beta`, `y_nm`, and `x_nm` as arguments. It scales the data, creates a matrix, and calculates the log-likelihood. The execution shows the function returning `[1] 0.8480306`.
- Environment:** Shows the current environment with variables: `beta.in_ num [1:3] 0 0 0`, `m 700`, `n 1000L`, `n1 300L`, `start 2023-02-16 15:57:36 I...`, and `train id int [1:700] 2 3 4 5 6...`.
- Files:** Lists files in the current directory, including `csi-chemai.jpg`, `image`, `Reg_Class_Lect_14_Part_a.pdf`, `Reg_Class_Lect_14_Part_a.Rnw`, and `Reg_Class_Lect_14_Part_b.R`.
- Console:** Shows the execution of the function and the assignment of `beta.init = c(0,0,0)`. The final output is `[1] 485.203`.



(Refer Slide Time: 02:41)



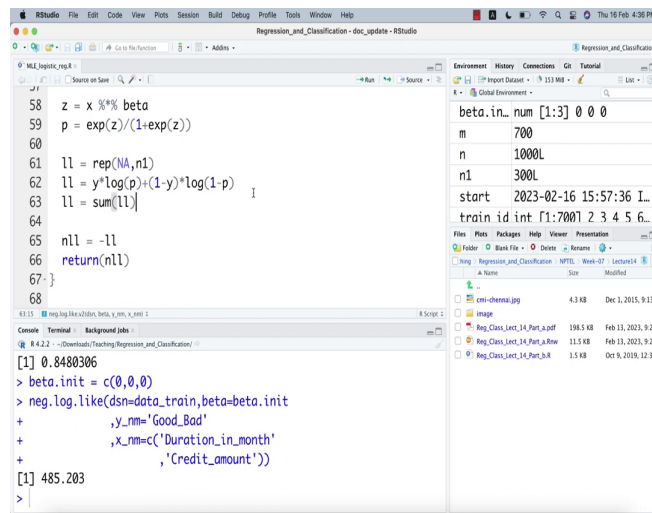
```
58 z = x %*% beta
59 p = exp(z)/(1+exp(z))
60
61 ll = rep(NA,n1)
62
63 ll = y*log(p)+(1-y)*log(1-p)
64
65 for(i in 1:n1){
66   ll = ll +(y[i]*log(p[i])+(1-y[i])*log(1-p[i]))
67 }
68
```

```
[1] 0.8480306
> beta.init = c(0,0,0)
> neg.log.like(dsn=data_train,beta=beta.init
+             ,y_nm='Good_Bad'
+             ,x_nm=c('Duration_in_month'
+                   , 'Credit_amount'))
[1] 485.203
>
```



Let me write a v2 version and v2 version will be instead of. So, basically what I want is instead of ll 0 I will just say replicate NA comma n1, n1 is the yeah and then ll equals to y times log p plus 1 minus y; 1 minus y times log 1 minus p and I do not need this piece of line and then I need to do ll equal to just some ll.

(Refer Slide Time: 03:24)



```
58 z = X %*% beta
59 p = exp(z)/(1+exp(z))
60
61 ll = rep(NA,n1)
62 ll = y*log(p)+(1-y)*log(1-p)
63 ll = sum(ll)
64
65 nll = -ll
66 return(nll)
67 }
68
```

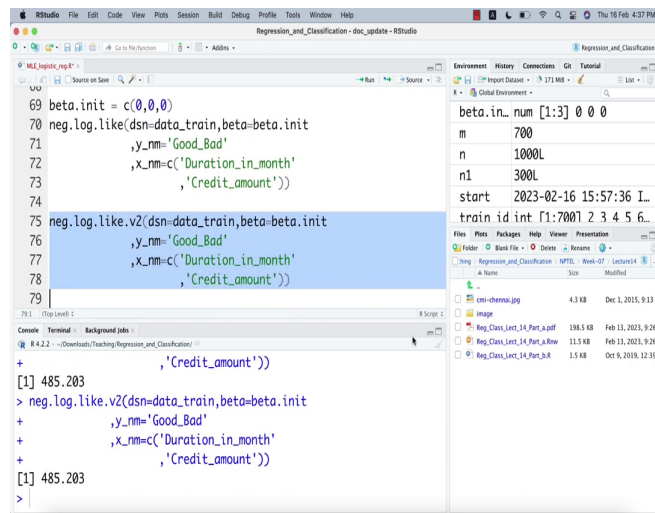
```
[1] 0.8480306
> beta.init = c(0,0,0)
> neg.log.like(dsn=data_train,beta=beta.init
+             ,y_rm='Good_Bad'
+             ,x_rm=c('Duration_in_month'
+                   , 'Credit_amount'))
[1] 485.203
>
```



