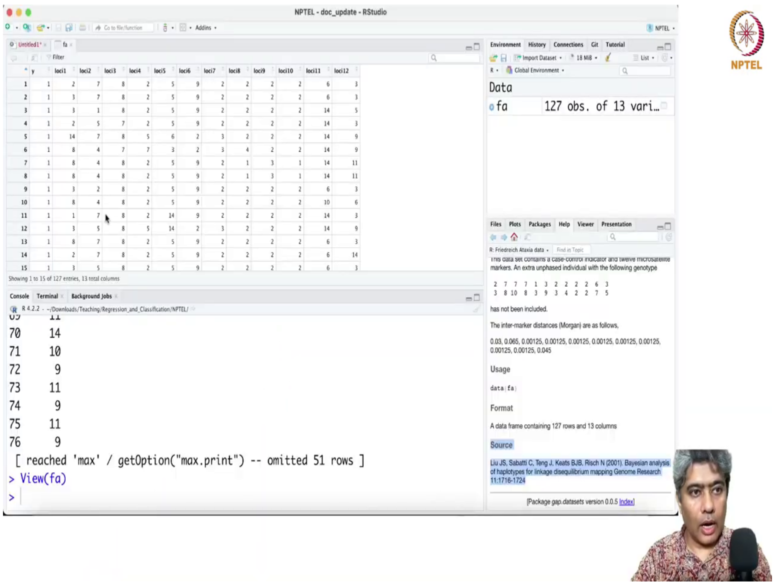
Predictive Analytics - Regression and Classification
Prof. Sourish Das
Department of Mathematics
Chennai Mathematical Institute

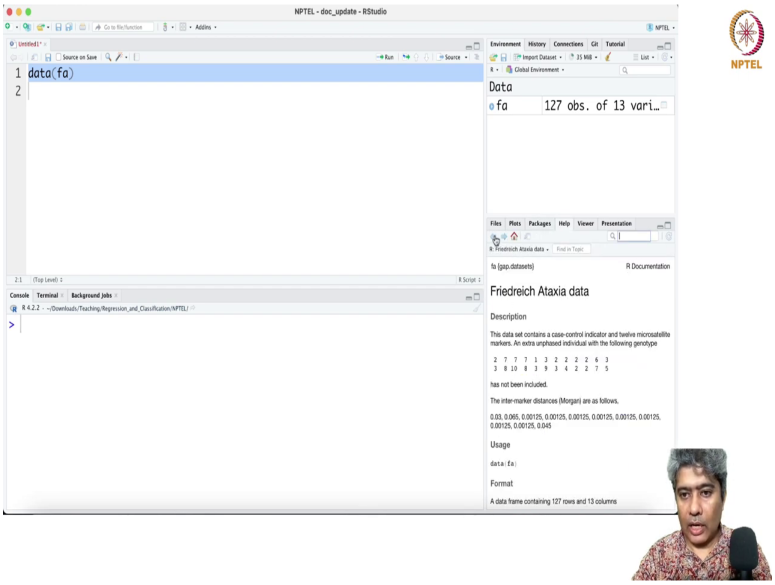
Lecture - 60
Hands on with Julia_Bayesian Logistic Regression with Horse Shoe Prior_Genetic Data Analysis

Hello all, welcome to the hands on exercise for this weeks. In this hands on exercise, we are going to do some logistic regression analysis using Julia from genetics data set that are available in R.

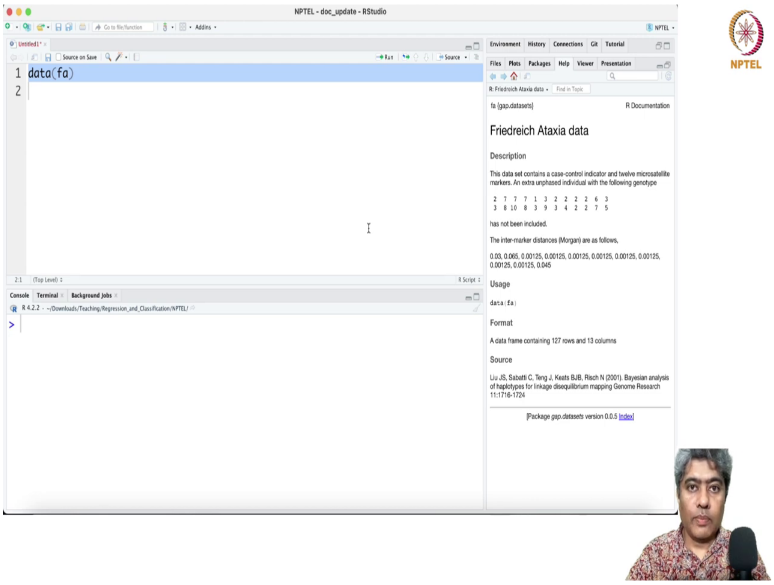
(Refer Slide Time: 00:39)



(Refer Slide Time: 00:41)



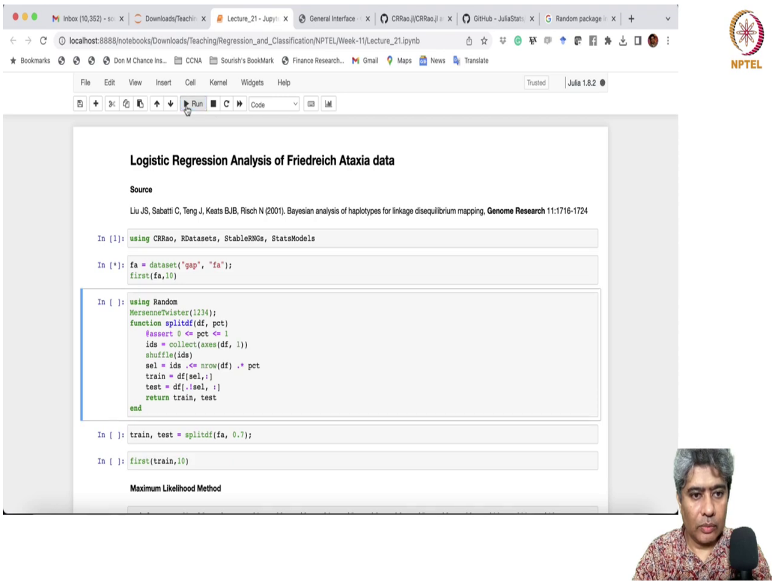
(Refer Slide Time: 00:55)



So, first I am going to start R. In R, you have this genetics in the package there is a called gap package Genetic Analysis Package. And in the genetic analysis package, there is a data called fa data, fa data, Friedreich Ataxia, data. We are going to use this data was first appeared in the genome research in 2001.

We are going to use this data to implement logistic regression using Julia. So, one of the advantage of Julia is many of the packages that are available in R are also available in Julia or from Julia you can call those data set using R data set from a package.

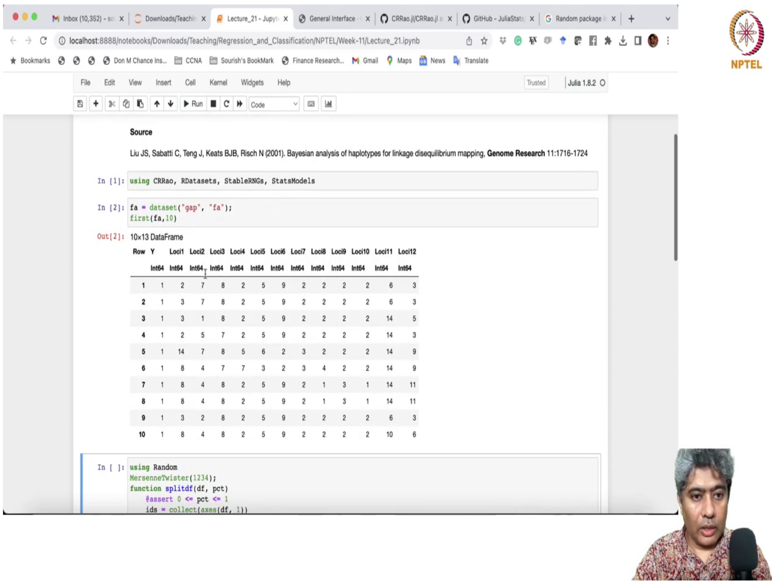
(Refer Slide Time: 01:37)



So, let me go to Julia my Jupyter Notebook. I have prepared this Jupyter Notebook to an extent to some extent and I will use this Jupyter Notebook to an extent this thing. So, the here is the in R data set you can see I have first what I am going to do let me just open this. So, maybe I will just do, ok. So, like here it is a Friedreich Ataxia data. So, I am going to give a name of you know.

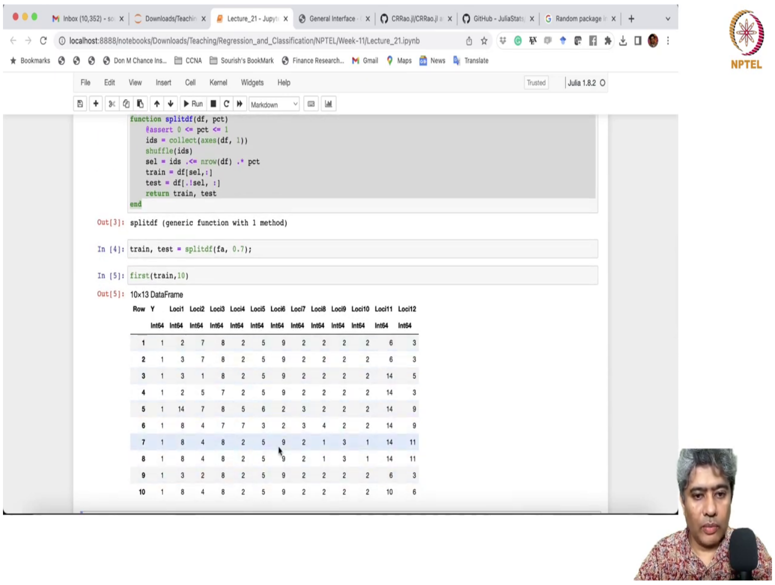
Ah So, maybe Logistic Regression Analysis of Ataxia data, ok. So, here is the source of the data I have given and so, maybe I will just turn it like this and this is the first thing I am going to call CRRao RDatasets StableRNG and StatModels, ok. So, this is first I am going to call this packages then I am going to from the RDatasets I am just calling RDatasets this is the name of the package gap package from R and fa is the name of the data set. So, I am let me just call this.

(Refer Slide Time: 03:09)



So, here we have the first column is the target column Y it has 0 1s and then there are 12 columns, ok. There are 12 columns which are we will be using as a predictor variable.

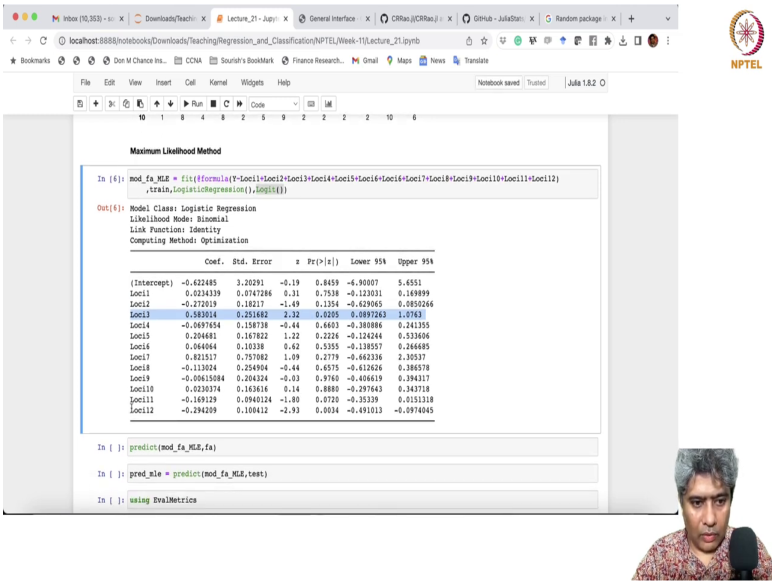
(Refer Slide Time: 03:41)



Now, what I am going to do in this if you just instead of actually run the first thing. So, here I am going to run this piece of code in this piece of code I am going to split the data frame into train and test. You can see that first time taking the percentage between 0 and 1 and then I am collecting the IDs of the data frame and then kind of shuffling them.

