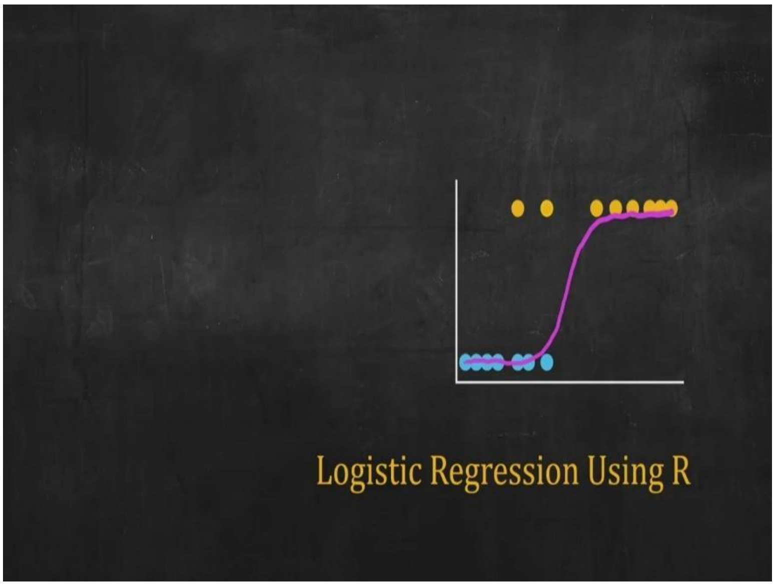
Data Analysis for Biologists
Professor Biplab Bose
Department of Bioscience & Bioengineering
Mehta Family School of Data Science & Artificial Intelligence
Indian Institute of Technology Guwahati
Lecture 36

Logistic regression using R

(Refer Slide Time: 0:29)



Logistic Regression Using R

Hello everyone, welcome back. In this lecture, we will learn to perform logistic regression using R. Logistic regression is a classification method. And today what we will do, we will try to classify tumour sample or rather images of tumour samples and we have and we are going to classify them into either malignant tumour or non-malignant tumours.

Before I go into the details of how to perform it using the logistic regression in R, let me give you a brief story behind this problem that we are dealing today. So, usually what people do, people do a biopsy for a tumour sample, for example, the data set that I will be using is a breast cancer data set. And here you have biopsy samples, and this biopsy sample pathologist will look under the microscope.

And then what they are looking for, they are looking for aberrations in the shape and morphology of cells and the types of different cells present there. Based on this information and the knowledge they have, they usually will decide, they will flag whether the tumour is a benign one, a nonmalignant one or a malignant one.

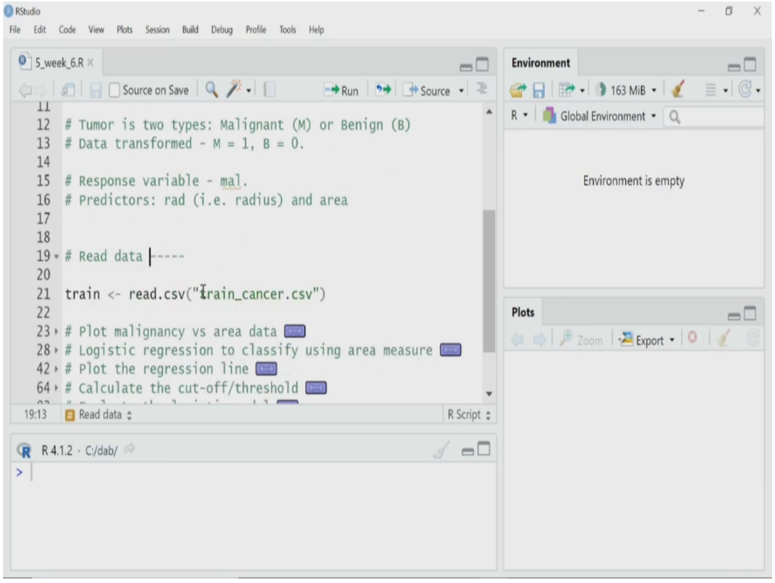
Nowadays, what are people are trying that if we can use machine learning to actually analyze these microscopic images and decide automatically whether a particular tumour sample is malignant or not. So, what I have done, I have downloaded a data set, I have given the link of that data set in the script, you can download it, it is a breast cancer tumour biopsy data set and it has lots of features.

What they have done? They have digitized these biopsy samples to microscopic images, and then from each of these digitized images, they have extracted different features, lots of them, for example, different features for morphology, area of the nucleus, radius of the nucleus, and then they have other features like gray and strict texture and all these things. And all these data are labelled, means they are labelled either as malignant or benign.

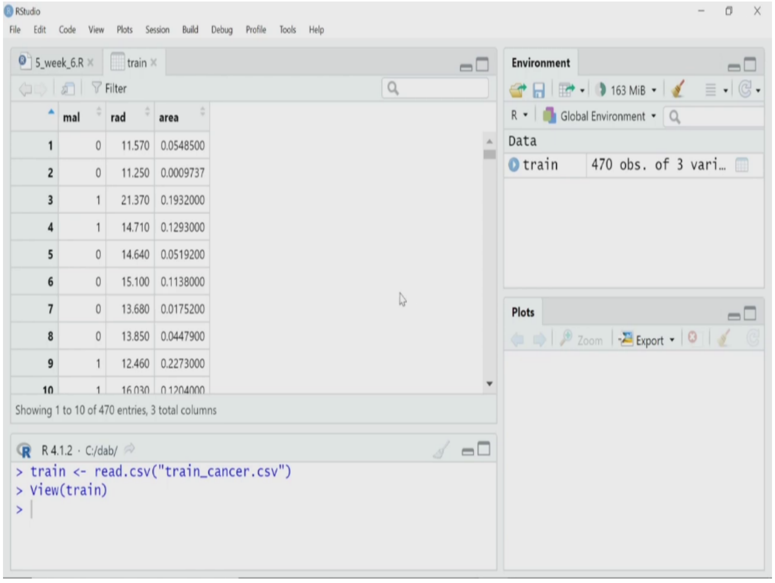
I have cleaned up the data and created a separate file a CSV file in that, and what I have kept, I have kept these malignancy label. So, here I call one as a label that means the sample is malignant, whereas, if the label is zero, that means the sample is benign or non-malignant, and I have only kept two features from the data set one is the area of the nucleus, the mean area of the nucleus of cells in that image, and the mean radius of a nucleus of cells in that image.

So, based on these two, I will try to classify tumour sample either malignant or non-malignant. Now, if you remember, during classification problem, it is what we do, we usually divide data set into a training data set and a test data set. And that is what I have done, I have randomly picked a handful of that data as a test data and kept that separate, I will create the logistic regression model classifier using the training data set only which is large in size, and then I will use it up on the test data. So, let us go into R studio and perform the analysis.

(Refer Slide Time: 3:50)



```r
# Tumor is two types: Malignant (M) or Benign (B)
# Data transformed - M = 1, B = 0.

# Response variable - mal.
# Predictors: rad (i.e. radius) and area


# Read data -----

train <- read.csv("train_cancer.csv")

# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line
# Calculate the cut-off/threshold
```



	mal	rad	area
1	0	11.570	0.0548500
2	0	11.250	0.0009737
3	1	21.370	0.1932000
4	1	14.710	0.1293000
5	0	14.640	0.0519200
6	0	15.100	0.1138000
7	0	13.680	0.0175200
8	0	13.850	0.0447900
9	1	12.460	0.2273000
10	1	16.030	0.1204000

Showing 1 to 10 of 470 entries, 3 total columns

```
> train <- read.csv("train_cancer.csv")
> View(train)
>
```

Data
train 470 obs. of 3 vari…

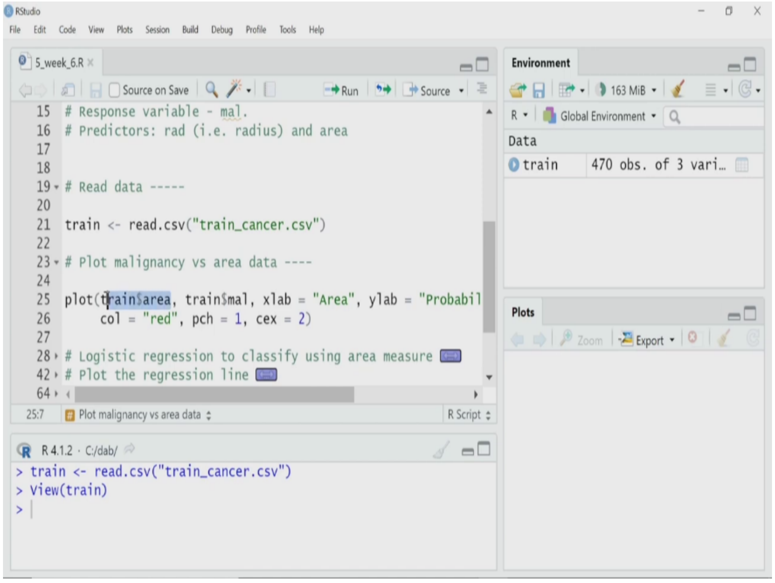
train ← read.csv(“train_cancer.csv”)

So, the first thing I will do, I will read that data and I will explain a bit. So, I have, as I said, I have separated the data into training set and test set. So, I have a CSV file, train underscore cancer dot CSV, I will read it using read dot CSV and assign that data to train variable. So, let us check the train variable data. So, it is three column and it has 470 observation and you can see the first column is mal, that is whether malignant or not, and it has the labels.

So, if it is 0, the first sample is 0, that means it is non-malignant, and when the third sample is 1, that means a malignant sample. And then the second column is the radius average radius of the nucleus in that sample and the third variable is area, or average area of nucleus of the cells in the sample. So, what I will do first I will try to classify these samples, whether they are malignant or not malignant based on the area data only.

So, I will have binary classification with respect to one single predictor that is the area. So, before I go into performing logistic regression for that, I want to plot the data so that we can understand how you want to perform the logistic regression.

(Refer Slide Time: 5:18)



```
# Response variable - mal.
# Predictors: rad (i.e. radius) and area


# Read data -----

train <- read.csv("train_cancer.csv")

# Plot malignancy vs area data ----

plot(train$area, train$mal, xlab = "Area", ylab = "Probabil
     col = "red", pch = 1, cex = 2)

# Logistic regression to classify using area measure
# Plot the regression line
```

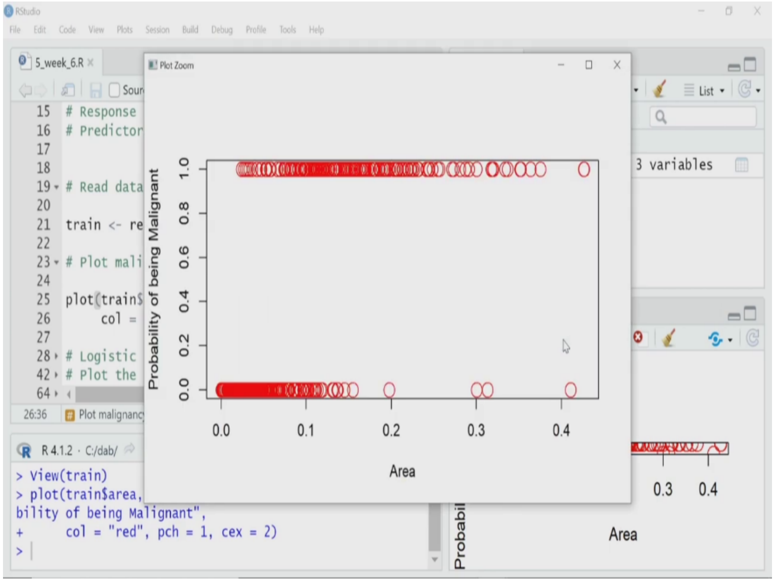
25:7 Plot malignancy vs area data R Script

R 4.1.2 · C:/dab/

```
> train <- read.csv("train_cancer.csv")
> View(train)
>
```

Data

train	470 obs. of 3 vari…



RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
5_week_6.R
Plot Zoom
Probability of being Malignant
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.1
0.2
0.3
0.4
Area
List
3 variables
Probabil

26:36 Plot malignanc

R 4.1.2 · C:/dab/

```
> View(train)
> plot(train$area,
bility of being Malignant",
+       col = "red", pch = 1, cex = 2)
>
```

plot(train$area, train$mal, xlab = “Area”, ylab = “Probability”,

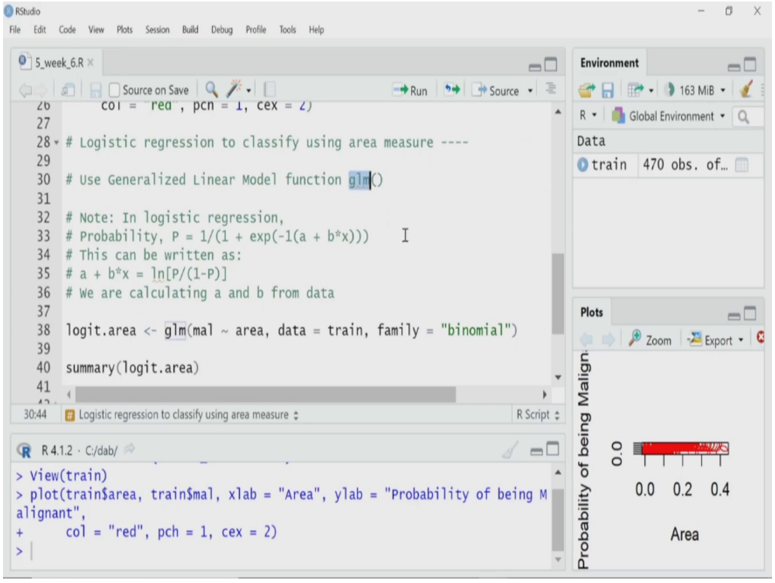
col = “red”, pch = 1, cex = 2)

So again, I am using the plot function and here it is, now, you must have learned how to plot it. So, on the horizontal axis, you want the area and the vertical axis I want the mal variable and I am labelling them and using color codes. So, let me plot it. So, if I zoom you can see, see there is some sort of segregation of data.

So, on the vertical axis, what I have done, see that sample which are labelled as non-malignant, they are 0, that means their probability of being malignant is 0. So, all those samples are along these 0 lines, 0 vertical line, whereas those samples which are labelled malignant, that means their probability of being malignant is 1. So that is why rather than putting the vertical axis as label, what I am writing here is that the vertical axis is probability of being malignant.

And that is what we do in logistic regression. If you want to recapitulate that, please go back to the lecture of logistic regression that we have done earlier. So, if you look into the data 470 data, you can see that when the area is small, most of the majority of them are non-malignant, whereas there is some overlap, but still majority of malignant tumours sample have larger area, mean area of a nucleus. So, I want to fit some sort of sigmoidal logistic curve here, and that is what I will perform using logistic regression.

(Refer Slide Time: 6:57)



```
    col = "red", pch = 1, cex = 2)

# Logistic regression to classify using area measure ----

# Use Generalized Linear Model function glm()

# Note: In logistic regression,
# Probability, P = 1/(1 + exp(-1(a + b*x)))
# This can be written as:
# a + b*x = ln[P/(1-P)]
# We are calculating a and b from data

logit.area <- glm(mal ~ area, data = train, family = "binomial")

summary(logit.area)
```

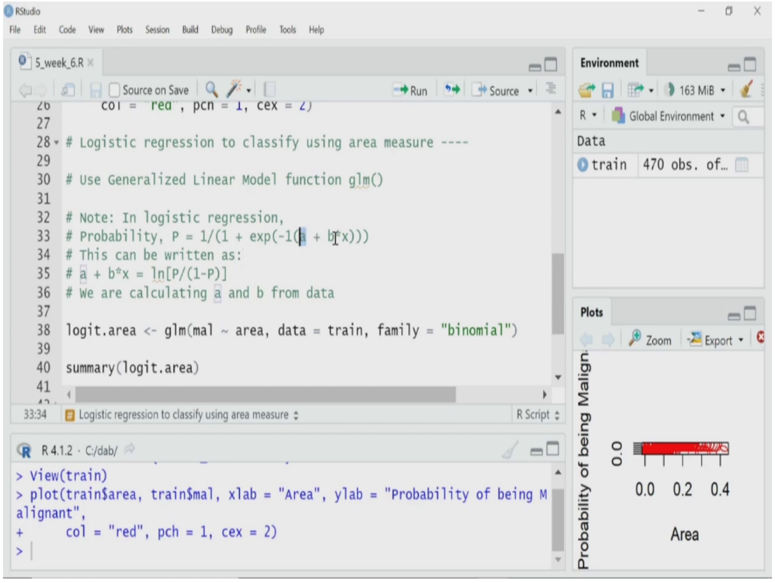
```
> View(train)
> plot(train$area, train$mal, xlab = "Area", ylab = "Probability of being M
alignant",
+      col = "red", pch = 1, cex = 2)
> 
```

logit.area ← glm(mal ~ area, data = train, family = “binomial”)

So, let me go back and now perform the logistic regression model. To perform logistic regression, what I will do I will use the GLM function, this is called generalized linear model function. And I will ask it to perform logistic regression. Now, just to remind you, in logistic regression, what we have?

We have an equation which is called logistic equation which is equal to something like this P the probability, in this case the probability of being malignant is equal to 1 divided by in denominator you have 1 plus e to the power minus a plus b into x. So, this one is actually my logistic equation and I want to fit this to the data.

(Refer Slide Time: 7:51)



```
      col = "red", pch = 1, cex = 2)

# Logistic regression to classify using area measure ----

# Use Generalized Linear Model function glm()

# Note: In logistic regression,
# Probability, P = 1/(1 + exp(-1(a + b*x)))
# This can be written as:
# a + b*x = ln[P/(1-P)]
# We are calculating a and b from data

logit.area <- glm(mal ~ area, data = train, family = "binomial")

summary(logit.area)
```

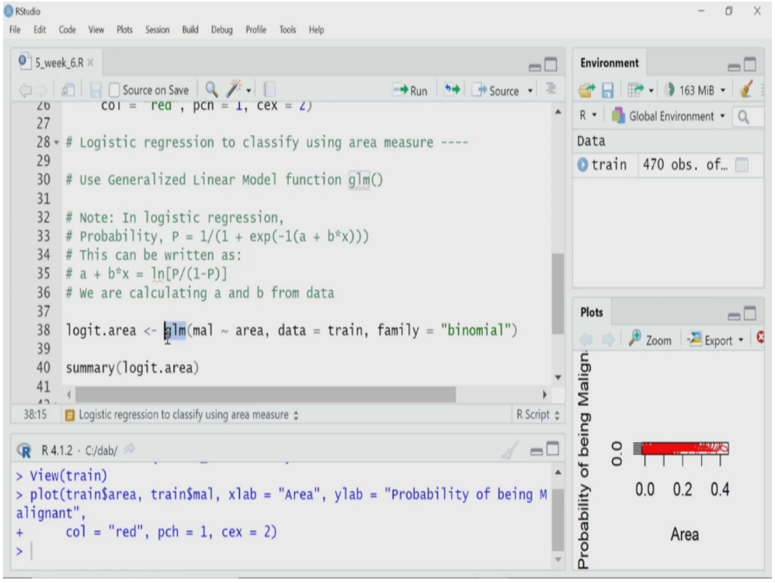
```
> View(train)
> plot(train$area, train$mal, xlab = "Area", ylab = "Probability of being M
alignant",
+       col = "red", pch = 1, cex = 2)
> |
```

logit.area ← glm(mal ~ area, data = train, family = “binomial”)

summary(logit.area)

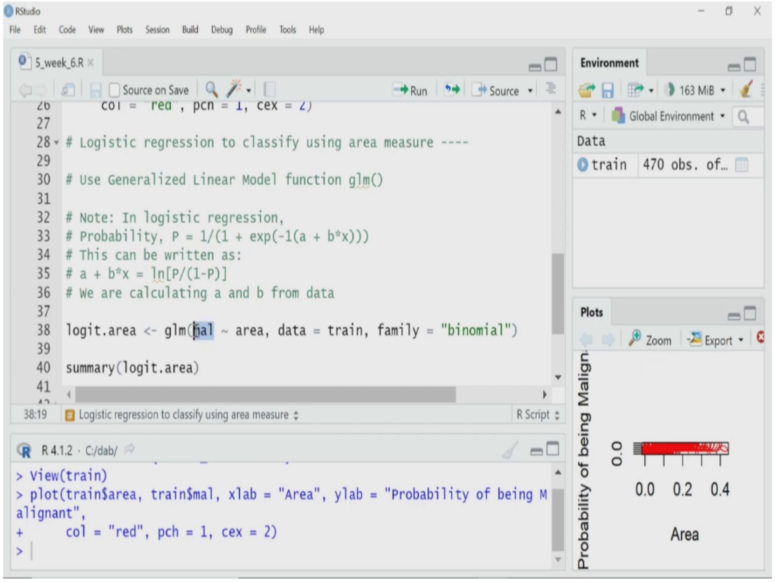
And what is unknown here, I have to calculate this a and b. If you do some sort of algebra, you can easily rearrange the term and you can see that it will become a equal to, a plus b into x equal to ln of P divided by 1 minus P, P divided by 1 minus P is the odds, so, a plus b into x is equal to log of or ln of odd. So, I have to calculate this value of a and b from that data.

(Refer Slide Time: 8:24)



```
      col = "red", pch = 1, cex = 2)

# Logistic regression to classify using area measure ----

# Use Generalized Linear Model function glm()

# Note: In logistic regression,
# Probability, P = 1/(1 + exp(-1(a + b*x)))
# This can be written as:
# a + b*x = ln[P/(1-P)]
# We are calculating a and b from data

logit.area <- glm(mal ~ area, data = train, family = "binomial")

summary(logit.area)
```

```
> View(train)
> plot(train$area, train$mal, xlab = "Area", ylab = "Probability of being M
alignant",
+       col = "red", pch = 1, cex = 2)
> |
```

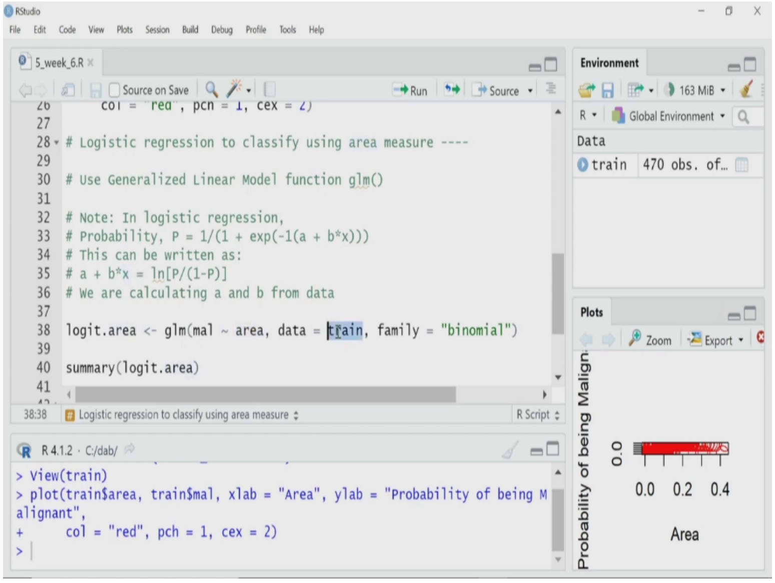


```
    col = "red", pch = 1, cex = 2)

# Logistic regression to classify using area measure ----

# Use Generalized Linear Model function glm()

# Note: In logistic regression,
# Probability, P = 1/(1 + exp(-1(a + b*x)))
# This can be written as:
# a + b*x = ln[P/(1-P)]
# We are calculating a and b from data

logit.area <- glm(mal ~ area, data = train, family = "binomial")

summary(logit.area)
```

```
> View(train)
> plot(train$area, train$mal, xlab = "Area", ylab = "Probability of being M
alignant",
+       col = "red", pch = 1, cex = 2)
>
```

So, how do I do that? As I said I will use the GLM function, generalized linear model function. And I have to define certain arguments for that. Just like the linear model the first thing I have to define, I have to define the model. So, what I have to define I have to define in that case what is the dependent variable and what is the independent variable. So, the MAL that is my dependent variable or response variable and after tilde I have written area, area variable is the independent one. So, area is the x in this equation.

(Refer Slide Time: 8:56)

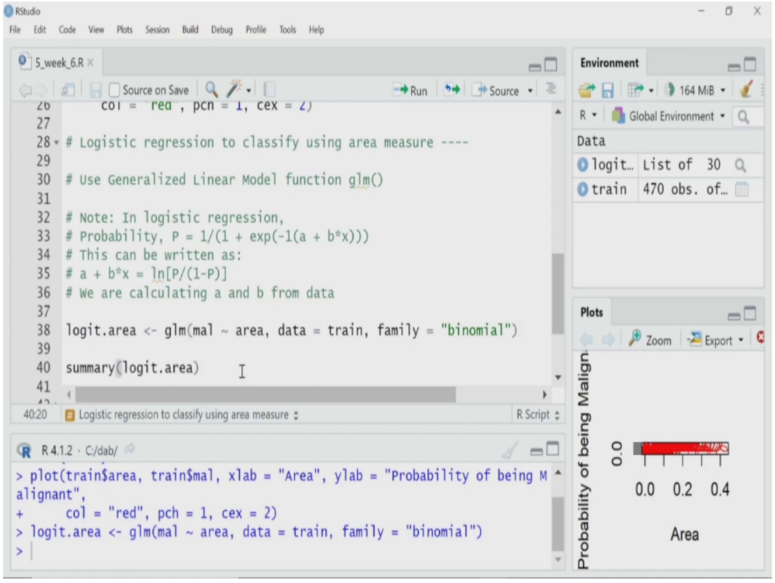


```
    col = "red", pch = 1, cex = 2)

# Logistic regression to classify using area measure ----

# Use Generalized Linear Model function glm()

# Note: In logistic regression,
# Probability, P = 1/(1 + exp(-1(a + b*x)))
# This can be written as:
# a + b*x = ln[P/(1-P)]
# We are calculating a and b from data

logit.area <- glm(mal ~ area, data = train, family = "binomial")

summary(logit.area)
```

```
> View(train)
> plot(train$area, train$mal, xlab = "Area", ylab = "Probability of being M
alignant",
+       col = "red", pch = 1, cex = 2)
>
```

What will be the data? Data will be my training data set that I have stored in train, and by family I am written binomial. So, by writing family equal to binomial you are telling GLM algorithm, GLM function, that see, you have to perform logistic regression. And this whole result of logistic regression will be assigned and stored at in the model called logit dot area.

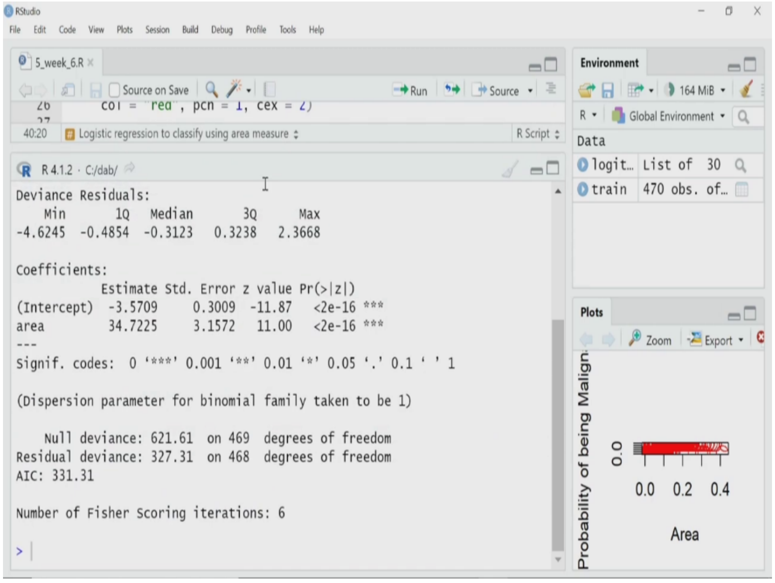
(Refer Slide Time: 9:20)


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
5_week_6.R
Source on Save
Run
Source
40:20
Logistic regression to classify using area measure
R Script
R 4.1.2 · C:/dab/
Environment
164 MiB
Global Environment
Plots
Zoom
Export
Probability of being Malign
0.0
0.0
0.2
0.4
Area

```
  col = "red", pch = 1, cex = 2)

# Logistic regression to classify using area measure ----

# Use Generalized Linear Model function glm()

# Note: In logistic regression,
# Probability, P = 1/(1 + exp(-1(a + b*x)))
# This can be written as:
# a + b*x = ln[P/(1-P)]
# We are calculating a and b from data

logit.area <- glm(mal ~ area, data = train, family = "binomial")

summary(logit.area)
```

```
> plot(train$area, train$mal, xlab = "Area", ylab = "Probability of being M
alignant",
+      col = "red", pch = 1, cex = 2)
> logit.area <- glm(mal ~ area, data = train, family = "binomial")
>
```

Data
logit... List of 30
train 470 obs. of...