That is if I write this if I just replace that two line with for loop with this vectorize this is called vectorization of code, ok. So, you avoid the. So, I am just doing vector operation here. So, if I do this then it suppose to work much better, it is supposed to do quick job. So, now I have in the negative log likelihood is the way I wrote the function previously and negative log likely log like dot v2 is the next one which I vectorized.

So, so what I am going to do or let me just run this code both of them and this is what I am getting from the older version.

(Refer Slide Time: 04:24)



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

```
69 beta.init = c(0,0,0)
70 neg.log.like(dsn=data_train,beta=beta.init
71             ,y_nm='Good_Bad'
72             ,x_nm=c('Duration_in_month'
73                   , 'Credit_amount'))
74
75 neg.log.like.v2(dsn=data_train,beta=beta.init
76                ,y_nm='Good_Bad'
77                ,x_nm=c('Duration_in_month'
78                      , 'Credit_amount'))
79
```

The console shows the execution of the code:

```
[1] 485.203
> neg.log.like.v2(dsn=data_train,beta=beta.init
+                ,y_nm='Good_Bad'
+                ,x_nm=c('Duration_in_month'
+                      , 'Credit_amount'))
[1] 485.203
>
```

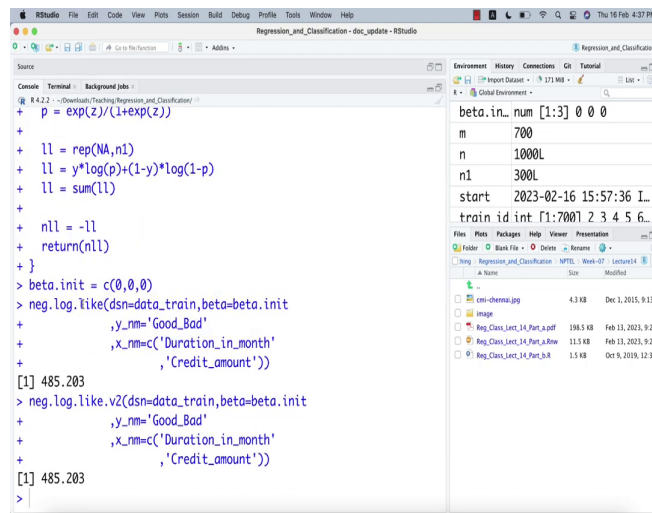
The Environment pane on the right shows the following variables:

Variable	Value
beta.in_ num	[1:3] 0 0 0
m	700
n	1000L
n1	300L
start	2023-02-16 15:57:36 I...
train id int	[1:700] 2 3 4 5 6

The Files pane shows a folder named 'Regression_and_Classification' containing several files, including 'cni-chennai.jpg', 'image', and three PDF files related to 'Reg.Class.Lect.14_Part_a'.