And 70 percent of that I am keeping it in the train and rest of them I am keeping in the test and then I am returning train and test. So, I am calling this split data frame function and splitting it into train and test. And so, this is the first 10 rows of the training data set.

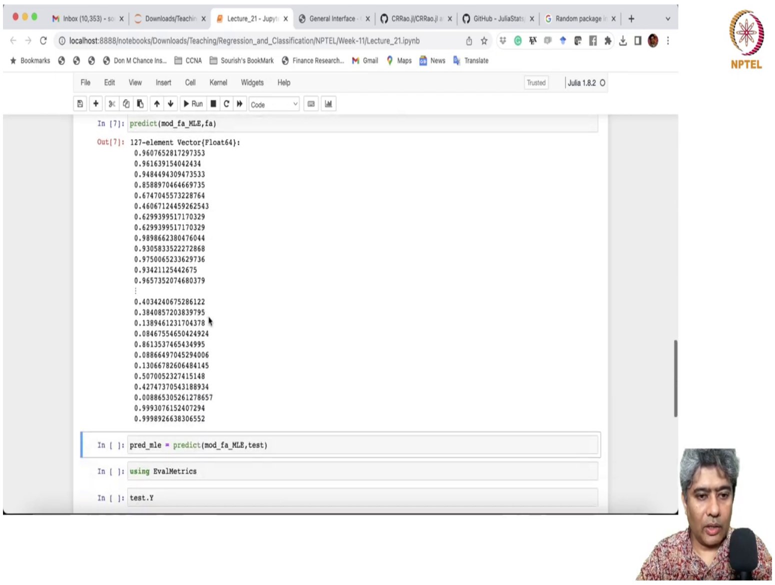
(Refer Slide Time: 04:34)



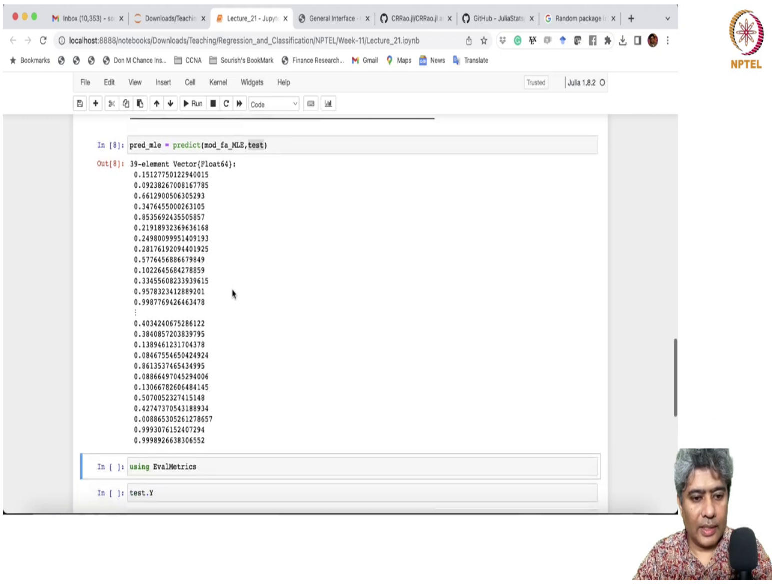
Then first I am going to implement the maximum likelihood of a method of logistic regression in that I am going to call fit from the CRRao. First, I have to provide the formula Y is the target and then all the predictors that I want to fit all the predictors that I have given here another data frame is train the I want to fit a logistic regression with Logit link. So, if I just run this. So, here is the output.

Now, if you see the P value for most of these Locis are not great except the third Loci third predictor Loci3 and then these are also not effective and Loci11 and 12 have effective here. So, these are the out of 12 Loci only three have effective kind of effect on the target variable.

(Refer Slide Time: 05:52)

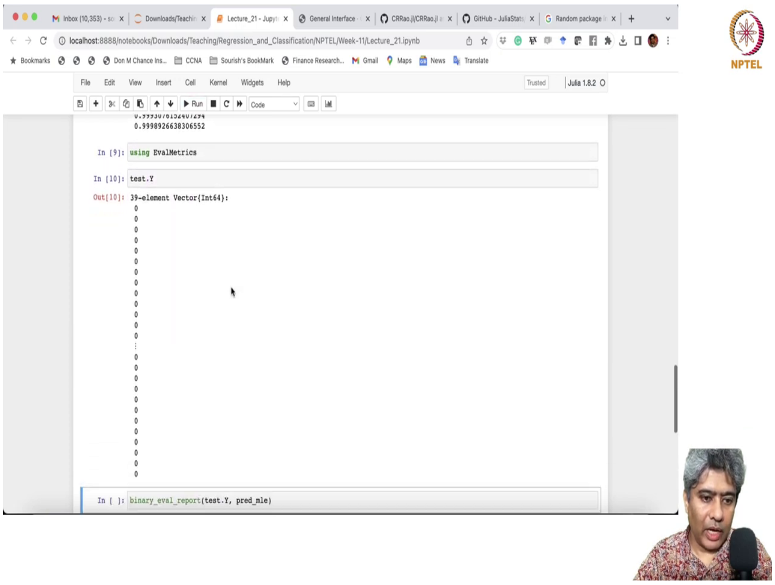


(Refer Slide Time: 06:00)



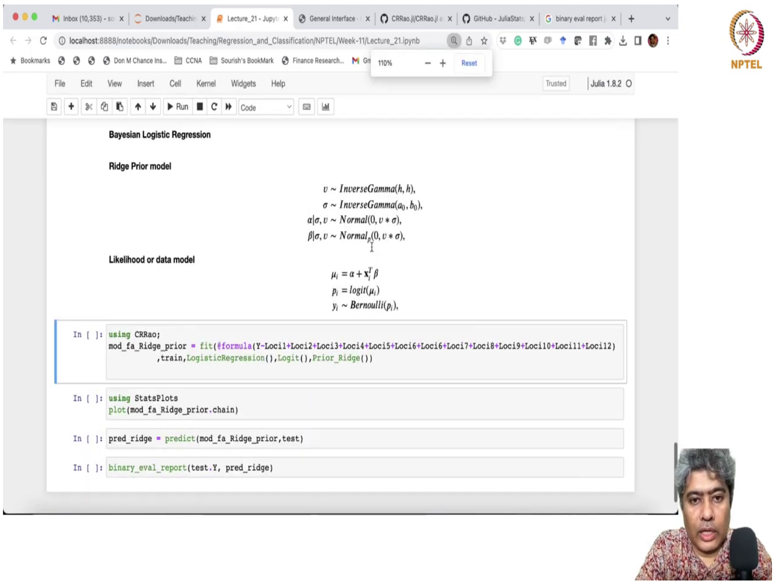
Now, using this model we can have a predictive we can predict the probability score on the you know on the train data set on the entire data set. Or similarly, if you just say predict you could call the predict function from CRRao and provide the model name and the test data set you can. So, you can I can I do not need this actually I need this one I need to calculate the predict on the test data set.

(Refer Slide Time: 06:32)



So, this gives me 39 the probability of a particular data point is 1 or 0 that will be based on this test data set. And then EvalMetrics is a package you can call and that will help us to do the evaluation and test dot Y is going to give me the either 0 or 1.

(Refer Slide Time: 06:52)

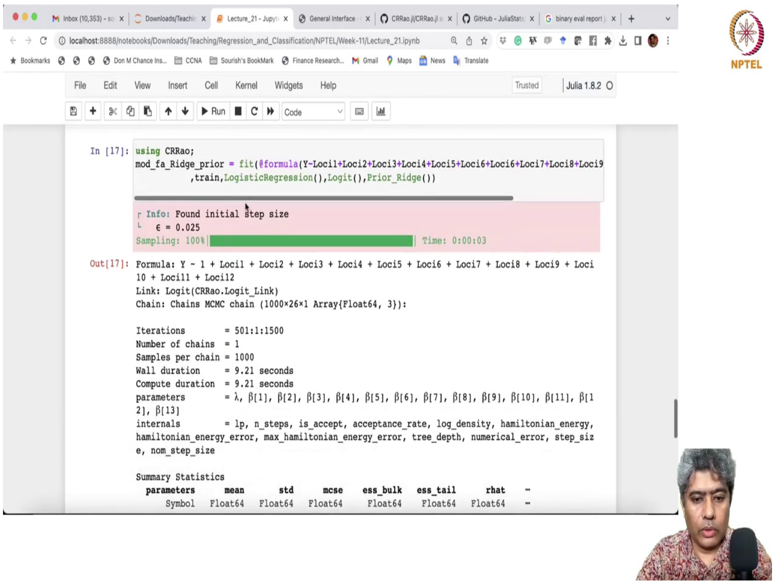


So, in the test from the test data set I am just taking the Y and binary eval report if I just give the fine test dot Y the actual Y values and the predicted score, ok. So, predict mle contains the all the probability scores, right. So, now if I just run this gives me the confusion matrix. Similarly, now if I have to if I want to implement the Bayesian Logistic Regression say Logistic Regression so, with say ridge prior model.

So, here if you see that Y follow Bernoulli mu i where mu i follow alpha plus x i transpose beta. And here is that that is my likelihood model and then well actually I should not say this actually mu i that should be like p i p i it should be p i actually. And then p i is (Refer Time: 08:07) equal to this logit mu i think now that is a right way of writing it and then mu i equal to x i transpose beta, which alpha plus x i transpose beta and then.

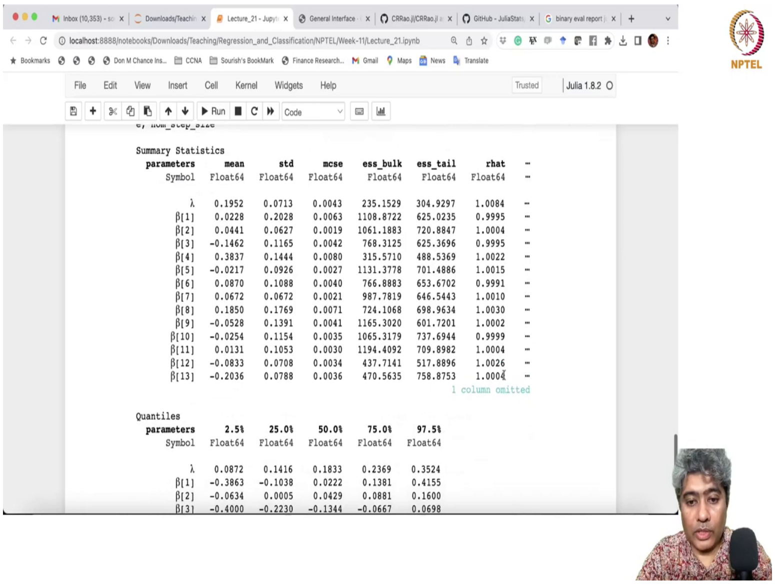
So, alpha follows say normal 0 and beta follow normal 0 v sigma follow inverse gamma and v follow some another inverse gamma then this model this is typically we will implement as ridge regression model. Now, what we can do if you see the way we have implemented the regression simple Logistic Regression model almost similar way we can implement it here except that at the end I am just using prior ridge.