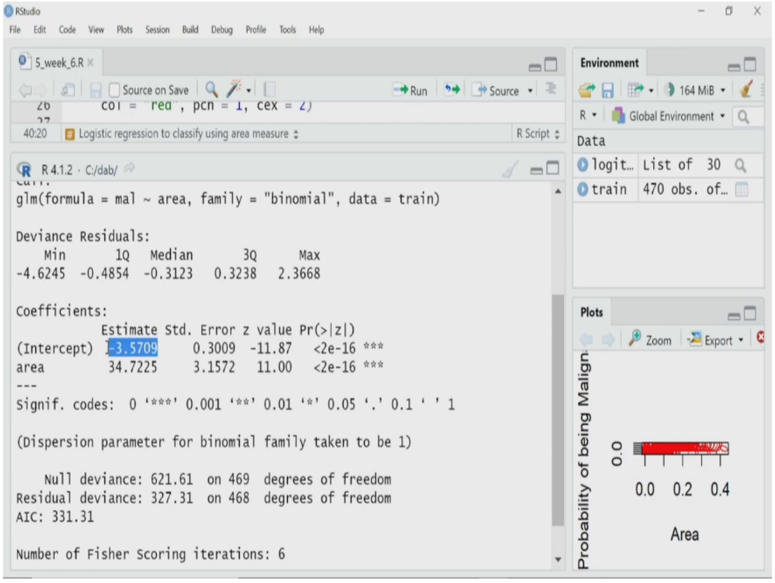

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
5_week_6.R
Source on Save
Run
Source
40:20
Logistic regression to classify using area measure
R Script
R 4.1.2 · C:/dab/
Environment
164 MiB
Global Environment
Plots
Zoom
Export
Probability of being Malign
0.0
0.0
0.2
0.4
Area

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.6245  -0.4854  -0.3123   0.3238   2.3668

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.5709     0.3009  -11.87   <2e-16 ***
area         34.7225     3.1572   11.00   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 327.31  on 468  degrees of freedom
AIC: 331.31

Number of Fisher Scoring iterations: 6

>
```

Data
logit... List of 30
train 470 obs. of...

So, if I execute it, I have executed it. And now, let me check the summary of the summary output of that. So, just like any linear regression model, linear model we have created here, here also we have the column for estimated values of coefficient, then statistical test data and I will focus on those.

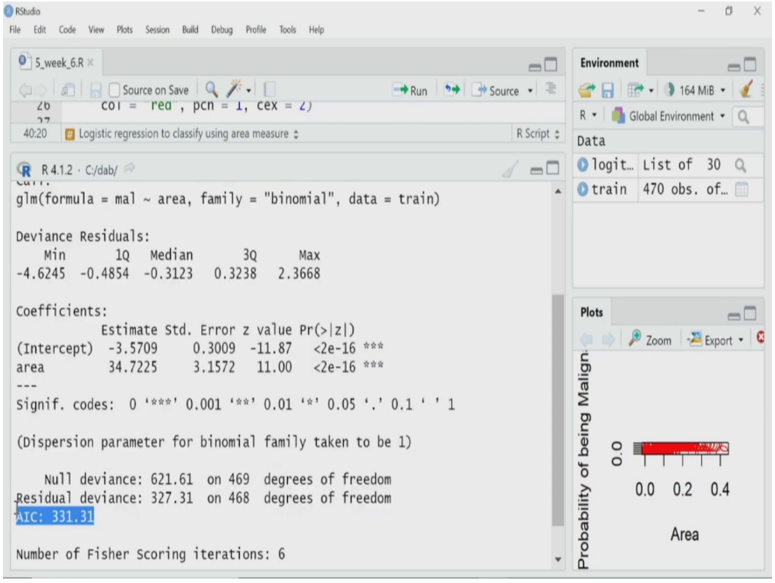
(Refer Slide Time: 9:45)



```
glm(formula = mal ~ area, family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.6245  -0.4854  -0.3123   0.3238   2.3668

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.5709     0.3009  -11.87   <2e-16 ***
area         34.7225     3.1572   11.00   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 327.31  on 468  degrees of freedom
AIC: 331.31

Number of Fisher Scoring iterations: 6
```

So, here in the first column you have the estimated variable. So, the intercept a is equal to minus 3.57, it has estimated and it has done a statistical test and the probability is very low, 2 e minus 16. So, that means this intercept is statistically significant one, the coefficient for area the b, in my equation is equal to 34.7225 and it is also statistically significant. So that is good enough for me to proceed.

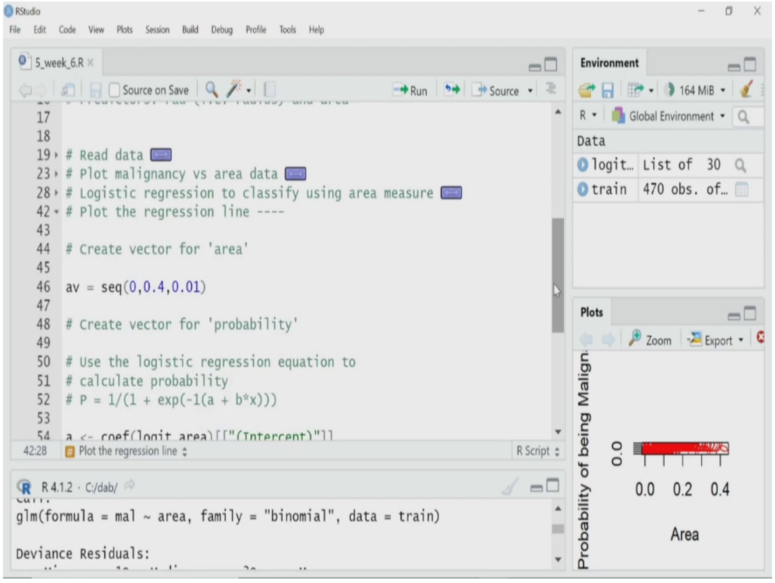
(Refer Slide Time: 10:18)



```
glm(formula = mal ~ area, family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.6245  -0.4854  -0.3123   0.3238   2.3668

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.5709     0.3009  -11.87   <2e-16 ***
area         34.7225     3.1572   11.00   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 327.31  on 468  degrees of freedom
AIC: 331.31

Number of Fisher Scoring iterations: 6
```

There is another interesting thing that it reports, it is called Akaike information criteria, AIC, I will come to that later on, when I use that for model comparison. So, this is what we have got. So that means I can say that it has performed logistic regression now, the both the coefficient should be there, they are statistically significant. So now, I would like to plot this model on an overlay on the original data.

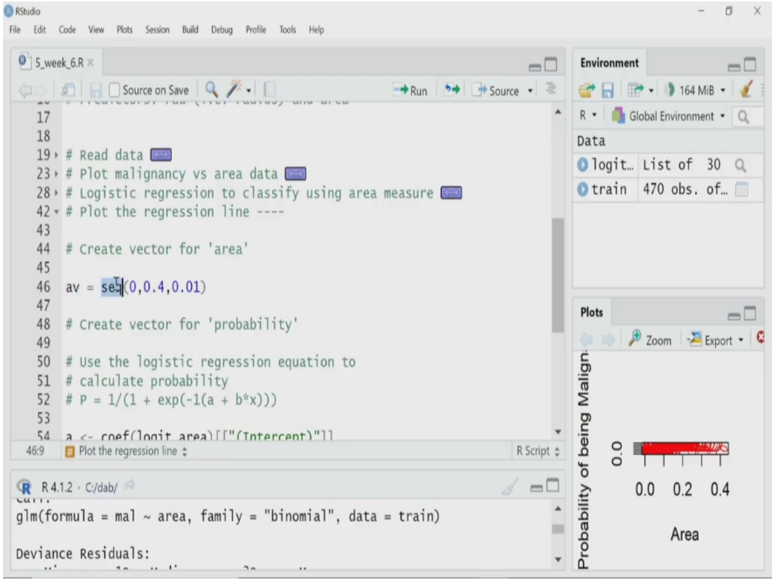
(Refer Slide Time: 10:42)



```
# Read data
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit_area)[["(Intercept)"]]
```

```
glm(formula = mal ~ area, family = "binomial", data = train)

Deviance Residuals:
```

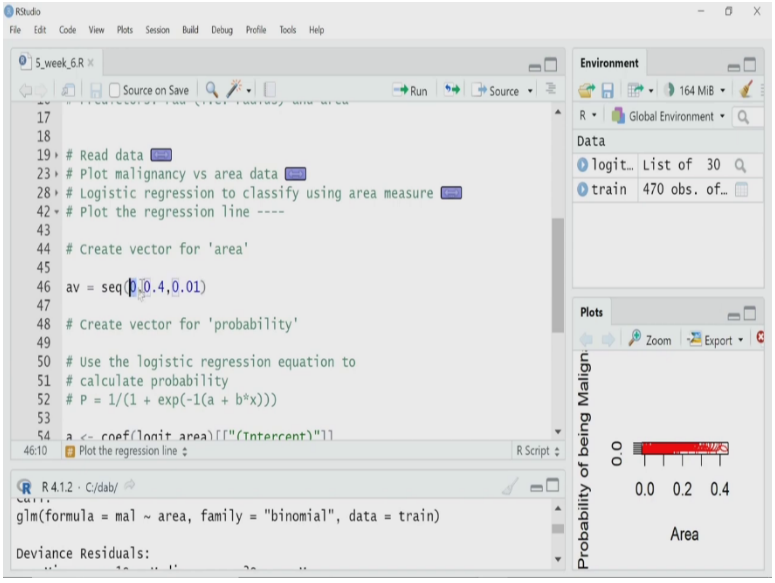
logit…	List of	30
train	470 obs. of…	



```
# Read data
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit_area)[["(Intercept)"]]
```

```
glm(formula = mal ~ area, family = "binomial", data = train)

Deviance Residuals:
```

logit…	List of	30
train	470 obs. of…	

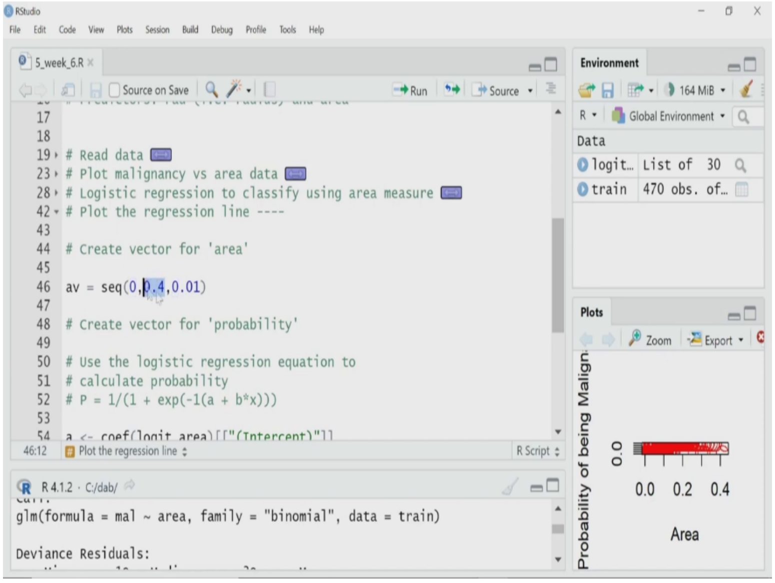


```
# Read data
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit_area)[["(Intercept)"]]
```

```
glm(formula = mal ~ area, family = "binomial", data = train)

Deviance Residuals:
```

Data

logit…	List of 30
train	470 obs. of…



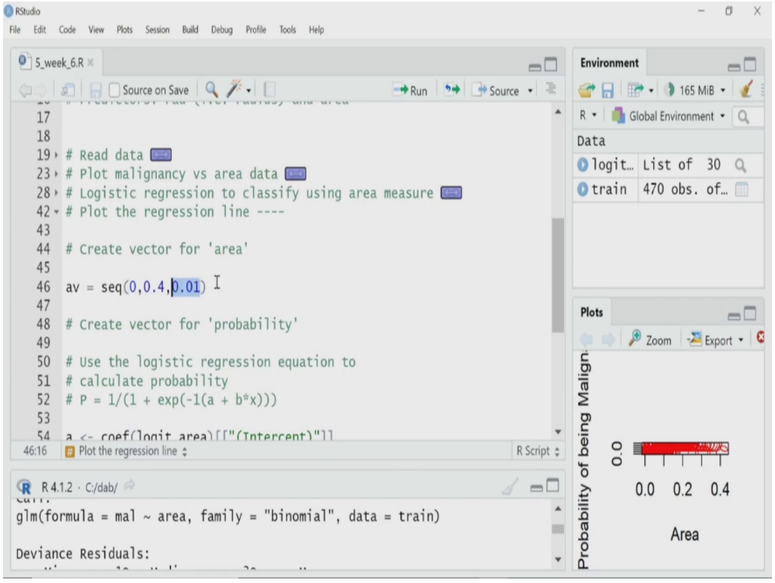
```
# Read data
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit_area)[["(Intercept)"]]
```

```
glm(formula = mal ~ area, family = "binomial", data = train)

Deviance Residuals:
```

av = seq(0, 0.4, 0.01)

To do that, I will again have to create vectors, and I have to create the call the predict function. So, what I will do? I will create a vector for input data. So, area is the input the independent variables, I am creating av vector using the sequence function, where I am setting a data set, creating a data set, starting from 0 to 0.4, why 0.4?

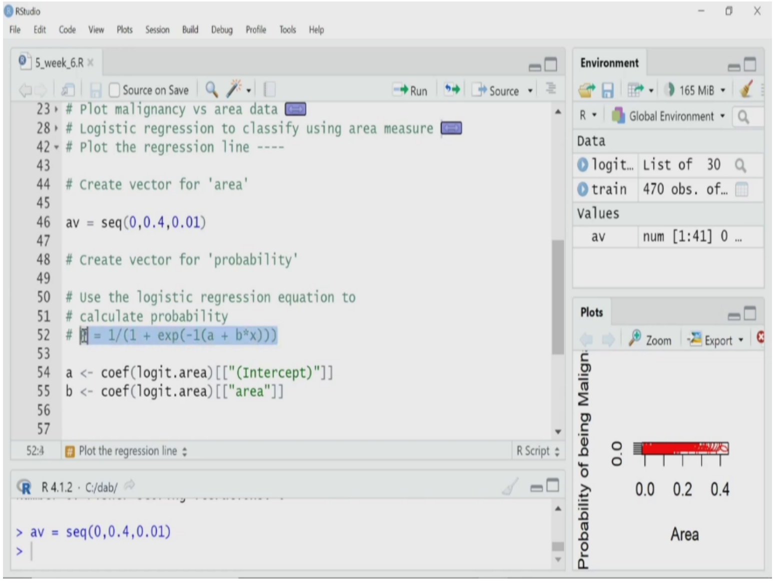
(Refer Slide Time: 11:08)



```
# Read data
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
```

```
glm(formula = mal ~ area, family = "binomial", data = train)

Deviance Residuals:
```

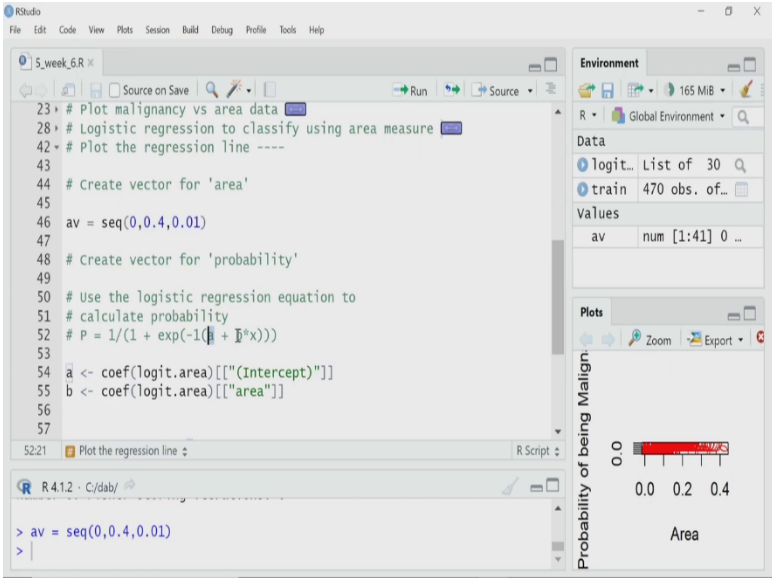
Data
logit… List of 30
train 470 obs. of…



```
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

```
> av = seq(0,0.4,0.01)
>
```

Data
logit… List of 30
train 470 obs. of…
Values
av num [1:41] 0 …



```r
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

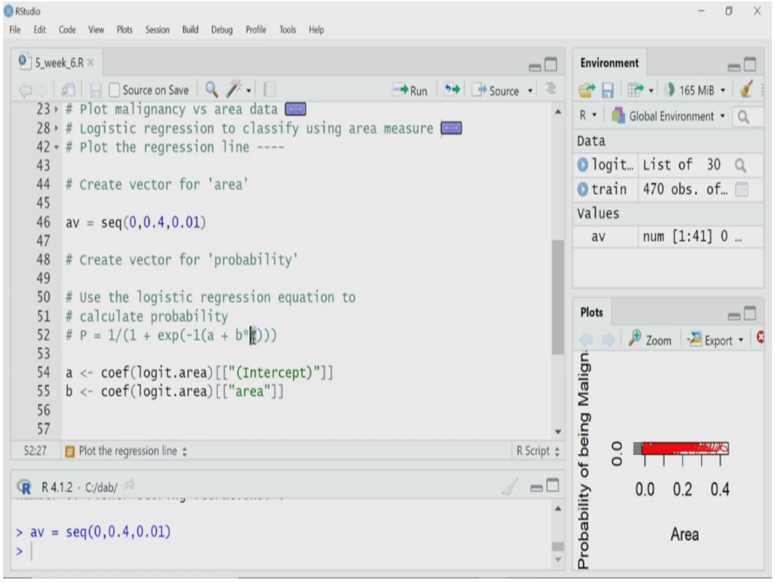
```
> av = seq(0,0.4,0.01)
>
```

Data

logit…	List of 30
train	470 obs. of…

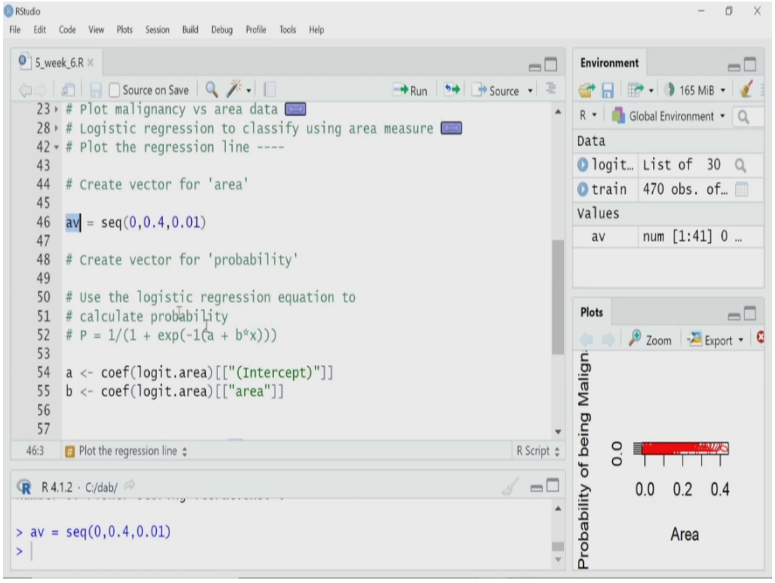
Values

av	num [1:41] 0 …



```r
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

```
> av = seq(0,0.4,0.01)
>
```

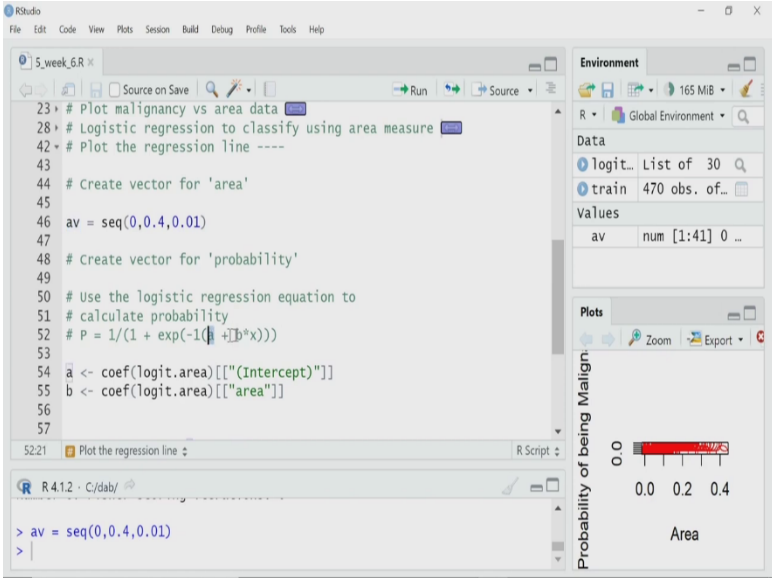


```r
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

```
> av = seq(0,0.4,0.01)
>
```

Let us zoom into the figure, see, we have data up to around 0.4, something like that. So, I want to create a list of data or vector of from 0 to 0.4 and with the increment of 0.01. So, I create av. Now, I will use this av, and I want to calculate the P. So, this x in this equation, this is my logistic equation, I want to calculate P, I know the value of a and b, I calculated them by a logistic regression, and this x. For x, I have decided different value and stored that here in av.

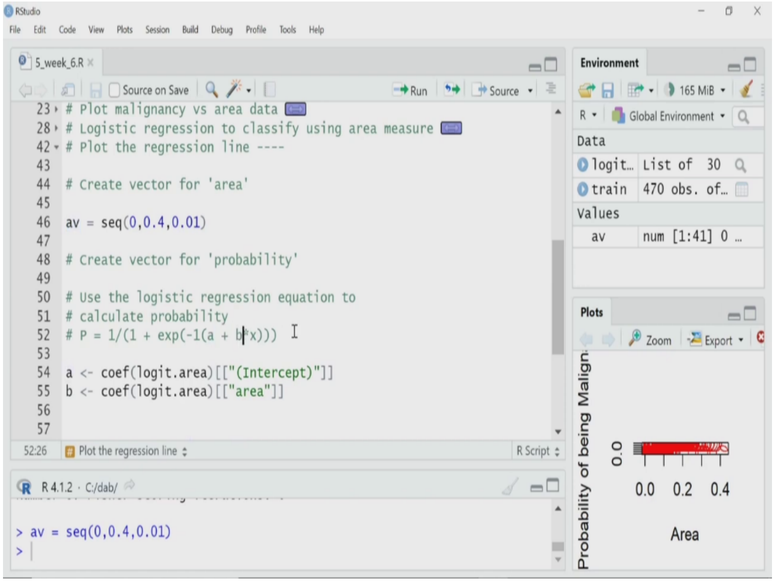
(Refer Slide Time: 11:48)



```
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a +b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

```
> av = seq(0,0.4,0.01)
>
```

```
Data
logit… List of 30
train  470 obs. of…
Values
av     num [1:41] 0 …
```



```
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

```
> av = seq(0,0.4,0.01)
>
```

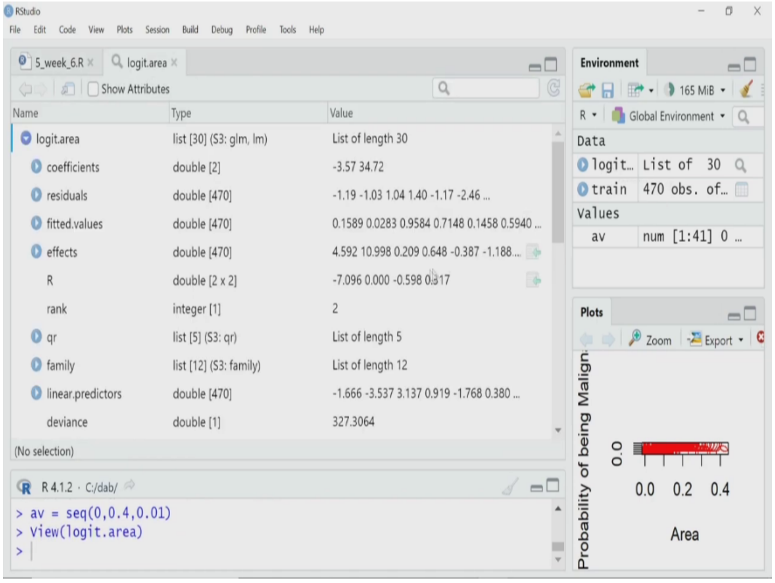
```
Data
logit… List of 30
train  470 obs. of…
Values
av     num [1:41] 0 …
```

a ⟵ coef(logit.area)[[“(Intercept)”]]

b ⟵ coef(logit.area)[[“area”]]

So first, I have to get the a and b I could have copied and pasted, but I have tried to generalize it so that if the value of a and b changes for regression to regression still, we can use the script.

(Refer Slide Time: 11:59)

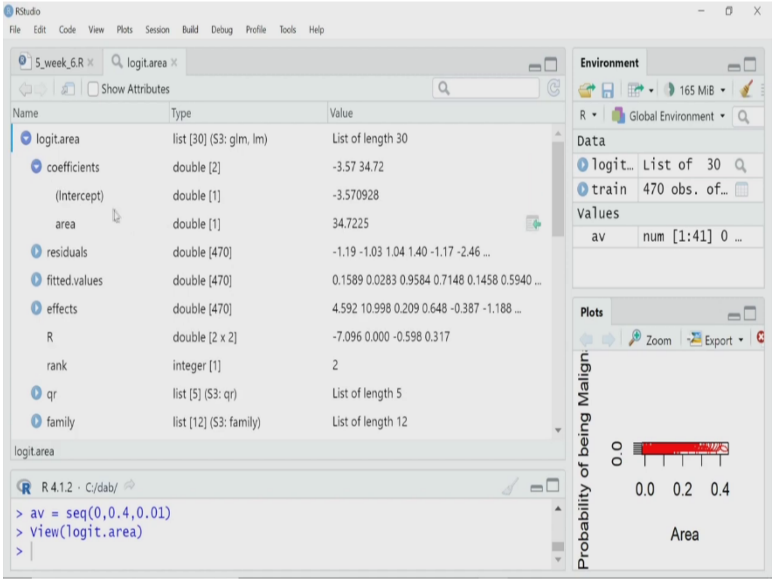


Name	Type	Value
logit.area	list [30] (S3: glm, lm)	List of length 30
coefficients	double [2]	-3.57 34.72
residuals	double [470]	-1.19 -1.03 1.04 1.40 -1.17 -2.46 ...
fitted.values	double [470]	0.1589 0.0283 0.9584 0.7148 0.1458 0.5940 ...
effects	double [470]	4.592 10.998 0.209 0.648 -0.387 -1.188 ...
R	double [2 x 2]	-7.096 0.000 -0.598 0.317
rank	integer [1]	2
qr	list [5] (S3: qr)	List of length 5
family	list [12] (S3: family)	List of length 12
linear.predictors	double [470]	-1.666 -3.537 3.137 0.919 -1.768 0.380 ...
deviance	double [1]	327.3064

(No selection)

```
> av = seq(0,0.4,0.01)
> View(logit.area)
>
```

Data
logit... List of 30
train 470 obs. of...
Values
av num [1:41] 0 ...

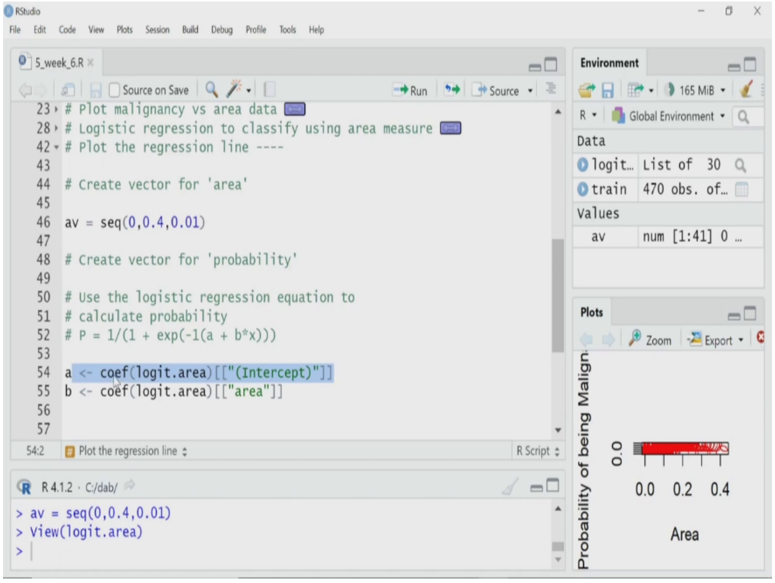


Name	Type	Value
logit.area	list [30] (S3: glm, lm)	List of length 30
coefficients	double [2]	-3.57 34.72
(Intercept)	double [1]	-3.570928
area	double [1]	34.7225
residuals	double [470]	-1.19 -1.03 1.04 1.40 -1.17 -2.46 ...
fitted.values	double [470]	0.1589 0.0283 0.9584 0.7148 0.1458 0.5940 ...
effects	double [470]	4.592 10.998 0.209 0.648 -0.387 -1.188 ...
R	double [2 x 2]	-7.096 0.000 -0.598 0.317
rank	integer [1]	2
qr	list [5] (S3: qr)	List of length 5
family	list [12] (S3: family)	List of length 12

logit.area

```
> av = seq(0,0.4,0.01)
> View(logit.area)
>
```

Data
logit... List of 30
train 470 obs. of...
Values
av num [1:41] 0 ...



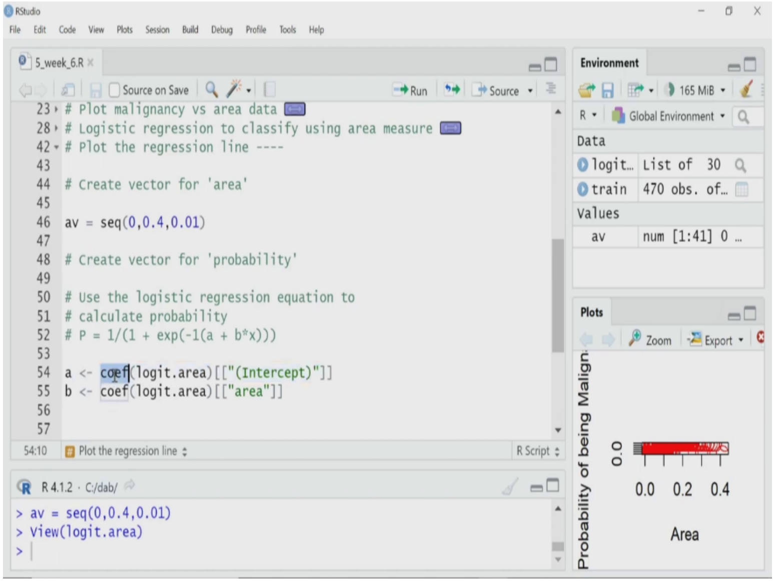
```
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

```
> av = seq(0,0.4,0.01)
> View(logit.area)
>
```

```
Data
logit… List of 30
train  470 obs. of…
Values
av     num [1:41] 0 …
```

So, what I am doing, see this logit model, if you look into it has lots of objects. So, in the coefficient, if you call, see in the coefficient, there are two variables, named variable, intercept and area. So, using the coefficient function, coef function, actually, I can fetch this data from my model, and it will fetch both this thing intercept and area and then again, I can from that, fetch intercept separately and area separately.

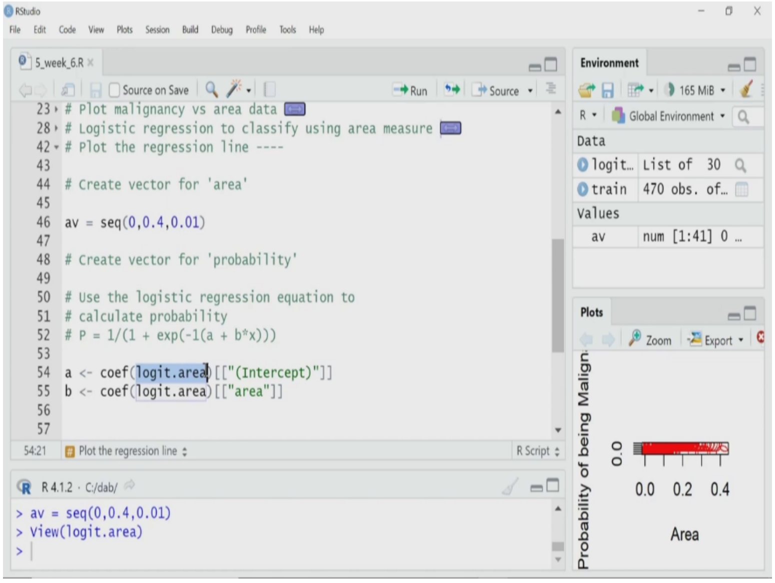
(Refer Slide Time: 12:30)



```
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

```
> av = seq(0,0.4,0.01)
> View(logit.area)
>
```

```
Data
logit… List of 30
train  470 obs. of…
Values
av     num [1:41] 0 …
```



```
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

54:21 Plot the regression line R Script

R 4.1.2 · C:/dab/

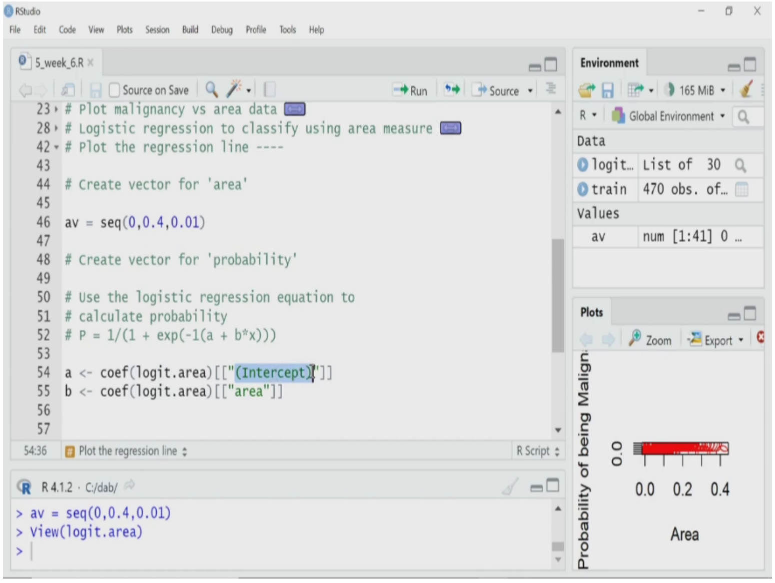
```
> av = seq(0,0.4,0.01)
> View(logit.area)
>
```

Data

logit...	List of 30
train	470 obs. of...