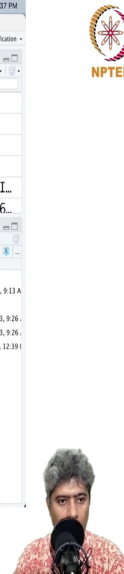
(Refer Slide Time: 04:40)



```
Source  
+ p = exp(z)/(1+exp(z))  
+  
+ ll = rep(NA,n1)  
+ ll = y*log(p)+(1-y)*log(1-p)  
+ ll = sum(ll)  
+  
+ nll = -ll  
+ return(nll)  
+ }  
> beta.init = c(0,0,0)  
> neg.log.like(dsn=data_train,beta=beta.init  
+ ,y_nm='Good_Bad'  
+ ,x_nm=c('Duration_in_month'  
+ , 'Credit_amount'))  
[1] 485.203  
> neg.log.like.v2(dsn=data_train,beta=beta.init  
+ ,y_nm='Good_Bad'  
+ ,x_nm=c('Duration_in_month'  
+ , 'Credit_amount'))  
[1] 485.203  
>
```

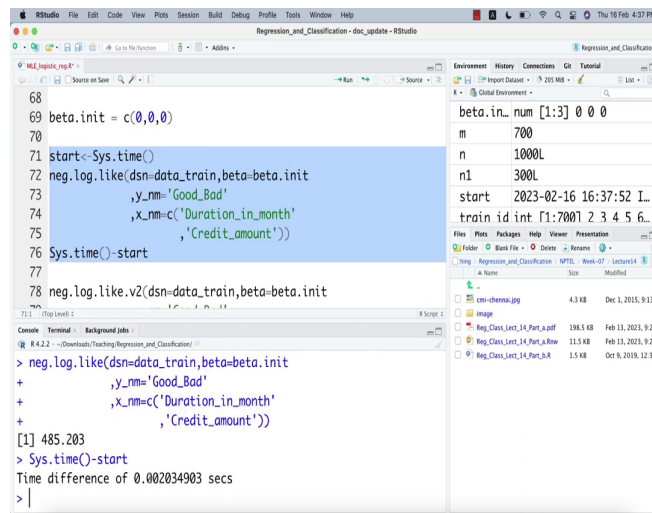
Environment: History Connections Git Tutorial
beta.in... num [1:3] 0 0 0
m 700
n 1000L
n1 300L
start 2023-02-16 15:57:36 I...
train id int [1:700] 2 3 4 5 6

Files Plots Packages Help Viewer Presentation
A Name Size Modified
cni-chemai.jpg 4.3 kB Dec 1, 2015, 9:33 A
image
Reg_Class_Lect_14_Part_a.pdf 198.5 kB Feb 13, 2023, 9:26
Reg_Class_Lect_14_Part_a_Row 11.5 kB Feb 13, 2023, 9:26
Reg_Class_Lect_14_Part_b.R 15.9 kB Oct 9, 2019, 12:39 P



Now, first I am if I it suppose to give me the exact same value for my newer version also the v2 1 dot v2 as well, right. Yeah, you see here I called the older version which give me 485.203 and this is my newer version which is also give me the exact same value. So, it returns the exact same value. But how much time it takes?

(Refer Slide Time: 04:57)



```
68
69 beta.init = c(0,0,0)
70
71 start<-Sys.time()
72 neg.log.like(dsn-data_train,beta=beta.init
73             ,y_nm='Good_Bad'
74             ,x_nm=c('Duration_in_month'
75                   , 'Credit_amount'))
76 Sys.time()-start
77
78 neg.log.like.v2(dsn-data_train,beta=beta.init
```

Environment: History Connections GR Tutorial
Global Environment

beta.in_ num [1:3]	0 0 0
m	700
n	1000L
n1	300L
start	2023-02-16 16:37:52 I...
train id int [1:700]	2 3 4 5 6

Files Plots Packages Help Viewer Presentation

A Name	Size	Modified
img	Regression_and_Classification	Week-07 - Lecture14
img	Blank File	Delete Rename
img	csi-chennai.jpg	4.3 KB Dec 1, 2015, 9:33 A
img	image	
img	Reg_Class_Lect_14_Part_a.pdf	198.5 KB Feb 13, 2023, 9:26
img	Reg_Class_Lect_14_Part_a_Row	11.5 KB Feb 13, 2023, 9:26
img	Reg_Class_Lect_14_Part_b-R	15.9 KB Oct 9, 2019, 12:39 I

Console Terminal Background jobs

```
> neg.log.like(dsn-data_train,beta=beta.init
+             ,y_nm='Good_Bad'
+             ,x_nm=c('Duration_in_month'
+                   , 'Credit_amount'))
[1] 485.203
> Sys.time()-start
Time difference of 0.002034903 secs
> |
```



Let me just do that start Sys dot time and Sys dot time minus start s t a r t, alright. So, if I run the older version. So, it is taking 0.00203 and similarly if I taking if I write this one of the (Refer Time: 05:40) sorry, but that s t a r t. So, this is the one I am going to do and then yeah.