(Refer Slide Time: 09:17)



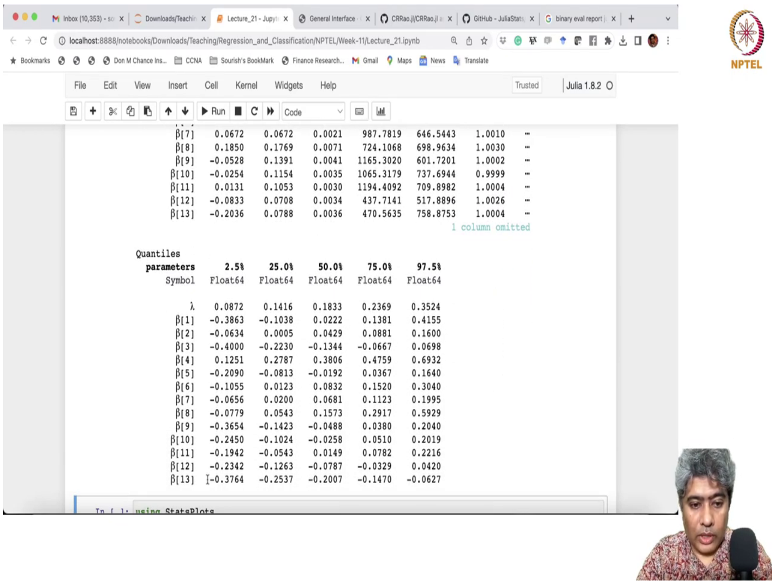
I am just calling ridge prior here and if I just run it. So, it will take few seconds maybe ok it is taking about few second it took about, alright.

(Refer Slide Time: 09:35)



So, it took most of the and if you see rhat all of them are kind of near one so; that means, if rhat is near one; that means, the convergence has taken place.

(Refer Slide Time: 09:51)

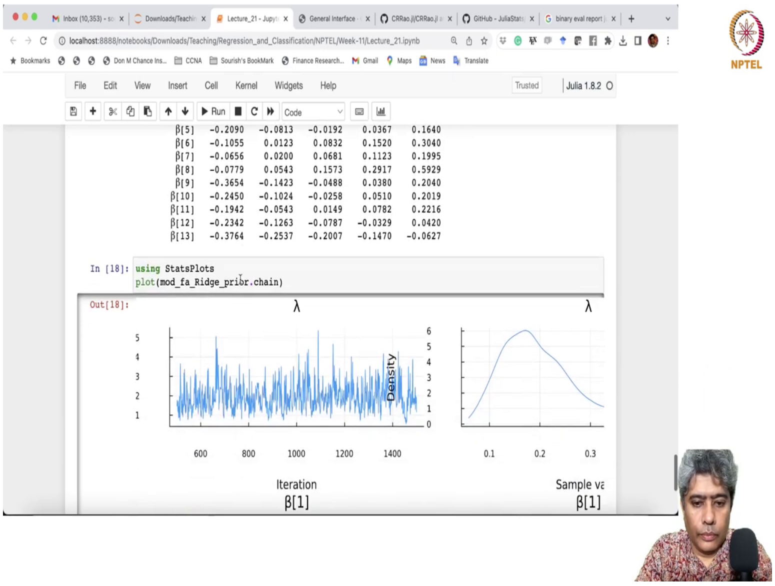


And if you see 13 entirely negative, but 12 is not negativity it includes the confidence interval if I see the 2.5 percentile point and 90. So, let me just explain you this Bayesian statistics a little bit. So, this is first the coefficient estimates and these are the standard deviations, if I have to compare it with the likelihood estimates, right.

So, these are the my coefficient estimates and these are the standard error. So, that is how you will see it here these are my coefficient estimates. So, beta 1 you will see including beta 1 it is there are 13 betas whereas; here I have actually 12 betas and alpha. So, total 13 betas I have. So, the first one is a intercept beta 1 this is intercept and then or here it is intercept and then you have 12 coefficient corresponds to each of the predictor.

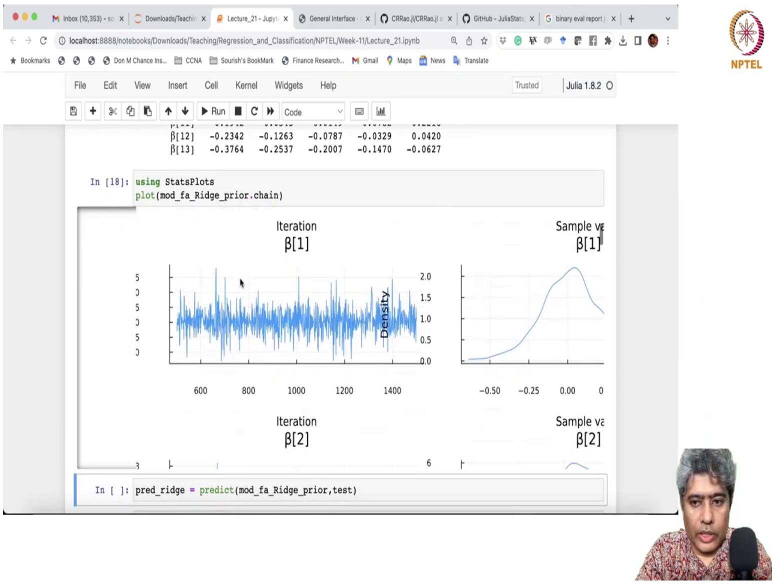
Now, these are the some Monte Carlo standard error and if smaller Monte Carlo standard error indicates that the most likely it has converged nicely. And another statistics is called rhat if rhat is close to one; that means, the mcmc convergence has taken place. So, Bayesian statistics in this case are being implemented using Markov chain Monte Carlo algorithm and not gradient descent algorithm.

(Refer Slide Time: 11:35)



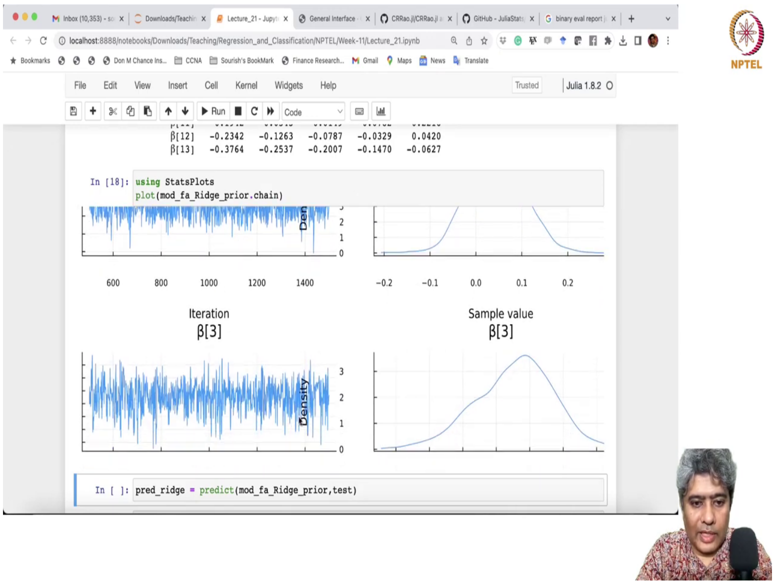
So, it is a Monte Carlo simulation based technique that they have increased and here for inference what they do they give a 95 percent confidence interval. And if you look into the 95 percent confidence that the last coefficient does have a effect sort of. And the third coefficient also does have a effect it is a positive and this is a negative coefficient and rest all of the coefficient includes 0 some way or other. So, there none of them have actually any effect on that.

(Refer Slide Time: 12:08)

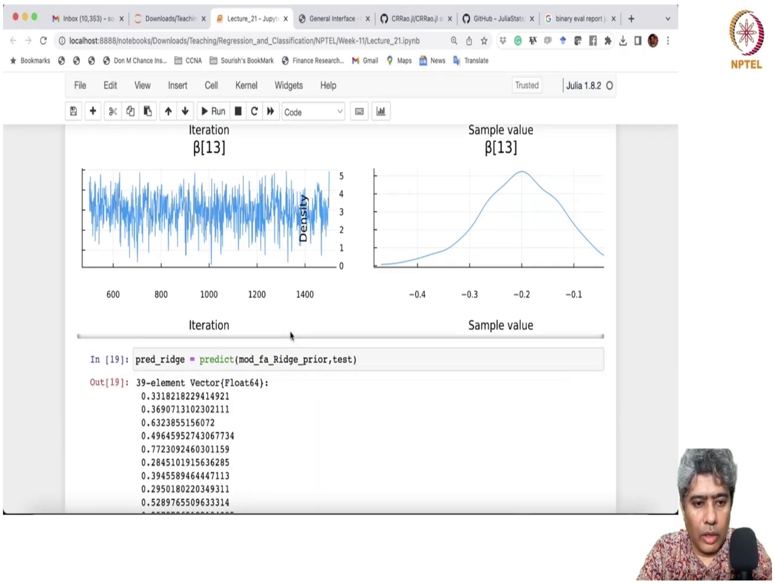


(Refer Slide Time: 12:10)

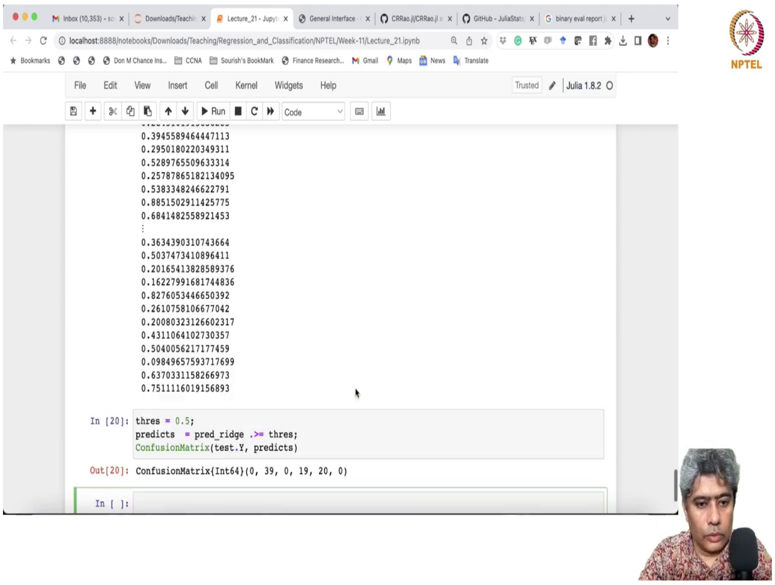
(Refer Slide Time: 12:13)



(Refer Slide Time: 12:24)



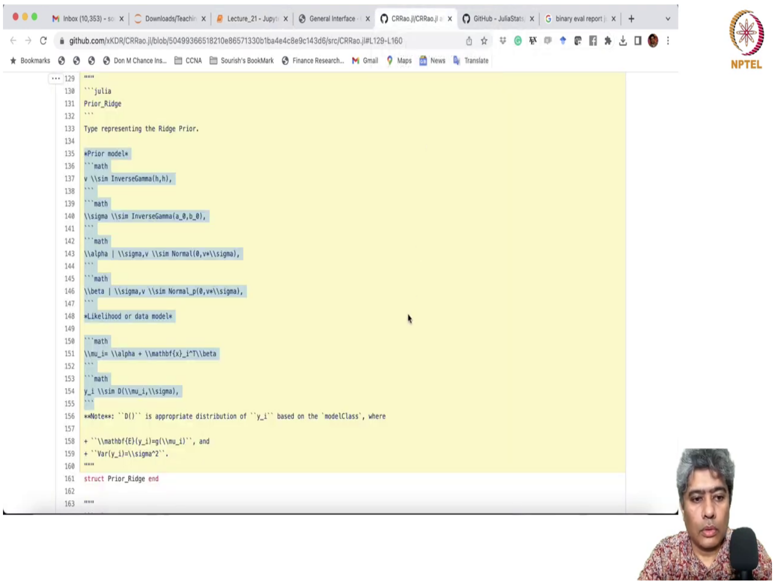
(Refer Slide Time: 12:29)