Values

av	num [1:41] 0 ...



```
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

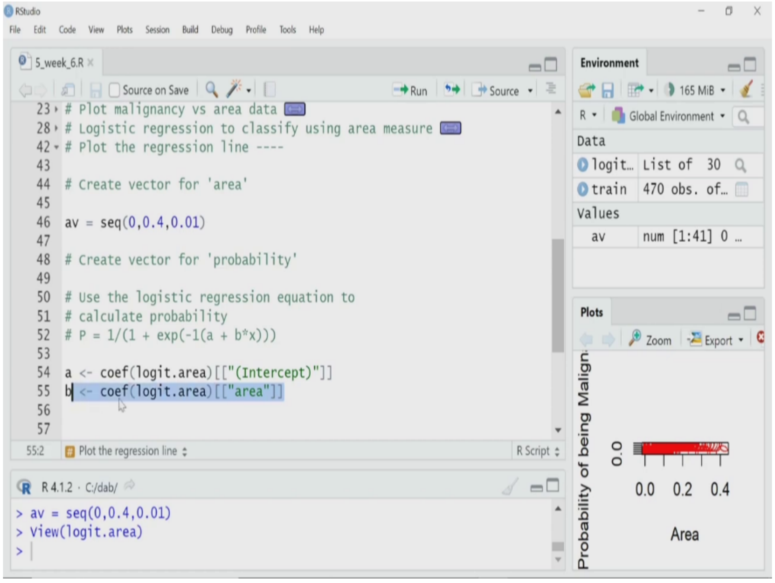
54:36 Plot the regression line R Script

R 4.1.2 · C:/dab/

```
> av = seq(0,0.4,0.01)
> View(logit.area)
>
```

So that is what I am doing here in this line of the script. I am using coef function to get the coefficient from my logistic regression model that I just created. And from that I am fetching only the data for the intercept variable. And it has written the name as in the round bracket that is why I have to write the round bracket also here within the apostrophe.

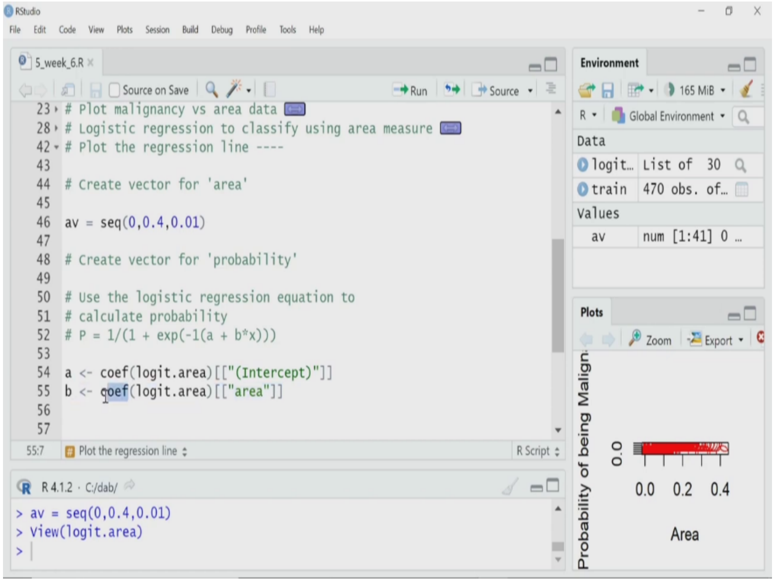
(Refer Slide Time: 12:52)



```
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

```
> av = seq(0,0.4,0.01)
> View(logit.area)
>
```

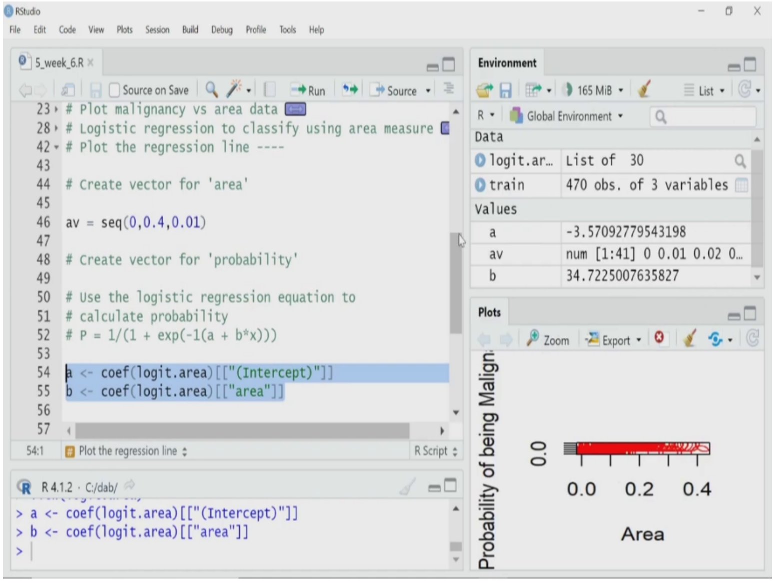
Data	
logit…	List of 30
train	470 obs. of…
Values	
av	num [1:41] 0 …



```
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

```
> av = seq(0,0.4,0.01)
> View(logit.area)
>
```

Data	
logit…	List of 30
train	470 obs. of…
Values	
av	num [1:41] 0 …



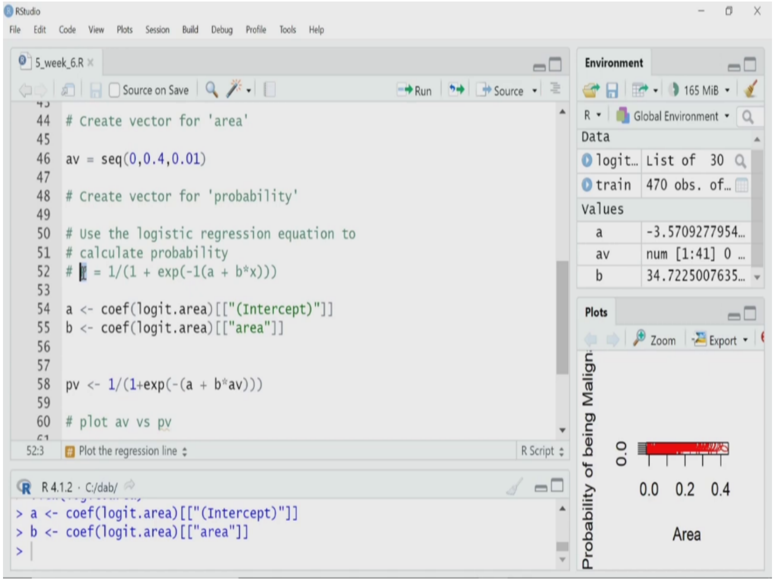
```
# Plot malignancy vs area data
# Logistic regression to classify using area measure
# Plot the regression line ----

# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]
```

```
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
>
```

Data	
logit.ar…	List of 30
train	470 obs. of 3 variables
Values	
a	-3.57092779543198
av	num [1:41] 0 0.01 0.02 0…
b	34.7225007635827

And the second line, I am facing data for b. How I am doing it? Again, I am using the coefficient function and applying it on my logistic model that I have created. And I am fetching only the area data. So, if I execute both of those, I get the value of a and b. Here, you can see a is minus 3.5, b is 34.7.

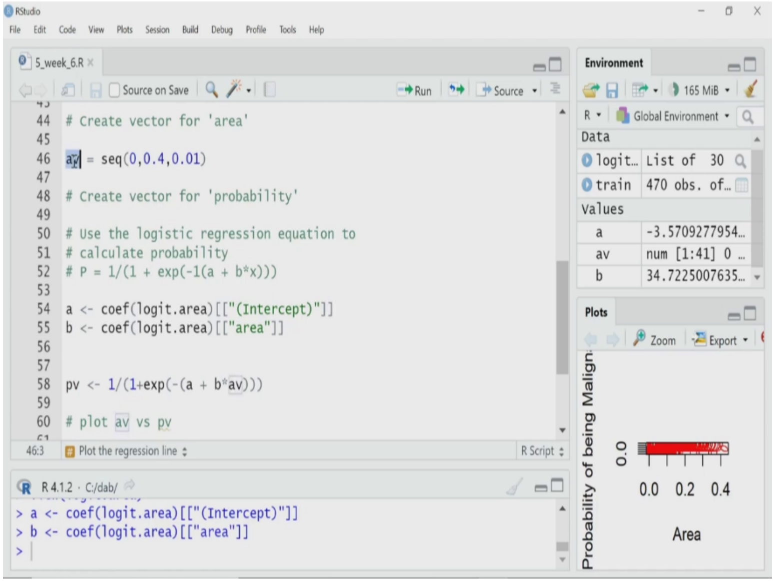
(Refer Slide Time: 13:16)



```
# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv
```

```
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
>
```

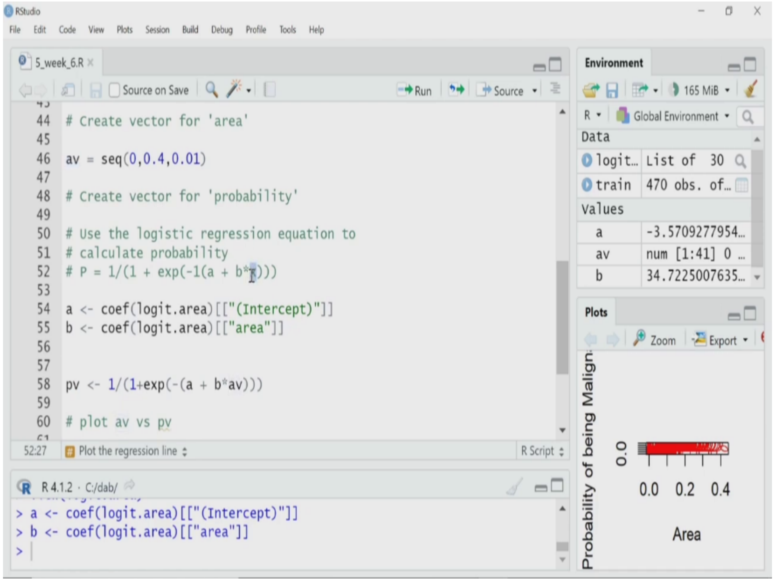
```
Data
logit…  List of  30
train   470 obs. of…
Values
a       -3.5709277954…
av      num [1:41] 0 …
b       34.7225007635…
```



```
# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv
```

```
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
>
```

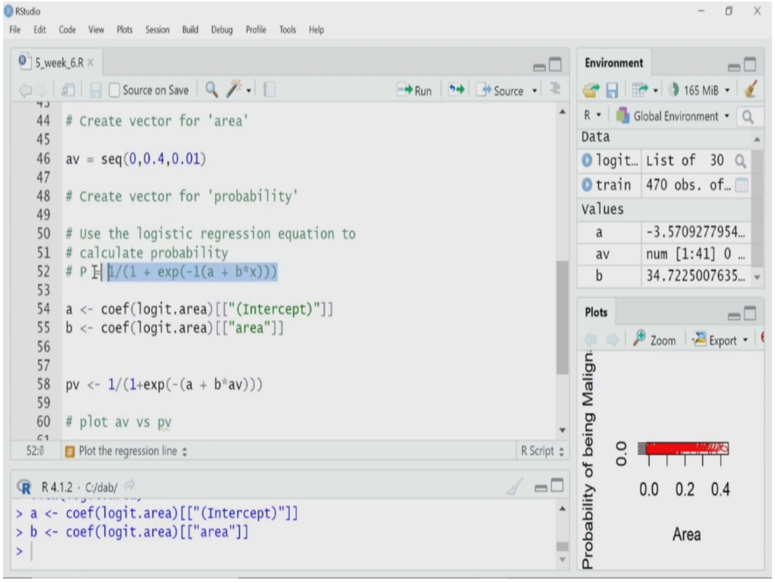
```
Data
logit…  List of  30
train   470 obs. of…
Values
a       -3.5709277954…
av      num [1:41] 0 …
b       34.7225007635…
```



```
# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv
```

```
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
>
```

```
Data
logit...  List of 30
train     470 obs. of...
Values
a         -3.5709277954...
av        num [1:41] 0 ...
b         34.7225007635...
```



```
# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv
```

```
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
>
```

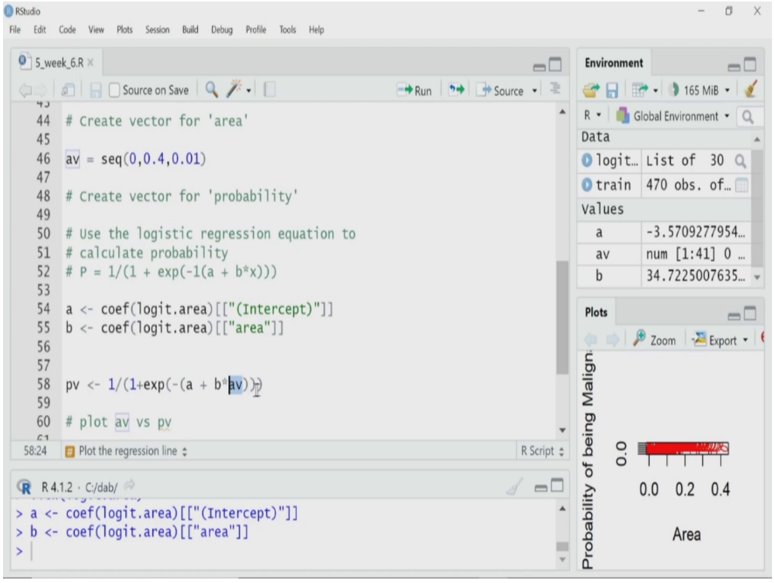
```
Data
logit...  List of 30
train     470 obs. of...
Values
a         -3.5709277954...
av        num [1:41] 0 ...
b         34.7225007635...
```

pv ← 1/(1 – exp(-(a + b*av)))

So, now, I have got a and b. So, what I will do? I will create a vector, which will store the value of P, and I will call that pv, and how I am doing that? So, I have I know the value of a and b, so I have to give each value of this a v vector as input as x and calculate the corresponding value of p using this equation. I can do that one by one.

But I have almost 40 data points. So that means I have to calculate these 40 times, I do not want to do that. R has the advantage that it can actually repeat these on a vector.

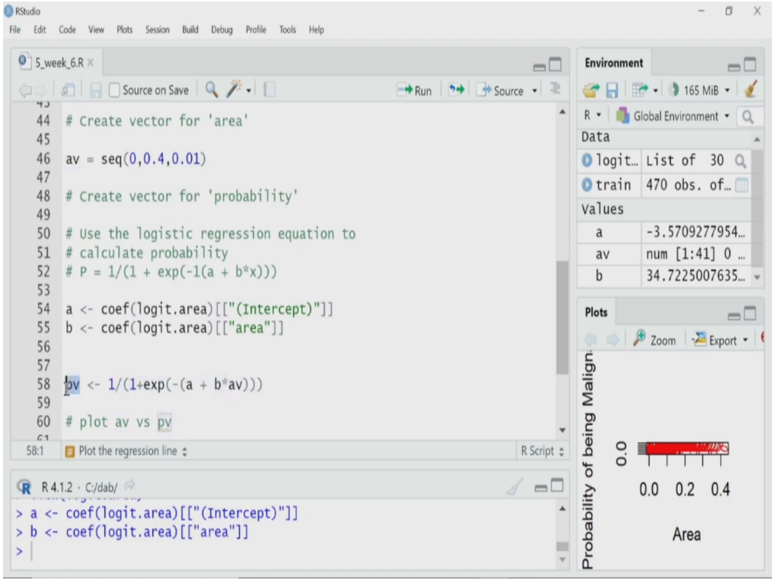
(Refer Slide Time: 13:58)



```
# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv
```

```
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
>
```

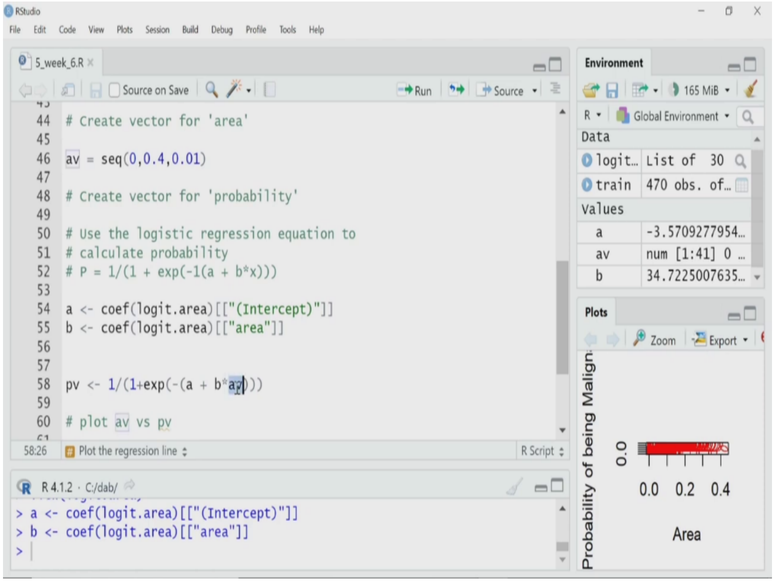
Data		
logit...	List of 30	
train	470 obs. of...	
Values		
a	-3.5709277954...	
av	num [1:41] 0 ...	
b	34.7225007635...	



```
# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv
```

```
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
>
```

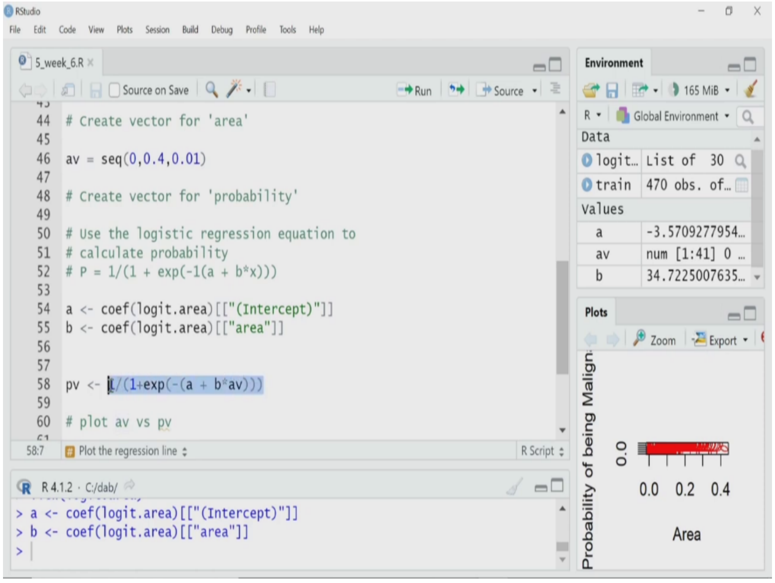
Data		
logit...	List of 30	
train	470 obs. of...	
Values		
a	-3.5709277954...	
av	num [1:41] 0 ...	
b	34.7225007635...	



```
# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv
```

```
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
>
```

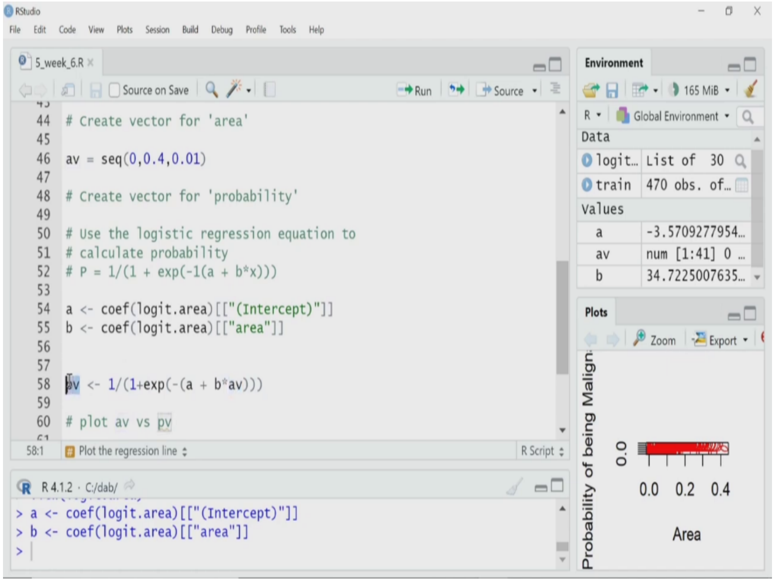
Data		
logit...	List of 30	
train	470 obs. of...	
Values		
a	-3.5709277954...	
av	num [1:41] 0 ...	
b	34.7225007635...	



```
# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv
```

```
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
>
```

```
Data
logit… List of 30
train  470 obs. of…
Values
a      -3.5709277954…
av     num [1:41] 0 …
b      34.7225007635…
```



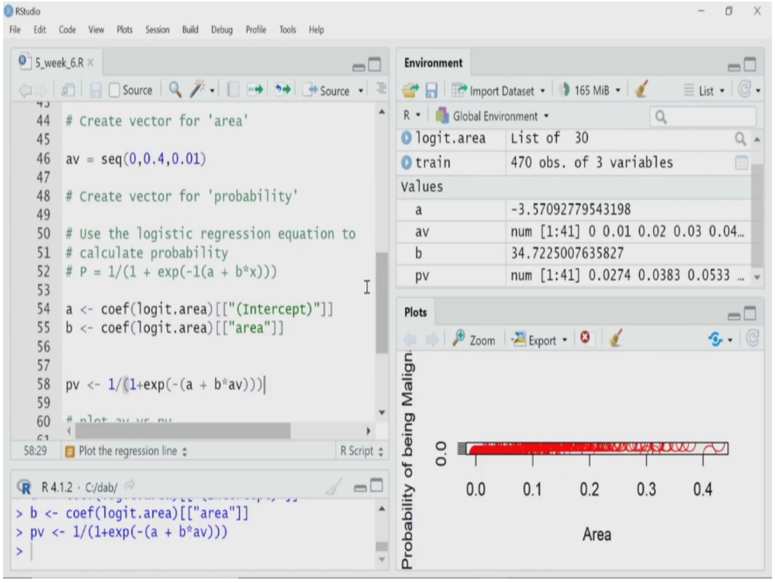
```
# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv
```

```
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
>
```

```
Data
logit… List of 30
train  470 obs. of…
Values
a      -3.5709277954…
av     num [1:41] 0 …
b      34.7225007635…
```

So, av is a vector and I give that as a vector itself. So, look at this line, I am saying pv is actually equivalent to or equal to, assigned to 1 divided by 1 plus exponential in the bracket I am writing minus a plus b into av. So, the input is av, the vector. So, when this equation will be implemented, it will throw the output also as a vector. So, for each value of av in av, I will get a corresponding value in pv.

(Refer Slide Time: 14:30)



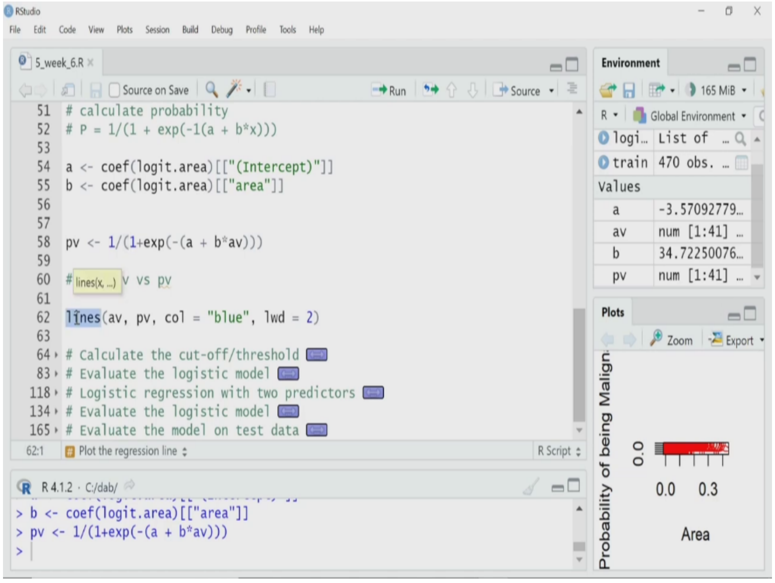
```
# Create vector for 'area'

av = seq(0,0.4,0.01)

# Create vector for 'probability'

# Use the logistic regression equation to
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv
```

```
logit.area   List of  30
train        470 obs. of 3 variables
Values
  a          -3.57092779543198
  av         num [1:41] 0 0.01 0.02 0.03 0.04…
  b          34.7225007635827
  pv         num [1:41] 0.0274 0.0383 0.0533 …
```

```
> b <- coef(logit.area)[["area"]]
> pv <- 1/(1+exp(-(a + b*av)))
>
```

So, if I execute it, you can see here, av has 1 to 41, 40 values. Similarly, for each of these values, it has calculated pv which has 40 values. So, these two vectors have same length. So now, what I will do, I will plot this av and pv and draw a smooth line and overlay on the existing plot.

(Refer Slide Time: 14:56)

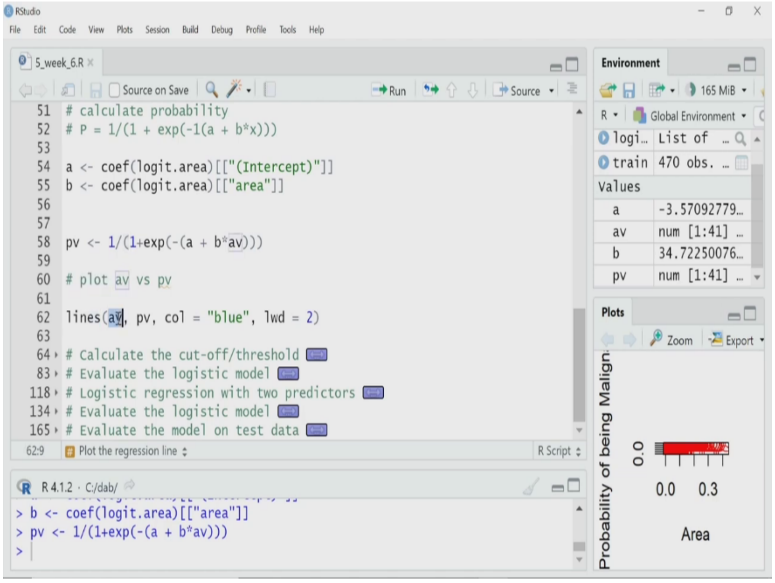


```
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# lines(x, ...) v vs pv

lines(av, pv, col = "blue", lwd = 2)

# Calculate the cut-off/threshold
# Evaluate the logistic model
# Logistic regression with two predictors
# Evaluate the logistic model
# Evaluate the model on test data
```

62:1 Plot the regression line ÷ R Script ÷

```
> b <- coef(logit.area)[["area"]]
> pv <- 1/(1+exp(-(a + b*av)))
>
```

logi…	List of …
train	470 obs. …
Values	
a	-3.57092779…
av	num [1:41] …
b	34.72250076…
pv	num [1:41] …

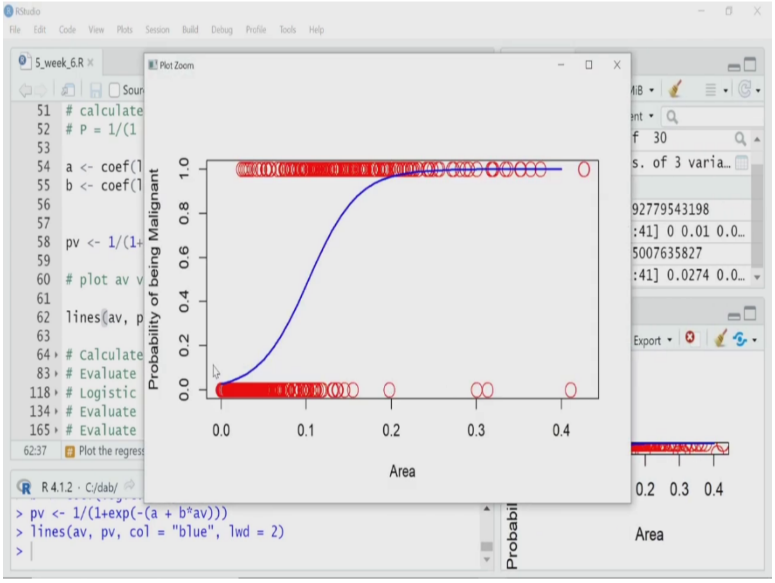


```
# calculate probability
# P = 1/(1 + exp(-1(a + b*x)))

a <- coef(logit.area)[["(Intercept)"]]
b <- coef(logit.area)[["area"]]


pv <- 1/(1+exp(-(a + b*av)))

# plot av vs pv

lines(av, pv, col = "blue", lwd = 2)

# Calculate the cut-off/threshold
# Evaluate the logistic model
# Logistic regression with two predictors
# Evaluate the logistic model
# Evaluate the model on test data
```

62:9 Plot the regression line ÷ R Script ÷

```
> b <- coef(logit.area)[["area"]]
> pv <- 1/(1+exp(-(a + b*av)))
>
```

logi…	List of …
train	470 obs. …
Values	
a	-3.57092779…
av	num [1:41] …
b	34.72250076…
pv	num [1:41] …



```
# calculate
# P = 1/(1

a <- coef(1
b <- coef(1


pv <- 1/(1+

# plot av v

lines(av, p

# Calculate
# Evaluate
# Logistic
# Evaluate
# Evaluate
```

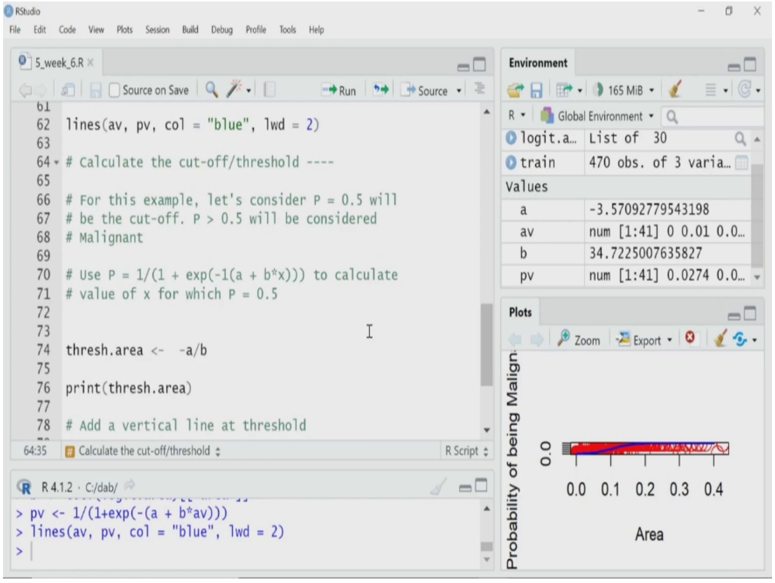
```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
>
```

lines(av, pv, col = “blue”, lwd = 2)

So, I am using the lines function, which will draw a smooth curve and my input data is av and pv, I am using color blue and line width 2. Let me zoom, you can see we got the standard logistic regression type equation curve here. So, now let me move to something important here, see I have got this curve.

Now, how should I now make a decision whether a particular sample is malignant or not, so to do that I have to decide the threshold, for example, I can say okay when the probability is 0.5, so then I have a particular value of area. So, if the area the mean area of nucleus in a particular tumour sample is bigger than that value of area, obviously, probability will be bigger than 0.5, and I will call it a malignant sample. So, that means I have to calculate the threshold area.

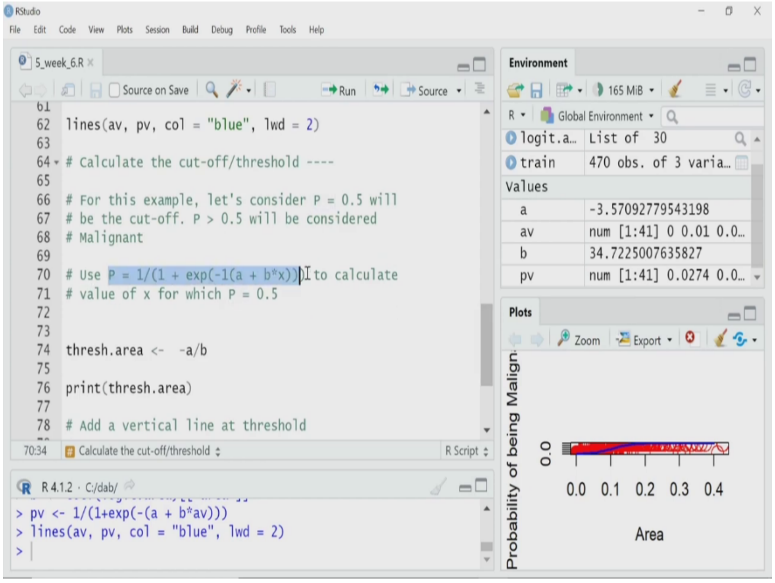
(Refer Slide Time: 15:55)



```
lines(av, pv, col = "blue", lwd = 2)

# Calculate the cut-off/threshold ----

# For this example, let's consider P = 0.5 will
# be the cut-off. P > 0.5 will be considered
# Malignant

# Use P = 1/(1 + exp(-1(a + b*x))) to calculate
# value of x for which P = 0.5


thresh.area <-  -a/b

print(thresh.area)

# Add a vertical line at threshold
```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
>
```

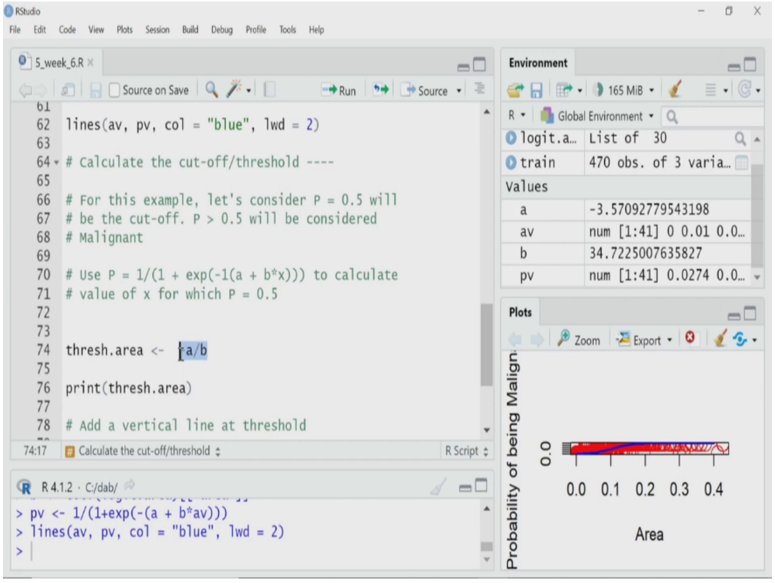
logit.a…	List of 30
train	470 obs. of 3 varia…
Values	
a	-3.57092779543198
av	num [1:41] 0 0.01 0.0…
b	34.7225007635827
pv	num [1:41] 0.0274 0.0…



```
lines(av, pv, col = "blue", lwd = 2)

# Calculate the cut-off/threshold ----

# For this example, let's consider P = 0.5 will
# be the cut-off. P > 0.5 will be considered
# Malignant

# Use P = 1/(1 + exp(-1(a + b*x))) to calculate
# value of x for which P = 0.5


thresh.area <-  -a/b

print(thresh.area)

# Add a vertical line at threshold
```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
>
```

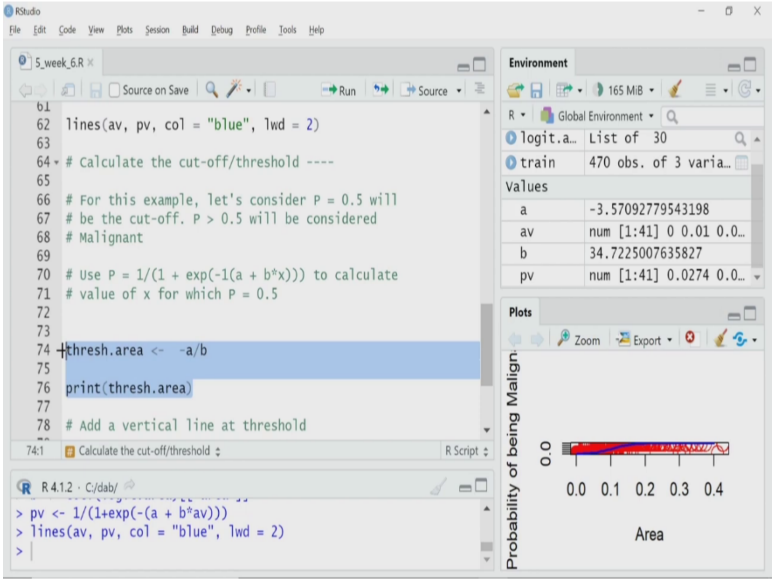
logit.a…	List of 30
train	470 obs. of 3 varia…
Values	
a	-3.57092779543198
av	num [1:41] 0 0.01 0.0…
b	34.7225007635827
pv	num [1:41] 0.0274 0.0…



```
lines(av, pv, col = "blue", lwd = 2)

# Calculate the cut-off/threshold ----

# For this example, let's consider P = 0.5 will
# be the cut-off. P > 0.5 will be considered
# Malignant

# Use P = 1/(1 + exp(-1(a + b*x))) to calculate
# value of x for which P = 0.5


thresh.area <-  -a/b

print(thresh.area)

# Add a vertical line at threshold
```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
>
```

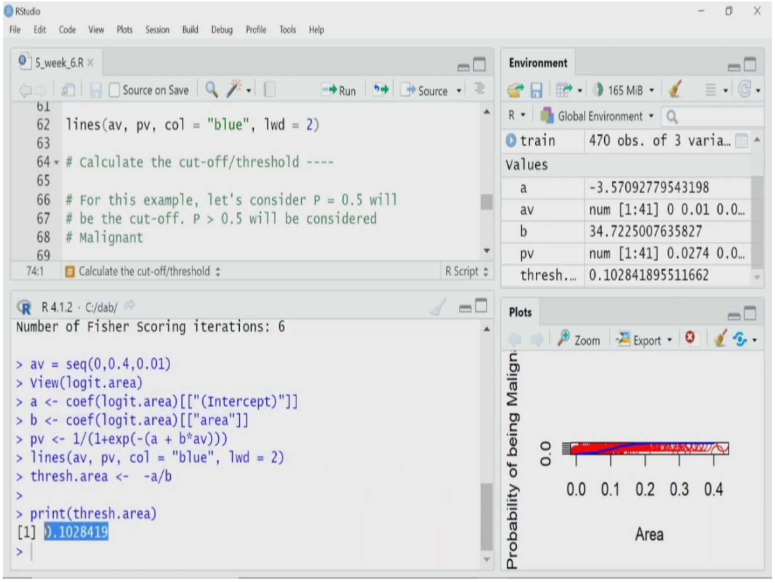
logit.a…	List of 30
train	470 obs. of 3 varia…
Values	
a	-3.57092779543198
av	num [1:41] 0 0.01 0.0…
b	34.7225007635827
pv	num [1:41] 0.0274 0.0…



```
lines(av, pv, col = "blue", lwd = 2)

# Calculate the cut-off/threshold ----

# For this example, let's consider P = 0.5 will
# be the cut-off. P > 0.5 will be considered
# Malignant

# Use P = 1/(1 + exp(-1(a + b*x))) to calculate
# value of x for which P = 0.5


thresh.area <-  -a/b

print(thresh.area)

# Add a vertical line at threshold
```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
>
```

logit.a...	List of 30
train	470 obs. of 3 varia...
Values	
a	-3.57092779543198
av	num [1:41] 0 0.01 0.0...
b	34.7225007635827
pv	num [1:41] 0.0274 0.0...