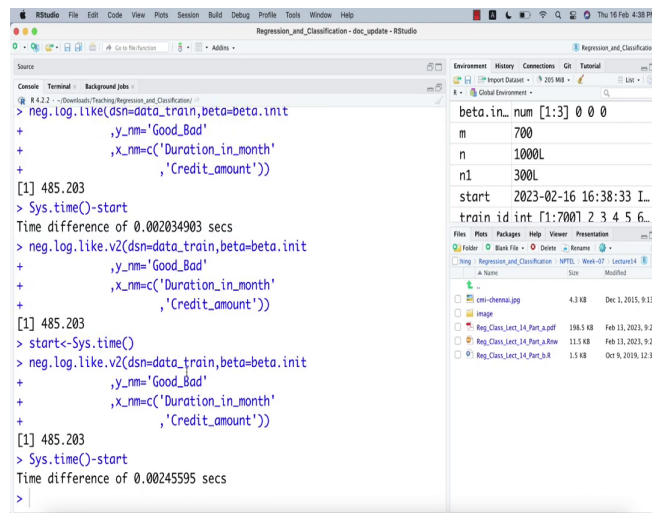
(Refer Slide Time: 06:01)

```
75     , 'Credit_amount'))
76 Sys.time()-start
77
78 start<-Sys.time()
79 neg.log.like.v2(dsn=data_train,beta=beta.init
80   ,y_nm='Good_Bad'
81   ,x_nm=c('Duration_in_month'
82   , 'Credit_amount'))
83 Sys.time()-start
84
85 ## call optimisation to estimate MLE of beta
```

```
> Sys.time()-start
Time difference of 0.002034903 secs
> neg.log.like.v2(dsn=data_train,beta=beta.init
+   ,y_nm='Good_Bad'
+   ,x_nm=c('Duration_in_month'
+   , 'Credit_amount'))
[1] 485.203
>
```



(Refer Slide Time: 06:13)



```
RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
Regression_and_Classification - doc_update - RStudio

Source
Console Terminal Background Jobs
R 4.2.2 ~\Downloads\Tracking\Regression_and_Classification\
> neg.log.like(dsn=data_train,beta=beta.init
+ ,y_nm='Good_Bad'
+ ,x_nm=c('Duration_in_month'
+ , 'Credit_amount'))
[1] 485.203
> Sys.time()-start
Time difference of 0.002034903 secs
> neg.log.like.v2(dsn=data_train,beta=beta.init
+ ,y_nm='Good_Bad'
+ ,x_nm=c('Duration_in_month'
+ , 'Credit_amount'))
[1] 485.203
> start<-Sys.time()
> neg.log.like.v2(dsn=data_train,beta=beta.init
+ ,y_nm='Good_Bad'
+ ,x_nm=c('Duration_in_month'
+ , 'Credit_amount'))
[1] 485.203
> Sys.time()-start
Time difference of 0.00245595 secs
>
```

