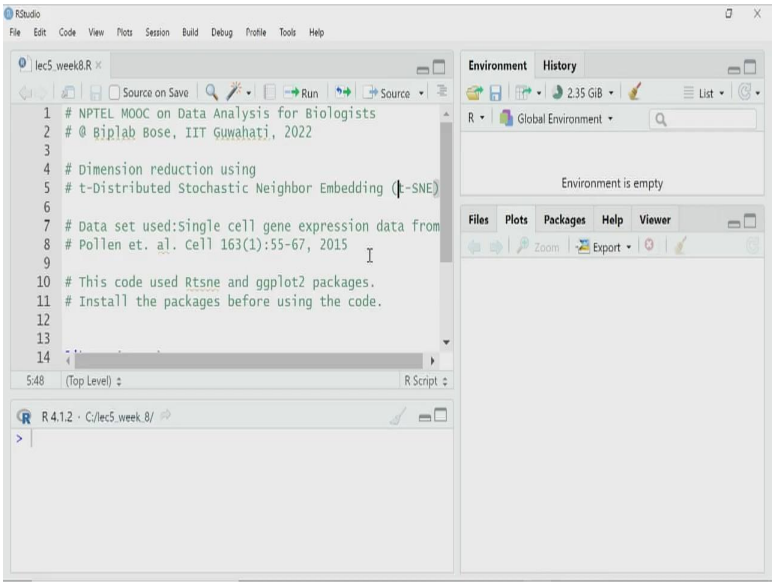
Data Analysis for Biologists
Professor Biplab Bose
Department of Biosciences & Bioengineering
Mehta Family School of Data Science and Artificial Intelligence
Indian Institute of Technology, Guwahati
Lecture 48
tSNE using R

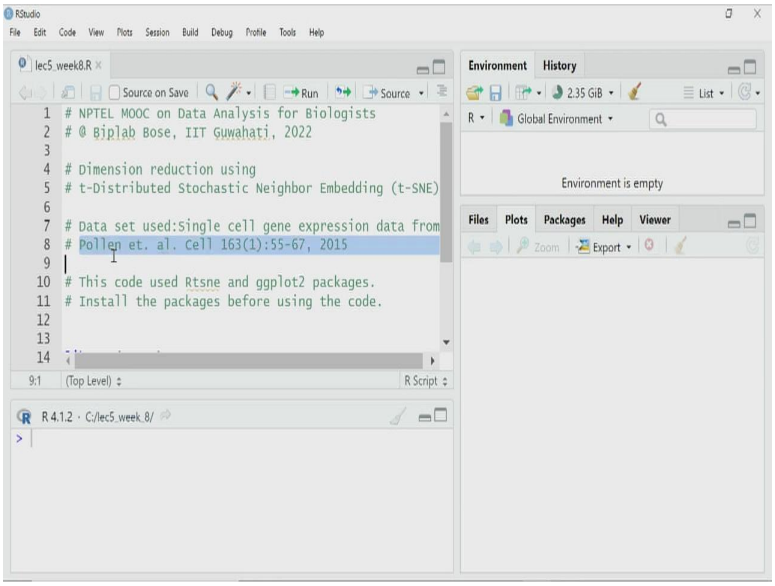
Welcome back. In the last lecture, we have learned about tSNE. The theory of it the mathematics behind that. tSNE is a dimension reduction technique and we use that to reduce the dimension and visualize the higher dimensional data in usually two dimension or in three dimensions. In this lecture, I will use our two part from tSNE on a single cell gene expression data.

(Refer Slide Time: 1:01)

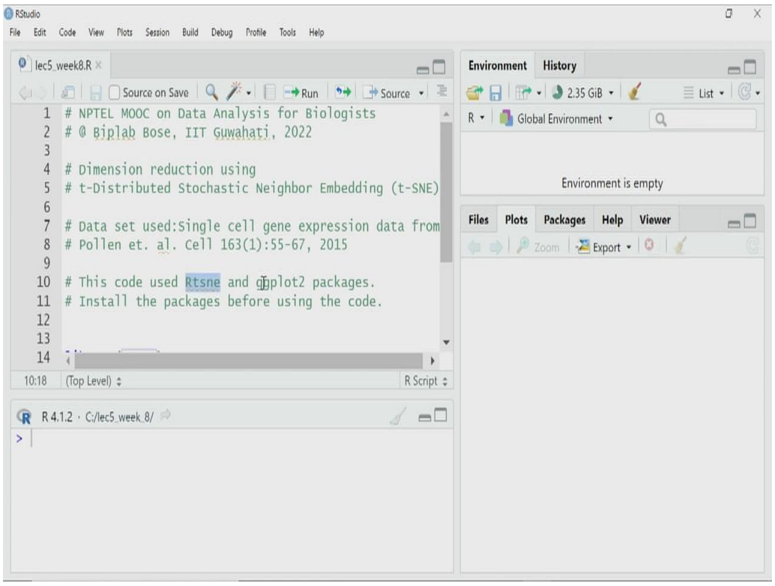


```
# NPTEL MOOC on Data Analysis for Biologists
# @ Biplab Bose, IIT Guwahati, 2022

# Dimension reduction using
# t-Distributed Stochastic Neighbor Embedding (t-SNE)

# Data set used:Single cell gene expression data from
# Pollen et. al. Cell 163(1):55-67, 2015

# This code used Rtsne and ggplot2 packages.
# Install the packages before using the code.
```

```
>
```



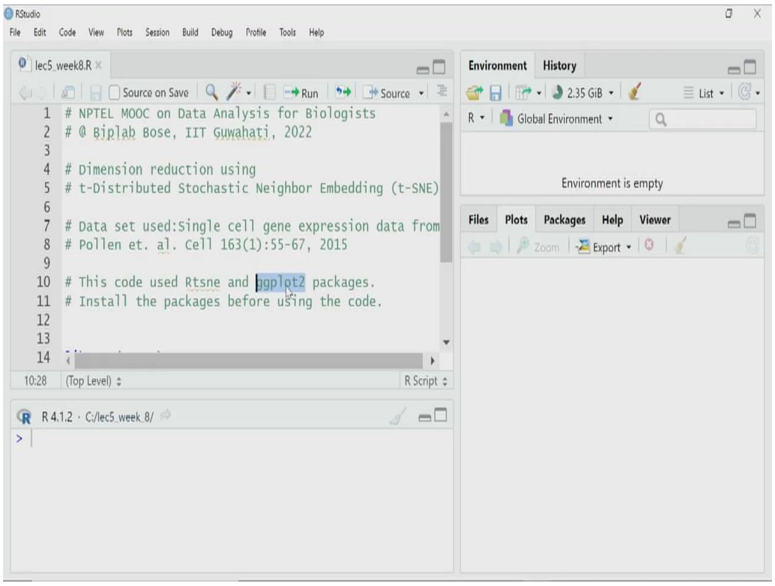
```
# NPTEL MOOC on Data Analysis for Biologists
# @ Biplab Bose, IIT Guwahati, 2022

# Dimension reduction using
# t-Distributed Stochastic Neighbor Embedding (t-SNE)

# Data set used:Single cell gene expression data from
# Pollen et. al. Cell 163(1):55-67, 2015

# This code used Rtsne and ggplot2 packages.
# Install the packages before using the code.
```



```
# NPTEL MOOC on Data Analysis for Biologists
# @ Biplab Bose, IIT Guwahati, 2022

# Dimension reduction using
# t-Distributed Stochastic Neighbor Embedding (t-SNE)

# Data set used:Single cell gene expression data from
# Pollen et. al. Cell 163(1):55-67, 2015

# This code used Rtsne and ggplot2 packages.
# Install the packages before using the code.
```