```
lines(av, pv, col = "blue", lwd = 2)

# Calculate the cut-off/threshold ----

# For this example, let's consider P = 0.5 will
# be the cut-off. P > 0.5 will be considered
# Malignant
```

```
Number of Fisher Scoring iterations: 6

> av = seq(0,0.4,0.01)
> View(logit.area)
> a <- coef(logit.area)[["(Intercept)"]]
> b <- coef(logit.area)[["area"]]
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
>
```

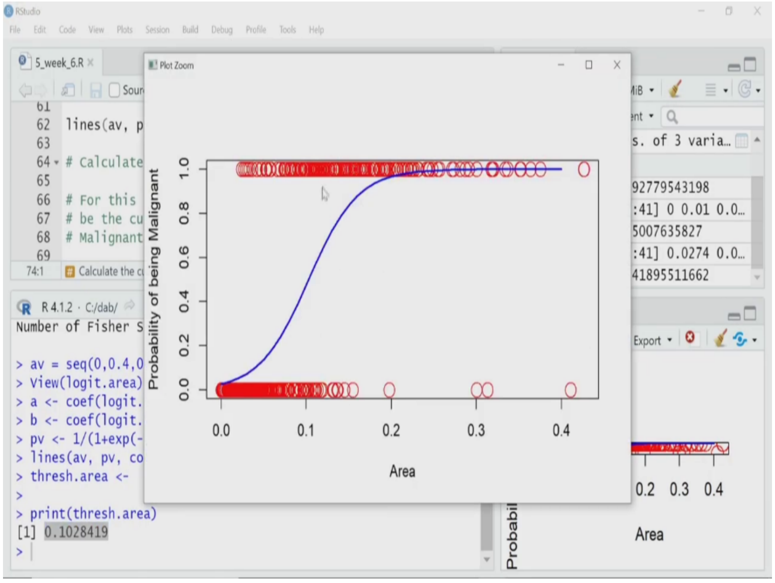
train	470 obs. of 3 varia...
Values	
a	-3.57092779543198
av	num [1:41] 0 0.01 0.0...
b	34.7225007635827
pv	num [1:41] 0.0274 0.0...
thresh...	0.102841895511662

thresh.area ← -a / b

print(thresh.area)

So, how I am doing that, here, I will do that. So, I know the equation p equal to this, the logistic equation. If you set P equal to 0.5 and do some algebra, it will come that this will be equal to this value of x will be equal to minus a by b, a and b, we have calculated. So, I will calculate that and print it. So, the value of that threshold area is 0.1028. If the mean area of a tumour samples nucleus, cell nucleus, is bigger than 0.10, then you will call that sample now, as a malignant sample as per your logistic model.

(Refer Slide Time: 16:38)



RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
5_week_6.R
Plot Zoom
Probability of being Malignant
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.1
0.2
0.3
0.4
Area
Export

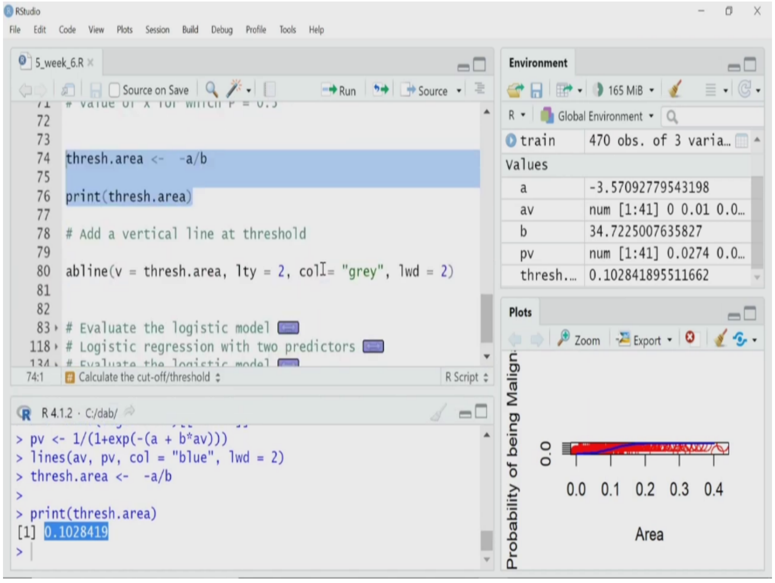
```
lines(av, p

# Calculate

# For this
# be the cu
# Malignant
```

74:1 Calculate the c

R 4.1.2 · C:/dab/

```
Number of Fisher S

> av = seq(0,0.4,0
> View(logit.area)
> a <- coef(logit.
> b <- coef(logit.
> pv <- 1/(1+exp(-
> lines(av, pv, co
> thresh.area <-
>
> print(thresh.area)
[1] 0.1028419
>
```

```
s. of 3 varia…
92779543198
:41] 0 0.01 0.0…
5007635827
:41] 0.0274 0.0…
41895511662
```



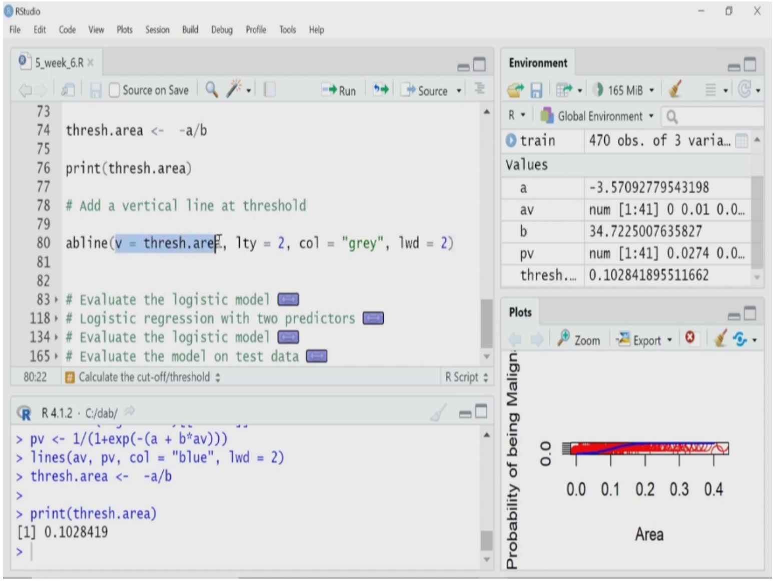
```
# Value of x for which P = 0.5

thresh.area <-  -a/b

print(thresh.area)

# Add a vertical line at threshold

abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)


# Evaluate the logistic model
# Logistic regression with two predictors
# Evaluate the logistic model
```

74:1 Calculate the cut-off/threshold R Script

R 4.1.2 · C:/dab/

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
>
```

```
train     470 obs. of 3 varia…
Values
  a       -3.57092779543198
  av      num [1:41] 0 0.01 0.0…
  b       34.7225007635827
  pv      num [1:41] 0.0274 0.0…
  thresh… 0.102841895511662
```



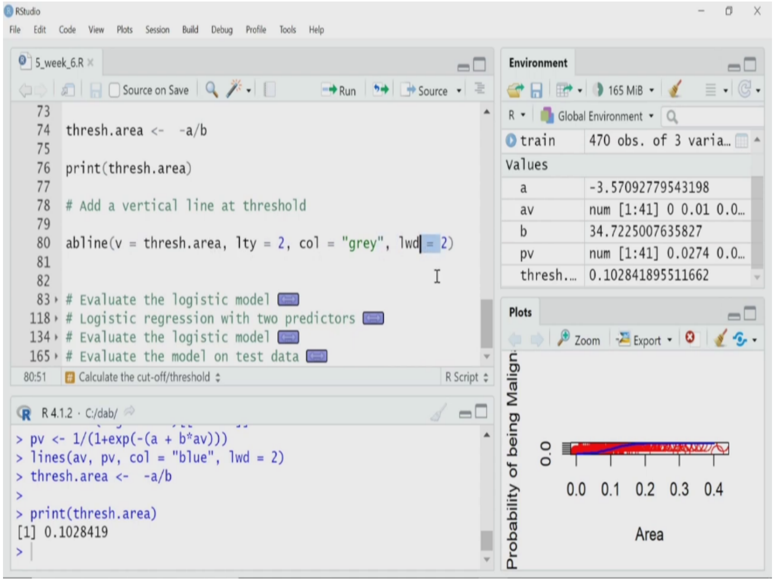
```

thresh.area <-  -a/b

print(thresh.area)

# Add a vertical line at threshold

abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)


# Evaluate the logistic model
# Logistic regression with two predictors
# Evaluate the logistic model
# Evaluate the model on test data
```

80:22 Calculate the cut-off/threshold R Script

R 4.1.2 · C:/dab/

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
>
```

```
train     470 obs. of 3 varia…
Values
  a       -3.57092779543198
  av      num [1:41] 0 0.01 0.0…
  b       34.7225007635827
  pv      num [1:41] 0.0274 0.0…
  thresh… 0.102841895511662
```



```
thresh.area <-  -a/b

print(thresh.area)

# Add a vertical line at threshold

abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)


# Evaluate the logistic model
# Logistic regression with two predictors
# Evaluate the logistic model
# Evaluate the model on test data
```

80:51 Calculate the cut-off/threshold R Script

R 4.1.2 · C:/dab/

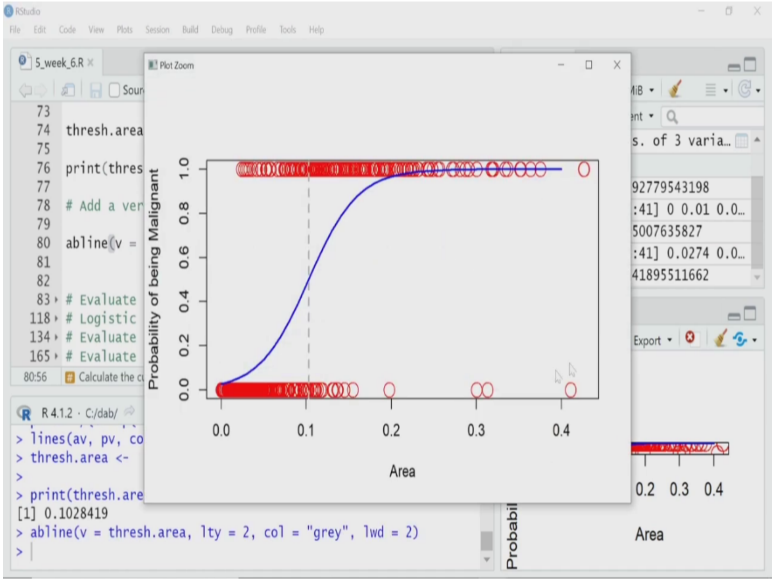
```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
>
```

```
train      470 obs. of 3 varia…
Values
  a        -3.57092779543198
  av       num [1:41] 0 0.01 0.0…
  b        34.7225007635827
  pv       num [1:41] 0.0274 0.0…
  thresh.… 0.102841895511662
```

abline(v = thresh.area, lty = 2, col = “grey”, lwd = 2)

So, I want to put a vertical line on this plot on this plot representing that threshold. So, what I will do? I will call the abline function, because abline overlay a straight line. And I will say that draw a vertical line and that vertical line will have a value thresh dot area, that threshold that I have calculated and I am using grey color for that and line length is 2.

(Refer Slide Time: 17:05)


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
5_week_6.R
Plot Zoom
Probability of being Malignant
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.1
0.2
0.3
0.4
Area
Export

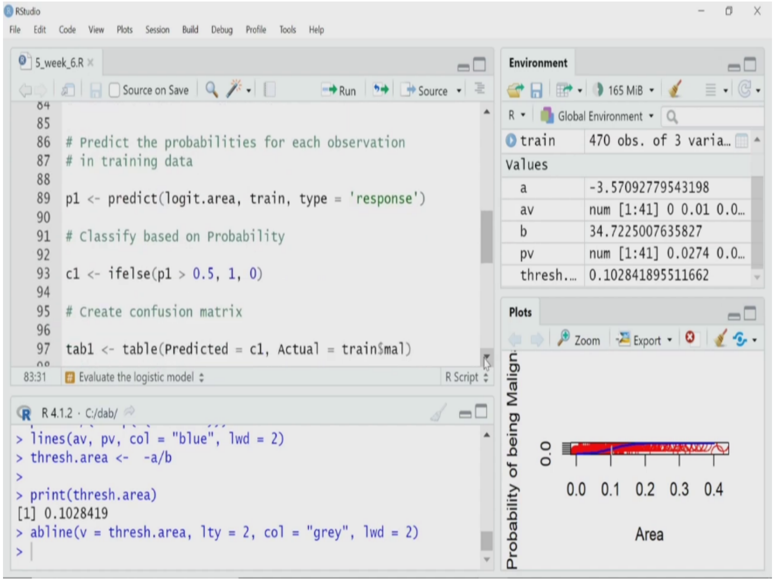
```
thresh.area

print(thres

# Add a ver

abline(v =


# Evaluate
# Logistic
# Evaluate
# Evaluate
```

80:56 Calculate the c

R 4.1.2 · C:/dab/

```
> lines(av, pv, co
> thresh.area <-
>
> print(thresh.are
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
>
```

```
s. of 3 varia…
92779543198
:41] 0 0.01 0.0…
5007635827
:41] 0.0274 0.0…
41895511662
```



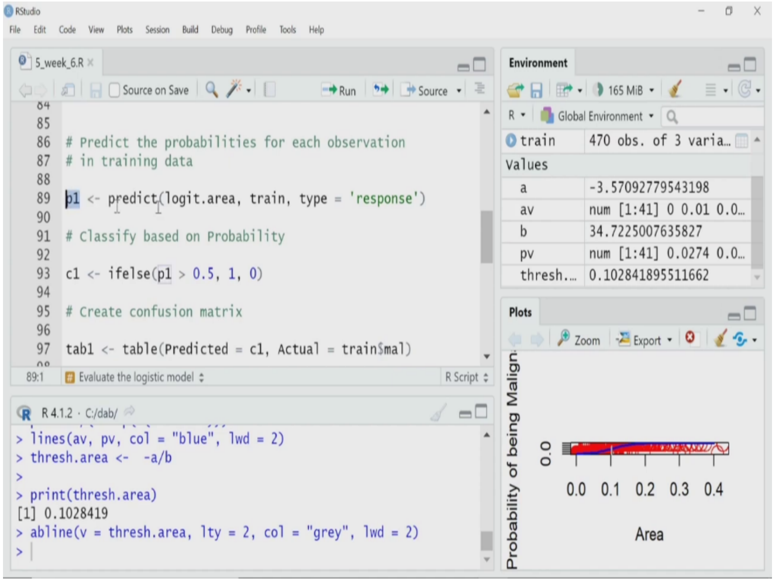
```

# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)
```

83:31 Evaluate the logistic model R Script

R 4.1.2 · C:/dab/

```
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
>
```

train	470 obs. of 3 varia…
Values	
a	-3.57092779543198
av	num [1:41] 0 0.01 0.0…
b	34.7225007635827
pv	num [1:41] 0.0274 0.0…
thresh.…	0.102841895511662



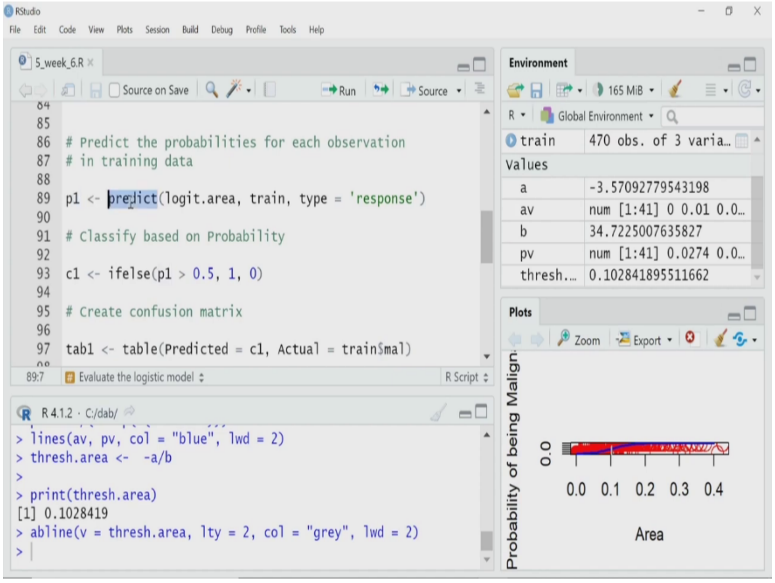
```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tabl <- table(Predicted = c1, Actual = train$mal)
```

89:1 Evaluate the logistic model R Script

R 4.1.2 · C:/dab/

```
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
>
```

train	470 obs. of 3 varia...
Values	
a	-3.57092779543198
av	num [1:41] 0 0.01 0.0...
b	34.7225007635827
pv	num [1:41] 0.0274 0.0...
thresh...	0.102841895511662

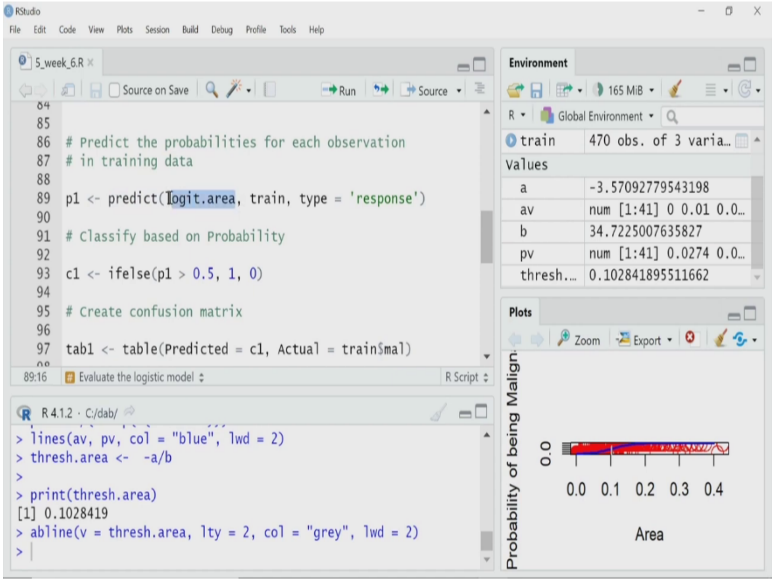


```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tabl <- table(Predicted = c1, Actual = train$mal)
```

89:7 Evaluate the logistic model R Script

R 4.1.2 · C:/dab/

```
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
>
```

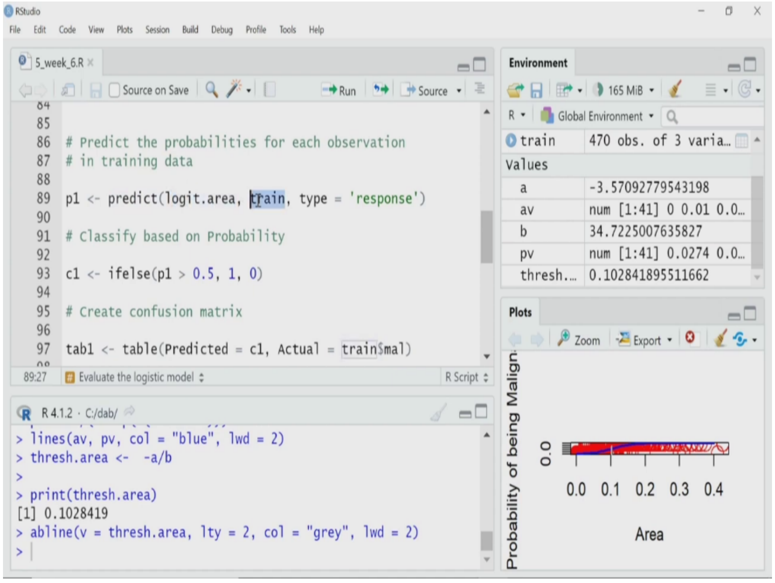


```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tabl <- table(Predicted = c1, Actual = train$mal)
```

89:16 Evaluate the logistic model R Script

R 4.1.2 · C:/dab/

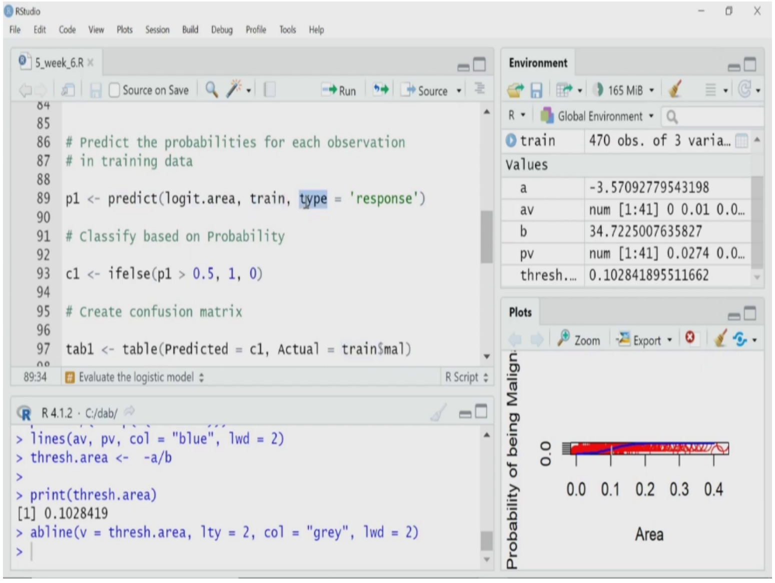
```
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
>
```



```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tabl <- table(Predicted = c1, Actual = train$mal)
```

```
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
>
```

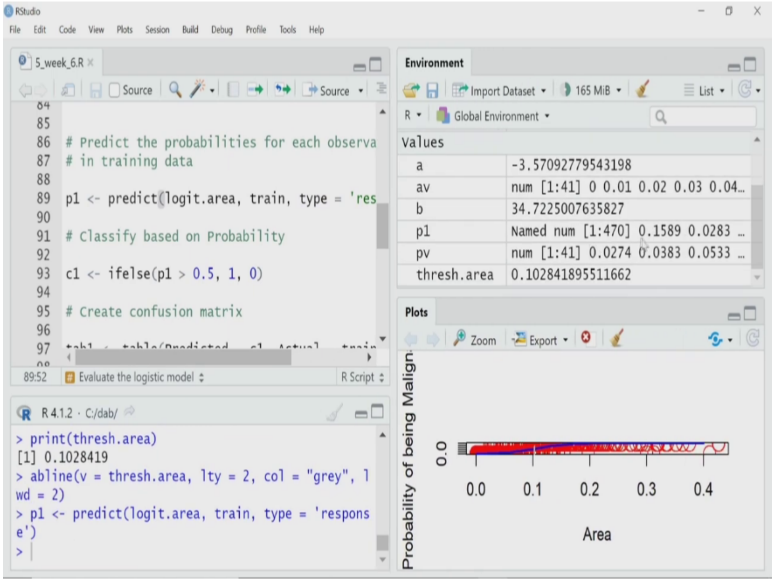
```
train      470 obs. of 3 varia…
Values
a          -3.57092779543198
av         num [1:41] 0 0.01 0.0…
b          34.7225007635827
pv         num [1:41] 0.0274 0.0…
thresh.…   0.102841895511662
```



```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tabl <- table(Predicted = c1, Actual = train$mal)
```

```
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
>
```

```
train      470 obs. of 3 varia…
Values
a          -3.57092779543198
av         num [1:41] 0 0.01 0.0…
b          34.7225007635827
pv         num [1:41] 0.0274 0.0…
thresh.…   0.102841895511662
```



```
# Predict the probabilities for each observa
# in training data

p1 <- predict(logit.area, train, type = 'res

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix
```

```
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", l
wd = 2)
> p1 <- predict(logit.area, train, type = 'respons
e')
>
```

```
Values
a            -3.57092779543198
av           num [1:41] 0 0.01 0.02 0.03 0.04…
b            34.7225007635827
p1           Named num [1:470] 0.1589 0.0283 …
pv           num [1:41] 0.0274 0.0383 0.0533 …
thresh.area  0.102841895511662
```

```
p1 ← predict(logit.area, train, type = “response”)
c1 ← ifelse(p1 > 0.5, 1, 0)
tab1 ← table(Predicted = c1, Actual = train$mal)
```

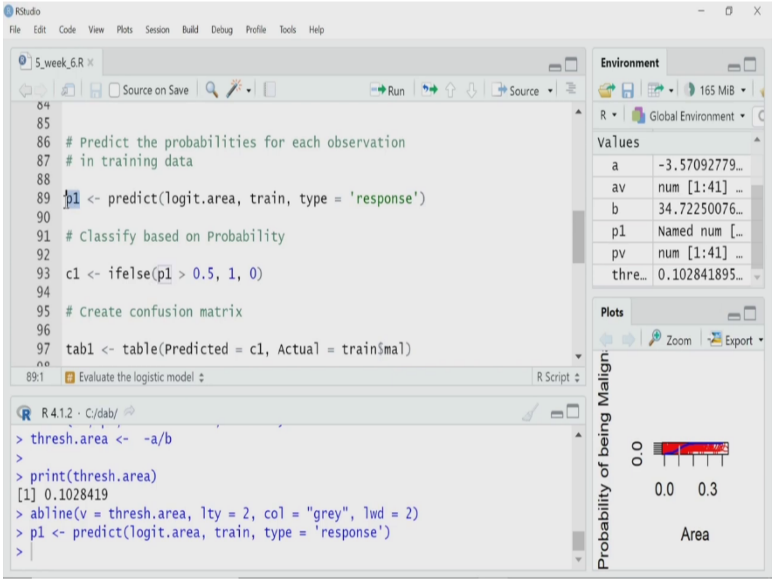
So, if I execute that, here you have, you have this vertical line. So, that means, any data beyond this vertical line on the right-hand side are all malignant below that vertical line everything is benign. Fine, good enough. So, now, visually I can see it, but I have to evaluate the goodness of this model. And usually, the goodness of a logistic model or any this type of machine learning model is done using what is called a classification model, we use usually what is called a confusion matrix.

And so, let us see how I can create that here. So, what I will do, I will use now the same training data set that I have used and I will put as an input to my logistic regression model, I have created the logistic regression model. Now, I will use that training data set itself as an input to that and calculate the probability, and that probability will be stored in p 1 variable.

So, that is what I am doing here? I am using predict function and I am using the logit dot area the logistic model and train as the input data and I am saying that okay calculate the response. So, it calculates thus those values. And if you see those values will be p 1, those values are some fractions, these are probabilities.

Now, I want to compare these values with respect to the value of mal for each sample. So, for each sample, now, I have calculated the probability. Now, the value of mal for each sample varies from 0 to 1, that is also probability is either 0 or 1.

(Refer Slide Time: 18:45)

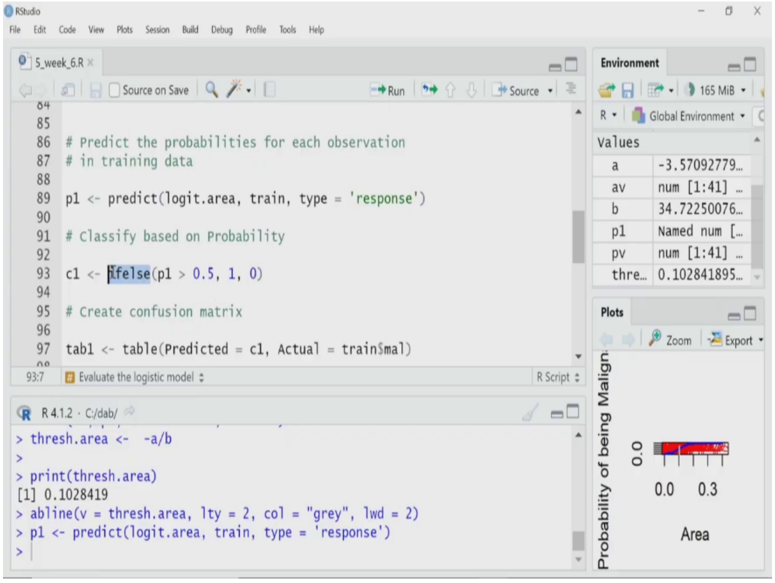


```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)
```

```
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
>
```

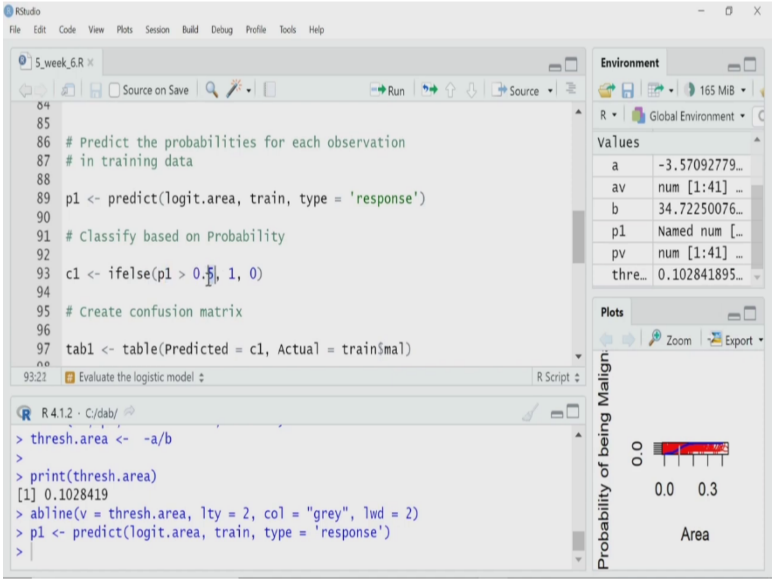
Values

a	-3.57092779…
av	num [1:41] …
b	34.72250076…
p1	Named num […
pv	num [1:41] …
thre…	0.102841895…



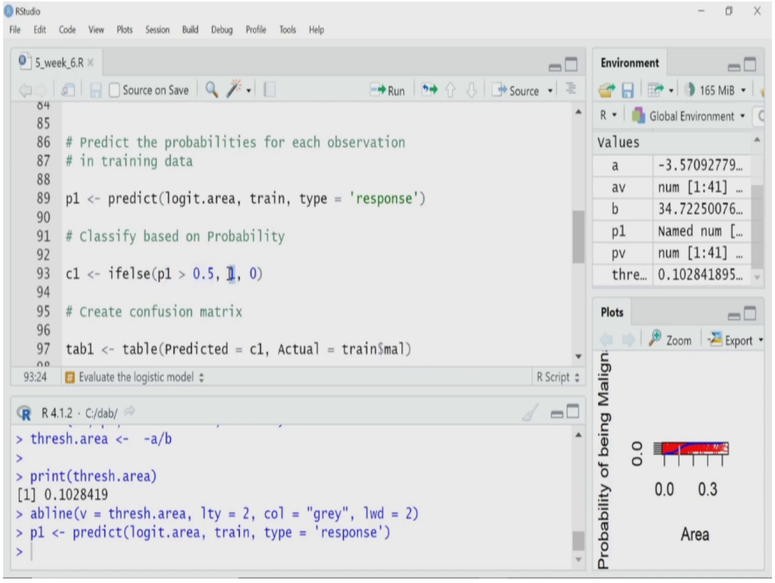
```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)
```

```
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
>
```



```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)
```

```
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
>
```



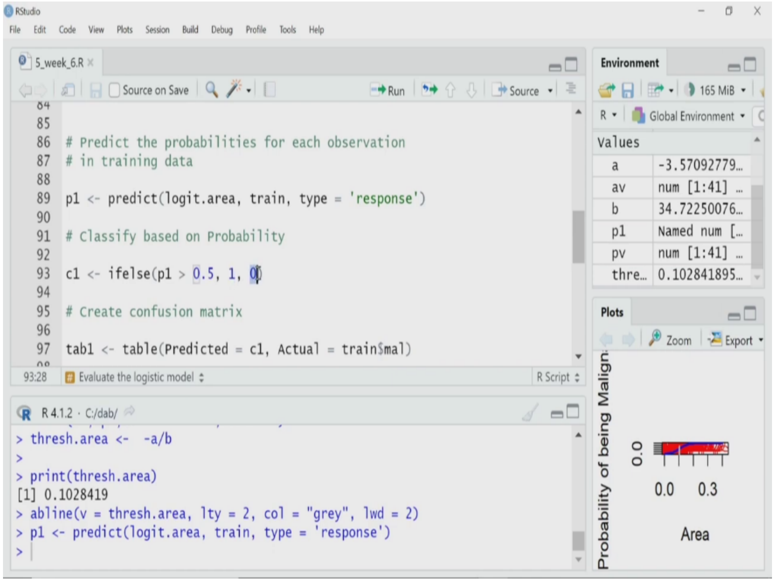
```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)
```

93:24 Evaluate the logistic model R Script

```
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
>
```

Values

a	-3.57092779...
av	num [1:41] ...
b	34.72250076...
p1	Named num [...
pv	num [1:41] ...
thre...	0.102841895...



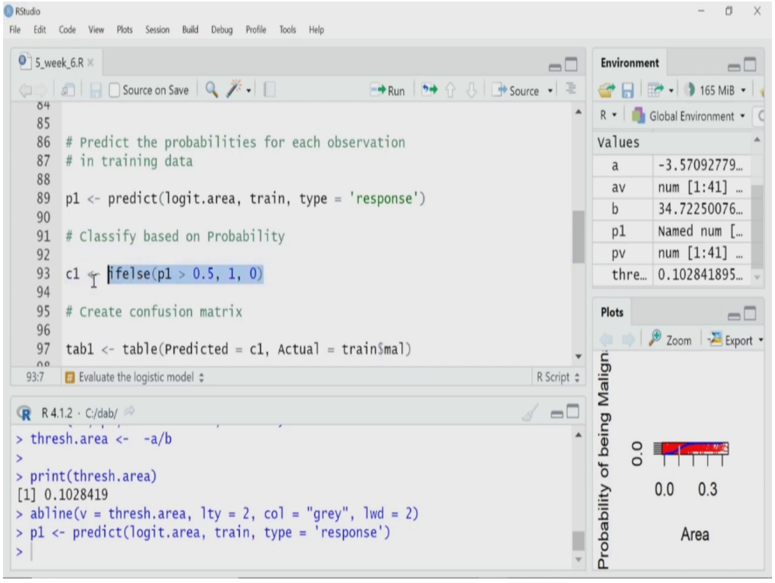
```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)
```

93:28 Evaluate the logistic model R Script

```
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
>
```

Values

a	-3.57092779...
av	num [1:41] ...
b	34.72250076...
p1	Named num [...
pv	num [1:41] ...
thre...	0.102841895...