Environment History Connections GR Tutorial

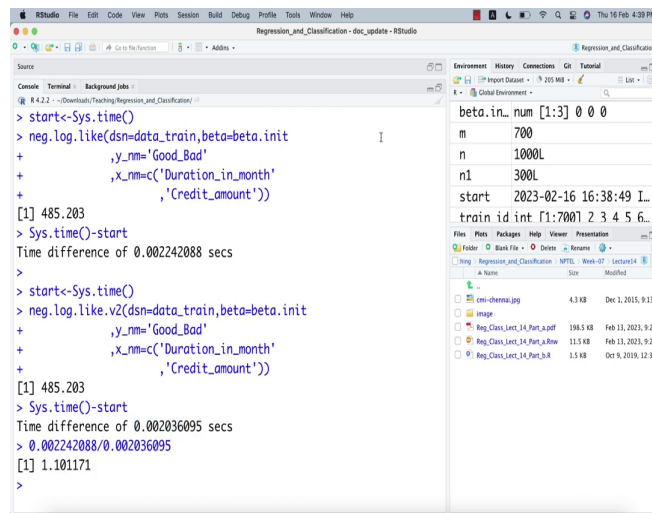
beta.in_ num [1:3]	0 0 0
m	700
n	1000L
n1	300L
start	2023-02-16 16:38:33 I...
train id int [1:700]	2 3 4 5 6...

Files Plots Packages Help Viewer Presentation

A Name	Size	Modified
cmi-chennai.jpg	4.3 KB	Dec 1, 2015, 9:33 A
image		
Reg_Class_Lect_14_Part_a.pdf	198.5 KB	Feb 13, 2023, 9:26
Reg_Class_Lect_14_Part_a.Rnw	11.5 KB	Feb 13, 2023, 9:26
Reg_Class_Lect_14_Part_b.R	15.9 KB	Oct 9, 2019, 12:39 P



(Refer Slide Time: 06:31)



```
> start<-Sys.time()
> neg.log.like(dsn=data_train,beta=beta.init
+             ,y_nm='Good_Bad'
+             ,x_nm=c('Duration_in_month'
+                   , 'Credit_amount'))
[1] 485.203
> Sys.time()-start
Time difference of 0.002242088 secs
>
> start<-Sys.time()
> neg.log.like.v2(dsn=data_train,beta=beta.init
+                ,y_nm='Good_Bad'
+                ,x_nm=c('Duration_in_month'
+                      , 'Credit_amount'))
[1] 485.203
> Sys.time()-start
Time difference of 0.002036095 secs
> 0.002242088/0.002036095
[1] 1.101171
>
```

Environment: History Connections GR Tutorial

beta.in._num [1:3]	0 0 0
m	700
n	1000L
n1	300L
start	2023-02-16 16:38:49 I...
train id int [1:700]	2 3 4 5 6

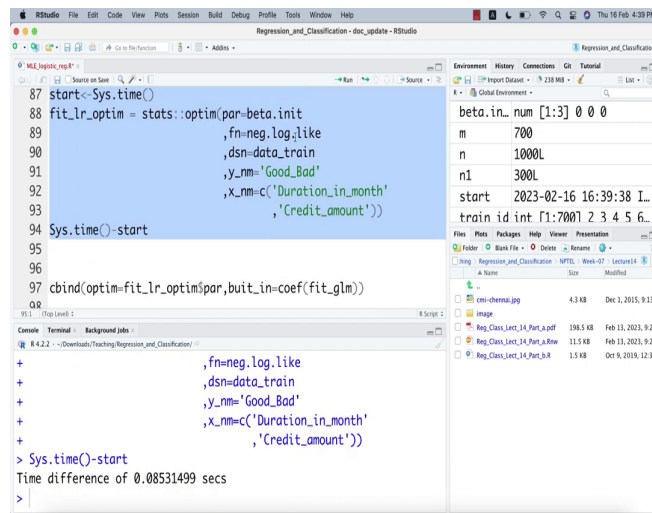
Files: Plots: Packages: Help: Viewer: Presentation

A Name	Size	Modified
cmi-chemai.jpg	4.3 KB	Dec 1, 2015, 9:33 A
image		
Reg_Class_Lect_14_Part_a.pdf	18.5 KB	Feb 13, 2023, 9:26
Reg_Class_Lect_14_Part_a.Rnw	11.5 KB	Feb 13, 2023, 9:26
Reg_Class_Lect_14_Part_b.R	1.5 KB	Oct 9, 2019, 12:39 I



So, if I run this. So, let me just run it once more. So, the first one alright let me see how much time it is taking. So, this is the old one and this is the new one. So, this is the old one divided by new one. So, it is looks like the new one is slightly faster. So, what I will do and same way what we can do is we can do the optimization will do.