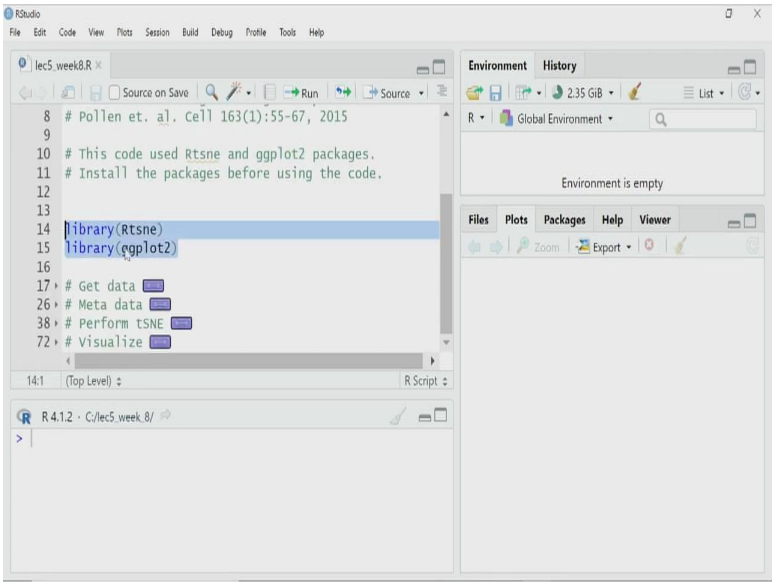
So, let us start doing that. So the data that I will be using is from a paper taken Pollen et. al. in 2015. And to perform these tSNE T distributed stochastic neighbor embedding technique, I require a package that is called Rtsne. So Rtsne is available, you should install it before you run this code. And I will visualize them I will draw the plot using gg plot 2. So you should have that installed in your machine.

(Refer Slide Time: 1:31)



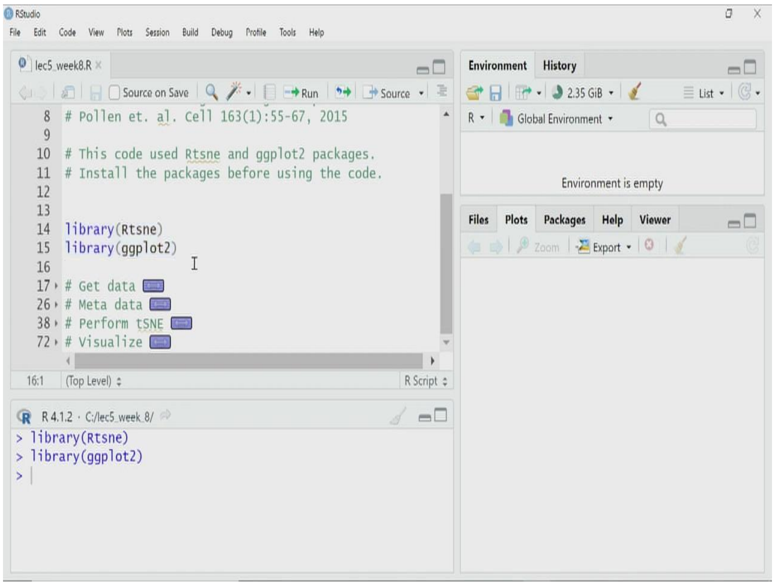
```r
# NPTEL MOOC on Data Analysis for Biologists
# @ Biplab Bose, IIT Guwahati, 2022

# Dimension reduction using
# t-Distributed Stochastic Neighbor Embedding (t-SNE)

# Data set used:Single cell gene expression data from
# Pollen et. al. Cell 163(1):55-67, 2015

# This code used Rtsne and ggplot2 packages.
# Install the packages before using the code.
```

```
>
```



```r
# Pollen et. al. Cell 163(1):55-67, 2015

# This code used Rtsne and ggplot2 packages.
# Install the packages before using the code.


library(Rtsne)
library(ggplot2)

# Get data
# Meta data
# Perform tSNE
# Visualize
```

```
>
```



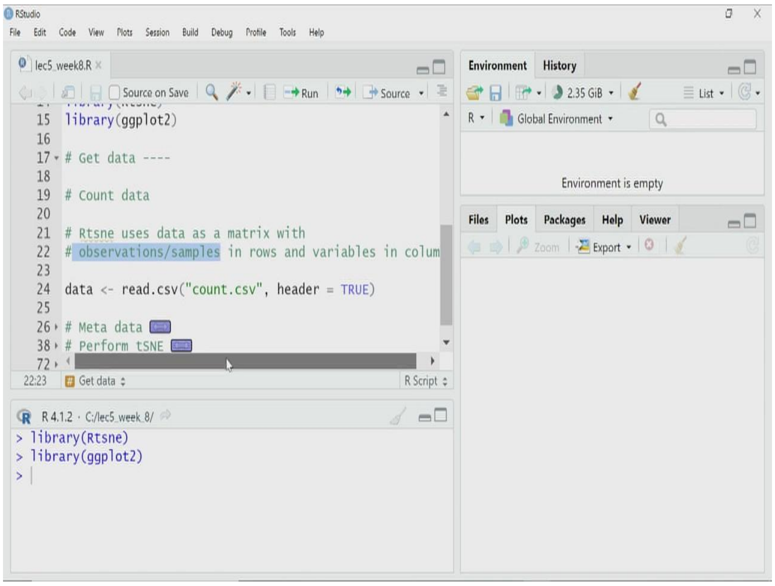
```r
# Pollen et. al. Cell 163(1):55-67, 2015

# This code used Rtsne and ggplot2 packages.
# Install the packages before using the code.


library(Rtsne)
library(ggplot2)

# Get data
# Meta data
# Perform tSNE
# Visualize
```

```
> library(Rtsne)
> library(ggplot2)
>
```

```
library(Rtsne)

library(ggplot2)
```

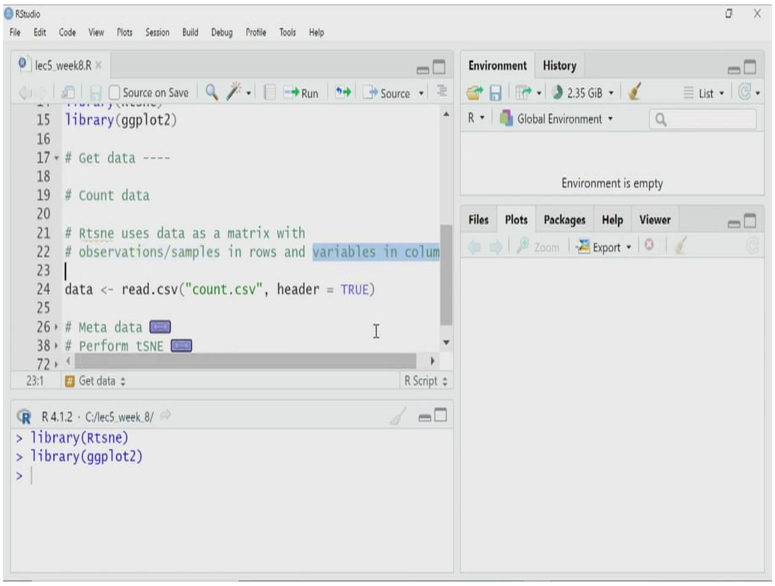
I have already installed them. So I do not need to install, what I will do, I will do first load the libraries. So I have loaded her both are Rtsne and gg plot 2. Now we will get the data, the data is in CSV format, I have downloaded cleaned the data in the right format, so that we can work on it. Many of the cases, many of the time when you will be doing it for your single cell or gene expression data, you may not have the clean data, you have to do some preprocessing to clean the data and arrange in the right format that is required for Rtsne to work on it.