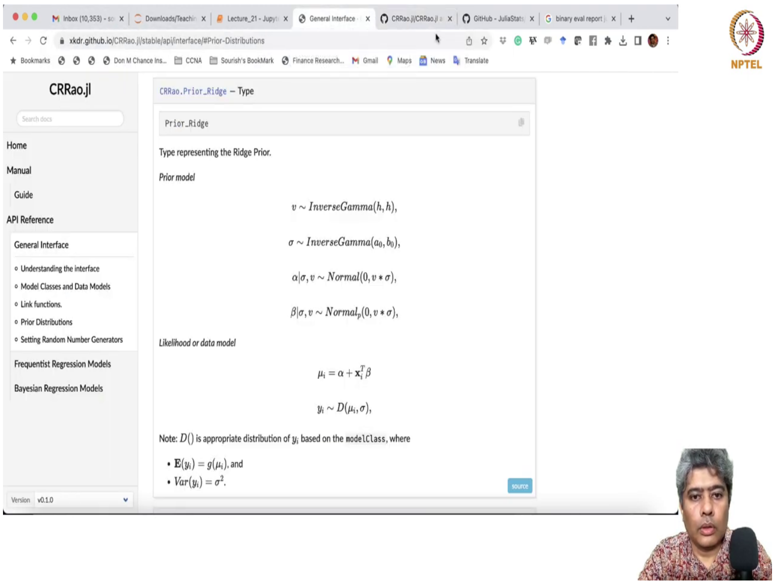
So, if I just do a plot simple stat plot. So, here is the plots you see they have decently converged nicely converged, ok. All of them pretty much decently converged here I am doing the predictive error. And here I have to do the confusion matrix I have to compute for some reason binary evaluation of the report is not working I will come back to you on the same.

So, I will say take predictive ridge as that and then if I just use that. So, these are the confusion matrix that have we have found. So, I am not sure if it is correctly done. So, I will check with this.

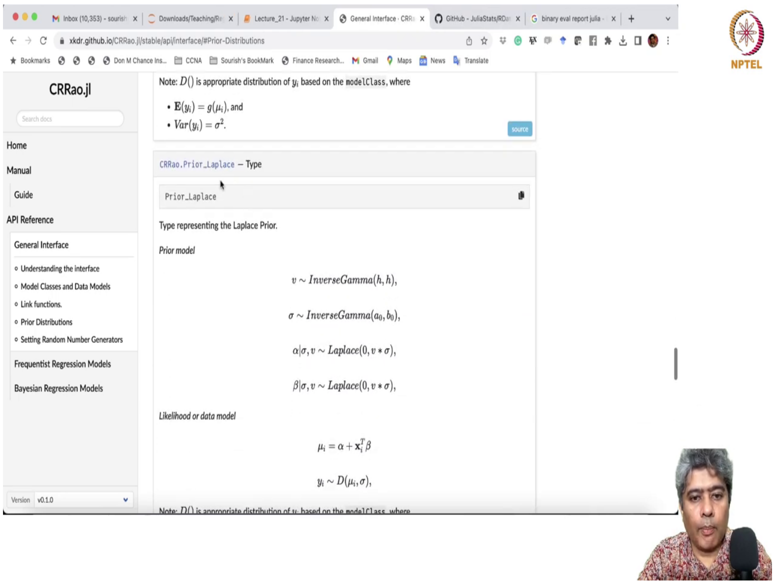
(Refer Slide Time: 13:11)



(Refer Slide Time: 13:23)

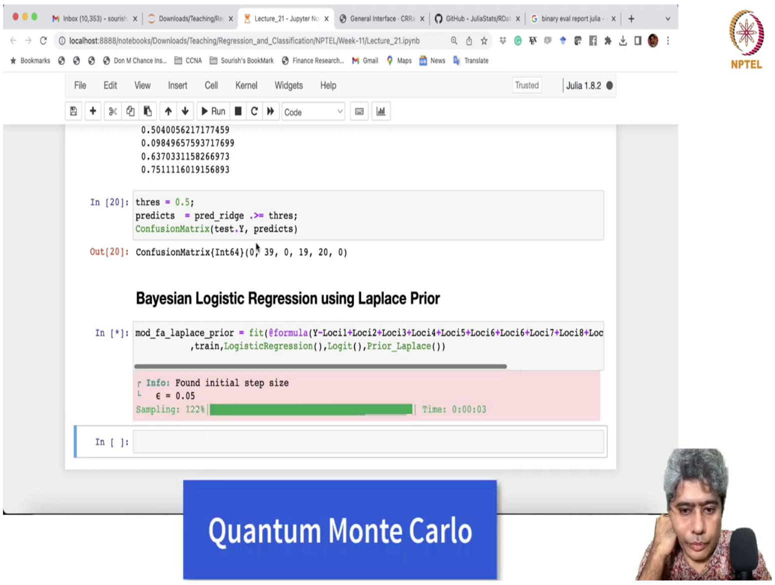


(Refer Slide Time: 13:28)



Now, if you look back if you go to the say if I am now interested in implementing say Laplace prior how do I do implement.

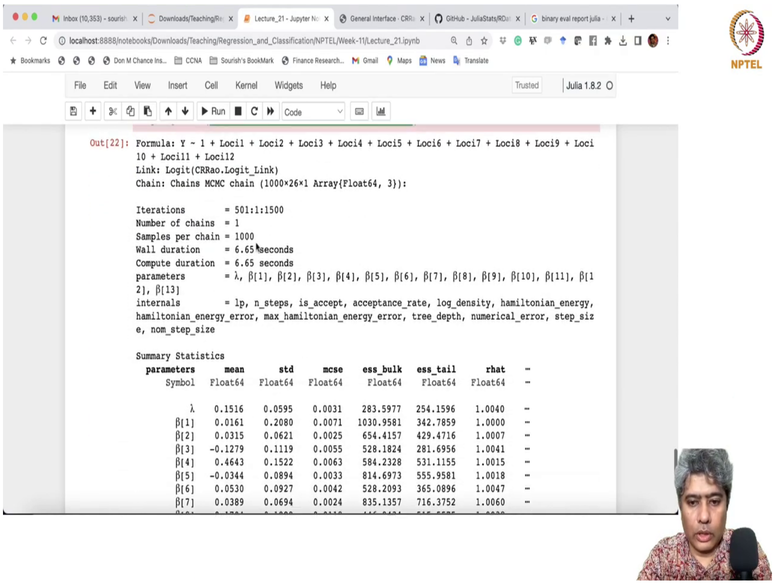
(Refer Slide Time: 13:33)



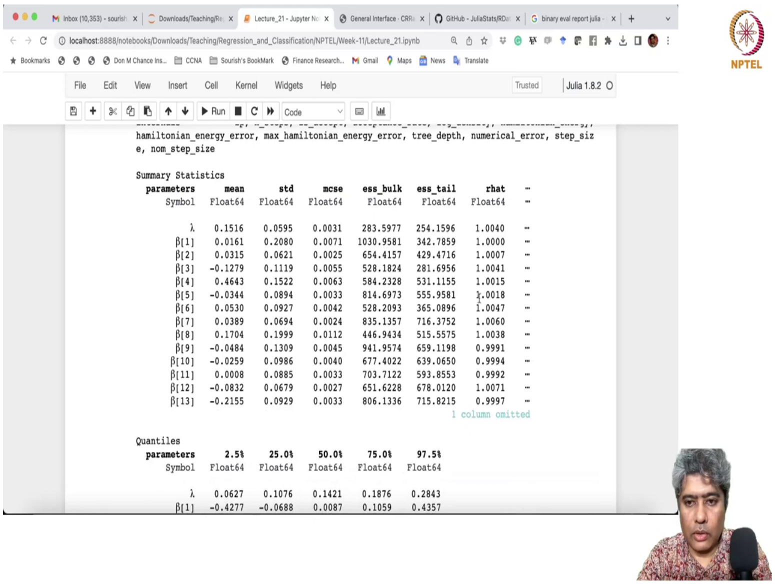
So, if I am interested in implementing say Bayesian Logistic Regression Bayesian Logistic Regression using Laplace prior, ok. So, if I have to do that all I have to do is just copy this guy from here and instead of I will just name it as Laplace and instead of prior ridge. I will just say Prior underscore Laplace.

So, Prior underscore Laplace ok and rest of the thing will be exactly same and now if you just run sorry, I think Laplace there was a spelling mistake yeah now I think it is done, yeah.

(Refer Slide Time: 14:58)



(Refer Slide Time: 15:02)



hamiltonian_energy_error, max_hamiltonian_energy_error, tree_depth, numerical_error, step_size, nom_step_size

Summary Statistics

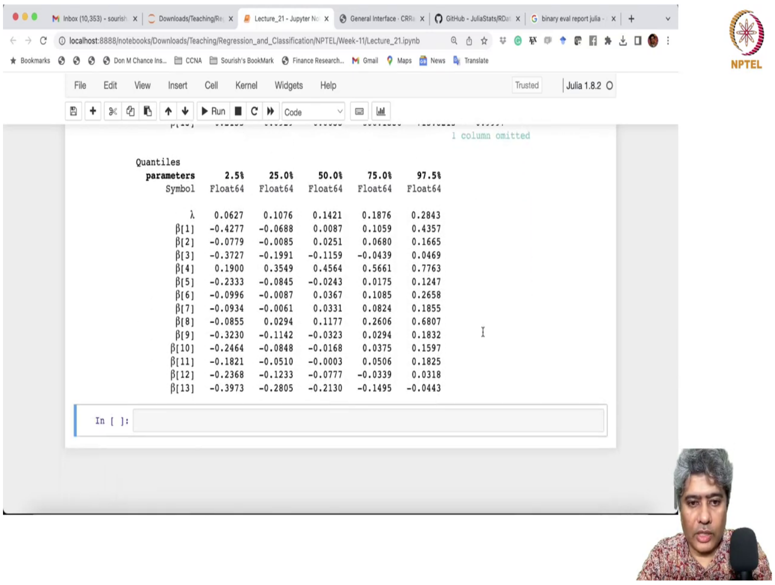
parameters	mean	std	mcse	ess_bulk	ess_tail	rhat	⋯
Symbol	Float64	Float64	Float64	Float64	Float64	Float64	⋯
λ	0.1516	0.0595	0.0031	283.5977	254.1596	1.0040	⋯
β[1]	0.0161	0.2080	0.0071	1030.9581	342.7859	1.0000	⋯
β[2]	0.0315	0.0621	0.0025	654.4157	429.4716	1.0007	⋯
β[3]	-0.1279	0.1119	0.0055	528.1824	281.6956	1.0041	⋯
β[4]	0.4643	0.1522	0.0063	584.2328	531.1155	1.0015	⋯
β[5]	-0.0344	0.0894	0.0033	814.6973	555.9581	1.0018	⋯
β[6]	0.0530	0.0927	0.0042	528.2093	365.0896	1.0047	⋯
β[7]	0.0389	0.0694	0.0024	835.1357	716.3752	1.0060	⋯
β[8]	0.1704	0.1999	0.0112	446.9434	515.5575	1.0038	⋯
β[9]	-0.0484	0.1309	0.0045	941.9574	659.1198	0.9991	⋯
β[10]	-0.0259	0.0986	0.0040	677.4022	639.0650	0.9994	⋯
β[11]	0.0008	0.0885	0.0033	703.7122	593.8553	0.9992	⋯
β[12]	-0.0832	0.0679	0.0027	651.6228	678.0120	1.0071	⋯
β[13]	-0.2155	0.0929	0.0033	806.1336	715.8215	0.9997	⋯