```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)
```

93:7 Evaluate the logistic model R Script

```
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
>
```

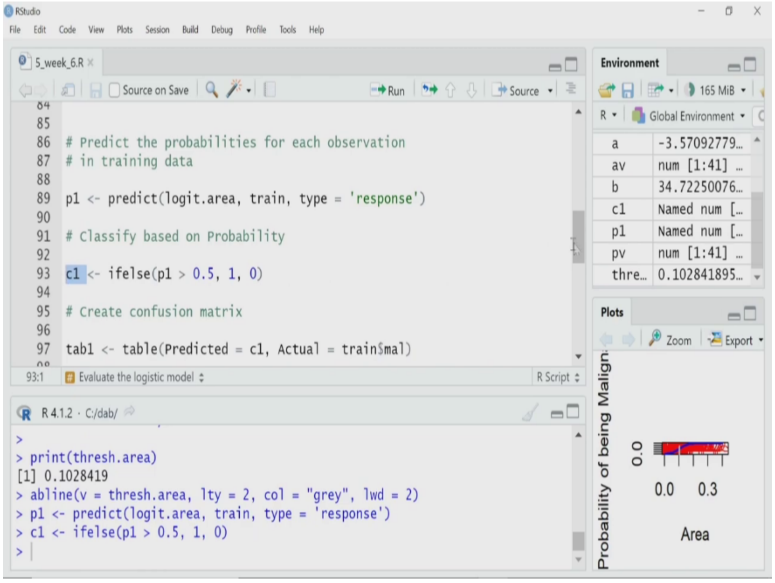
Values

a	-3.57092779...
av	num [1:41] ...
b	34.72250076...
p1	Named num [...
pv	num [1:41] ...
thre...	0.102841895...

So, what I will do is that I will now based on this p value, the probability value, I will classify my training set sample whether it is malignant or not. So, what I will do? I have decided that if the probability is bigger than 0.5, it is malignant. So, I am calling if else function and I am giving some argument I am telling that if p 1, for a sample, is bigger than 0.5, then you call it 1, that means a malignant otherwise you call it 0.

And this whole data you store in a vector c 1. So, I am using a if else function to decide whether each of the sample for which I predicted the probability whether they are 1 or 0, malignant or benign. So, I do that.

(Refer Slide Time: 19:30)



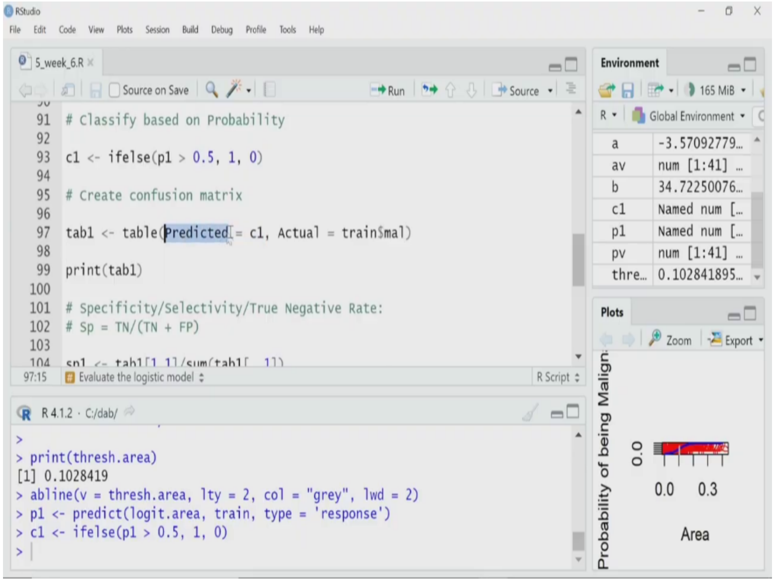
```
# Predict the probabilities for each observation
# in training data

p1 <- predict(logit.area, train, type = 'response')

# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)
```

```
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
> |
```

a	-3.57092779…
av	num [1:41] …
b	34.72250076…
c1	Named num […
p1	Named num […
pv	num [1:41] …
thre…	0.102841895…

Now this c 1 value, these are either 1 or 0. And already in my original training data, these tumours are labelled as 1 and 0. So, I will compare whether my original labelling of 1 matches with my predicted value of that sample, predicted label of that sample or not, so I will create a table. To create that table, I will use a table function. Now I will not go in details of what the table function is doing. Once I execute it will be very easy for you to understand this, let me tell you what is the arguments here.

(Refer Slide Time: 20:04)



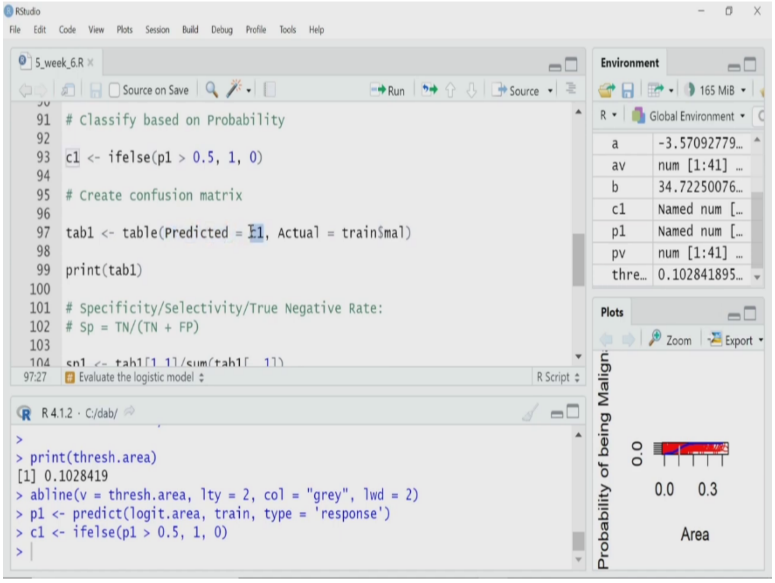
```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[, 1])
```

97:15 Evaluate the logistic model R Script

R 4.1.2 · C:/dab/

```
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
>
```

a	-3.57092779...
av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
thre...	0.102841895...



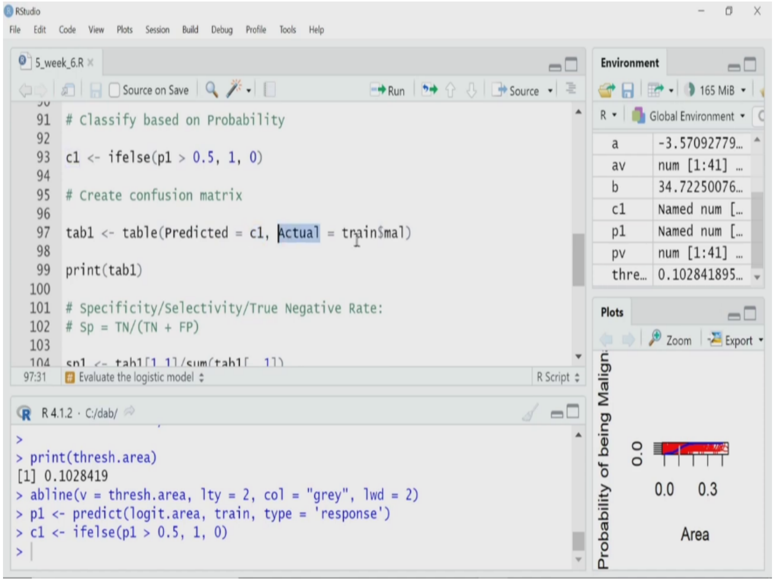
```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[, 1])
```

97:27 Evaluate the logistic model R Script

R 4.1.2 · C:/dab/

```
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
>
```

a	-3.57092779...
av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
thre...	0.102841895...



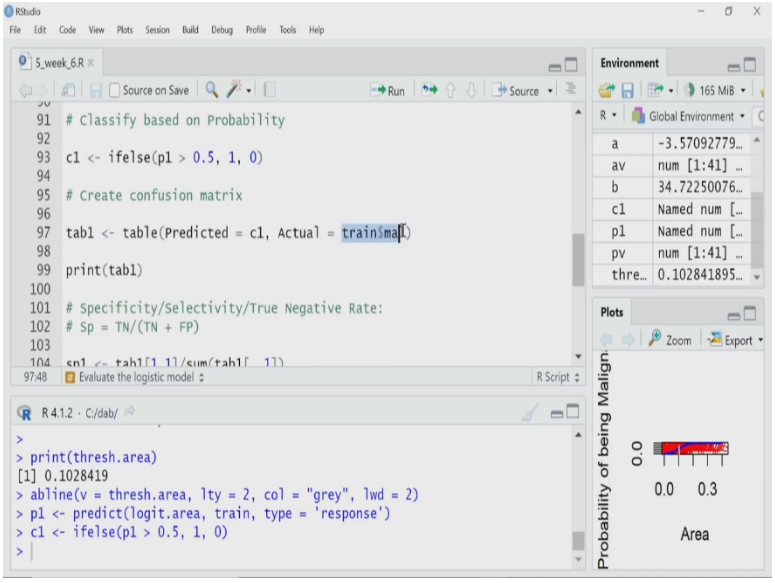
```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[, 1])
```

97:31 Evaluate the logistic model R Script

R 4.1.2 · C:/dab/

```
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
>
```

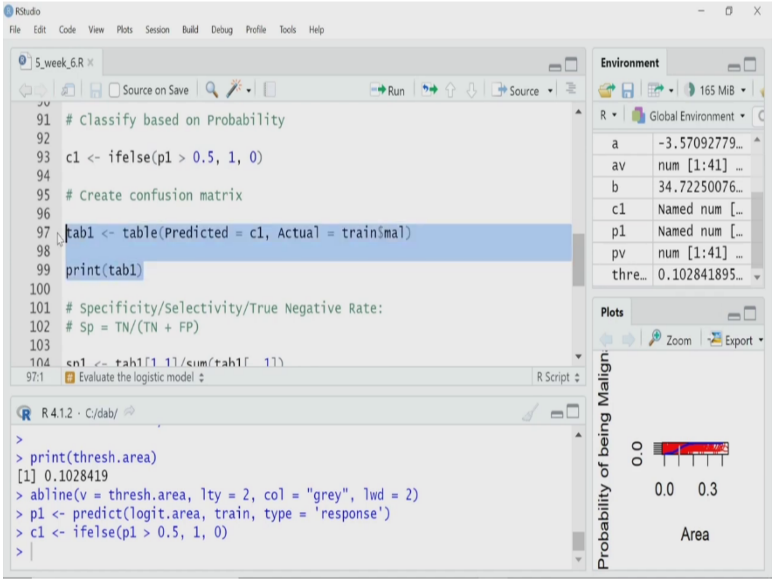
a	-3.57092779...
av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
thre...	0.102841895...



```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[, 1])
```

```
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
>
```

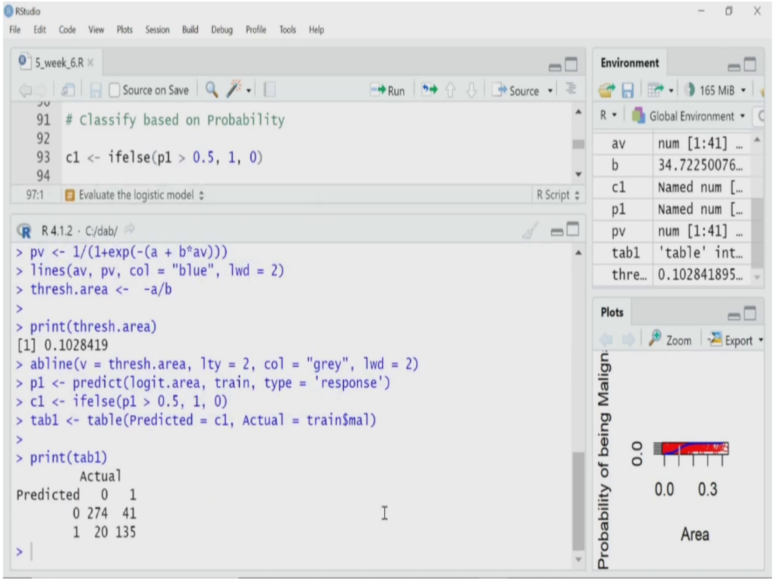
a	-3.57092779...
av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
thre...	0.102841895...



```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

# Create confusion matrix

tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[, 1])
```

```
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
>
```

a	-3.57092779...
av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
thre...	0.102841895...



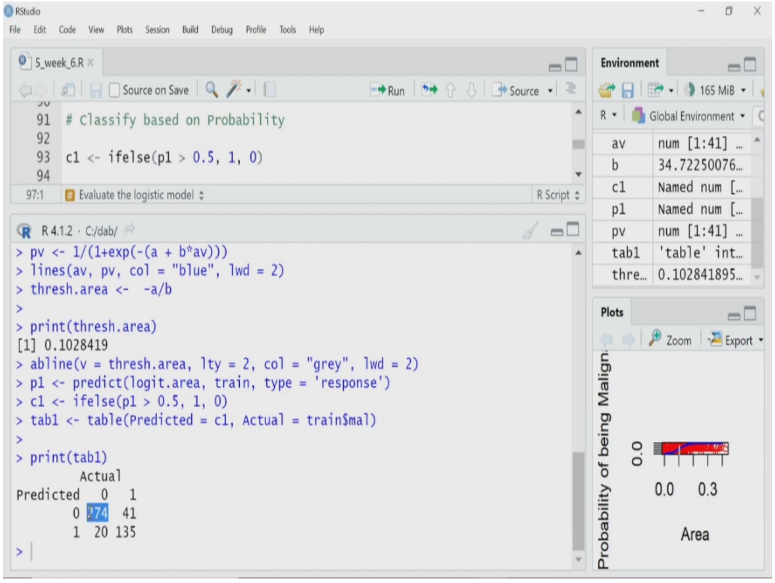
```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
> tab1 <- table(Predicted = c1, Actual = train$mal)
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
>
```

av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
tab1	'table' int...
thre...	0.102841895...

The first argument is a named argument I am giving to the table, I am saying the predicted variable is c 1, yes, that is my predicted thing, either 0 or 1 that I am predicted for the sample. And the actual one is the mal variable in my training data set, that is why train dollar sign mal, and I want to create a table and I want to print the table. So, I am printing that creating and printing it here, let us look into the table it will be much clearer.

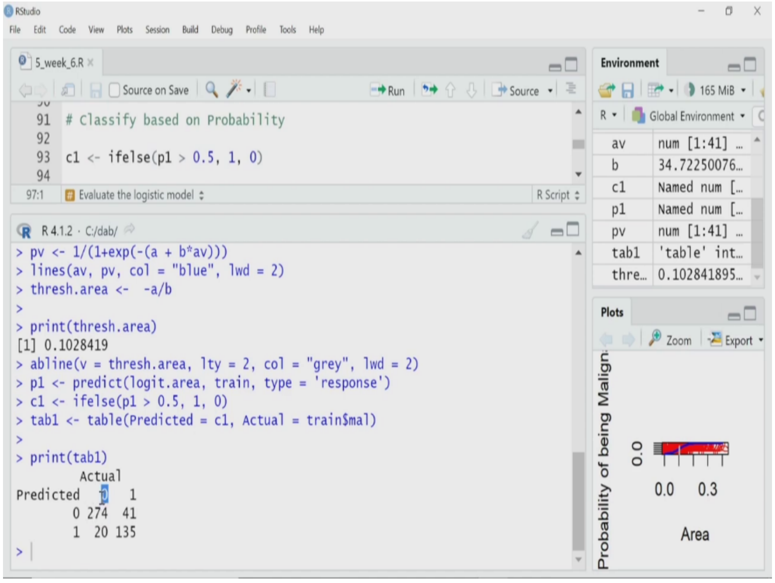
(Refer Slide Time: 20:33)



```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)
```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
> tabl <- table(Predicted = c1, Actual = train$mal)
>
> print(tabl)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
> |
```

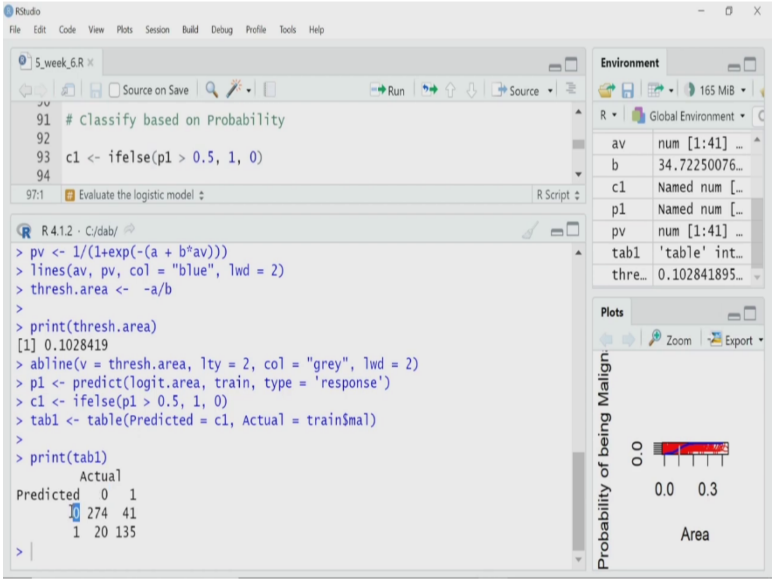
```
av     num [1:41] ...
b      34.72250076...
c1     Named num [...
p1     Named num [...
pv     num [1:41] ...
tabl   'table' int...
thre... 0.102841895...
```



```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)
```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
> tabl <- table(Predicted = c1, Actual = train$mal)
>
> print(tabl)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
> |
```

```
av     num [1:41] ...
b      34.72250076...
c1     Named num [...
p1     Named num [...
pv     num [1:41] ...
tabl   'table' int...
thre... 0.102841895...
```



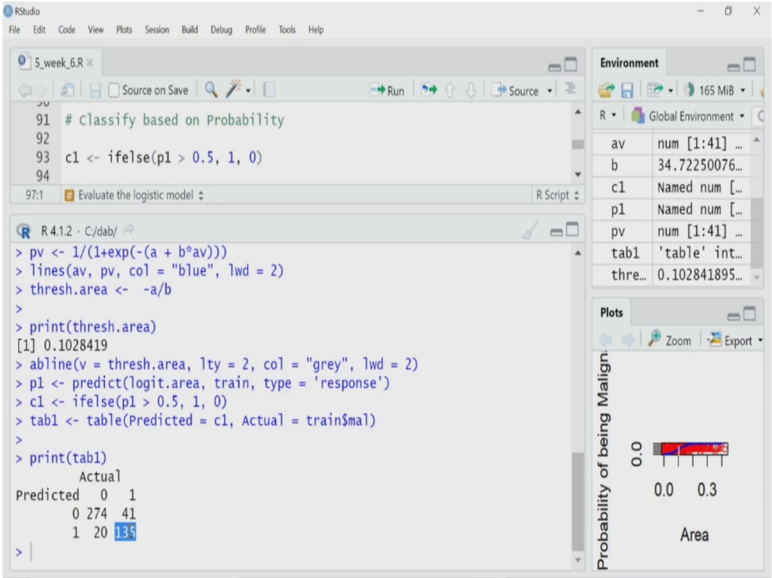
```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)

```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
> tab1 <- table(Predicted = c1, Actual = train$mal)
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
> 
```

```
av     num [1:41] …
b      34.72250076…
c1     Named num […
p1     Named num […
pv     num [1:41] …
tab1   'table' int…
thre…  0.102841895…
```

So, it is a 2 by 2 matrix, 2 by 2 table. And in actually, there are two factors 0 and 1, because my samples are either 0 or 1, malignant or non-malignant, and the predicted label is also 0, 1. So, you can see the first one is 274, actual is also 0, predicted label is also 0, that means there are 274 samples in my training data set, which are actually 0, benign. And my prediction is also 0, that means they are benign.

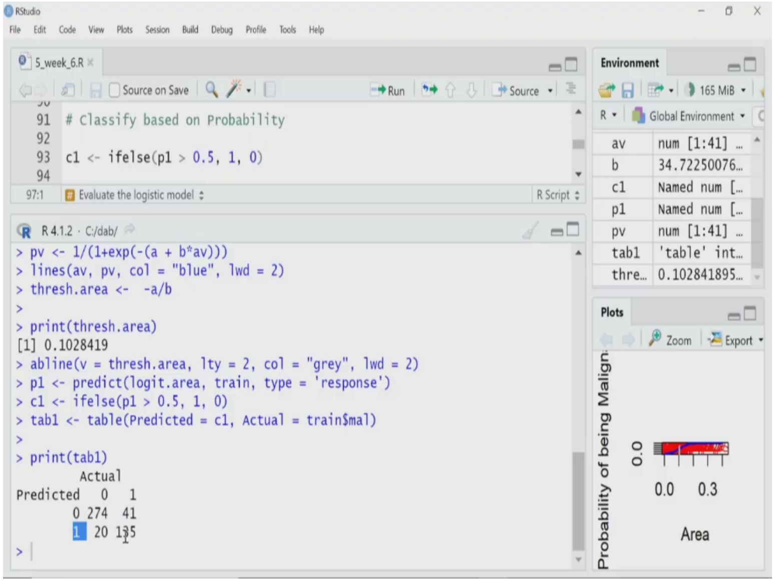
(Refer Slide Time: 21:03)



```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)
```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
> tabl <- table(Predicted = c1, Actual = train$mal)
>
> print(tabl)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
>
```

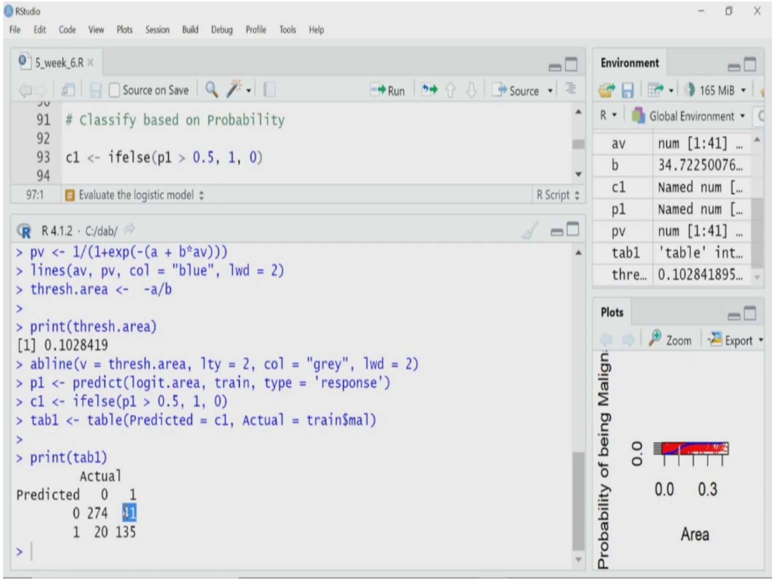
```
av      num [1:41] ...
b       34.72250076...
c1      Named num [...
p1      Named num [...
pv      num [1:41] ...
tabl    'table' int...
thre... 0.102841895...
```



```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)
```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
> tabl <- table(Predicted = c1, Actual = train$mal)
>
> print(tabl)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
>
```

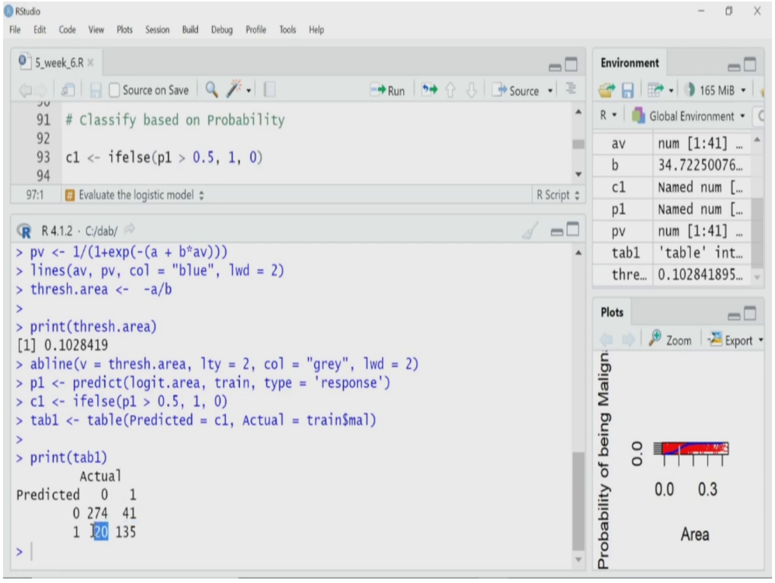
```
av      num [1:41] ...
b       34.72250076...
c1      Named num [...
p1      Named num [...
pv      num [1:41] ...
tabl    'table' int...
thre... 0.102841895...
```



```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)
```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
> tabl <- table(Predicted = c1, Actual = train$mal)
>
> print(tabl)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
> |
```

```
av      num [1:41] ...
b       34.72250076...
c1      Named num [...
p1      Named num [...
pv      num [1:41] ...
tabl    'table' int...
thre... 0.102841895...
```



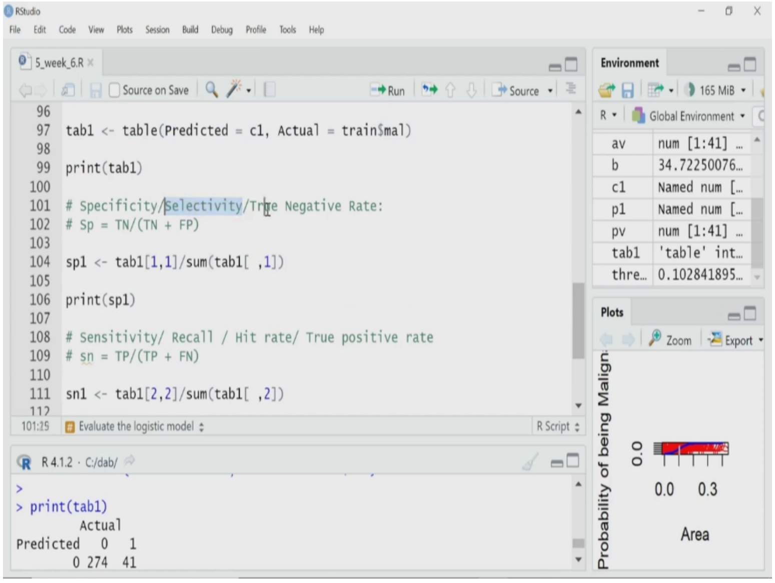
```
# Classify based on Probability

c1 <- ifelse(p1 > 0.5, 1, 0)
```

```
> pv <- 1/(1+exp(-(a + b*av)))
> lines(av, pv, col = "blue", lwd = 2)
> thresh.area <-  -a/b
>
> print(thresh.area)
[1] 0.1028419
> abline(v = thresh.area, lty = 2, col = "grey", lwd = 2)
> p1 <- predict(logit.area, train, type = 'response')
> c1 <- ifelse(p1 > 0.5, 1, 0)
> tabl <- table(Predicted = c1, Actual = train$mal)
>
> print(tabl)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
> |
```

```
av      num [1:41] ...
b       34.72250076...
c1      Named num [...
p1      Named num [...
pv      num [1:41] ...
tabl    'table' int...
thre... 0.102841895...
```

Whereas, 135 samples are there, which are actually labelled in my original data set as malignant and my predicted is also malignant. So, there are some false negatives and there are some false positives here. So, based on these actually, I can calculate my sensitivity and specificity.

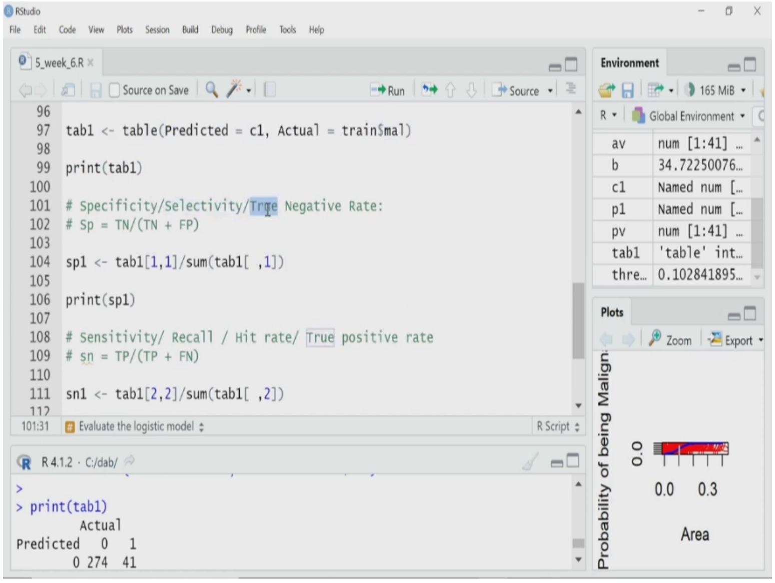
(Refer Slide Time: 21:24)



```
tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])
```

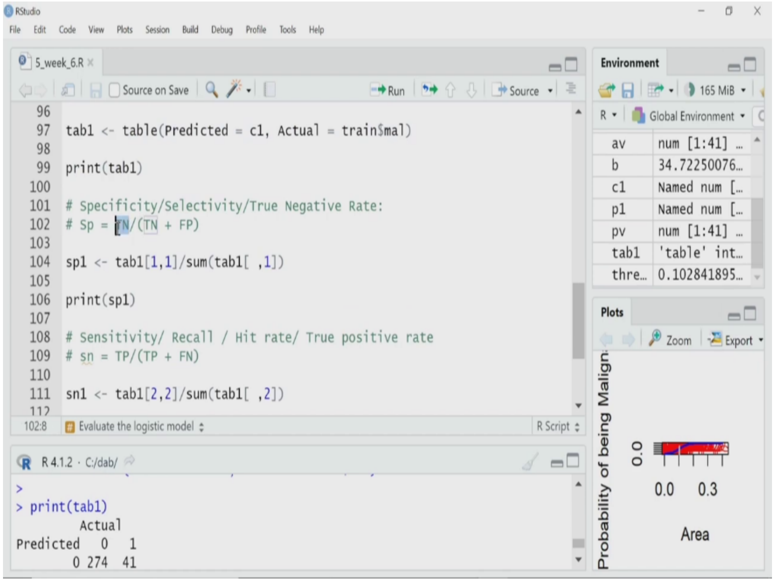
```
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
```

av	num [1:41] …
b	34.72250076…
c1	Named num […
p1	Named num […
pv	num [1:41] …
tab1	'table' int…
thre…	0.102841895…



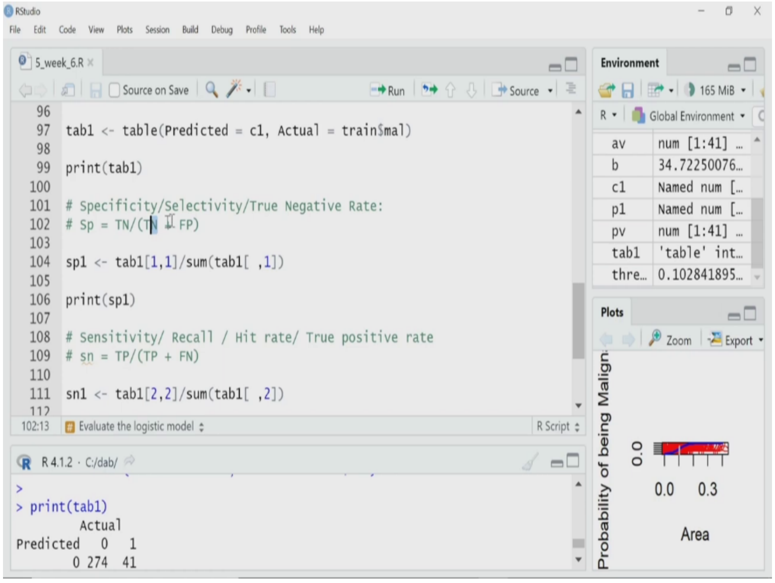
```
tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])
```

```
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
```



```
tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])
```

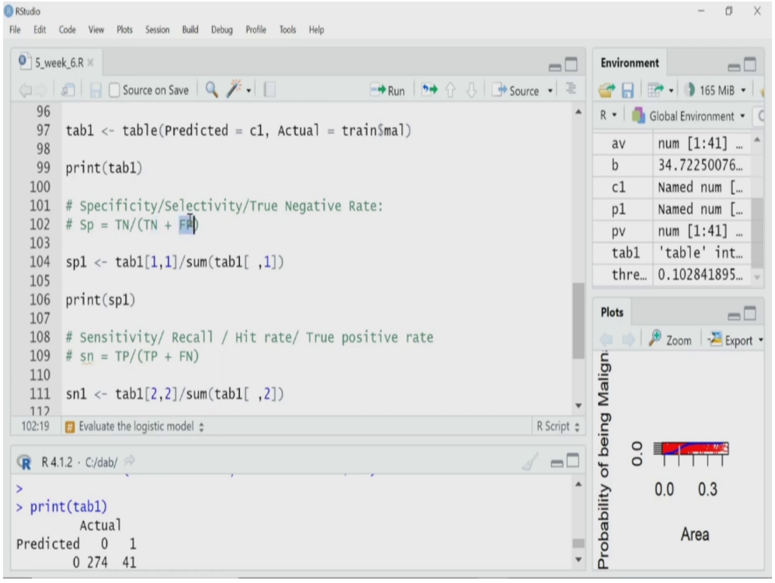
```
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
```



```
tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])
```

```
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
```

av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
tab1	'table' int...
thre...	0.102841895...