(Refer Slide Time: 2:07)



```
library(ggplot2)

# Get data ----

# Count data

# Rtsne uses data as a matrix with
# observations/samples in rows and variables in colum

data <- read.csv("count.csv", header = TRUE)

# Meta data
# Perform tSNE
```

```
> library(Rtsne)
> library(ggplot2)
>
```



```
library(ggplot2)

# Get data ----

# Count data

# Rtsne uses data as a matrix with
# observations/samples in rows and variables in colum

data <- read.csv("count.csv", header = TRUE)

# Meta data
# Perform tSNE
```

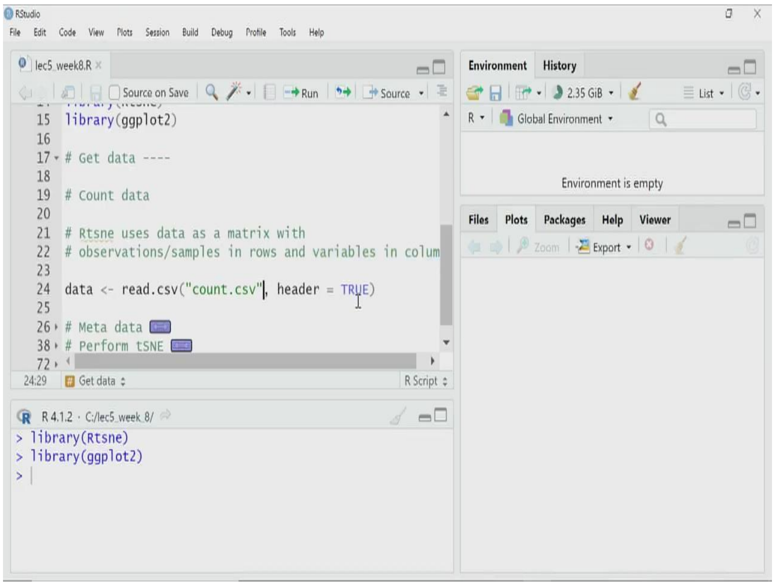
```
> library(Rtsne)
> library(ggplot2)
>
```

data ← read.csv("count.csv”, header = TRUE)

So let us look into the data. So what I have here is a data count data. And I will load it is a CSV file, and then will see what we have got there. Now remember, Rtsne require the data in a particular format. In this format, the data should be a matrix where observations or samples will be in the rows, whereas the variables should be in the column.

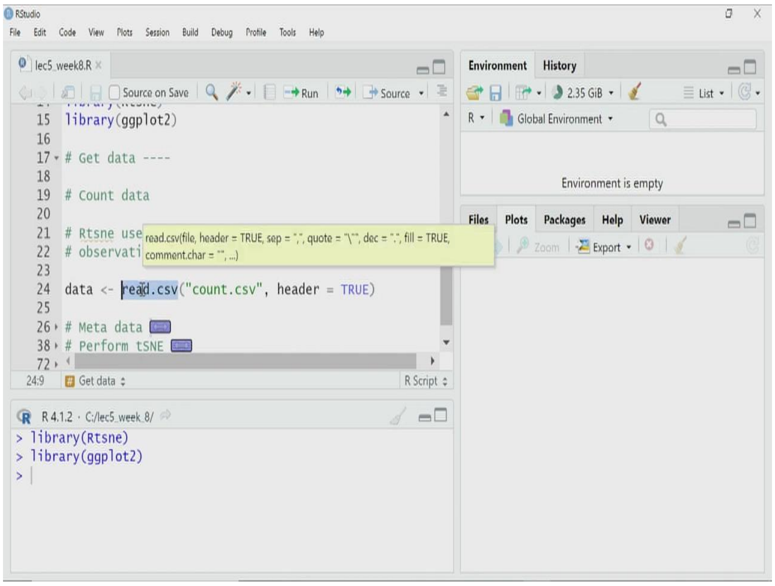
What do I mean by this, in this particular example that I am dealing today, variables are genes, and we have count of each gene. And so they should be in the columns, whereas the observation or samples are individual sales, right? So they should be in the rows.

(Refer Slide Time: 2:49)



```
library(ggplot2)

# Get data ----

# Count data

# Rtsne uses data as a matrix with
# observations/samples in rows and variables in colum

data <- read.csv("count.csv", header = TRUE)

# Meta data
# Perform tSNE
```

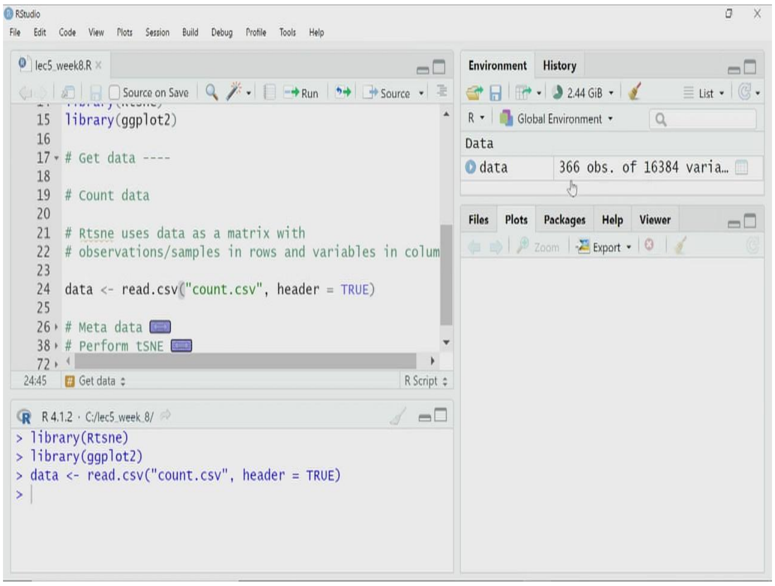
```
> library(Rtsne)
> library(ggplot2)
>
```



```
library(ggplot2)

# Get data ----

# Count data

# Rtsne use
# observati

data <- read.csv("count.csv", header = TRUE)

# Meta data
# Perform tSNE
```