(Refer Slide Time: 06:55)



```
87 start<-Sys.time()
88 fit_lr_optim = stats::optim(par=beta.init
89   ,fn=neg.log.like
90   ,dsn=data_train
91   ,y_nm='Good_Bad'
92   ,x_nm=c('Duration_in_month'
93   , 'credit_amount'))
94 Sys.time()-start
95
96
97 cbind(optim=fit_lr_optim$par,buit_in=coef(fit_glm))
98
99
```

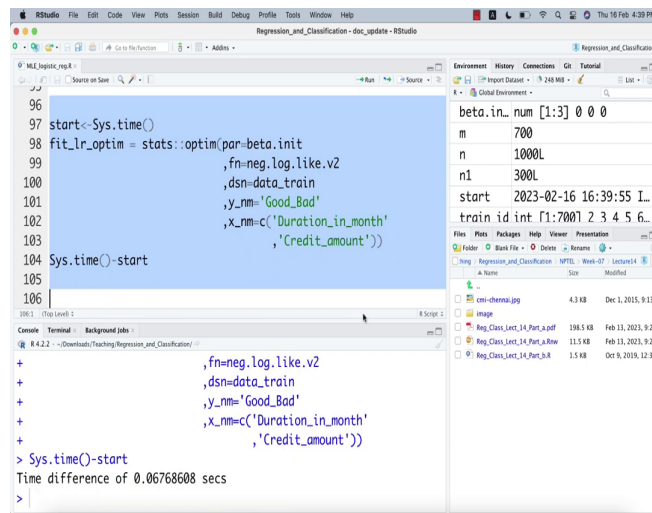
```
+ ,fn=neg.log.like
+ ,dsn=data_train
+ ,y_nm='Good_Bad'
+ ,x_nm=c('Duration_in_month'
+ , 'credit_amount'))
> Sys.time()-start
Time difference of 0.08531499 secs
>
```

beta.in_ num [1:3]	0 0 0
m	700
n	1000L
n1	300L
start	2023-02-16 16:39:38 I...
train id int [1:700]	2 3 4 5 6



So, let me just do the optimization with the old version and the new version. So, an optimization will have to do some computation additional computation. So, if I just run this with. So, it takes about 0.8 08 second with the older version.

(Refer Slide Time: 07:22)



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

```
96  
97 start<-Sys.time()  
98 fit_lr_optim = stats::optim(par=beta.init  
99 ,fn=neg.log.like.v2  
100 ,dsn=data_train  
101 ,y_nm='Good_Bad'  
102 ,x_nm=c('Duration_in_month'  
103 , 'Credit_amount'))  
104 Sys.time()-start  
105  
106
```

The console output shows the execution of the code and the time taken:

```
106.1 | Top Level | R Script |  
+ ,fn=neg.log.like.v2  
+ ,dsn=data_train  
+ ,y_nm='Good_Bad'  
+ ,x_nm=c('Duration_in_month'  
+ , 'Credit_amount'))  
> Sys.time()-start  
Time difference of 0.06768608 secs  
>
```

The Environment pane on the right shows the following variables:

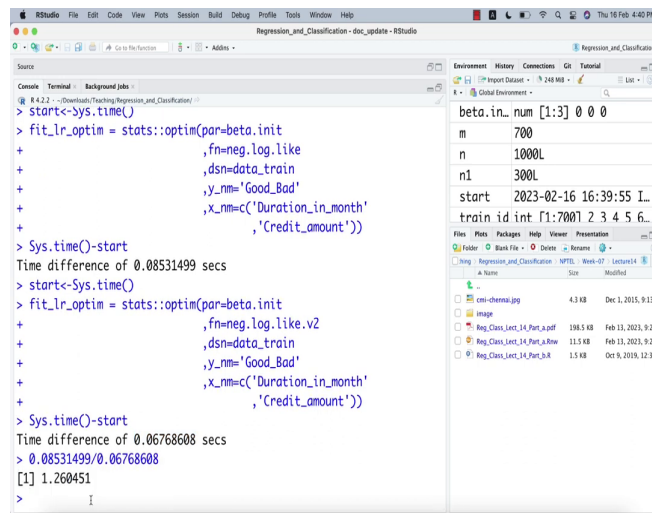
Variable	Value
beta.in_	num [1:3] 0 0 0
m	700
n	1000L
n1	300L
start	2023-02-16 16:39:55 I...
train id	int [1:700] 2 3 4 5 6...