```
tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])
```

```
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
```

av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
tab1	'table' int...
thre...	0.102841895...

sp1 ← tab1[1,1] / sum(tab1[,1])

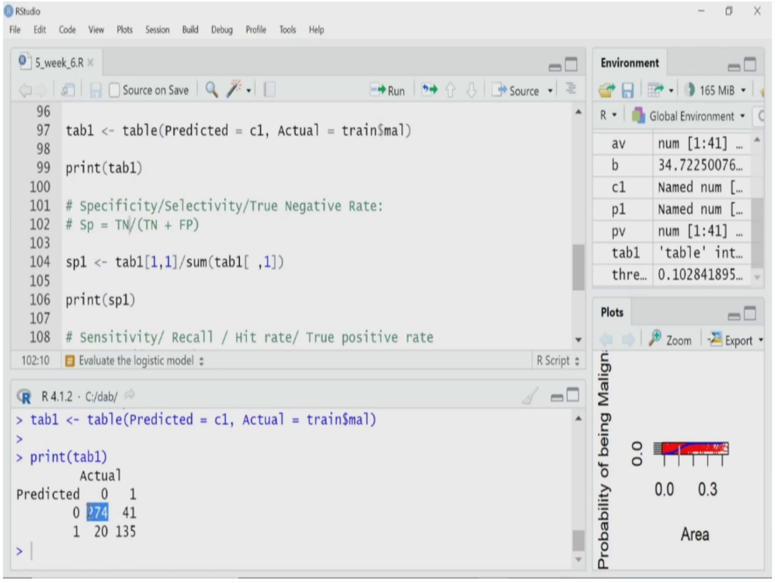
print(sp1)

sn1 ← tab1[2,2] / sum(tab1[,2])

print(sn1)

What is sensitivity and specificity. Specificity is also called selectivity or true negative rate, that is equal to true negative number of true negative in my confusion table divided by true negative plus false positive. And that is what I calculate here. Table is a matrix, so this is a matrix.

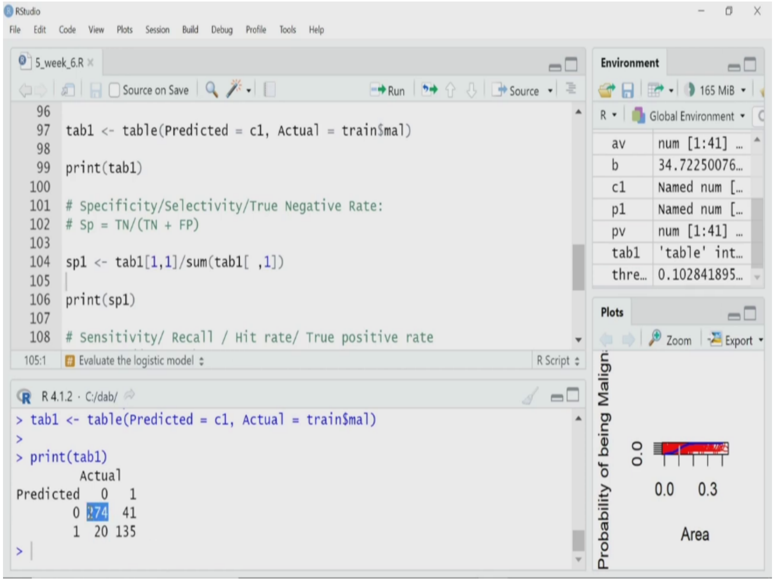
(Refer Slide Time: 21:45)



av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
tab1	'table' int...
thre...	0.102841895...

```
tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
```

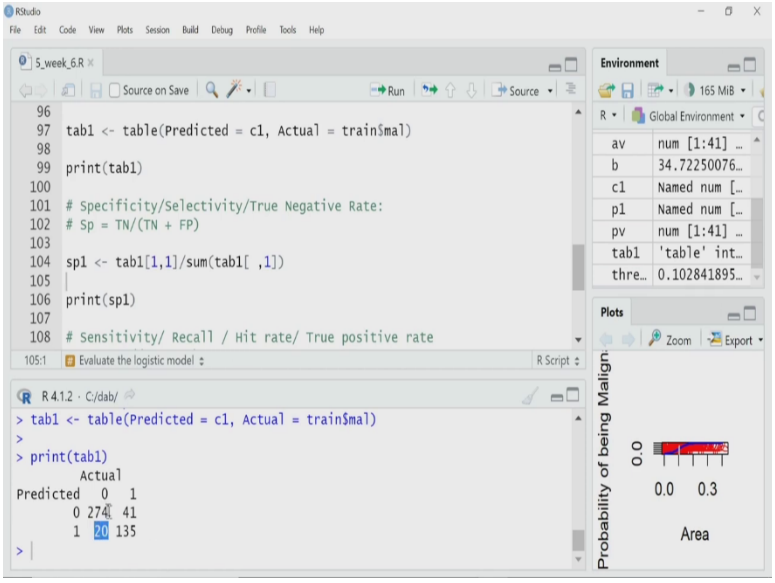
```
> tab1 <- table(Predicted = c1, Actual = train$mal)
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
>
```



av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
tab1	'table' int...
thre...	0.102841895...

```
tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
```

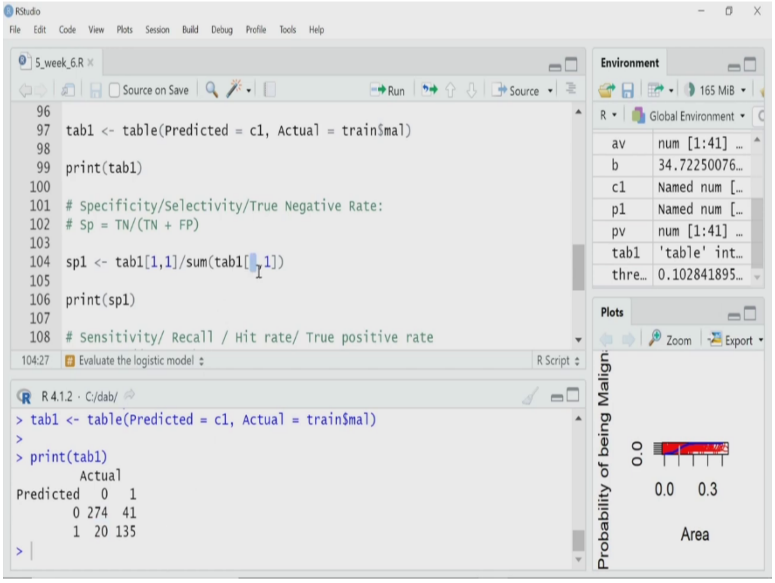
```
> tab1 <- table(Predicted = c1, Actual = train$mal)
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
>
```



av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
tab1	'table' int...
thre...	0.102841895...

```
tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
```

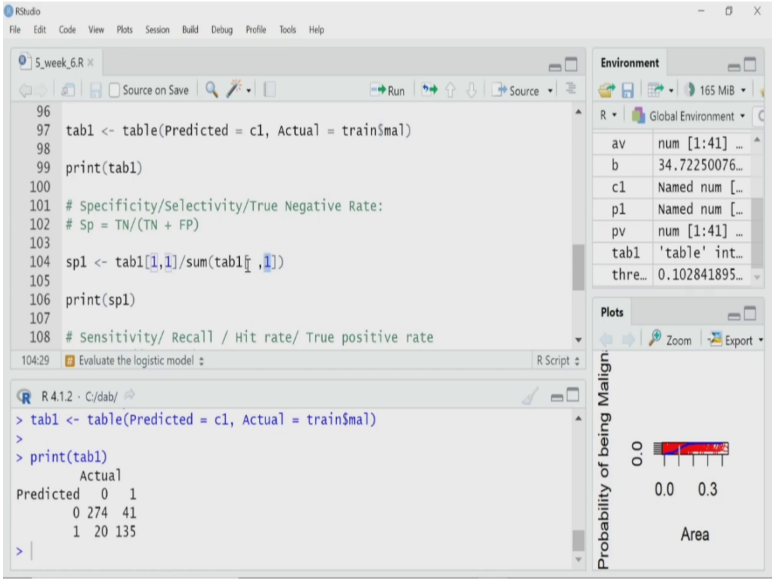
```
> tab1 <- table(Predicted = c1, Actual = train$mal)
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
>
```



```
tabl <- table(Predicted = c1, Actual = train$mal)

print(tabl)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

spl <- tabl[1,1]/sum(tabl[ ,1])

print(spl)

# Sensitivity/ Recall / Hit rate/ True positive rate
```

```
> tabl <- table(Predicted = c1, Actual = train$mal)
>
> print(tabl)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
>
```

av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
tabl	'table' int...
thre...	0.102841895...



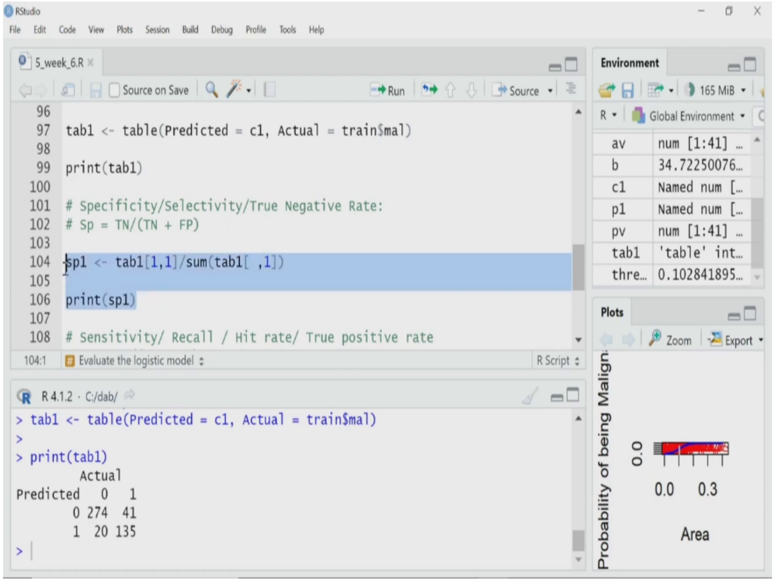
```
tabl <- table(Predicted = c1, Actual = train$mal)

print(tabl)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

spl <- tabl[1,1]/sum(tabl[ ,1])

print(spl)

# Sensitivity/ Recall / Hit rate/ True positive rate
```

```
> tabl <- table(Predicted = c1, Actual = train$mal)
>
> print(tabl)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
>
```

av	num [1:41] ...
b	34.72250076...
c1	Named num [...
p1	Named num [...
pv	num [1:41] ...
tabl	'table' int...
thre...	0.102841895...

So, I am calling true negative is at 1 1, one row, column, so that is true negative because both of them are 0, actually is also 0, predictive is also 0, so true negative. So, I am calling tab 1, I am fetching 1 1, value, that means we will fetch 274. And then what I am doing, I am summing all these two value of the first column 274 plus 20. That is why I am writing tab, no row number is given, the column number is given. So, it will sum all the rows for column 1, and I will print that data.

(Refer Slide Time: 22:18)



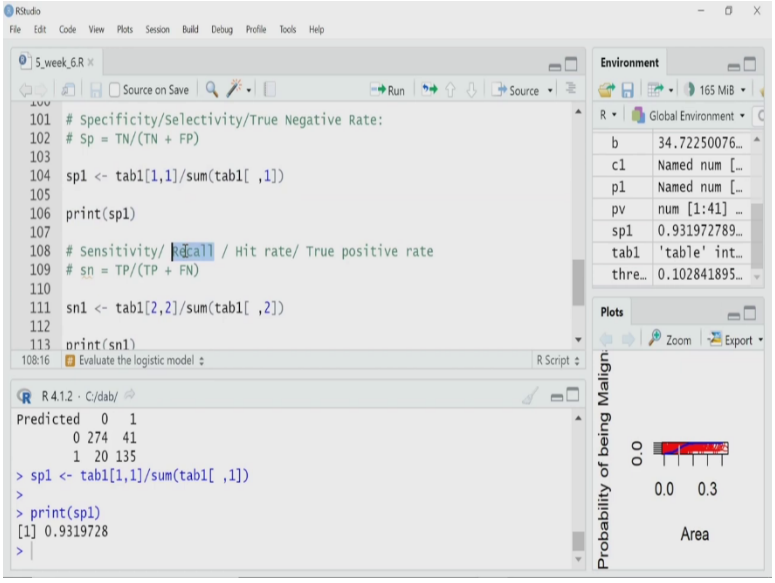
```
tab1 <- table(Predicted = c1, Actual = train$mal)

print(tab1)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
```

```
> tab1 <- table(Predicted = c1, Actual = train$mal)
>
> print(tab1)
         Actual
Predicted   0   1
        0 274  41
        1  20 135
>
```

av	num [1:41] …
b	34.72250076…
c1	Named num […
p1	Named num […
pv	num [1:41] …
tab1	'table' int…
thre…	0.102841895…

So, let me check what is the specificity? So, specificity 0.93, that means the specificity is 93 percent, is quite a good one.

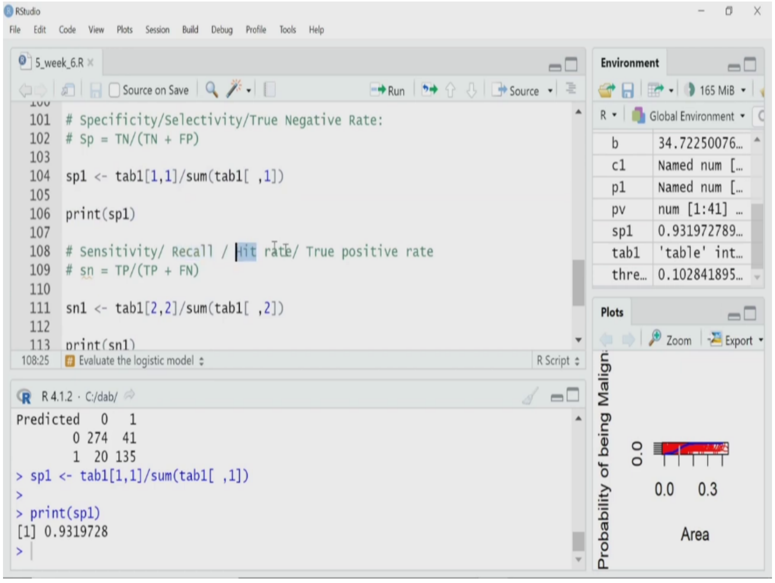
(Refer Slide Time: 22:28)



```
# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])

print(sn1)
```

```
Predicted   0   1
        0 274  41
        1  20 135
> sp1 <- tab1[1,1]/sum(tab1[ ,1])
>
> print(sp1)
[1] 0.9319728
>
```

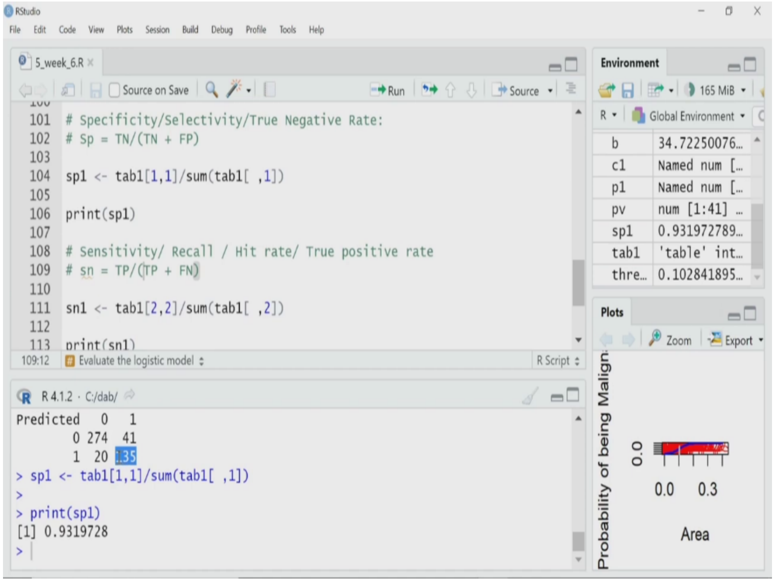
b	34.72250076…
c1	Named num […
p1	Named num […
pv	num [1:41] …
sp1	0.931972789…
tab1	'table' int…
thre…	0.102841895…



```
# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])

print(sn1)
```

```
Predicted   0   1
        0 274  41
        1  20 135
> sp1 <- tab1[1,1]/sum(tab1[ ,1])
>
> print(sp1)
[1] 0.9319728
>
```

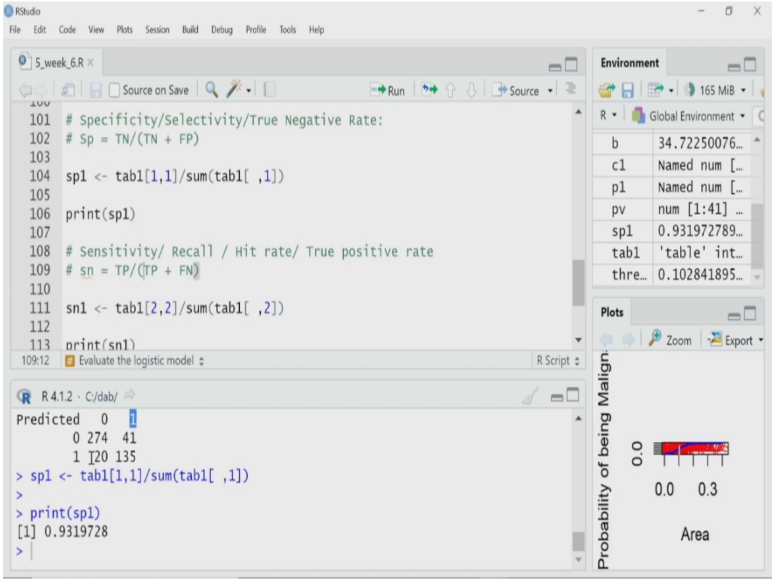
b	34.72250076…
c1	Named num […
p1	Named num […
pv	num [1:41] …
sp1	0.931972789…
tab1	'table' int…
thre…	0.102841895…



```
# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])

print(sn1)
```

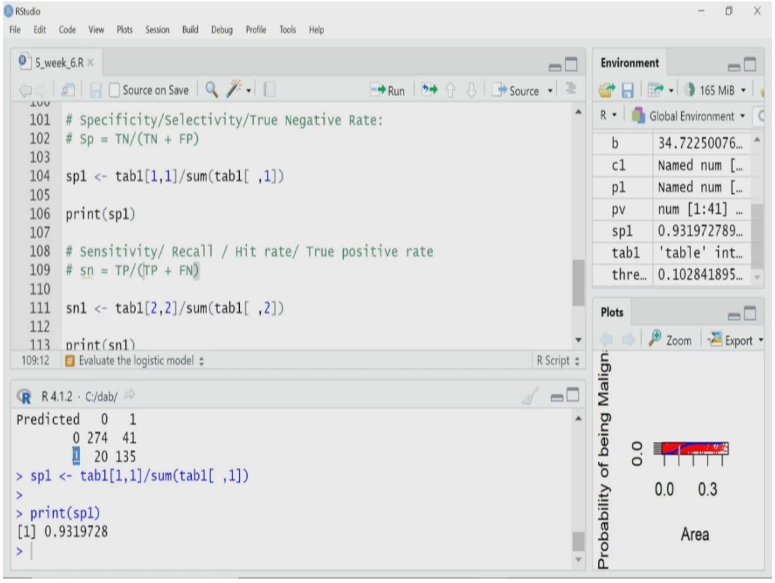
```
Predicted   0   1
        0 274  41
        1  20 135
> sp1 <- tab1[1,1]/sum(tab1[ ,1])
>
> print(sp1)
[1] 0.9319728
>
```

b	34.72250076…
c1	Named num […
p1	Named num […
pv	num [1:41] …
sp1	0.931972789…
tab1	'table' int…
thre…	0.102841895…



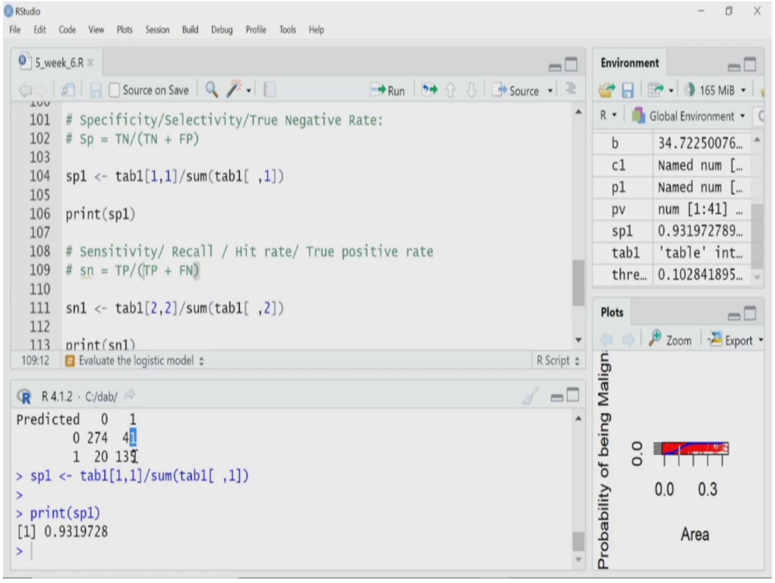
```
# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])

print(sn1)
```

```
Predicted   0   1
        0 274  41
        1  20 135
> sp1 <- tab1[1,1]/sum(tab1[ ,1])
>
> print(sp1)
[1] 0.9319728
>
```



```
# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])

print(sn1)
```

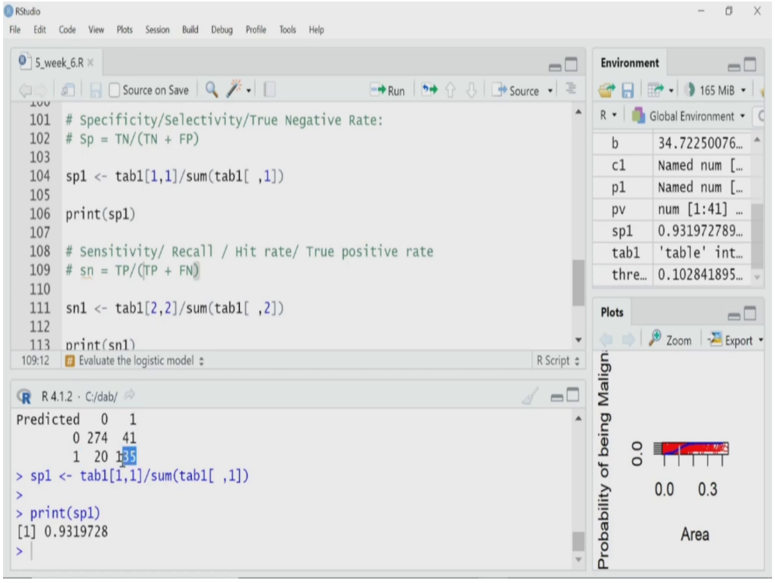
```
Predicted   0   1
        0 274  41
        1  20 135
> sp1 <- tab1[1,1]/sum(tab1[ ,1])
>
> print(sp1)
[1] 0.9319728
>
```



```
# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])

print(sn1)
```

```
Predicted   0   1
        0 274  41
        1  20 135
> sp1 <- tab1[1,1]/sum(tab1[ ,1])
>
> print(sp1)
[1] 0.9319728
>
```

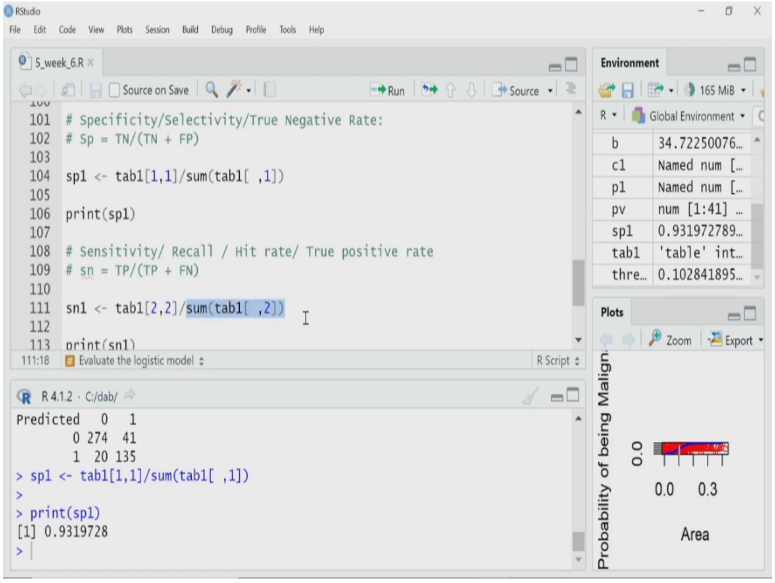
b	34.72250076…
c1	Named num […
p1	Named num […
pv	num [1:41] …
sp1	0.931972789…
tab1	'table' int…
thre…	0.102841895…



```
# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])

print(sn1)
```

```
Predicted   0   1
        0 274  41
        1  20 135
> sp1 <- tab1[1,1]/sum(tab1[ ,1])
>
> print(sp1)
[1] 0.9319728
>
```

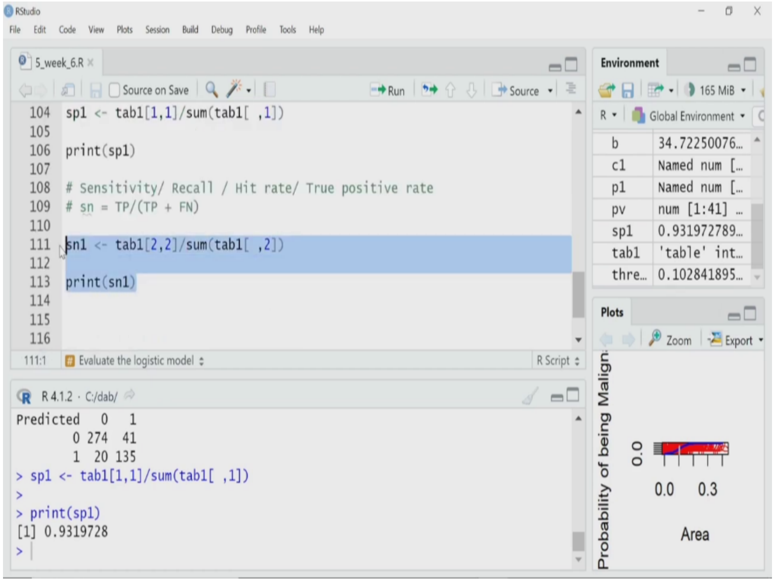
b	34.72250076…
c1	Named num […
p1	Named num […
pv	num [1:41] …
sp1	0.931972789…
tab1	'table' int…
thre…	0.102841895…



```
# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp1 <- tab1[1,1]/sum(tab1[ ,1])

print(sp1)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn1 <- tab1[2,2]/sum(tab1[ ,2])

print(sn1)
```

```
Predicted   0   1
        0 274  41
        1  20 135
> sp1 <- tab1[1,1]/sum(tab1[ ,1])
>
> print(sp1)
[1] 0.9319728
>
```

b	34.72250076…
c1	Named num […
p1	Named num […
pv	num [1:41] …
sp1	0.931972789…
tab1	'table' int…
thre…	0.102841895…



```
spl <- tabl[1,1]/sum(tabl[ ,1])

print(spl)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

snl <- tabl[2,2]/sum(tabl[ ,2])

print(snl)
```

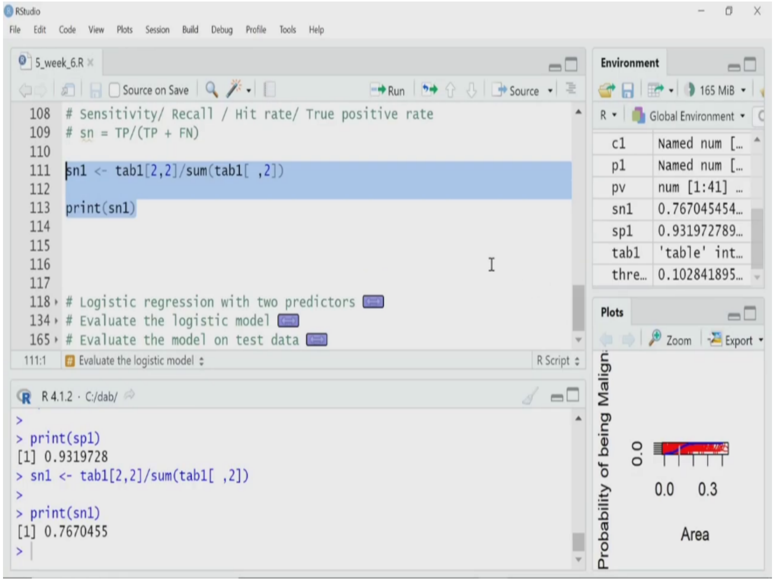
```
Predicted   0   1
        0 274  41
        1  20 135
> spl <- tabl[1,1]/sum(tabl[ ,1])
>
> print(spl)
[1] 0.9319728
>
```

b	34.72250076…
c1	Named num […
p1	Named num […
pv	num [1:41] …
spl	0.931972789…
tabl	'table' int…
thre…	0.102841895…

Let us check the sensitivity. Sensitivity is also called a recall value or hit rate or true positive rate. It is defined as the ratio between true positives and true positives plus false negative. So, how do I calculate it again, I use the same indexing of table so, I am taking table 2 2, that means table 2 2, means, second row, second column, this is 137, this is the true positive, because actual is also 1, predicted is also 1, divided by sum of the values of this second column, because this is true positive this one is false negative.

So, I have to sum this together, and I am doing the sum together and that is in the denominator. Let me calculate and print that. So, the sensitivity is 0.76, that means 76 percent is not very high, but with this data set that I have curated I believe this is quite good enough.

(Refer Slide Time: 23:28)



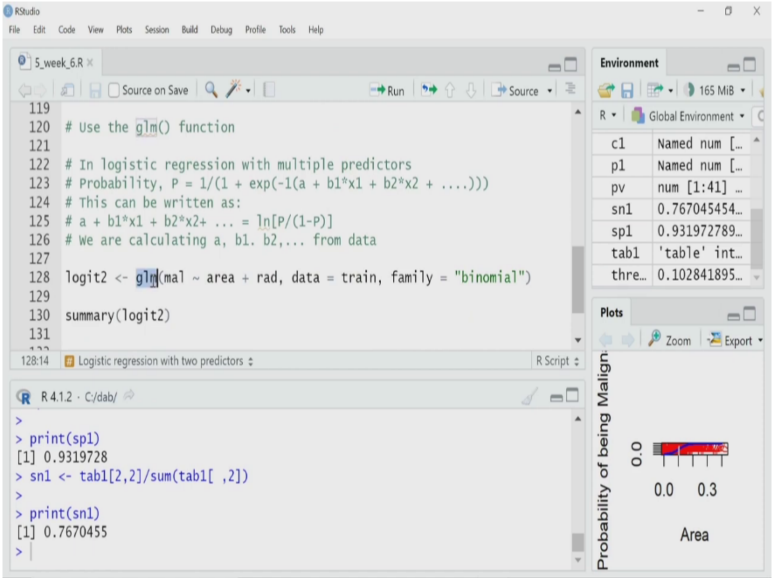
```
# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

snl <- tabl[2,2]/sum(tabl[ ,2])

print(snl)