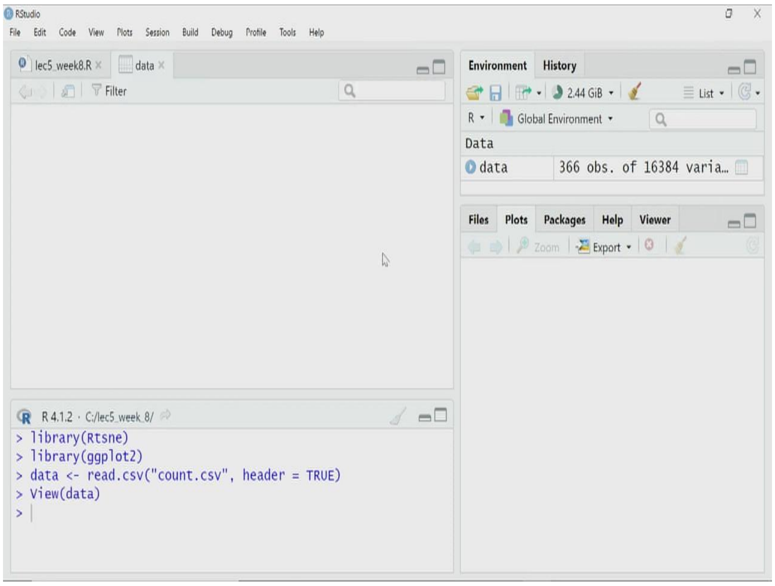
```
read.csv(file, header = TRUE, sep = ",", quote = "\"", dec = ".", fill = TRUE,
comment.char = "", ...)
```

```
> library(Rtsne)
> library(ggplot2)
>
```

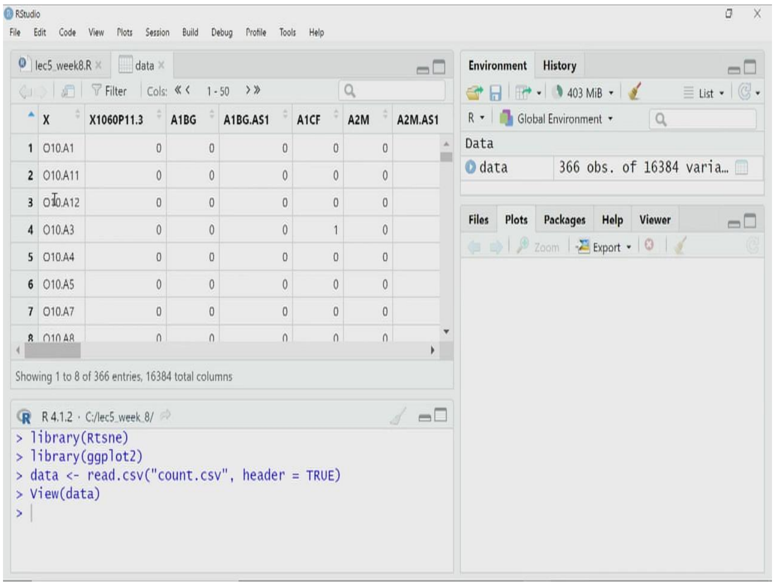

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
lec5_week8.R
Source on Save
Run
Source
Environment History
2.44 GiB
List
Global Environment
Data
data 366 obs. of 16384 varia...
Files Plots Packages Help Viewer
Zoom
Export
24:45
Get data
R Script
R 4.1.2 · C:/lec5_week_8/

```
library(ggplot2)

# Get data ----

# Count data

# Rtsne uses data as a matrix with
# observations/samples in rows and variables in colum

data <- read.csv("count.csv", header = TRUE)

# Meta data
# Perform tSNE
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
>
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
lec5_week8.R
data
Filter
Environment History
2.44 GiB
List
Global Environment
Data
data 366 obs. of 16384 varia...
Files Plots Packages Help Viewer
Zoom
Export
R 4.1.2 · C:/lec5_week_8/

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
>
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
lec5_week8.R
data
Filter
Cols: « ‹ 1 - 50 › »
Environment History
403 MiB
List
Global Environment
Data
data 366 obs. of 16384 varia...
Files Plots Packages Help Viewer
Zoom
Export
R 4.1.2 · C:/lec5_week_8/

	X	X1060P11.3	A1BG	A1BG.AS1	A1CF	A2M	A2M.AS1
1	O10.A1	0	0	0	0	0	
2	O10.A11	0	0	0	0	0	
3	O10.A12	0	0	0	0	0	
4	O10.A3	0	0	0	1	0	
5	O10.A4	0	0	0	0	0	
6	O10.A5	0	0	0	0	0	
7	O10.A7	0	0	0	0	0	
8	O10.A8	0	0	0	0	0	

Showing 1 to 8 of 366 entries, 16384 total columns

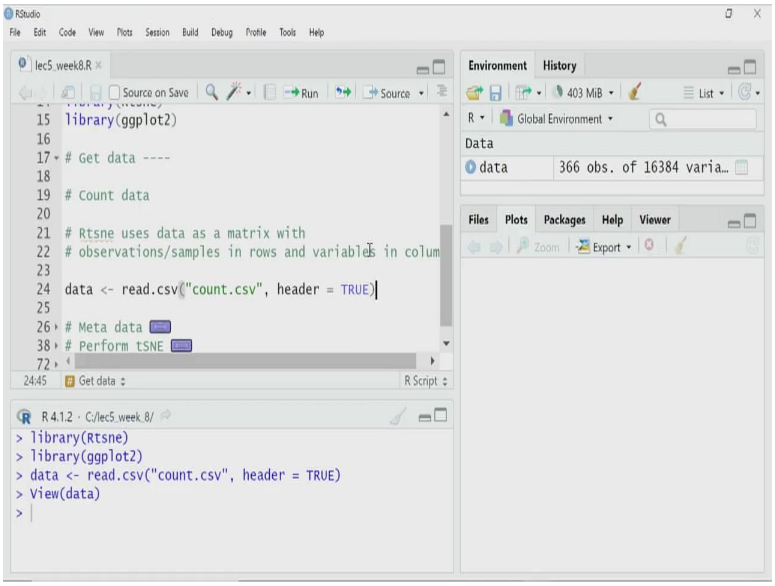
```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
>
```

So let me read it is a huge file. So it may take a few minutes depending upon your machine. So I am using the red dot CSV function, the standard one that I use to read CSV file, and I am keeping header equal to true and I am storing that data in a variable called data, it took some time to read the data and the data is now read, it has 366 observations that means 366 different sales data I have and 16,384 variables that means 16,384 genes expression data, we have so these 366 observations, they belong to different cell types.

So, I will go until what type of cell are those, so, you can double click on it and you can visualize this data in a matrix format it is a large file, so it may take some time. So here I have, we have different gene names on the column headers, and sample names are the numbers are in the row numbers. Now, these row numbers are some numbers.

Now, what these people have done when they have submitted provided the data, they have done a nice thing they have created a metadata file where for each of these samples, they have provided detailed information, what type of sales these are, where are they coming from whatever anatomical location, they have collected this data. So, all these are they are in a metadata file. So that is also in a CSV format file. So, I will read that and I will look into that.

(Refer Slide Time: 4:24)

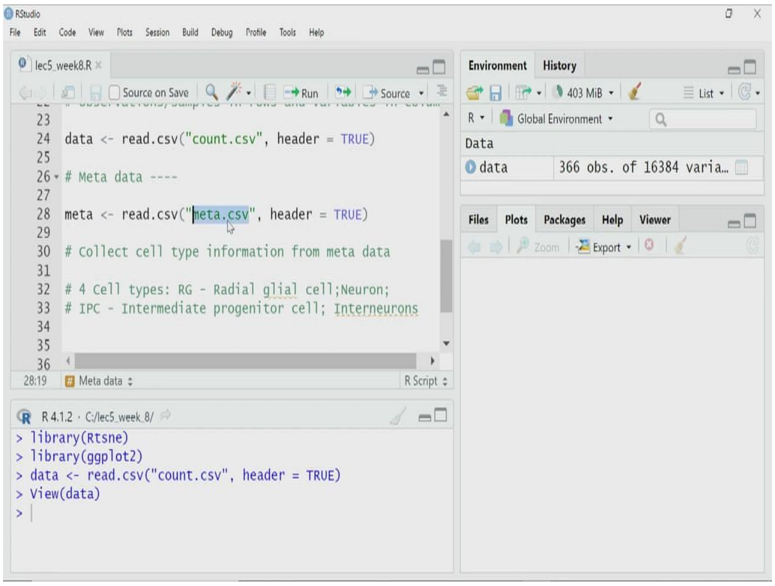