The Files pane shows the following files:

Name	Size	Modified
cmi-chemai.jpg	4.3 KB	Dec 1, 2015, 9:33 A
image		
Reg_Class_Lect_14_Part_a.pdf	198.5 KB	Feb 13, 2023, 9:26
Reg_Class_Lect_14_Part_a.Rnw	11.5 KB	Feb 13, 2023, 9:26
Reg_Class_Lect_14_Part_b.R	15.9 KB	Oct 9, 2019, 12:39 I



(Refer Slide Time: 07:34)



```
R 4.2.2 -- Downloads/Teaching/Regression_and_Classification/
> start<-Sys.time()
> fit_lr_optim = stats::optim(par=beta.init
+                               ,fn=neg.log.like
+                               ,dsn=data_train
+                               ,y_nm='Good_Bad'
+                               ,x_nm=c('Duration_in_month'
+                                       , 'Credit_amount'))
> Sys.time()-start
Time difference of 0.08531499 secs
> start<-Sys.time()
> fit_lr_optim = stats::optim(par=beta.init
+                               ,fn=neg.log.like.v2
+                               ,dsn=data_train
+                               ,y_nm='Good_Bad'
+                               ,x_nm=c('Duration_in_month'
+                                       , 'Credit_amount'))
> Sys.time()-start
Time difference of 0.06768608 secs
> 0.08531499/0.06768608
[1] 1.260451
>
```



And if I just run the same thing with the newer version it takes little faster it is little faster now clearly 0.067. So, about; so, you know this divided by this one is 1.26 almost 25 percent faster. So, 25 percent faster is this means if in this way you can make your code efficient.

Now, you can tell me that ok sir why if I am whether one code is doing it 0.08 second or 0.06 second do, I really care in this case. So, yes if your data is small probably you do not care that much, but as a developer you always have aspire for writing code for very big data you want to handle very big data very big models right. Now imagine you have such a big data that your one code you have written two code; one code or two programmer has written two code.

One programmer's code take 8.5 seconds another programmer's code take 6.7 seconds which code 8.5 minutes and you know 8.5 hours this could be like 8.5 hours and this could be 6.7 hours who naturally the same code it will run the same code will run on very big large data

also like you know millions of data points. Here you have only 10 1000 data point actually 700 data points on which you are trying to train this model.

So, over if, but if you have millions of data points it might take 8.5 hours or 6.7 hours. Now if you are running it on the cloud; that means, and cloud services typically charge you by minutes sometimes even by seconds. So, you are reducing your model training cost by almost 25 percent if you use this code the older code it will cost you 125 dollars and if you use the efficient code it will cost you 100 dollars.

So; that means, you are saving 25 dollars in training very big large data on cloud and there is another aspect to it is; that means, your algorithm has less carbon footprint because these cloud machines take like you know burn takes lot of electricity and those electricity are typically has you know at least in country like India uses lot of fossil fuel.

So; that means, effectively a efficient code means less carbon footprint. So, you better write your code which runs faster and gives you code as accurate as possible. So, this time complexity unfortunately we statistic students our statistic students do not get to read much, but it is extremely important to study algorithms and time complexity, space complexity all these things. So, yeah.

So, this was a small video I thought I should show you guys that it is extremely important to write an efficient important efficient code everyone should watch this video and everyone should you know understand these concepts of time complexity.

Thank you very much, see you in the next video.