# Logistic regression with two predictors
# Evaluate the logistic model
# Evaluate the model on test data
```

```
>
> print(spl)
[1] 0.9319728
> snl <- tabl[2,2]/sum(tabl[ ,2])
>
> print(snl)
[1] 0.7670455
>
```

c1	Named num […
p1	Named num […
pv	num [1:41] …
snl	0.767045454…
spl	0.931972789…
tabl	'table' int…
thre…	0.102841895…

Now, what I have done till now, I have created a model logistic model where I have a binary problem, yes no, malignant or not and I have only one predictor. But remember in my data set I have two predictor, mean area of the nucleus of cell as well as the radius also. So, now, I want to create a model, regression model, again logistic regression model again, using both the predictors.

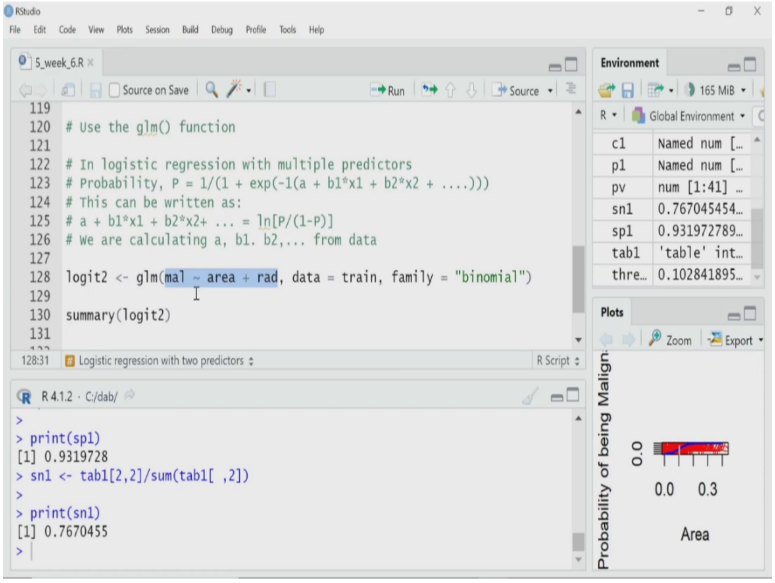
(Refer Slide Time: 23:50)



```
# Use the glm() function

# In logistic regression with multiple predictors
# Probability, P = 1/(1 + exp(-1(a + b1*x1 + b2*x2 + ....)))
# This can be written as:
# a + b1*x1 + b2*x2+ ... = ln[P/(1-P)]
# We are calculating a, b1. b2,... from data

logit2 <- glm(mal ~ area + rad, data = train, family = "binomial")

summary(logit2)
```

```
>
> print(spl)
[1] 0.9319728
> snl <- tabl[2,2]/sum(tabl[ ,2])
>
> print(snl)
[1] 0.7670455
>
```

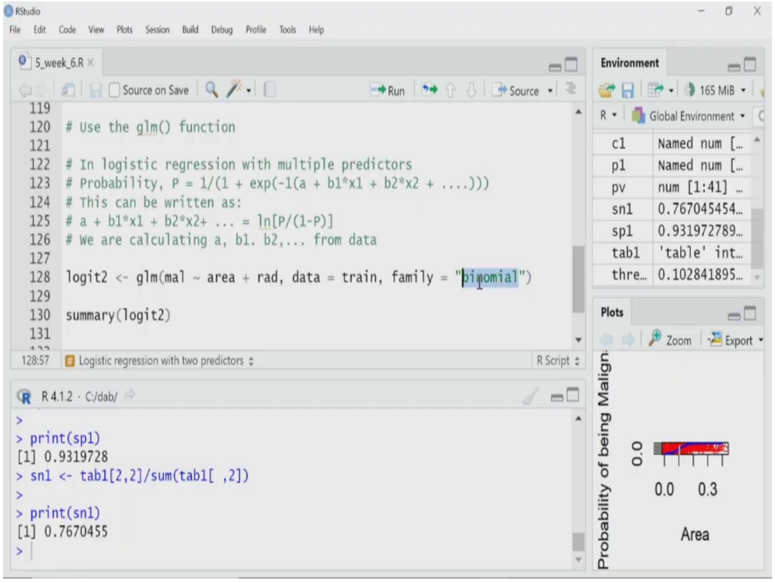
cl	Named num [...
pl	Named num [...
pv	num [1:41] ...
snl	0.767045454...
spl	0.931972789...
tabl	'table' int...
thre...	0.102841895...



```
# Use the glm() function

# In logistic regression with multiple predictors
# Probability, P = 1/(1 + exp(-1(a + b1*x1 + b2*x2 + ....)))
# This can be written as:
# a + b1*x1 + b2*x2+ ... = ln[P/(1-P)]
# We are calculating a, b1. b2,... from data

logit2 <- glm(mal ~ area + rad, data = train, family = "binomial")

summary(logit2)
```

```
>
> print(spl)
[1] 0.9319728
> snl <- tabl[2,2]/sum(tabl[ ,2])
>
> print(snl)
[1] 0.7670455
>
```

cl	Named num [...
pl	Named num [...
pv	num [1:41] ...
snl	0.767045454...
spl	0.931972789...
tabl	'table' int...
thre...	0.102841895...



RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
5_week_6.R
Source on Save
Run
Source
Environment
165 MiB
Global Environment
Logistic regression with two predictors
R Script
R 4.1.2 · C:/dab/
Plots
Zoom
Export
Probability of being Malign
0.0
0.0
0.3
Area

```
# Use the glm() function

# In logistic regression with multiple predictors
# Probability, P = 1/(1 + exp(-1(a + b1*x1 + b2*x2 + ....)))
# This can be written as:
# a + b1*x1 + b2*x2+ ... = ln[P/(1-P)]
# We are calculating a, b1. b2,... from data

logit2 <- glm(mal ~ area + rad, data = train, family = "binomial")

summary(logit2)
```

```
c1     Named num [...
p1     Named num [...
pv     num [1:41] ...
sn1    0.767045454...
sp1    0.931972789...
tab1   'table' int...
thre...  0.102841895...
```

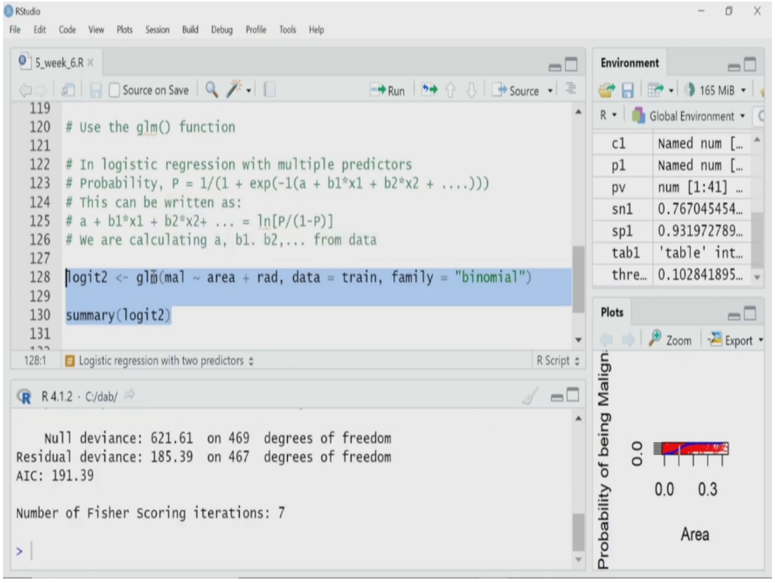
```
>
> print(sp1)
[1] 0.9319728
> sn1 <- tab1[2,2]/sum(tab1[ ,2])
>
> print(sn1)
[1] 0.7670455
>
```

logit2 ← glm(mal ~ area + rad, data = train, family = “binomial”)

summary(logit2)

I will use the same GLM function. But what I will do I will define the model differently. So, what I am doing here, so here I am calling the GLM function, and I am telling the model, model is mal tilda area plus red, the radius variable. So, I am asking the GLM function that see now you have to do the logistic regression with respect to two predictor area and radius, data is the same training data set and the family's binomial because I want it to perform logistic regression.

(Refer Slide Time: 24:21)

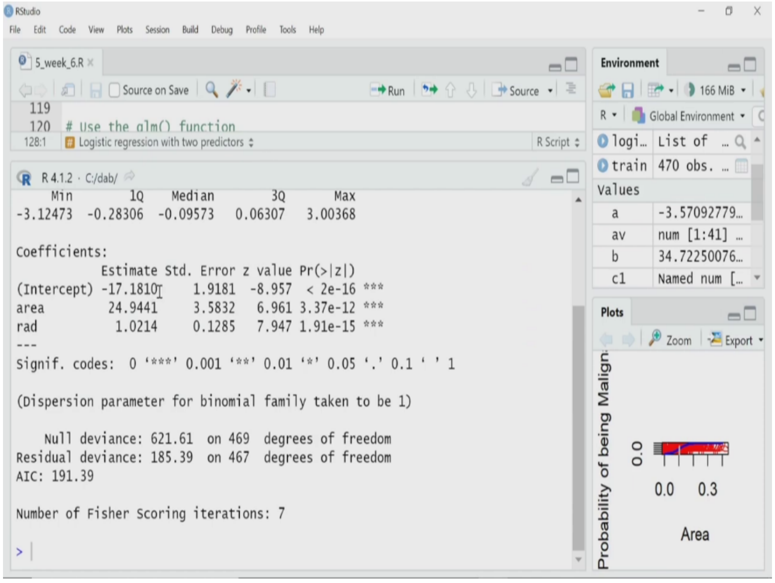


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
5_week_6.R
Source on Save
Run
Source
Environment
165 MiB
Global Environment
Logistic regression with two predictors
R Script
R 4.1.2 · C:/dab/
Plots
Zoom
Export
Probability of being Malign
0.0
0.0
0.3
Area

```
# Use the glm() function

# In logistic regression with multiple predictors
# Probability, P = 1/(1 + exp(-1(a + b1*x1 + b2*x2 + ....)))
# This can be written as:
# a + b1*x1 + b2*x2+ ... = ln[P/(1-P)]
# We are calculating a, b1. b2,... from data

logit2 <- glm(mal ~ area + rad, data = train, family = "binomial")

summary(logit2)
```

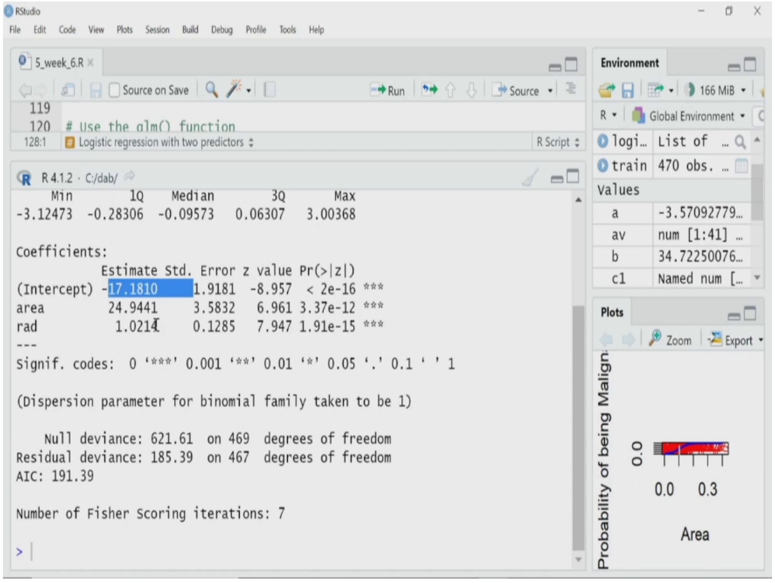
```
c1     Named num [...
p1     Named num [...
pv     num [1:41] ...
sn1    0.767045454...
sp1    0.931972789...
tab1   'table' int...
thre...  0.102841895...
```

```
    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7

>
```



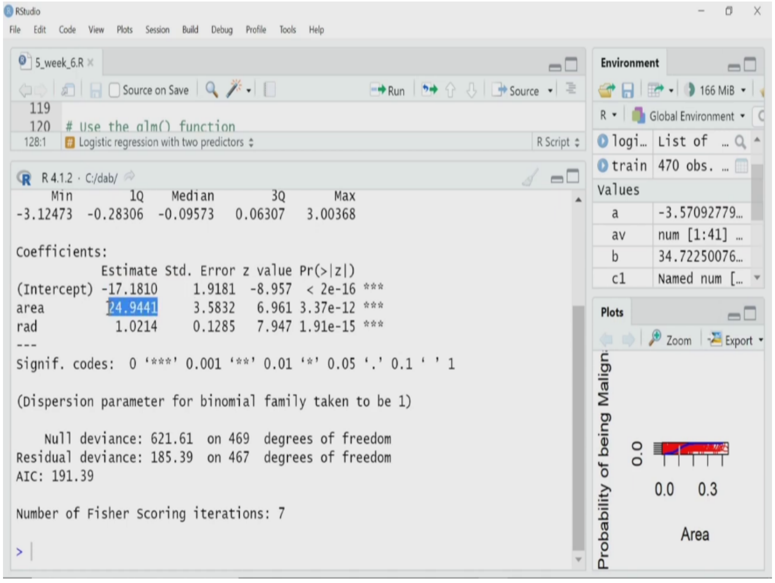
```
# Use the glm() function
```

```
     Min        1Q    Median        3Q       Max
-3.12473  -0.28306  -0.09573   0.06307   3.00368

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -17.1810     1.9181  -8.957  < 2e-16 ***
area         24.9441     3.5832   6.961 3.37e-12 ***
rad           1.0214     0.1285   7.947 1.91e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7

>
```



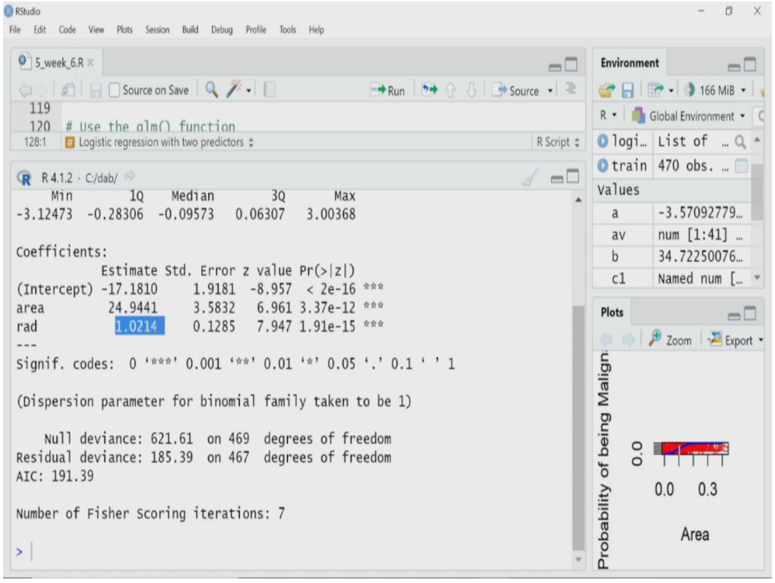
```
# Use the glm() function
```

```
     Min        1Q    Median        3Q       Max
-3.12473  -0.28306  -0.09573   0.06307   3.00368

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -17.1810     1.9181  -8.957  < 2e-16 ***
area         24.9441     3.5832   6.961 3.37e-12 ***
rad           1.0214     0.1285   7.947 1.91e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7

>
```



```
# Use the glm() function
```

```
     Min        1Q    Median        3Q       Max
-3.12473  -0.28306  -0.09573   0.06307   3.00368

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -17.1810     1.9181  -8.957  < 2e-16 ***
area         24.9441     3.5832   6.961 3.37e-12 ***
rad           1.0214     0.1285   7.947 1.91e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7

>
```



```
# Use the glm() function
```

logi…	List of …
train	470 obs. …

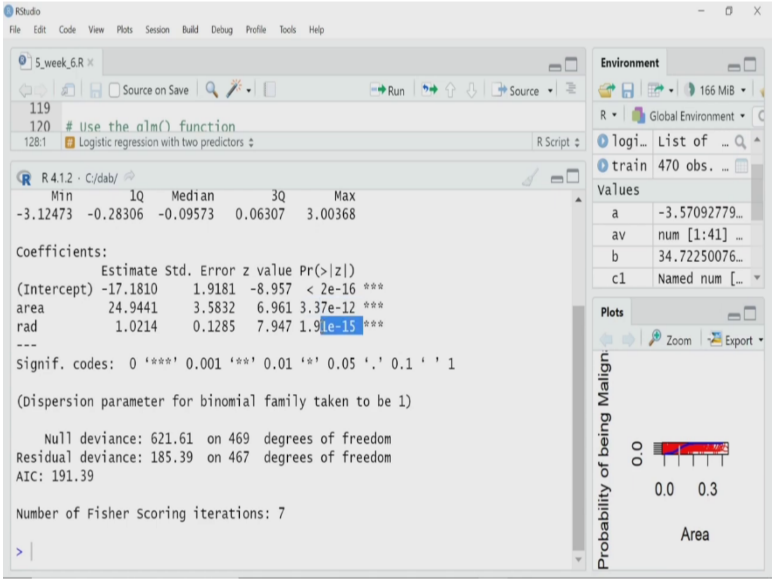
Values

a	-3.57092779…
av	num [1:41] …
b	34.72250076…
c1	Named num […

```
     Min       1Q    Median       3Q      Max
-3.12473  -0.28306  -0.09573  0.06307  3.00368

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -17.1810     1.9181  -8.957  < 2e-16 ***
area         24.9441     3.5832   6.961 3.37e-12 ***
rad           1.0214     0.1285   7.947 1.91e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7

>
```

So, here I execute it, let me clean the console a bit. So, I execute the regression logistic regression and I print the summary. So, here the first column is for the estimated parameter value. So, intercept is minus 17.18. Area coefficient is 24. And the coefficient for radius is 1.02, That means both area and radius are positively affecting whether a sample will be malignant or not. If the area increases, and the radius also increases that means it will be more and more malignant.

(Refer Slide Time: 25:02)



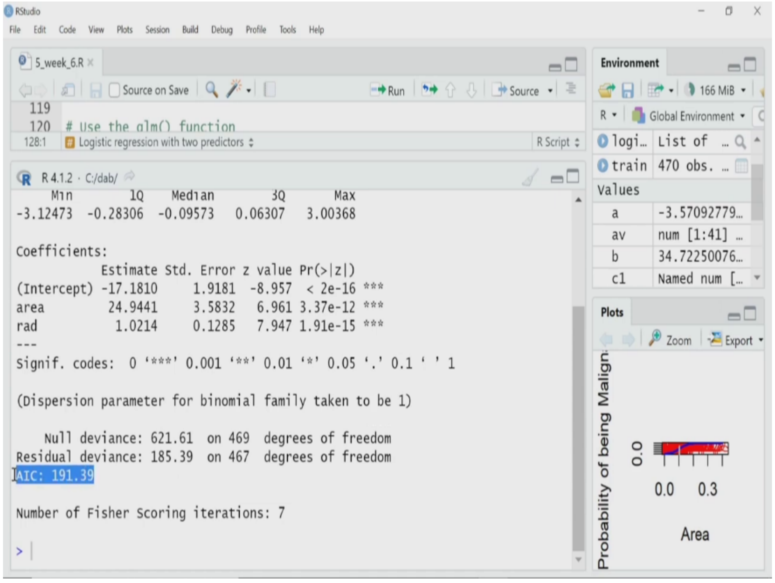
```
# Use the glm() function
```

logi…	List of …
train	470 obs. …

Values

a	-3.57092779…
av	num [1:41] …
b	34.72250076…
c1	Named num […

```
     Min       1Q    Median       3Q      Max
-3.12473  -0.28306  -0.09573  0.06307  3.00368

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -17.1810     1.9181  -8.957  < 2e-16 ***
area         24.9441     3.5832   6.961 3.37e-12 ***
rad           1.0214     0.1285   7.947 1.91e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7

>
```



```
# Use the glm() function
```

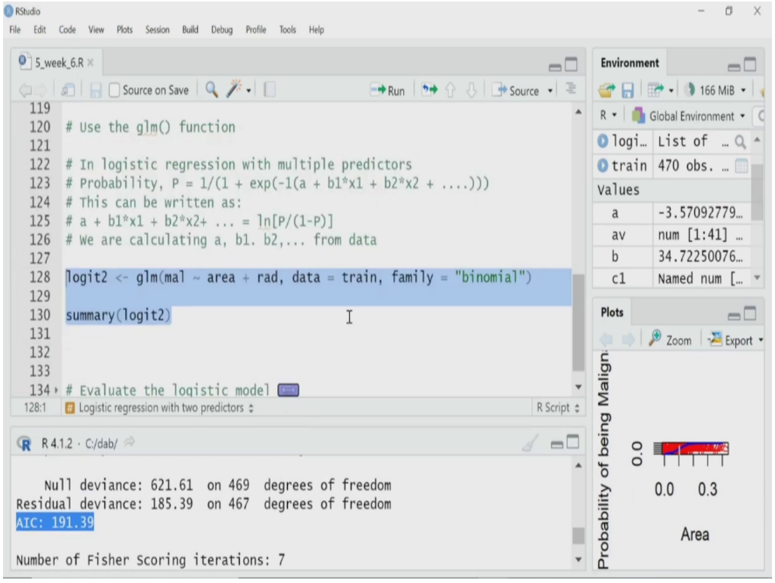
```
     Min        1Q    Median        3Q       Max
-3.12473  -0.28306  -0.09573   0.06307   3.00368

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -17.1810     1.9181  -8.957  < 2e-16 ***
area         24.9441     3.5832   6.961 3.37e-12 ***
rad           1.0214     0.1285   7.947 1.91e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7

>
```

logi…	List of …
train	470 obs. …
Values	
a	-3.57092779…
av	num [1:41] …
b	34.72250076…
c1	Named num […

And statistical test shows that all these three coefficients are also stastistically significant. And now I have to go back to something called the AIC. AIC as I said, is a criteria that I can use to compare between two models. So, Akaike information criteria can be used to compare between two models, which has different number of parameters.

In my previous model, it was something 300 something, I missed it, I have already cleared it, but you can try and check it again, it is around something around 300. Whereas for this model, I have a one extra parameter AIC is 191. The lower AIC value is a better value, that means I can say from this AIC value this second logistic model with two predictor, both the area and the radius of the, radius of nucleus is a better model than the previous one.

(Refer Slide Time: 26:02)



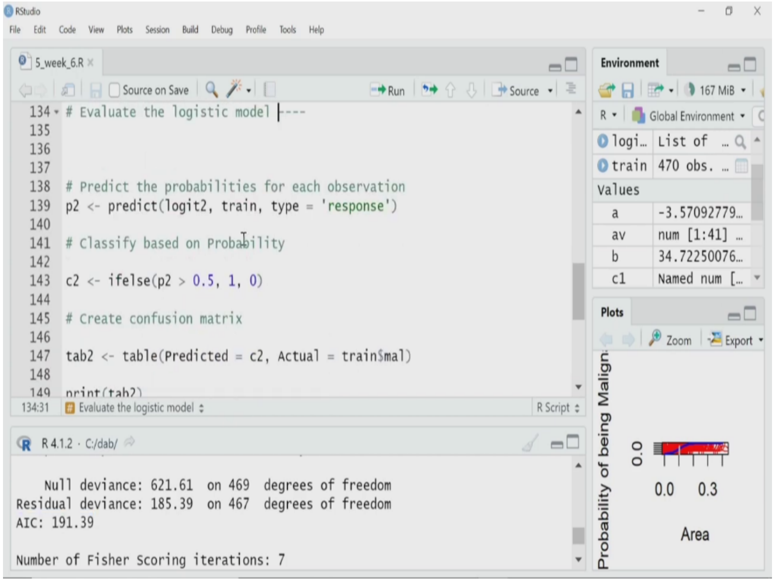
```
# Use the glm() function

# In logistic regression with multiple predictors
# Probability, P = 1/(1 + exp(-1(a + b1*x1 + b2*x2 + ....)))
# This can be written as:
# a + b1*x1 + b2*x2+ ... = ln[P/(1-P)]
# We are calculating a, b1. b2,... from data

logit2 <- glm(mal ~ area + rad, data = train, family = "binomial")

summary(logit2)



# Evaluate the logistic model
```

```
    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7
```

logi…	List of …
train	470 obs. …
Values	
a	-3.57092779…
av	num [1:41] …
b	34.72250076…
c1	Named num […

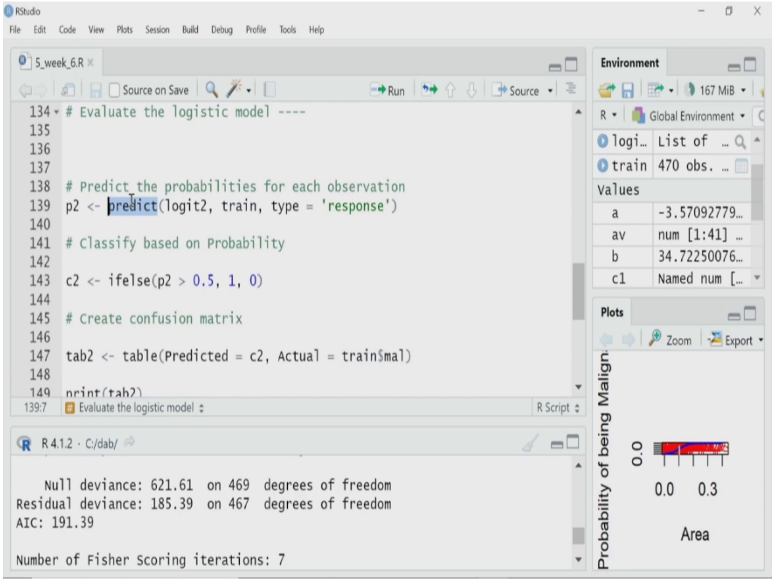
So, I have created the second logistic regression model, logistic regression classifier. And these are two predictor things, so, I will not be able to plot this the way I have plotted the previous one, but I can do the evaluation of the model using the same confusion matrix. So, I will do that.

(Refer Slide Time: 26:20)



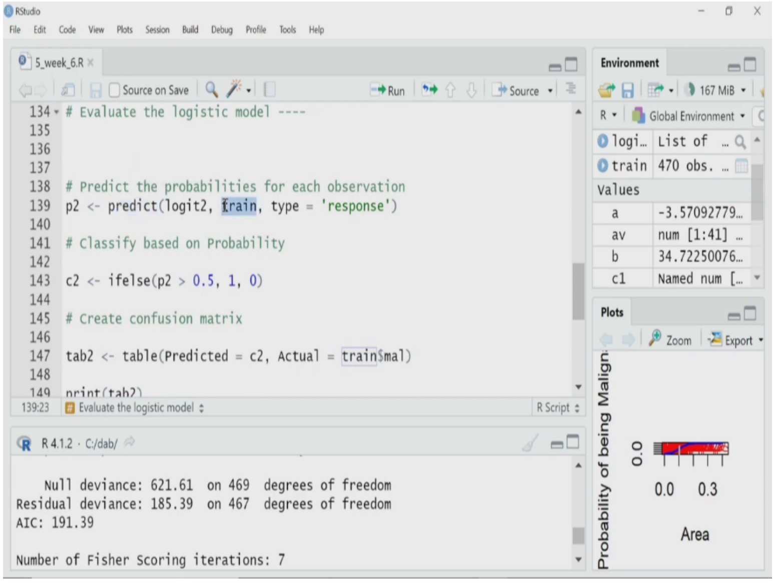
```
# Evaluate the logistic model ----



# Predict the probabilities for each observation
p2 <- predict(logit2, train, type = 'response')

# Classify based on Probability

c2 <- ifelse(p2 > 0.5, 1, 0)

# Create confusion matrix

tab2 <- table(Predicted = c2, Actual = train$mal)

print(tab2)
```

```
    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7
```



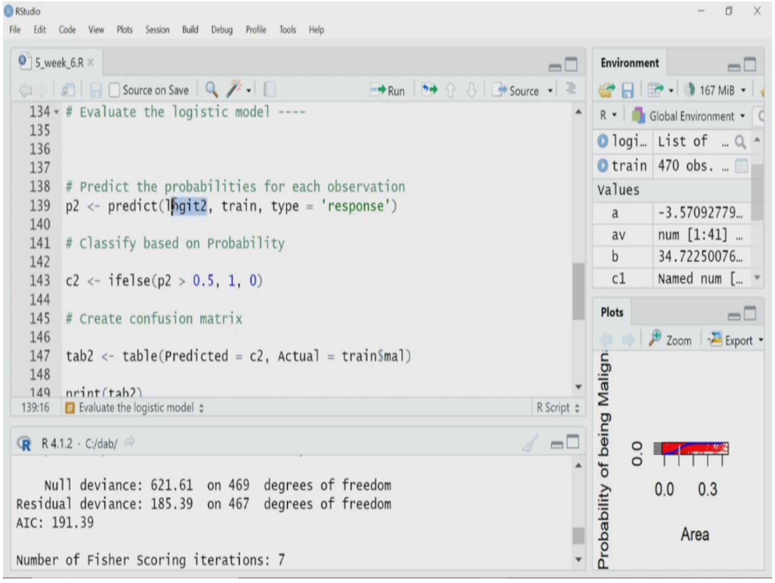
```
# Evaluate the logistic model ----



# Predict the probabilities for each observation
p2 <- predict(logit2, train, type = 'response')

# Classify based on Probability

c2 <- ifelse(p2 > 0.5, 1, 0)

# Create confusion matrix

tab2 <- table(Predicted = c2, Actual = train$mal)

print(tab2)
```

```
    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7
```



```
# Evaluate the logistic model ----



# Predict the probabilities for each observation
p2 <- predict(logit2, train, type = 'response')

# Classify based on Probability

c2 <- ifelse(p2 > 0.5, 1, 0)

# Create confusion matrix

tab2 <- table(Predicted = c2, Actual = train$mal)

print(tab2)
```

```
    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7
```



```
# Evaluate the logistic model ----



# Predict the probabilities for each observation
p2 <- predict(logit2, train, type = 'response')

# Classify based on Probability

c2 <- ifelse(p2 > 0.5, 1, 0)

# Create confusion matrix

tab2 <- table(Predicted = c2, Actual = train$mal)

print(tab2)
```

```
    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7
```

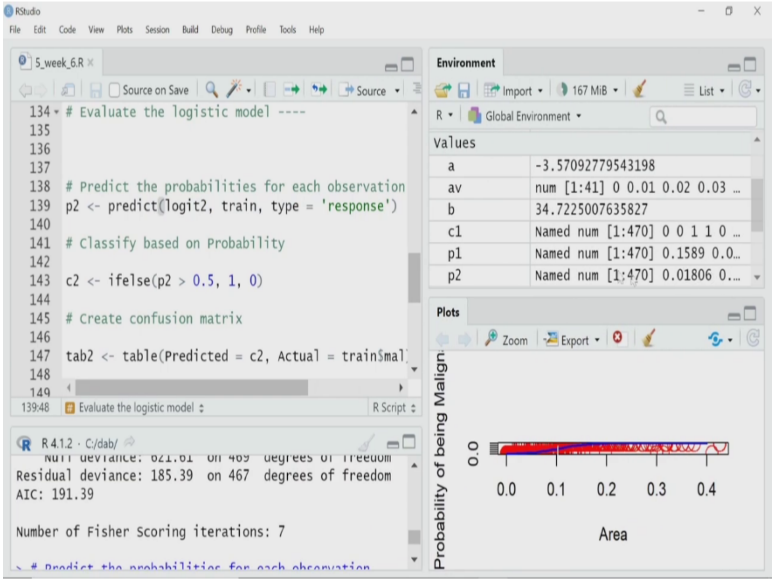
p2 ← predict(logit2, train, type = “response”)

c1 ← ifelse(p2 > 0.5, 1, 0)

tab2 ← table(Predicted = c2, Actual = train$mal)

So now, I will create the p 2 vector, the probability of being malignant for my training data set again. And what I will use? I will use the predict function. So, the input will be training data set train, but the model now will be logit 2, because this is the second model just now, I created and I want to calculate the response.

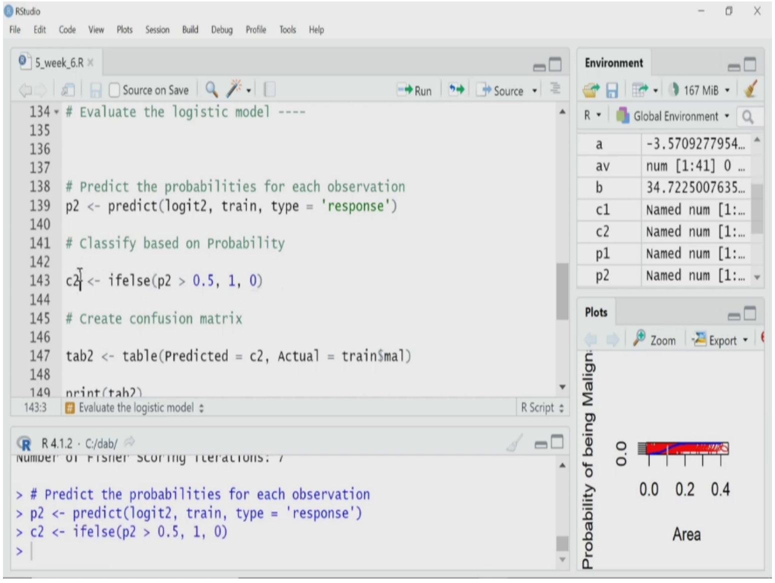
(Refer Slide Time: 26:45)