```
library(ggplot2)

# Get data ----

# Count data

# Rtsne uses data as a matrix with
# observations/samples in rows and variables in colum

data <- read.csv("count.csv", header = TRUE)

# Meta data
# Perform tSNE
```

Data

data	366 obs. of 16384 varia...

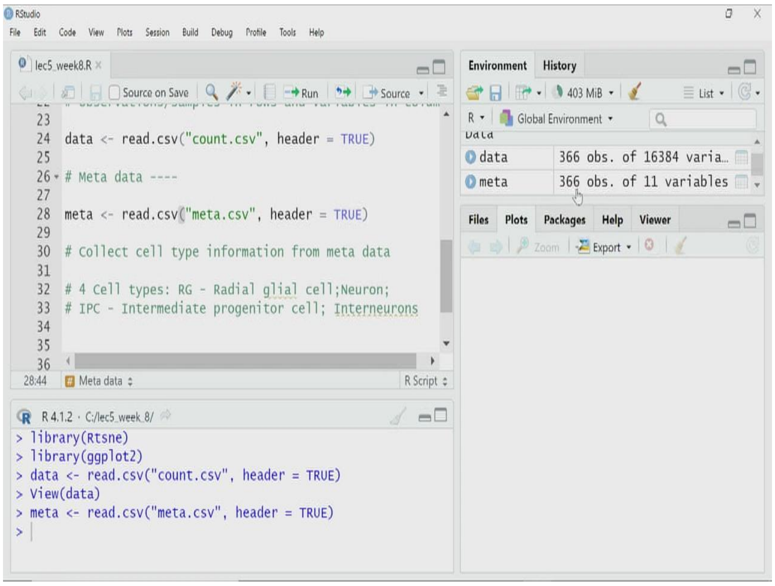
```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
>
```



```r
data <- read.csv("count.csv", header = TRUE)

# Meta data ----

meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons
```

```
R 4.1.2 · C:/lec5_week_8/
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> 
```

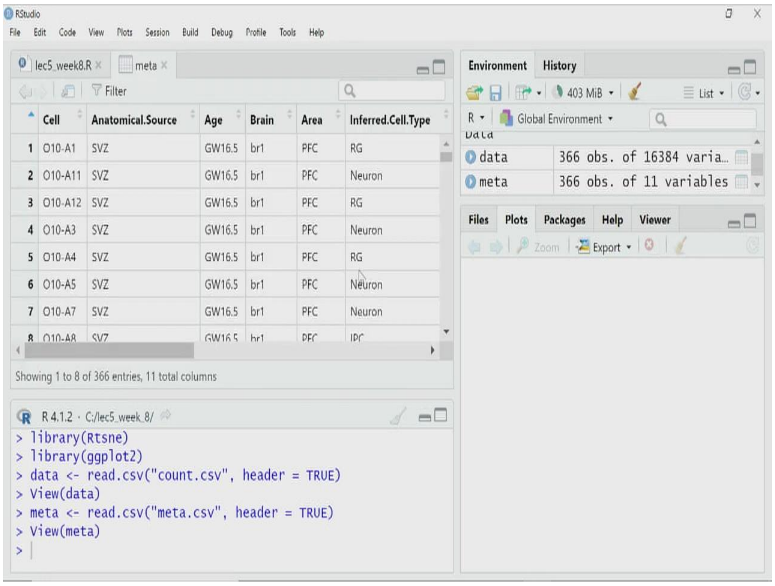
```
Data
data    366 obs. of 16384 varia…
```



```r
data <- read.csv("count.csv", header = TRUE)

# Meta data ----

meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons
```

```
R 4.1.2 · C:/lec5_week_8/
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> 
```

```
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```

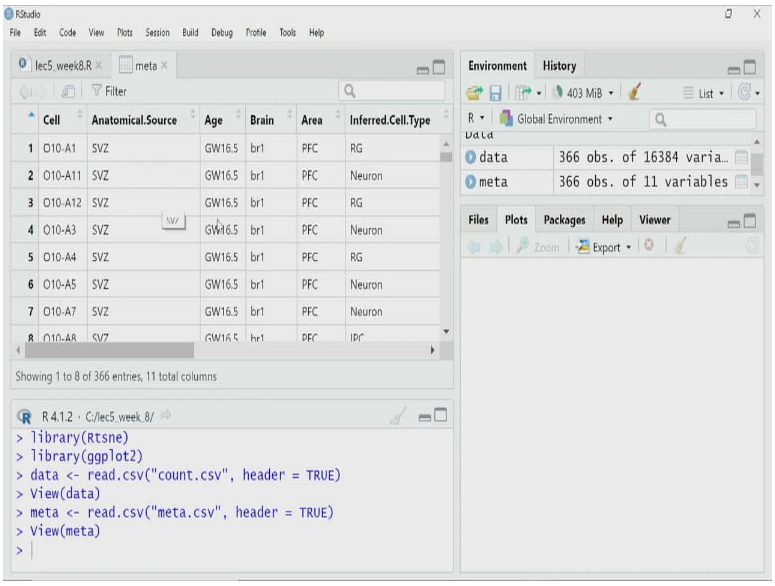


	Cell	Anatomical.Source	Age	Brain	Area	Inferred.Cell.Type
1	O10-A1	SVZ	GW16.5	br1	PFC	RG
2	O10-A11	SVZ	GW16.5	br1	PFC	Neuron
3	O10-A12	SVZ	GW16.5	br1	PFC	RG
4	O10-A3	SVZ	GW16.5	br1	PFC	Neuron
5	O10-A4	SVZ	GW16.5	br1	PFC	RG
6	O10-A5	SVZ	GW16.5	br1	PFC	Neuron
7	O10-A7	SVZ	GW16.5	br1	PFC	Neuron
8	O10-A8	SVZ	GW16.5	br1	PFC	IPC