1 column omitted

Quantiles

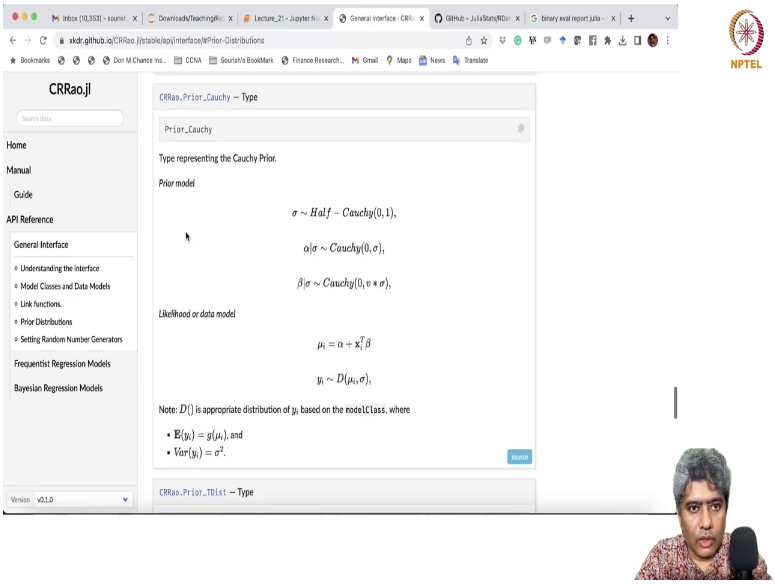
parameters	2.5%	25.0%	50.0%	75.0%	97.5%
Symbol	Float64	Float64	Float64	Float64	Float64
λ	0.0627	0.1076	0.1421	0.1876	0.2843
β[1]	-0.4277	-0.0688	0.0087	0.1059	0.4357

(Refer Slide Time: 15:04)



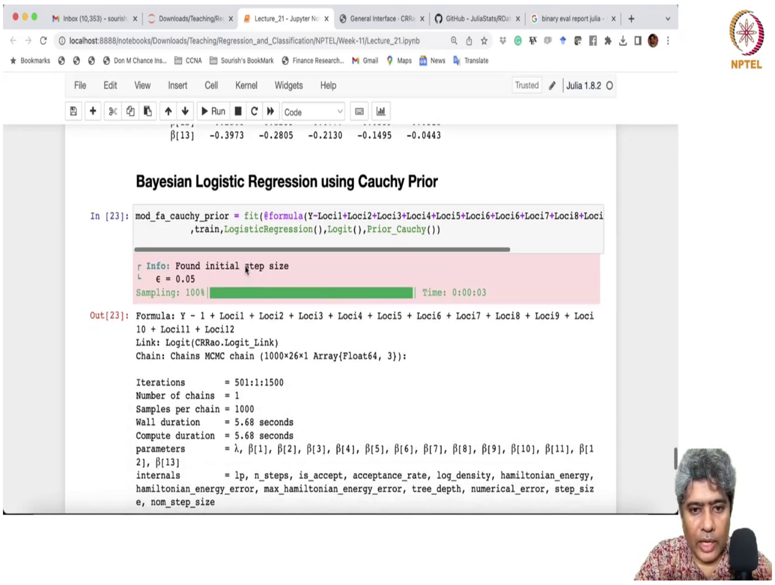
So, ok looks like Laplace also has been implemented very fast and it is kind of 1500 simulation samples have been created from 1500 samples first 500 is being considered as a burning and from 501 to 1500, 1000 samples based on 1000 samples it is being created. And similar kind of inference you can see you can do based on the credible interval.

(Refer Slide Time: 15:40)



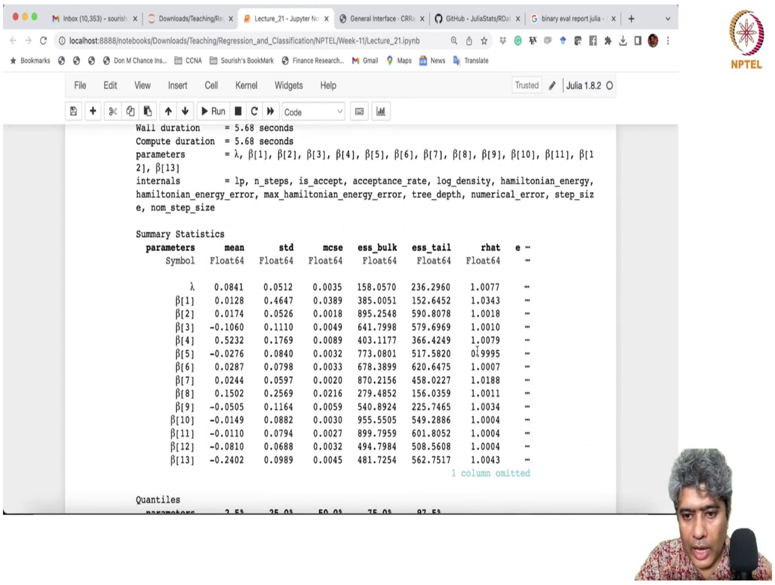
Now, if you interested in say doing Cauchy interval Cauchy prior. So, all you have to do is just take this information.

(Refer Slide Time: 15:51)



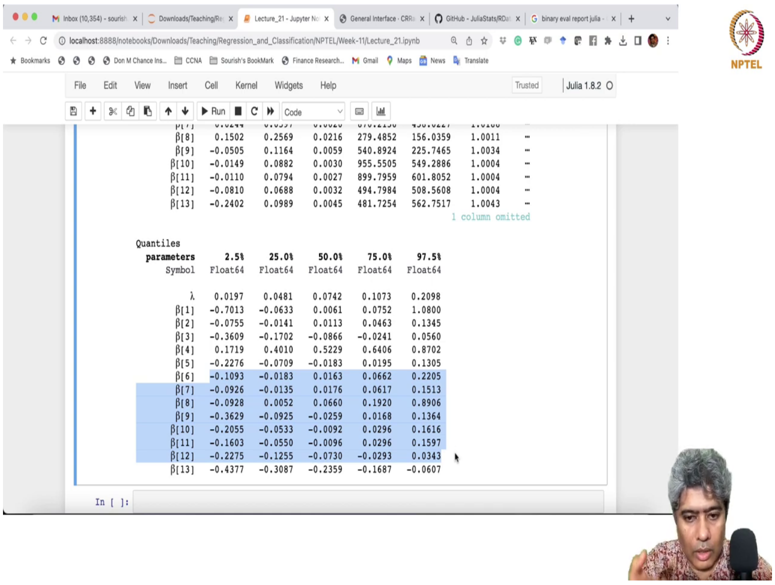
And now if I want to introduce say all I have to do just take this and instead of Laplace Prior I want Cauchy prior, ok C a u c h y Cauchy Prior, but not Laplace. So, I will just instead of Laplace I will say Cauchy Prior. So, I will just effectively if I just write the C and then tab will fill up the rest of the thing if I just write the C and then tab if you just press tab it will fill up the rest.

(Refer Slide Time: 16:56)



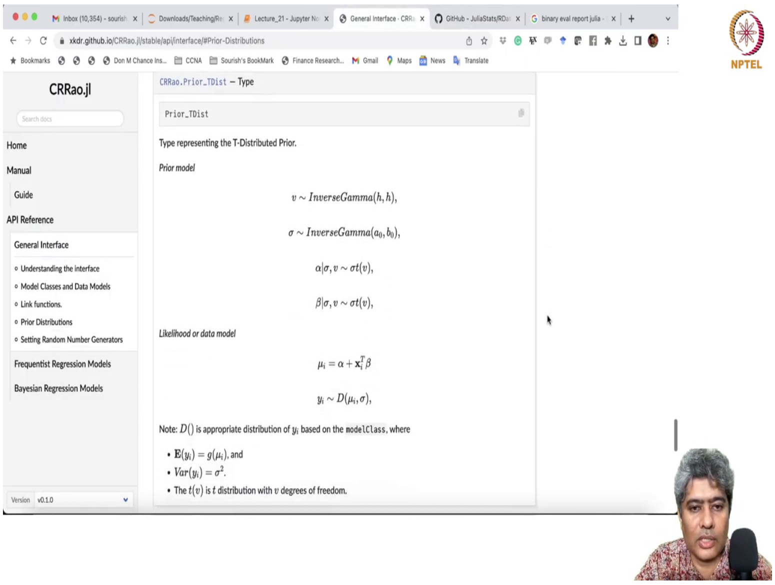
So, instead of Laplace I want to name it as a say Cauchy prior. So, for some reason the evaluation methods are not working properly I will correct those and I will share the correct Notebook with you guys. For some reason it is not working, but as you see here also pretty much all rhats are first thing you should check in the Bayesian statistics with all rhats are very close to one or not if it is indeed, one close to one then you are in good shape.

(Refer Slide Time: 17:09)

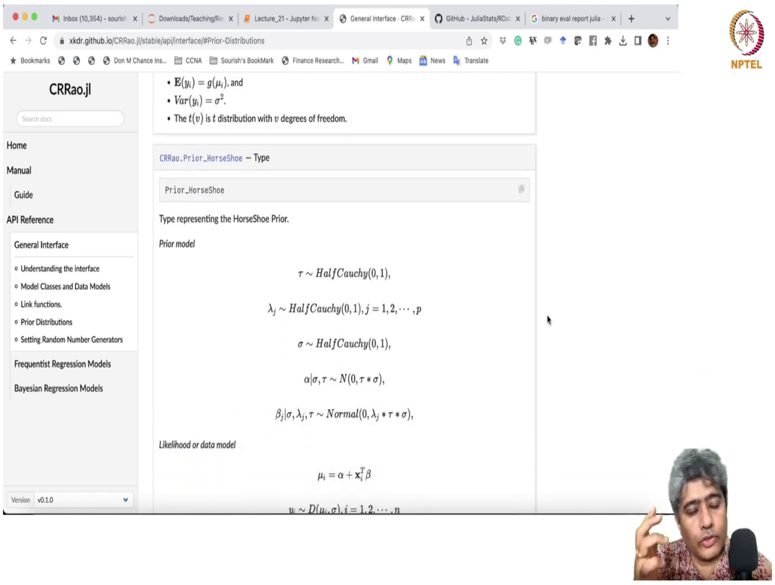


And here also you can see the third predictor is all positive and the last predictor is all negative rest of the all predictors are essentially either or including 0 then 95 percent confidence interval is including 0. So, similar consistent inference that we are getting from all of the Cauchy Priors.