```
# Evaluate the logistic model ----



# Predict the probabilities for each observation
p2 <- predict(logit2, train, type = 'response')

# Classify based on Probability

c2 <- ifelse(p2 > 0.5, 1, 0)

# Create confusion matrix

tab2 <- table(Predicted = c2, Actual = train$mal
```

```
    Null deviance: 621.61  on 469  degrees of freedom
Residual deviance: 185.39  on 467  degrees of freedom
AIC: 191.39

Number of Fisher Scoring iterations: 7

> # Predict the probabilities for each observation
```

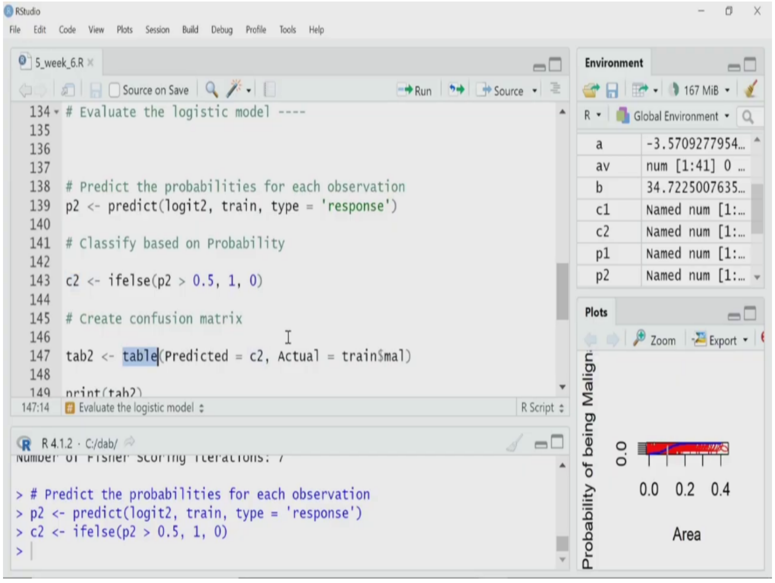
```
Values
a     -3.57092779543198
av    num [1:41] 0 0.01 0.02 0.03 ...
b     34.7225007635827
c1    Named num [1:470] 0 0 1 1 0 ...
p1    Named num [1:470] 0.1589 0.0...
p2    Named num [1:470] 0.01806 0....
```



```
# Evaluate the logistic model ----



# Predict the probabilities for each observation
p2 <- predict(logit2, train, type = 'response')

# Classify based on Probability

c2 <- ifelse(p2 > 0.5, 1, 0)

# Create confusion matrix

tab2 <- table(Predicted = c2, Actual = train$mal)

print(tab2)
```

```
Number of Fisher Scoring iterations: 7

> # Predict the probabilities for each observation
> p2 <- predict(logit2, train, type = 'response')
> c2 <- ifelse(p2 > 0.5, 1, 0)
> 
```

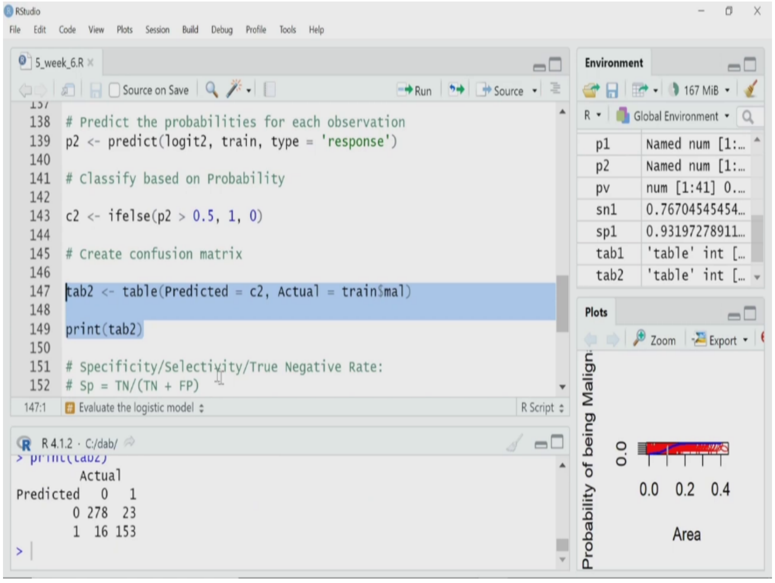
```
a     -3.5709277954...
av    num [1:41] 0 ...
b     34.7225007635...
c1    Named num [1:...
c2    Named num [1:...
p1    Named num [1:...
p2    Named num [1:...
```



```
# Evaluate the logistic model ----



# Predict the probabilities for each observation
p2 <- predict(logit2, train, type = 'response')

# Classify based on Probability

c2 <- ifelse(p2 > 0.5, 1, 0)

# Create confusion matrix

tab2 <- table(Predicted = c2, Actual = train$mal)

print(tab2)
```

```
Number of Fisher Scoring iterations: 7

> # Predict the probabilities for each observation
> p2 <- predict(logit2, train, type = 'response')
> c2 <- ifelse(p2 > 0.5, 1, 0)
> 
```

```
a     -3.5709277954...
av    num [1:41] 0 ...
b     34.7225007635...
c1    Named num [1:...
c2    Named num [1:...
p1    Named num [1:...
p2    Named num [1:...
```



```r
# Predict the probabilities for each observation
p2 <- predict(logit2, train, type = 'response')

# Classify based on Probability

c2 <- ifelse(p2 > 0.5, 1, 0)

# Create confusion matrix

tab2 <- table(Predicted = c2, Actual = train$mal)

print(tab2)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)
```

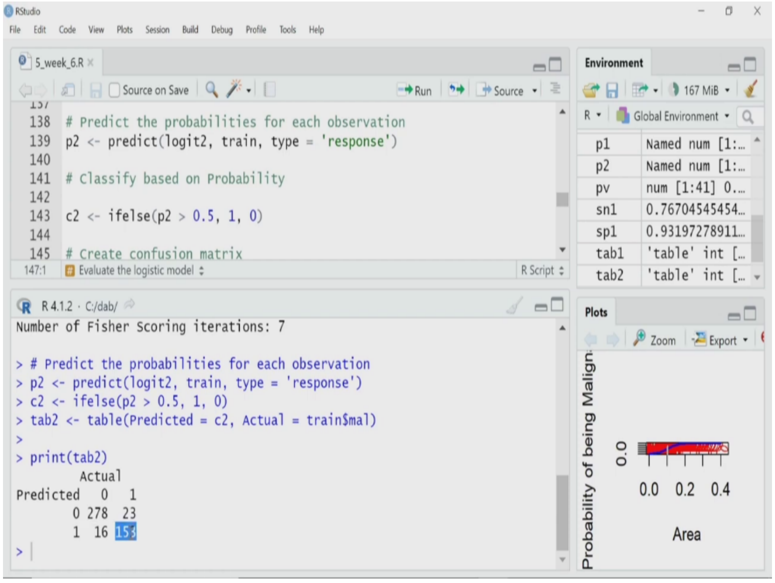
```
> print(tab2)
         Actual
Predicted   0   1
        0 278  23
        1  16 153
>
```

p1	Named num [1:…
p2	Named num [1:…
pv	num [1:41] 0.…
sn1	0.76704545454…
sp1	0.93197278911…
tab1	'table' int […
tab2	'table' int […

So, I perform that prediction. So, the p 2 is created, you can see p 2 are fractions, those are the probability for each of the sample to be malignant as per this new model. Now, I am again calling if else function. If this probability for a sample is bigger than 0.5, my threshold that I have decided, I will call it malignant, that means 1, otherwise, I will call it 0. So, the labels are 1 and 0.

So, I will store that label data in c 2. Now, I will compare this c 2, the labels for each sample as per this new logistic regression model with respect to the original labels present in my data set. So, I will use that table function to create the matrix the confusion matrix that we have done earlier, and I will print it.

(Refer Slide Time: 27:37)



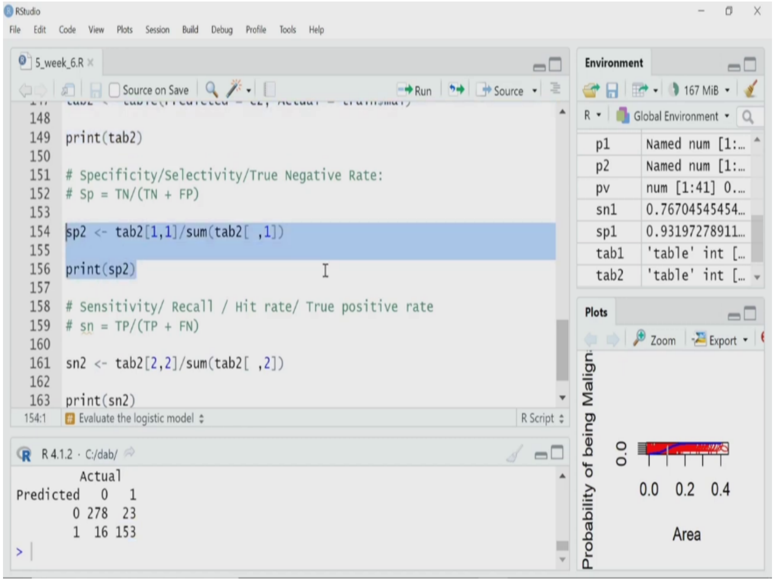
```
# Predict the probabilities for each observation
p2 <- predict(logit2, train, type = 'response')

# Classify based on Probability

c2 <- ifelse(p2 > 0.5, 1, 0)

# Create confusion matrix
```

```
Number of Fisher Scoring iterations: 7

> # Predict the probabilities for each observation
> p2 <- predict(logit2, train, type = 'response')
> c2 <- ifelse(p2 > 0.5, 1, 0)
> tab2 <- table(Predicted = c2, Actual = train$mal)
>
> print(tab2)
         Actual
Predicted   0   1
        0 278  23
        1  16 153
>
```

```
p1    Named num [1:...
p2    Named num [1:...
pv    num [1:41] 0...
sn1   0.76704545454...
sp1   0.93197278911...
tab1  'table' int [...
tab2  'table' int [...
```

So, now, look at the confusion matrix. Now, 278 samples, which were originally benign, are also considered as labelled as benign by the new model, whereas 153 samples which are earlier, we know they are malignant are predicted as a malignant by new model. So, again here, in this case, I can calculate sensitivity and specificity.

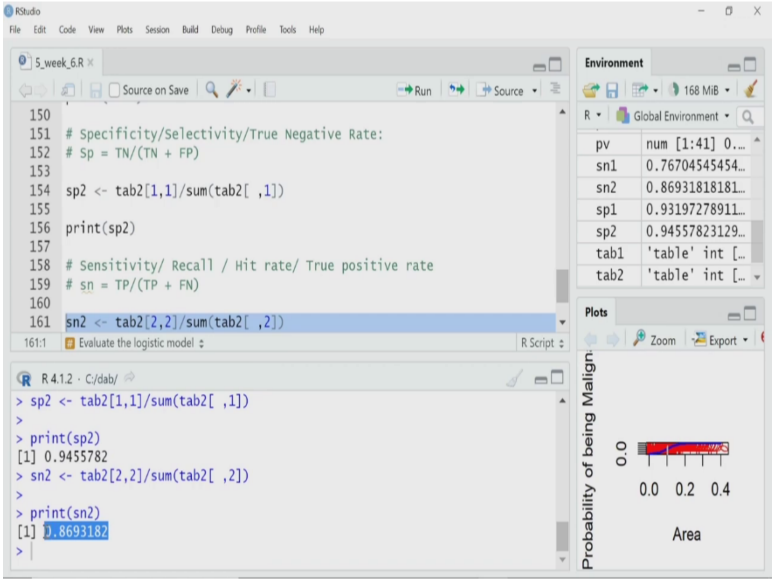
(Refer Slide Time: 27:58)



```
print(tab2)

# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp2 <- tab2[1,1]/sum(tab2[ ,1])

print(sp2)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn2 <- tab2[2,2]/sum(tab2[ ,2])

print(sn2)
```

```
         Actual
Predicted   0   1
        0 278  23
        1  16 153
>
```

```
p1    Named num [1:...
p2    Named num [1:...
pv    num [1:41] 0...
sn1   0.76704545454...
sp1   0.93197278911...
tab1  'table' int [...
tab2  'table' int [...
```



```
# Specificity/Selectivity/True Negative Rate:
# Sp = TN/(TN + FP)

sp2 <- tab2[1,1]/sum(tab2[ ,1])

print(sp2)

# Sensitivity/ Recall / Hit rate/ True positive rate
# sn = TP/(TP + FN)

sn2 <- tab2[2,2]/sum(tab2[ ,2])
```

```
> sp2 <- tab2[1,1]/sum(tab2[ ,1])
>
> print(sp2)
[1] 0.9455782
> sn2 <- tab2[2,2]/sum(tab2[ ,2])
>
> print(sn2)
[1] 0.8693182
>
```

```
pv    num [1:41] 0...
sn1   0.76704545454...
sn2   0.86931818181...
sp1   0.93197278911...
sp2   0.94557823129...
tab1  'table' int [...
tab2  'table' int [...
```

sp2 ← tab2[1,1] / sum(tab2[,1])

print(sp2)

sn2 ← tab2[2,2] / sum(tab2[,2])

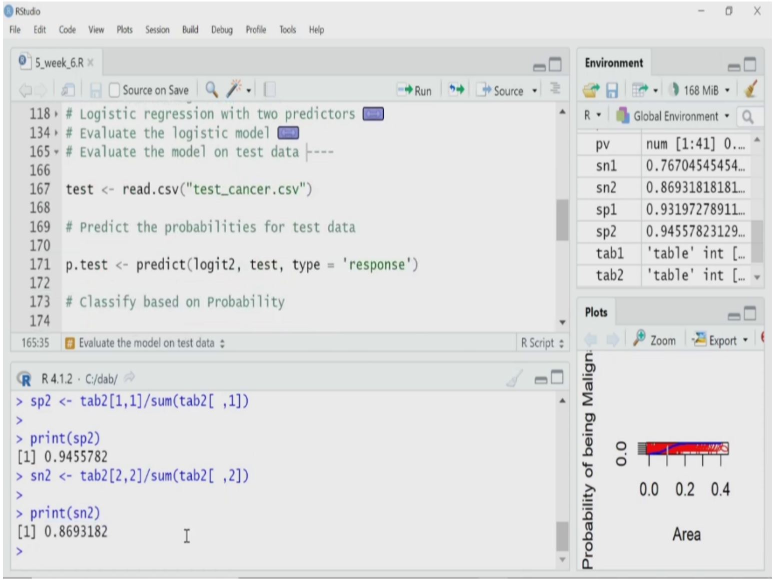
print(sn2)

So, I will calculate that first the specificity, that is true negative rate, true negative rate is 94.55. It was earlier 93 percent, now it is 94.5 percent, slightly increased. Let me calculate the sensitivity for this new model where I have two predictors, sensitivity is nothing but true positive rate, earlier, it was around 76 percent.

Now, you can see here it is 86.9. So that means by adding a new predictor, my sensitivity true positive rate has increased. And in fact, the AIC has also said my model fit is also better, because it has a lower value. So that means I should proceed with my this new second model, second logistic regression where I have to predictors.

And I will end this lecture now by applying this regression model with two predictor logistic regression model on my test data. If you remember I said at the very beginning that some of the data set I have separated and stored as a test sample.

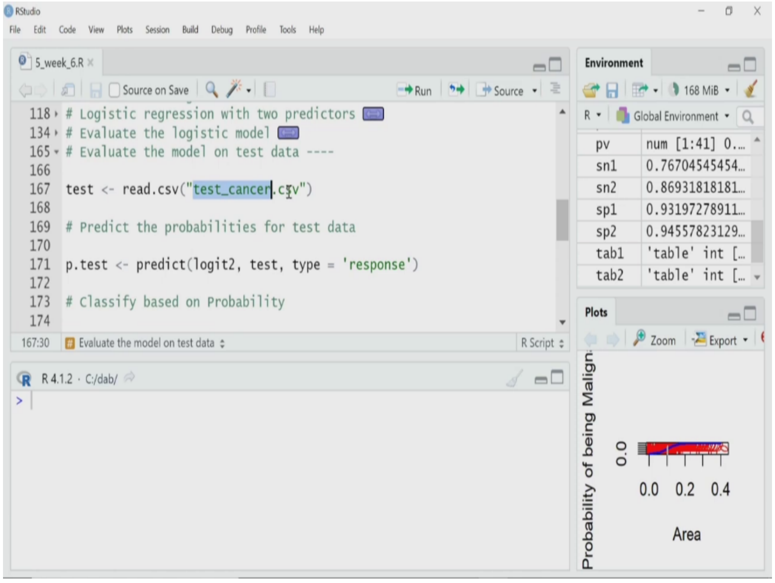
(Refer Slide Time: 29:00)



```
# Logistic regression with two predictors
# Evaluate the logistic model
# Evaluate the model on test data ----

test <- read.csv("test_cancer.csv")

# Predict the probabilities for test data

p.test <- predict(logit2, test, type = 'response')

# Classify based on Probability

```

```
> sp2 <- tab2[1,1]/sum(tab2[ ,1])
>
> print(sp2)
[1] 0.9455782
> sn2 <- tab2[2,2]/sum(tab2[ ,2])
>
> print(sn2)
[1] 0.8693182
>
```

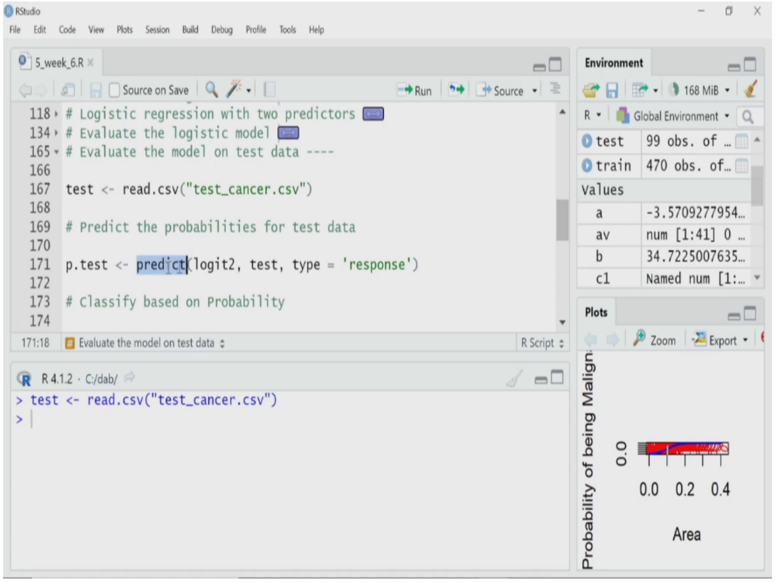
pv	num [1:41] 0....
sn1	0.76704545454...
sn2	0.86931818181...
sp1	0.93197278911...
sp2	0.94557823129...
tab1	'table' int [...
tab2	'table' int [...



```
# Logistic regression with two predictors
# Evaluate the logistic model
# Evaluate the model on test data ----

test <- read.csv("test_cancer.csv")

# Predict the probabilities for test data

p.test <- predict(logit2, test, type = 'response')

# Classify based on Probability

```

```
>
```

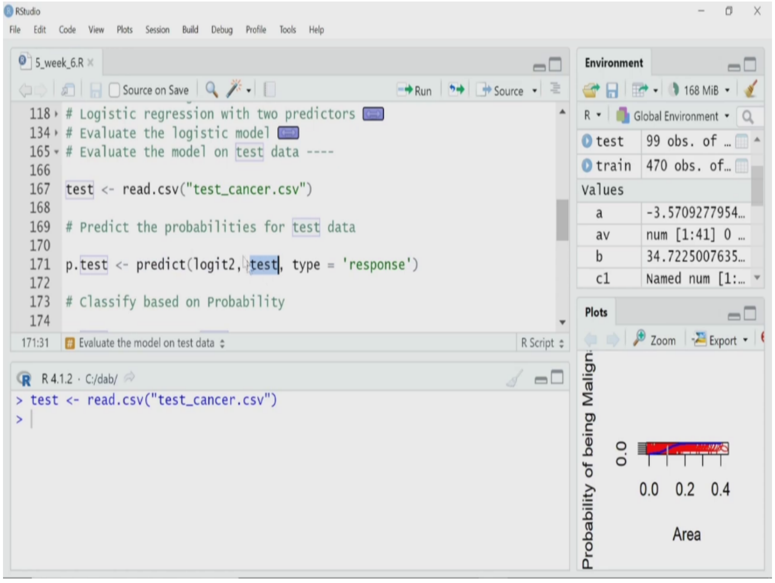
pv	num [1:41] 0....
sn1	0.76704545454...
sn2	0.86931818181...
sp1	0.93197278911...
sp2	0.94557823129...
tab1	'table' int [...
tab2	'table' int [...



```
# Logistic regression with two predictors
# Evaluate the logistic model
# Evaluate the model on test data ----

test <- read.csv("test_cancer.csv")

# Predict the probabilities for test data

p.test <- predict(logit2, test, type = 'response')

# Classify based on Probability
```

```
> test <- read.csv("test_cancer.csv")
> 
```

test	99 obs. of ...
train	470 obs. of...
Values	
a	-3.5709277954...
av	num [1:41] 0 ...
b	34.7225007635...
c1	Named num [1:...



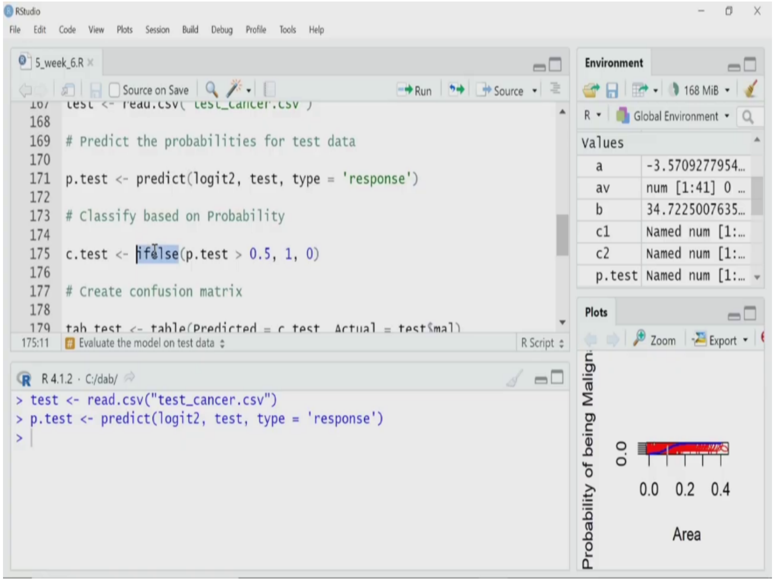
```
# Logistic regression with two predictors
# Evaluate the logistic model
# Evaluate the model on test data ----

test <- read.csv("test_cancer.csv")

# Predict the probabilities for test data

p.test <- predict(logit2, test, type = 'response')

# Classify based on Probability
```

```
> test <- read.csv("test_cancer.csv")
> 
```

test	99 obs. of ...
train	470 obs. of...
Values	
a	-3.5709277954...
av	num [1:41] 0 ...
b	34.7225007635...
c1	Named num [1:...

So, I will apply this model this classifier, logistic regression classifier on that test sample. So, the first thing that I have to read is I have to read the data, so that file is read underscore cancer dot CSV, so I am reading that using read dot CSV. Now, I will use again the predict function, use the logit 2 model, the second model, use test data as the input to predict the probabilities. So that will be stored in p dot test.

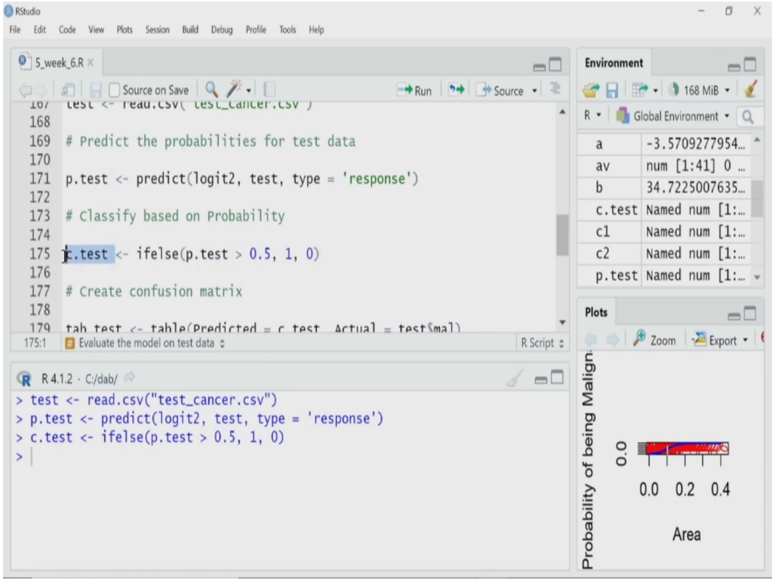
(Refer Slide Time: 29:36)



```
test <- read.csv( test_cancer.csv )

# Predict the probabilities for test data

p.test <- predict(logit2, test, type = 'response')

# Classify based on Probability

c.test <- ifelse(p.test > 0.5, 1, 0)

# Create confusion matrix

tab.test <- table(Predicted = c.test, Actual = test$mal)
```

```
Values
a       -3.5709277954…
av      num [1:41] 0 …
b       34.7225007635…
c1      Named num [1:…
c2      Named num [1:…
p.test  Named num [1:…
```

```
> test <- read.csv("test_cancer.csv")
> p.test <- predict(logit2, test, type = 'response')
> 
```



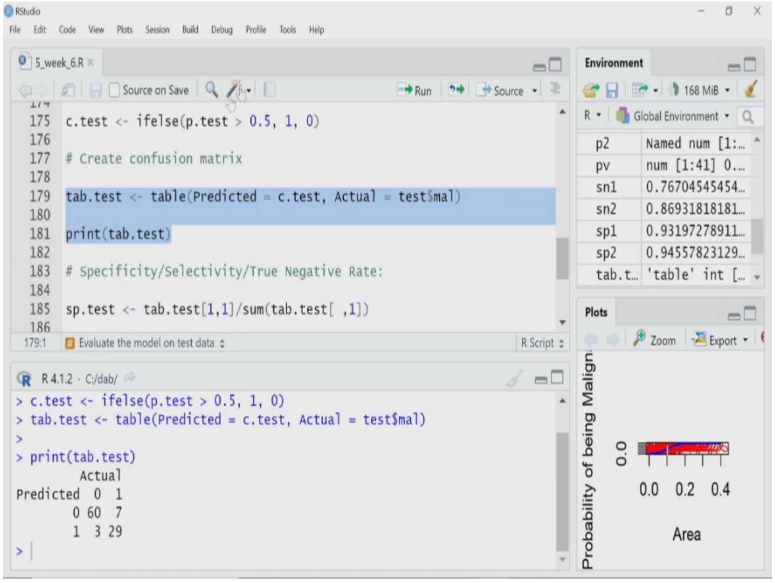
```
test <- read.csv( test_cancer.csv )

# Predict the probabilities for test data

p.test <- predict(logit2, test, type = 'response')

# Classify based on Probability

c.test <- ifelse(p.test > 0.5, 1, 0)

# Create confusion matrix

tab.test <- table(Predicted = c.test, Actual = test$mal)
```

```
a       -3.5709277954…
av      num [1:41] 0 …
b       34.7225007635…
c.test  Named num [1:…
c1      Named num [1:…
c2      Named num [1:…
p.test  Named num [1:…
```

```
> test <- read.csv("test_cancer.csv")
> p.test <- predict(logit2, test, type = 'response')
> c.test <- ifelse(p.test > 0.5, 1, 0)
> 
```

Next, I will classify these samples, this test sample based on the p dot test, the probabilities either as 1 or 0, malignant and non-malignant using the if else function. So, I have got the c dot test vector, where the labels, predicted labels are there. And I will create a confusion matrix for this now by comparing the original label of the sample in the test data.

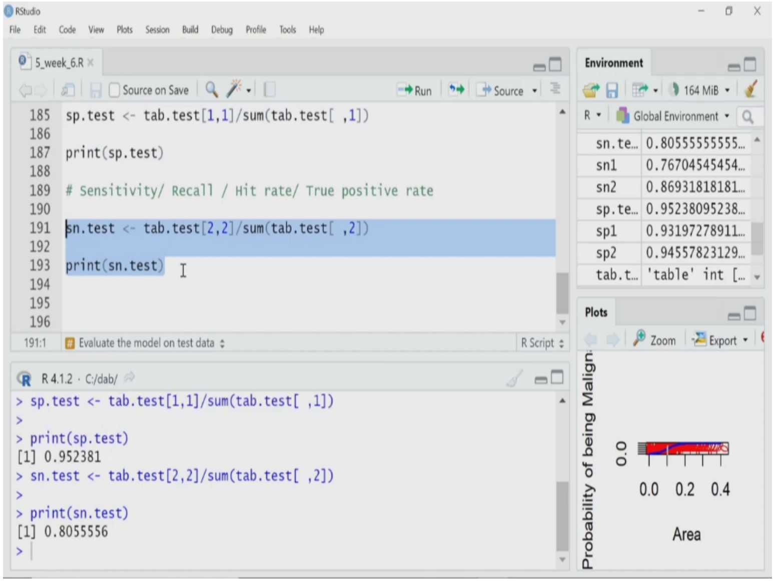
(Refer Slide Time: 29:59)



```
c.test <- ifelse(p.test > 0.5, 1, 0)

# Create confusion matrix

tab.test <- table(Predicted = c.test, Actual = test$mal)

print(tab.test)

# Specificity/Selectivity/True Negative Rate:

sp.test <- tab.test[1,1]/sum(tab.test[ ,1])
```

```
> c.test <- ifelse(p.test > 0.5, 1, 0)
> tab.test <- table(Predicted = c.test, Actual = test$mal)
>
> print(tab.test)
         Actual
Predicted  0  1
        0 60  7
        1  3 29
> |
```

```
p2     Named num [1:...
pv     num [1:41] 0....
sn1    0.76704545454...
sn2    0.86931818181...
sp1    0.93197278911...
sp2    0.94557823129...
tab.t... 'table' int [...
```



```
sp.test <- tab.test[1,1]/sum(tab.test[ ,1])

print(sp.test)

# Sensitivity/ Recall / Hit rate/ True positive rate

sn.test <- tab.test[2,2]/sum(tab.test[ ,2])

print(sn.test)
```

```
> sp.test <- tab.test[1,1]/sum(tab.test[ ,1])
>
> print(sp.test)
[1] 0.952381
> sn.test <- tab.test[2,2]/sum(tab.test[ ,2])
>
> print(sn.test)
[1] 0.8055556
> |
```

```
sn.te... 0.80555555555...
sn1    0.76704545454...
sn2    0.86931818181...
sp.te... 0.95238095238...
sp1    0.93197278911...
sp2    0.94557823129...
tab.t... 'table' int [...
```

So, I will create that. So, this is my now confusion matrix for my test data, I will calculate the specificity and sensitivity the same way we have done. The specificity true negative raise is 95 percent quite good, it is retaining specificity that we have found for our training data set itself, for test data set also the specificity is retained the true negative rate is retained similar value.

Let me check the sensitivity or the true positive value. Here also it is 80 percent, earlier also it was around 84 percent, so, here also it has retained that or retained something similar sensitivity that means my classifier, the logistic regression classifier that I have created is good enough to actually predict whether a sample in the test case whether it is a malignant or not.

So, that brings me to the end of this lecture. In this lecture, I have shown you how you can create a logistic regression model, we have started with a single predictor model, then we have shown how to create the confusion matrix, then we have shown how to calculate specificity and sensitivity of that model.

Next, I added another predictor and created another logistic regression model or classifier and check the confusion matrix and sensitivity and specificity of that. We observe that as I add another predictor the model has become better it is performance has become better. And then eventually we applied that classifier that we created using a training data set on a test data set and we have seen it has performed quite well. That is all for this video. Thank you for learning with me today.