Showing 1 to 8 of 366 entries, 11 total columns

```
R 4.1.2 · C:/lec5_week_8/
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> 
```

```
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```



	Cell	Anatomical.Source	Age	Brain	Area	Inferred.Cell.Type
1	O10-A1	SVZ	GW16.5	br1	PFC	RG
2	O10-A11	SVZ	GW16.5	br1	PFC	Neuron
3	O10-A12	SVZ	GW16.5	br1	PFC	RG
4	O10-A3	SVZ	GW16.5	br1	PFC	Neuron
5	O10-A4	SVZ	GW16.5	br1	PFC	RG
6	O10-A5	SVZ	GW16.5	br1	PFC	Neuron
7	O10-A7	SVZ	GW16.5	br1	PFC	Neuron
8	O10-A8	SVZ	GW16.5	br1	PFC	IPC

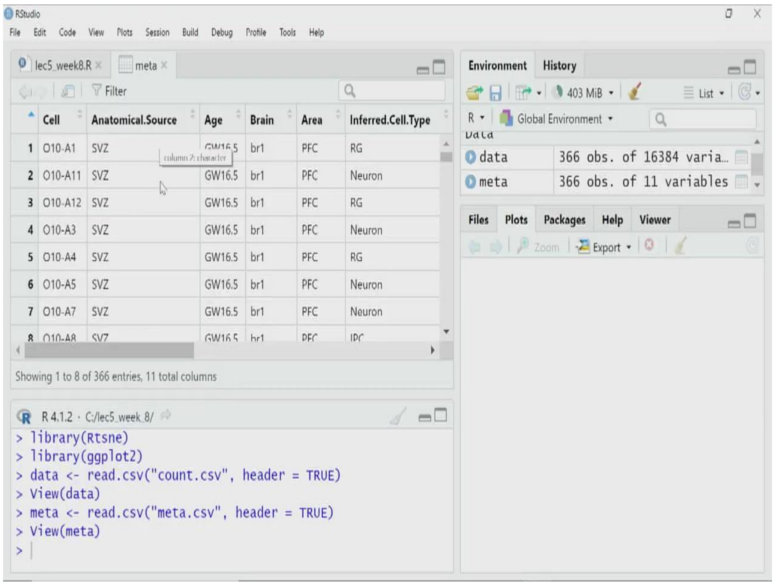
Showing 1 to 8 of 366 entries, 11 total columns

```
Data
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> |
```

So let us read the metadata. So, I am right there is a file called maker dot CSV and I am reading that using Read dot CSV. So, meta has been read the data metadata is in metavariable. So again, you have 366 observations, because we have 362 observations in the data file also. Let me click and open that you can see the cell numbers which are present in the data file are here and see these are all neuronal cell. What they have done.

(Refer Slide Time: 5:02)

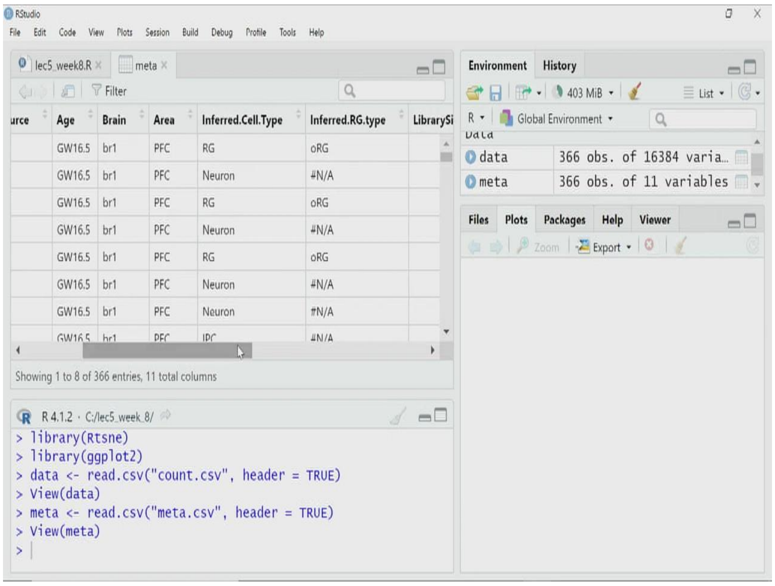


	Cell	Anatomical.Source	Age	Brain	Area	Inferred.Cell.Type
1	O10-A1	SVZ	GW16.5	br1	PFC	RG
2	O10-A11	SVZ	GW16.5	br1	PFC	Neuron
3	O10-A12	SVZ	GW16.5	br1	PFC	RG
4	O10-A3	SVZ	GW16.5	br1	PFC	Neuron
5	O10-A4	SVZ	GW16.5	br1	PFC	RG
6	O10-A5	SVZ	GW16.5	br1	PFC	Neuron
7	O10-A7	SVZ	GW16.5	br1	PFC	Neuron
8	O10-A8	SVZ	GW16.5	br1	PFC	IPC

Showing 1 to 8 of 366 entries, 11 total columns

```
Data
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> |
```



urce	Age	Brain	Area	Inferred.Cell.Type	Inferred.RG.type	LibrarySi
	GW16.5	br1	PFC	RG	oRG	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	RG	oRG	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	RG	oRG	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	IPC	#N/A	

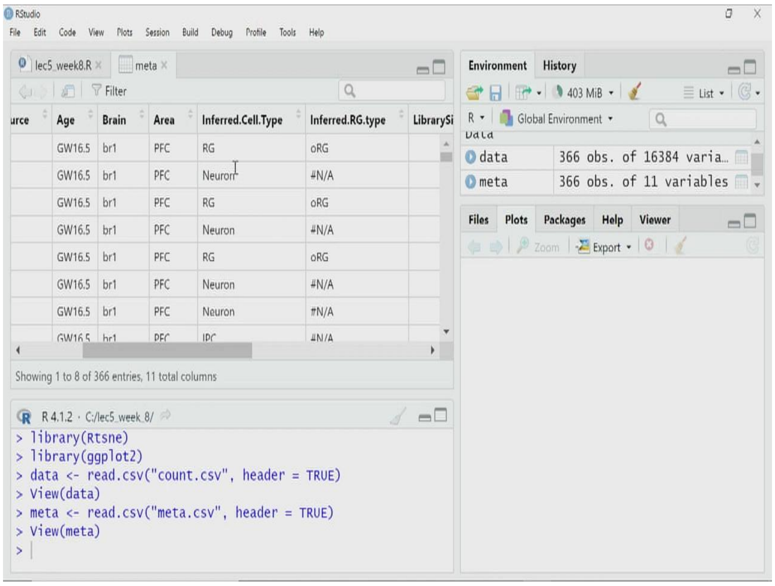
Showing 1 to 8 of 366 entries, 11 total columns

data 366 obs. of 16384 varia...
meta 366 obs. of 11 variables

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> |
```

They have taken samples from brain sections so their anatomical location of those section are written here is in the second column where is anatomical dot source, then you have different other information like age, area, brain and all these things and based upon the gene expression signature, they have already has flat each of these sample each of these cells in different 4 different cell types,

(Refer Slide Time: 5:23)