(Refer Slide Time: 17:46)

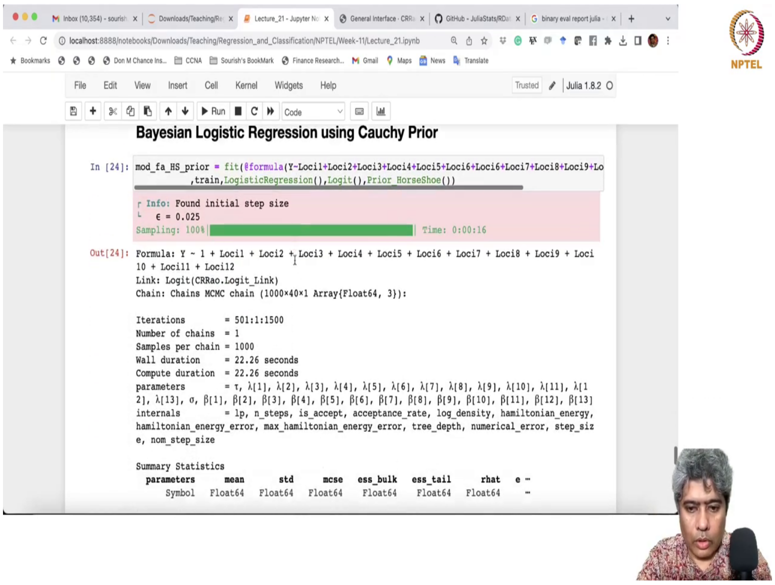


(Refer Slide Time: 17:48)



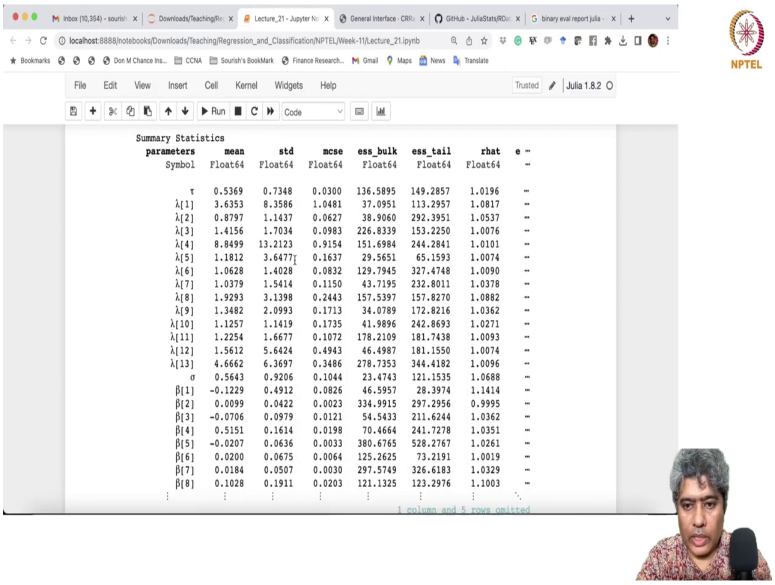
Now, let us see what else are we do we have other than Cauchy Prior we have also T distributed prior and then HorseShoe Prior HorseShoe Prior is very popular. Let us try to implement the HorseShoe Prior, ok. So, let us try to implement the HorseShoe Prior.

(Refer Slide Time: 18:02)



So, let me just do that and here is the. So, all I have to do just go copy this model and now instead of in the prior area just instead of Cauchy Prior just say HorseShoe. So, I will just say H and then tab it will fill up the rest and I am pretty much done. Of course, I do I want to change the name to HorseShoe Prior and run. HorseShoe Prior may require a little bit of more run maybe thousand simulation is not enough because it has lot of parameters in it.

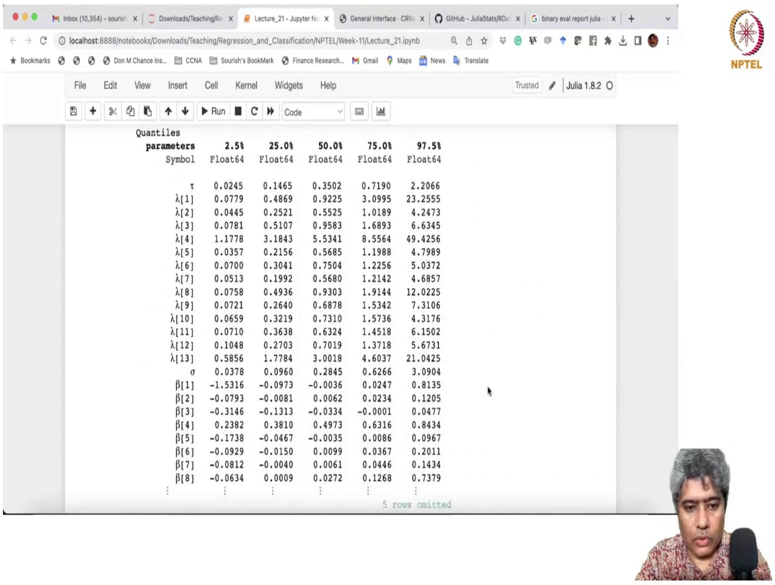
(Refer Slide Time: 19:01)


Summary Statistics

parameters	mean	std	mcse	ess_bulk	ess_tail	rhat	e ⋯
Symbol	Float64	Float64	Float64	Float64	Float64	Float64	⋯
τ	0.5369	0.7348	0.0300	136.5895	149.2857	1.0196	⋯
λ[1]	3.6353	8.3586	1.0481	37.0951	113.2957	1.0817	⋯
λ[2]	0.8797	1.1437	0.0627	38.9060	292.3951	1.0537	⋯
λ[3]	1.4156	1.7034	0.0983	226.8339	153.2250	1.0076	⋯
λ[4]	8.8499	13.2123	0.9154	151.6984	244.2841	1.0101	⋯
λ[5]	1.1812	3.6477	0.1637	29.5651	65.1593	1.0074	⋯
λ[6]	1.0628	1.4028	0.0832	129.7945	327.4748	1.0090	⋯
λ[7]	1.0379	1.5414	0.1150	43.7195	232.8011	1.0378	⋯
λ[8]	1.9293	3.1398	0.2443	157.5397	157.8270	1.0882	⋯
λ[9]	1.3482	2.0993	0.1713	34.0789	172.8216	1.0362	⋯
λ[10]	1.1257	1.1419	0.1735	41.9896	242.8693	1.0271	⋯
λ[11]	1.2254	1.6677	0.1072	178.2109	181.7438	1.0093	⋯
λ[12]	1.5612	5.6424	0.4943	46.4987	181.1550	1.0074	⋯
λ[13]	4.6662	6.3697	0.3486	278.7353	344.4182	1.0096	⋯
σ	0.5643	0.9206	0.1044	23.4743	121.1535	1.0688	⋯
β[1]	-0.1229	0.4912	0.0826	46.5957	28.3974	1.1414	⋯
β[2]	0.0099	0.0422	0.0023	334.9915	297.2956	0.9995	⋯
β[3]	-0.0706	0.0979	0.0121	54.5433	211.6244	1.0362	⋯
β[4]	0.5151	0.1614	0.0198	70.4664	241.7278	1.0351	⋯
β[5]	-0.0207	0.0636	0.0033	380.6765	528.2767	1.0261	⋯
β[6]	0.0200	0.0675	0.0064	125.2625	73.2191	1.0019	⋯
β[7]	0.0184	0.0507	0.0030	297.5749	326.6183	1.0329	⋯
β[8]	0.1028	0.1911	0.0203	121.1325	123.2976	1.1003	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

1 column and 5 rows omitted

(Refer Slide Time: 19:07)



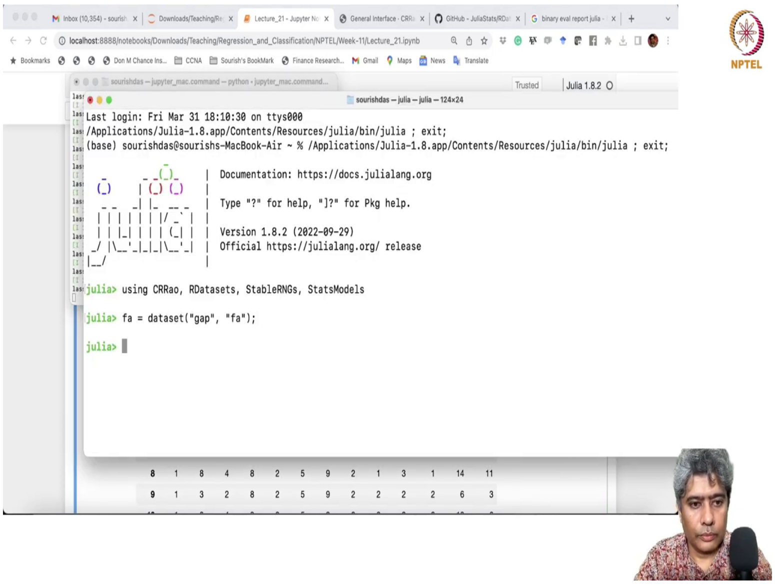
Quantiles

parameters	2.5%	25.0%	50.0%	75.0%	97.5%
Symbol	Float64	Float64	Float64	Float64	Float64
τ	0.0245	0.1465	0.3502	0.7190	2.2066
λ[1]	0.0779	0.4869	0.9225	3.0995	23.2555
λ[2]	0.0445	0.2521	0.5525	1.0189	4.2473
λ[3]	0.0781	0.5107	0.9583	1.6893	6.6345
λ[4]	1.1778	3.1843	5.5341	8.5564	49.4256
λ[5]	0.0357	0.2156	0.5685	1.1988	4.7989
λ[6]	0.0700	0.3041	0.7504	1.2256	5.0372
λ[7]	0.0513	0.1992	0.5680	1.2142	4.6857
λ[8]	0.0758	0.4936	0.9303	1.9144	12.0225
λ[9]	0.0721	0.2640	0.6878	1.5342	7.3106
λ[10]	0.0659	0.3219	0.7310	1.5736	4.3176
λ[11]	0.0710	0.3638	0.6324	1.4518	6.1502
λ[12]	0.1048	0.2703	0.7019	1.3718	5.6731
λ[13]	0.5856	1.7784	3.0018	4.6037	21.0425
σ	0.0378	0.0960	0.2845	0.6266	3.0904
β[1]	-1.5316	-0.0973	-0.0036	0.0247	0.8135
β[2]	-0.0793	-0.0081	0.0062	0.0234	0.1205
β[3]	-0.3146	-0.1313	-0.0334	-0.0001	0.0477
β[4]	0.2382	0.3810	0.4973	0.6316	0.8434
β[5]	-0.1738	-0.0467	-0.0035	0.0086	0.0967
β[6]	-0.0929	-0.0150	0.0099	0.0367	0.2011
β[7]	-0.0812	-0.0040	0.0061	0.0446	0.1434
β[8]	-0.0634	0.0009	0.0272	0.1268	0.7379
⋮	⋮	⋮	⋮	⋮	⋮

5 rows omitted

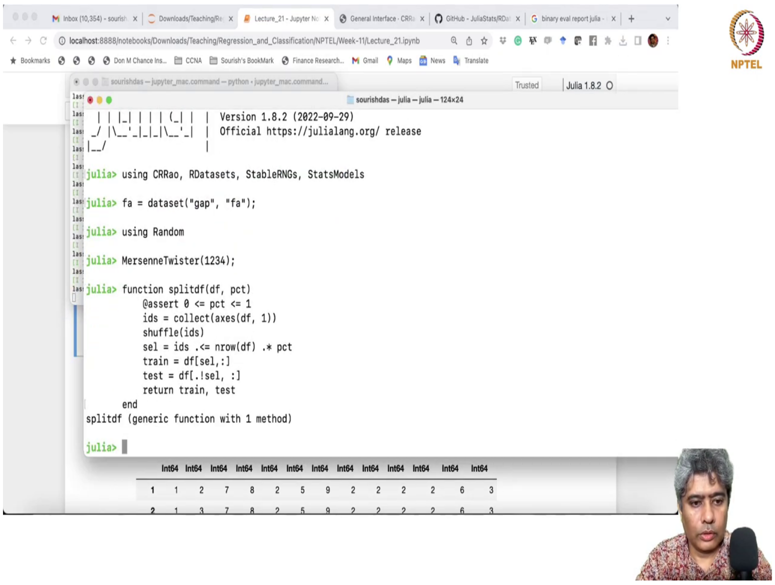
So, we will see interesting, but we are seeing that lot of most of the parameters have been converged. So, ok, but unfortunately it is not printing all the parameters ok, that is in Jupyter Notebook that sometimes a problem.