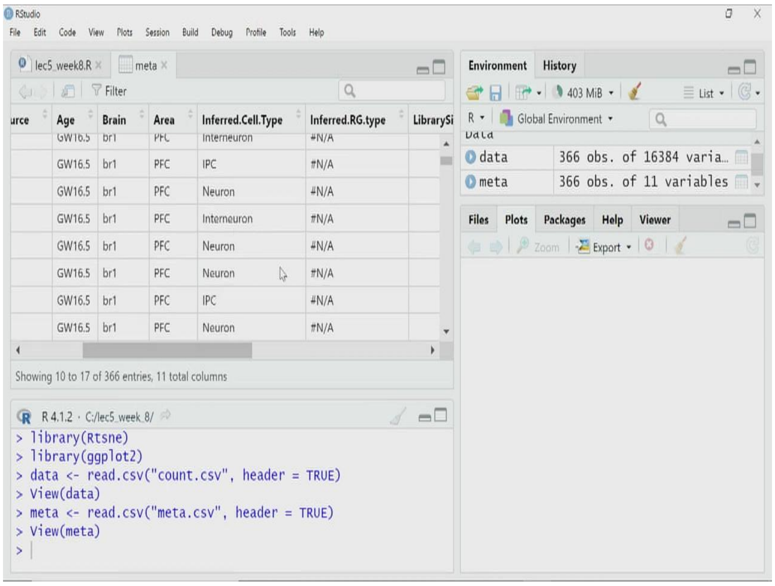


urce	Age	Brain	Area	Inferred.Cell.Type	Inferred.RG.type	LibrarySi
	GW16.5	br1	PFC	RG	oRG	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	RG	oRG	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	RG	oRG	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	IPC	#N/A	

Showing 1 to 8 of 366 entries, 11 total columns

data 366 obs. of 16384 varia...
meta 366 obs. of 11 variables

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> |
```



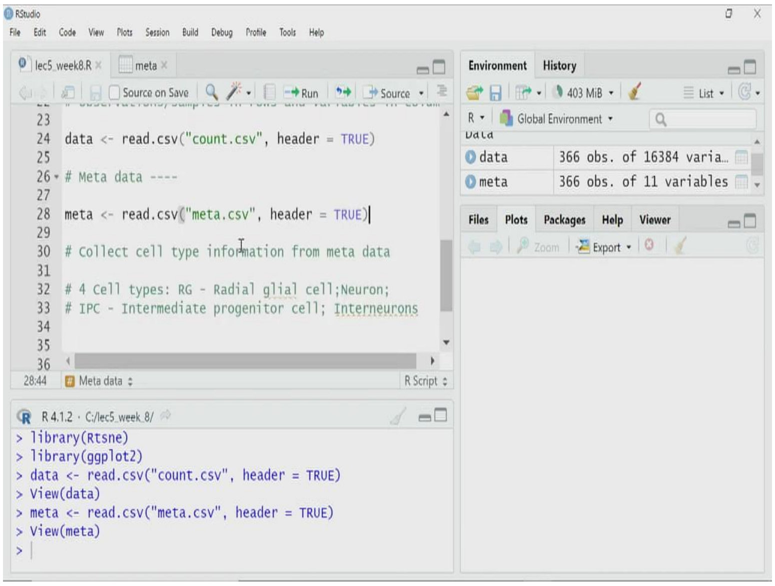
urce	Age	Brain	Area	Inferred.Cell.Type	Inferred.RG.type	LibrarySi
	GW16.5	br1	PFC	Interneuron	#N/A	
	GW16.5	br1	PFC	IPC	#N/A	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	Interneuron	#N/A	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	Neuron	#N/A	
	GW16.5	br1	PFC	IPC	#N/A	
	GW16.5	br1	PFC	Neuron	#N/A	

Showing 10 to 17 of 366 entries, 11 total columns

```
Data
data    366 obs. of 16384 varia...
meta    366 obs. of 11 variables
```

R 4.1.2 · C:/lec5_week_8/

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> |
```



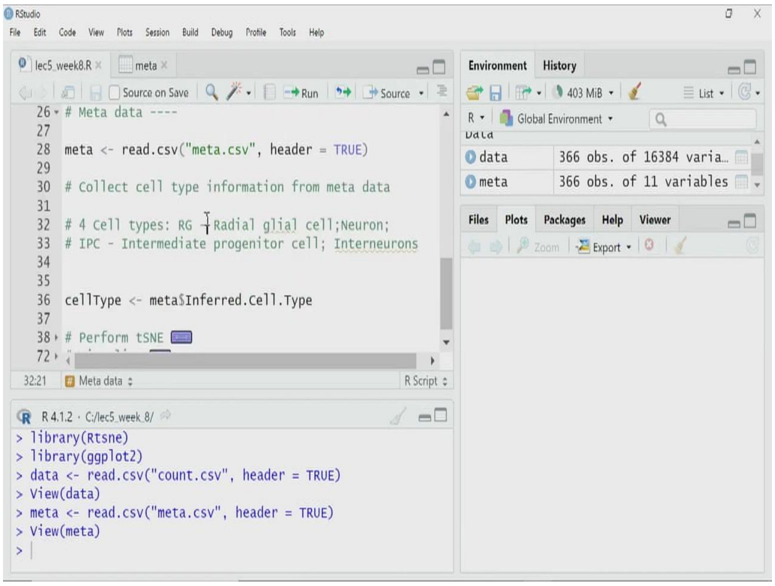
```
data <- read.csv("count.csv", header = TRUE)

# Meta data ----

meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons
```

28:44 Meta data R Script

```
Data
data    366 obs. of 16384 varia...
meta    366 obs. of 11 variables
```

R 4.1.2 · C:/lec5_week_8/

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> |
```



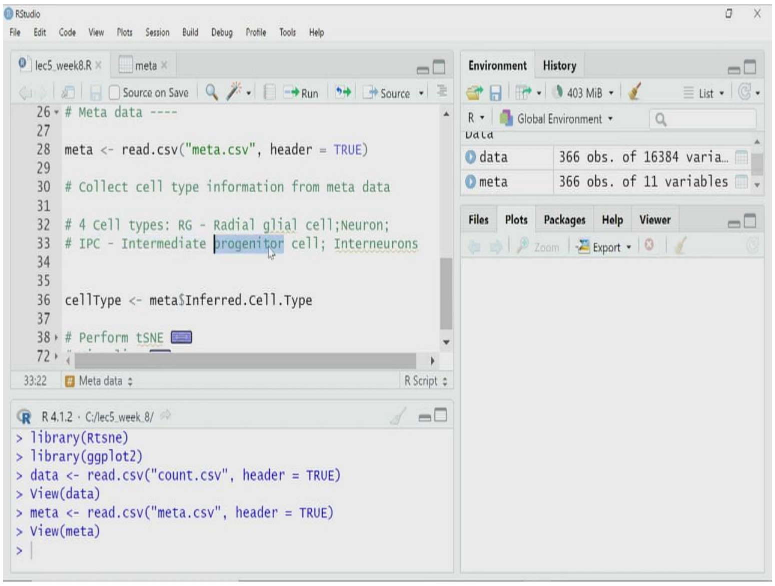
```
# Meta data ----

meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons


cellType <- meta$Inferred.Cell.Type

# Perform tSNE
```

32:21 Meta data R Script

```
Data
data    366 obs. of 16384 varia...
meta    366 obs. of 11 variables
```

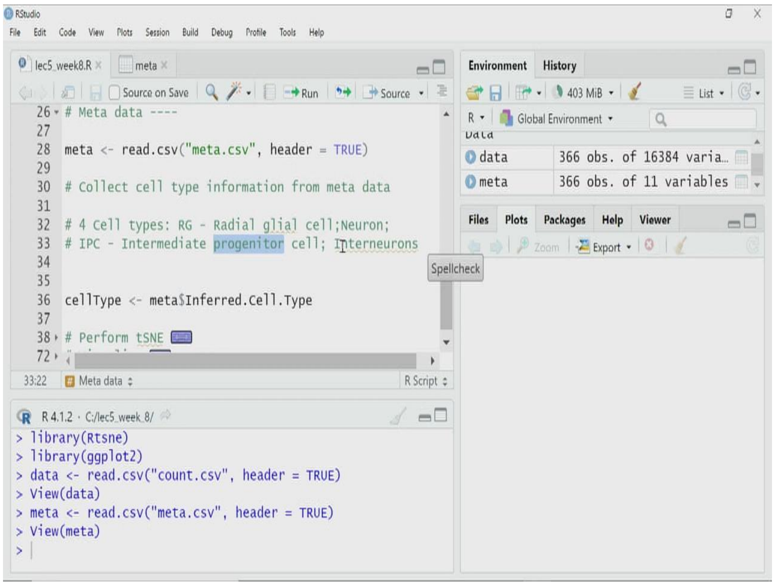
R 4.1.2 · C:/lec5_week_8/

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> |
```



```
# Meta data ----

meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons


cellType <- meta$Inferred.Cell.Type

# Perform tSNE
```

```
R 4.1.2 · C:/lec5_week_8/
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```