(Refer Slide Time: 19:32)

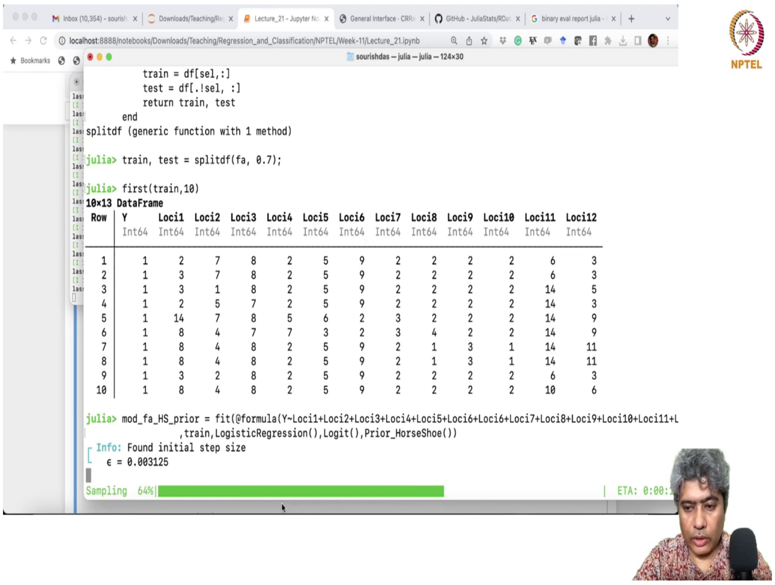


So, what I will do I will I will run it in the maybe code Jupyter Notebook co code Julia. So, I will open code Julia so, that we can see the results, ok. So, I will just call the you know let me just run this few lines there, ok. And then if I just I will just call this guy, ok.

(Refer Slide Time: 20:08)

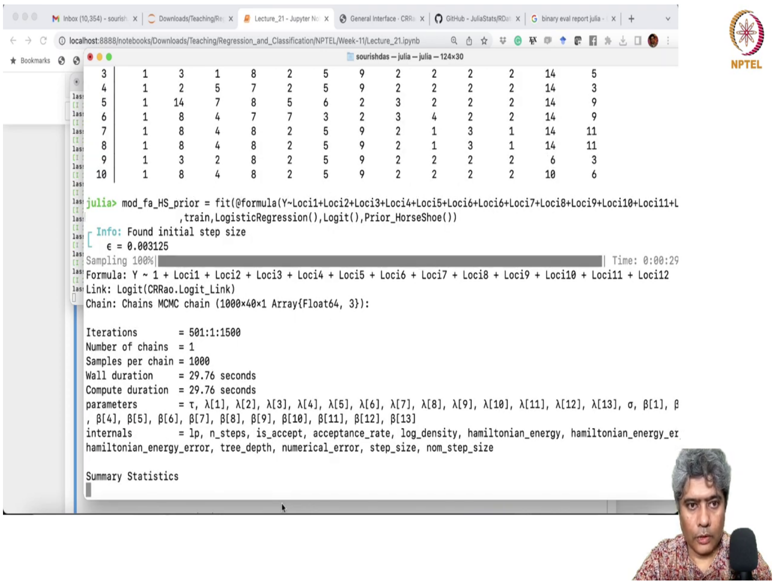


(Refer Slide Time: 20:23)



And I will call this split function train and test and then, ok then finally, I will just use the run the last data set last model, ok. Let us see how much time it takes yeah, its relatively faster actually, ok.

(Refer Slide Time: 21:11)



(Refer Slide Time: 21:14)

β[13]	-0.2512	0.1022	0.0099	112.2952	138.6843	1.0100	3.7727

Quantiles

parameters	2.5%	25.0%	50.0%	75.0%	97.5%
Symbol	Float64	Float64	Float64	Float64	Float64
τ	0.0448	0.1858	0.3741	0.6854	2.4024
λ[1]	0.0686	0.3783	0.9776	2.0487	10.3659
λ[2]	0.0160	0.1999	0.4936	1.0608	3.6866
λ[3]	0.0896	0.4442	1.0361	2.3558	8.8854
λ[4]	1.1317	3.0449	5.3431	8.7824	30.4613
λ[5]	0.0435	0.3840	0.7849	1.6689	4.3605
λ[6]	0.0561	0.2818	0.6471	1.2940	5.7463
λ[7]	0.0490	0.2006	0.5525	1.1298	4.7579
λ[8]	0.0926	0.5026	1.0171	2.0604	9.4914
λ[9]	0.0380	0.3298	0.6857	1.4236	5.7221
λ[10]	0.0679	0.3682	0.8375	1.4703	5.9469
λ[11]	0.0398	0.2101	0.5745	1.1829	4.3867
λ[12]	0.1013	0.4366	0.8015	1.4370	4.7194
λ[13]	0.4595	1.4928	2.5101	4.4102	14.2099
σ	0.0260	0.1057	0.2357	0.5544	1.9331
β[1]	-0.4893	-0.0390	0.0015	0.0676	0.4693
β[2]	-0.0772	-0.0107	0.0035	0.0331	0.1224
β[3]	-0.3291	-0.1268	-0.0345	0.0039	0.0557
β[4]	0.2264	0.4056	0.5106	0.6402	0.8183
β[5]	-0.2086	-0.0865	-0.0183	0.0082	0.0896
β[6]	-0.1097	-0.0274	0.0010	0.0392	0.1788
β[7]	-0.0656	-0.0097	0.0061	0.0421	0.1362
β[8]	-0.0793	0.0026	0.0336	0.1589	0.7289
β[9]	-0.2728	-0.0610	-0.0099	0.0074	0.1029
β[10]	-0.2158	-0.0573	-0.0038	0.0188	0.1154
β[11]	-0.1492	-0.0282	-0.0034	0.0154	0.1203
β[12]	-0.2074	-0.1010	-0.0451	-0.0140	0.0304
β[13]	-0.4347	-0.3244	-0.2532	-0.1836	-0.0293

(Refer Slide Time: 21:38)

Summary Statistics

parameters	mean	std	mcse	ess_bulk	ess_tail	rhat	ess_per_sec
Symbol	Float64	Float64	Float64	Float64	Float64	Float64	Float64
τ	0.6057	1.1656	0.0485	213.7638	279.2180	1.0098	7.1817
λ[1]	2.6673	20.9406	0.6698	65.2821	24.4273	1.0147	2.1933
λ[2]	0.8842	1.4997	0.0725	50.9558	19.2813	1.0183	1.7119
λ[3]	2.1101	2.7614	0.4457	59.7784	61.1237	1.0034	2.0083
λ[4]	7.8439	10.9166	0.6629	232.1859	301.8317	1.0171	7.8006
λ[5]	1.2158	1.4750	0.0764	171.6405	301.7291	1.0130	5.7665
λ[6]	2.0104	14.1987	0.8775	127.5979	168.4914	1.0218	4.2868
λ[7]	0.9570	1.4002	0.0899	141.1977	377.9457	1.0083	4.7437
λ[8]	2.1867	7.7913	0.3080	207.1347	364.7326	1.0046	6.9590
λ[9]	1.3076	2.2644	0.1557	146.8121	158.6430	1.0112	4.9324
λ[10]	1.3518	1.8582	0.0956	264.7104	354.4881	0.9996	8.8933
λ[11]	0.9866	1.3688	0.0905	146.0681	340.2527	1.0108	4.9074
λ[12]	1.2057	1.3025	0.0592	362.3571	441.7075	1.0043	12.1739
λ[13]	3.8526	4.7133	0.2952	147.1950	271.1536	1.0140	4.9452
σ	0.4555	0.5912	0.0350	183.0016	314.3923	1.0042	6.1482
β[1]	0.0082	0.2333	0.0125	406.6616	274.4591	0.9992	13.6624
β[2]	0.0124	0.0478	0.0030	298.8766	223.0706	0.9997	10.0412
β[3]	-0.0710	0.1056	0.0060	210.9314	299.9385	1.0005	7.0866
β[4]	0.5221	0.1637	0.0184	82.3646	321.5270	1.0108	2.7672
β[5]	-0.0396	0.0766	0.0101	70.1478	293.5113	1.0217	2.3567
β[6]	0.0069	0.0745	0.0076	94.3500	91.0072	1.0725	3.1698
β[7]	0.0169	0.0475	0.0031	210.9477	495.1743	1.0016	7.0871
β[8]	0.1140	0.1987	0.0200	190.2772	95.3975	1.0136	6.3926
β[9]	-0.0365	0.0989	0.0065	337.8955	285.8960	0.9990	11.3521
β[10]	-0.0215	0.0806	0.0051	263.1513	258.6645	1.0072	8.8410
β[11]	-0.0086	0.0621	0.0040	326.2493	131.9959	1.0019	10.9608
β[12]	-0.0610	0.0647	0.0048	198.6906	370.8219	1.0010	6.6753
β[13]	-0.2512	0.1022	0.0099	112.2952	138.6843	1.0100	3.7727

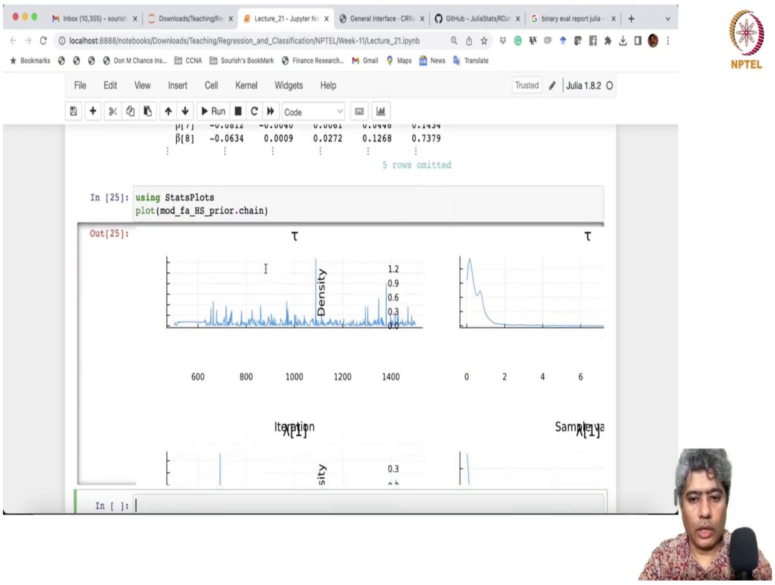
Quantiles

parameters	2.5%	25.0%	50.0%	75.0%	97.5%

So, alright so, if I just run right this yeah now, I can see that all the parameters here are being printed nicely you know rhat is all the rhat are very close to one means convergence did have taken place that is a very good news for us. And then for each parameters if we will see how this parameter for each coefficient there will be a scale parameter in the HorseShoe Prior.

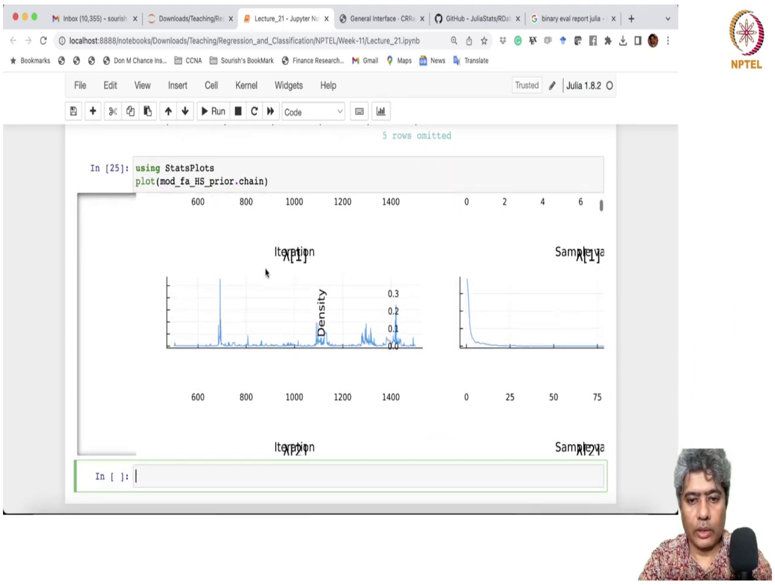
So, each scale parameter perform confidence interval are also being printed. And now what you can see that the third one as expected is also very strongly positive and the last one has a very strongly negative whereas, these ones are not you know very strongly positive or negative.

(Refer Slide Time: 22:41)

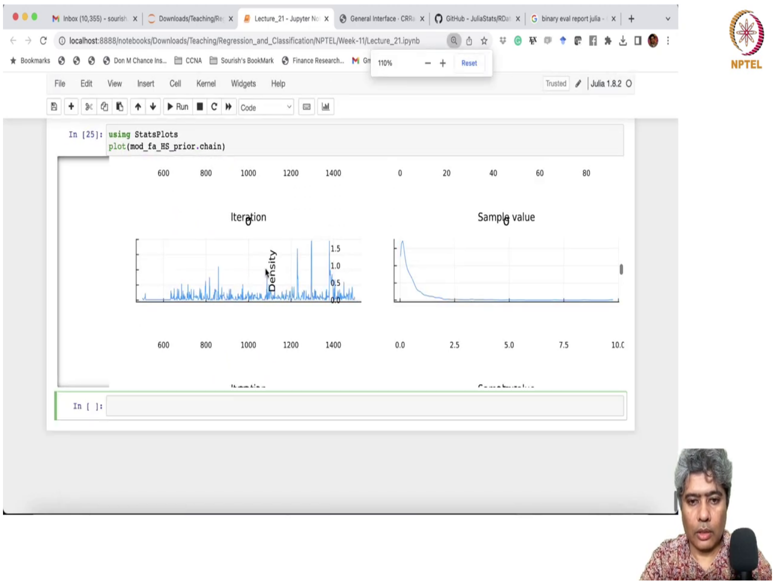


So, if I go back and if I just do the plot let me do a plotting and you will see typically HorseShoe Prior behaves a very interesting way. So, let me just try to do the plotting. So, (Refer Time: 22:46) ok let me just see if I can create this plot here HorseShoe prior chain.

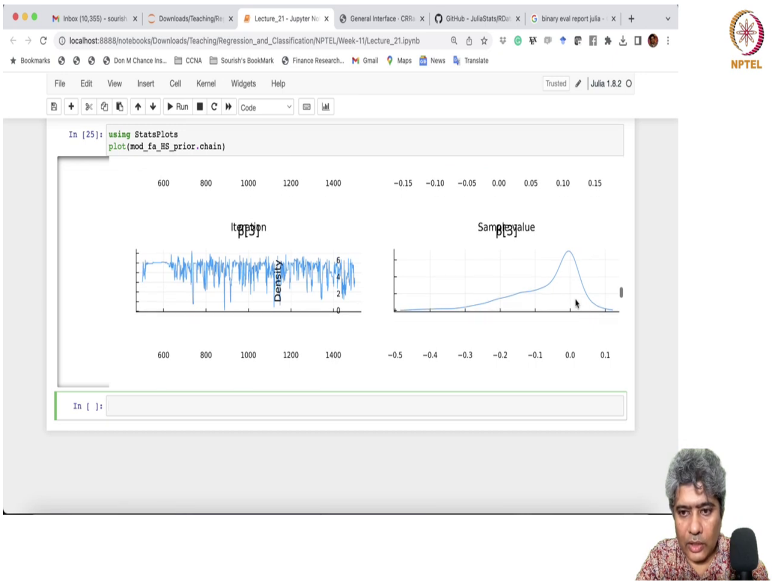
(Refer Slide Time: 23:15)



(Refer Slide Time: 23:25)

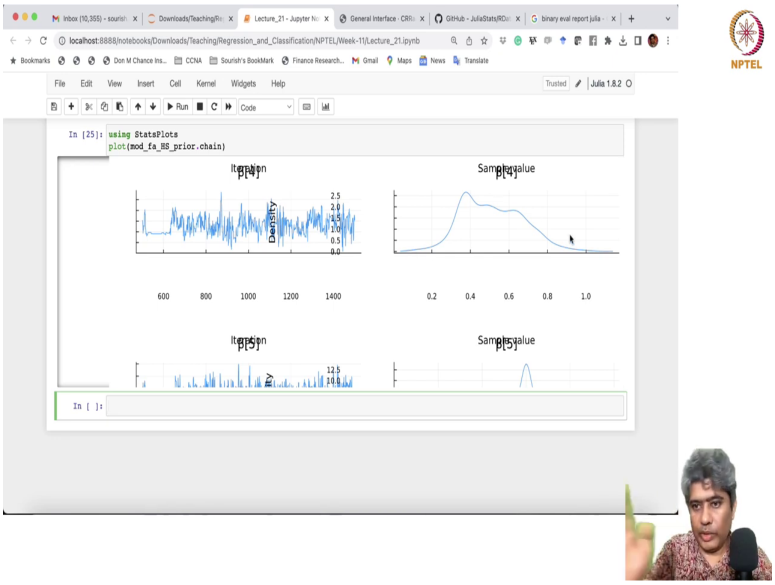


(Refer Slide Time: 23:45)

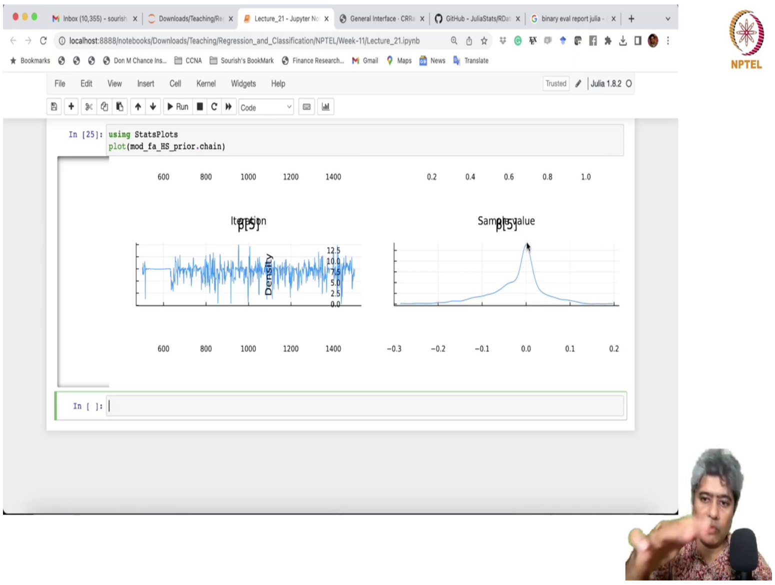


Yeah, it is better. So, first are all lambda parameters and then I will we will see here the betas. Now you can see that betas are all very close to 0 they are bit close to 0s and they are very close to sharp to the 0 and not too much. So, they see there are lot of they are not sticking and then the third one this is the second one and then this is the third one they are all positive and the entire thing is positive being shifted, ok.

(Refer Slide Time: 23:52)

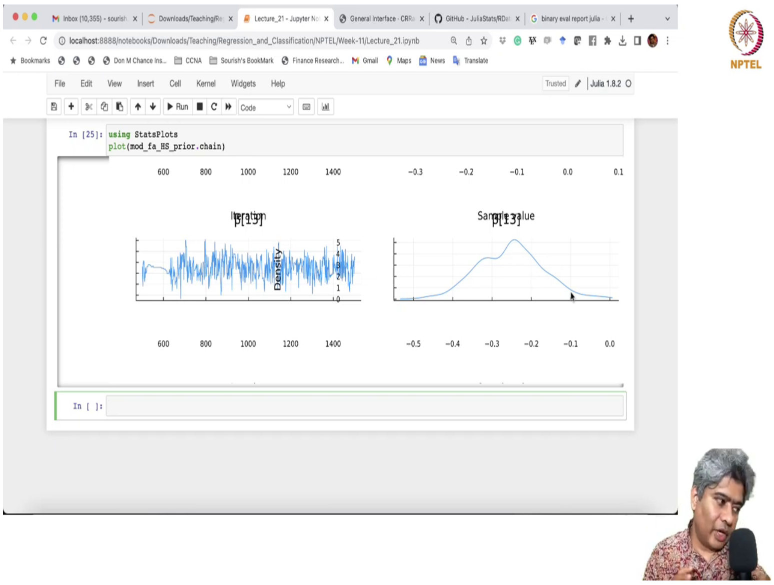


(Refer Slide Time: 23:55)



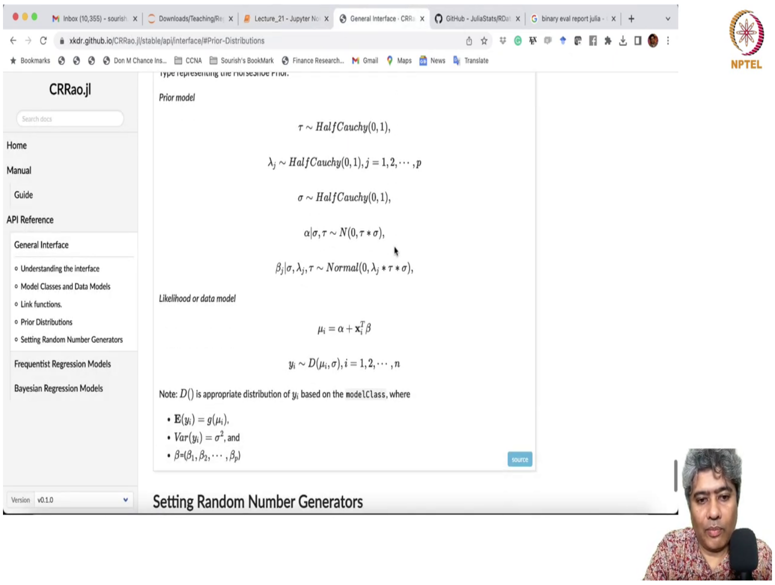
Where the others are like you know very stick to the like you know very stick to the 0 and then they have a flat distribution.

(Refer Slide Time: 24:14)



So, this is a very peculiar of the HorseShoe prior they have lot of mass on the 0 it tries to put. And similarly, but when it has a because it has a lot of mass on the tail in the prior itself when the posterior if it have any coefficient has true positive mass on the non-zero place then it gets you know it is nicely picked up that behavior also. So, you can implement HorseShoe prior in genetic you know in logistic regression very easily with just one line of code using CRRao package.

(Refer Slide Time: 24:46)



The detail about the HorseShoe Prior are given here I have given that the all these things like you know this is if D is a distribution it distribution could be normal Bernoulli anything mu i is x transpose beta.

So, you have to be mu i is in with distribution with respect to a mean and variance or standard deviation. And then beta follow Normal alpha follow Normal, but conditional normal the scale parameters are follow HalfCauchy distribution this lambda js follow HalfCauchy sigma follow HalfCauchy and the tau distribution also follow HalfCauchy.

So, with this is a very peculiar kind of you know very popular actually in the Bayesian literature and you can implement HorseShoe Prior using CRRao very easily. So, I hope you enjoyed this video and we will keep doing more such analysis using CRRaO in the coming videos.

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