```
# Meta data ----

meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons


cellType <- meta$Inferred.Cell.Type

# Perform tSNE
```

```
R 4.1.2 · C:/lec5_week_8/
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

meta ← read.csv("meta.csv”, header=TRUE)

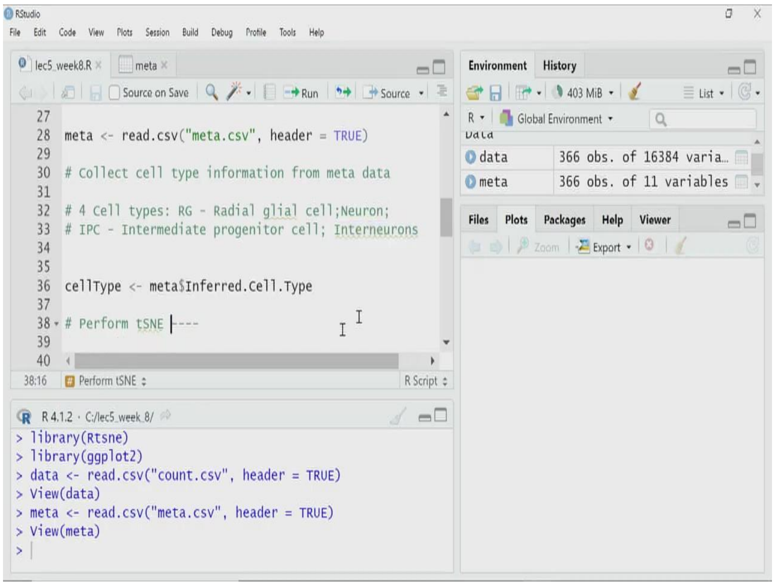
cellType ← meta$Inferred.Cell.Type

For example, it is written as RG neuron and then some cells are inter neuron something like that. So, what they are, RG for radial glial cell neurons are neurons, IPC is are intermediate progenitor cells from, you get other cell types and inter neuron cells. So, these there are four types of cells in these data. So, all those rows in my data variable are actually one of these four types of cells.

So, this is my data, I have 366 observation involving four types of sale and there are 16,000 Something gene expression data and I want to visualize this data into 2 dimension. So, I have 16,000 A 384 genes expression data, count data that means, the dimension of the data is

16,384 I cannot visualize it. So, I have to reduce the dimension, there are many new dimension reduction techniques like PCA and others and we will be using tSNE here.

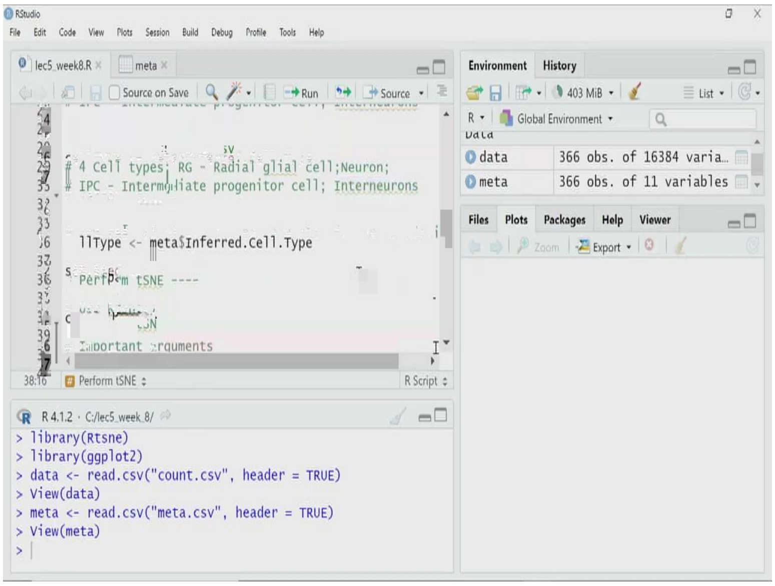
(Refer Slide Time: 6:23)



```
meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons


cellType <- meta$Inferred.Cell.Type

# Perform tSNE ----
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

```
data   366 obs. of 16384 varia…
meta   366 obs. of 11 variables
```



```
# 4 Cell types; RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons

llType <- meta$Inferred.Cell.Type

Perform tSNE ----

important arguments
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

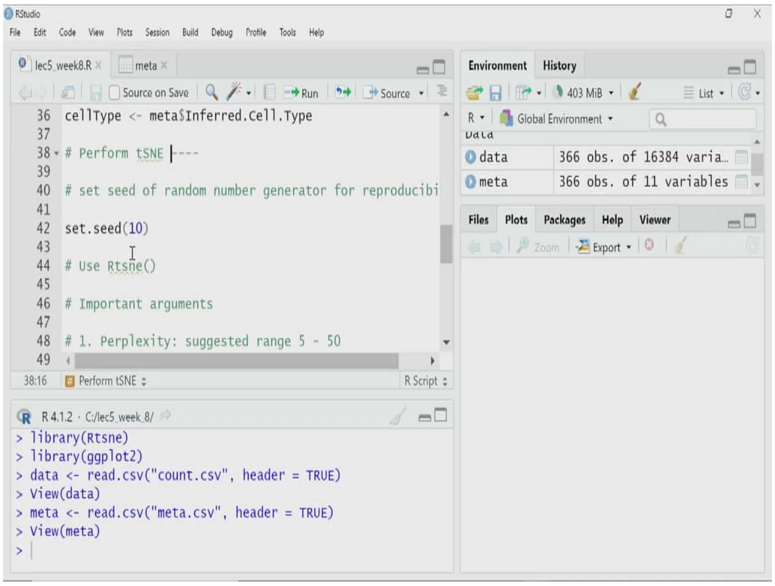
```
data   366 obs. of 16384 varia…
meta   366 obs. of 11 variables
```

cellType ← meta$Inferred.Cell.Type

So, as I said to perform tSNE, I will be using Rtsne function. And remember tSNE uses a random process, right? It is a stochastic method algorithm. So, that means it has to use random number generator. So it will use the default random number generator in R and it is a good practice.

Whenever you use some algorithm or method where random numbers will be generated and used, you decide and fix the seed value for random number generation so that every time you repeat the same analysis, you will use the same seed value and you will get reproducible result. Otherwise, if the random number sets are changed the result the final result may change slightly.

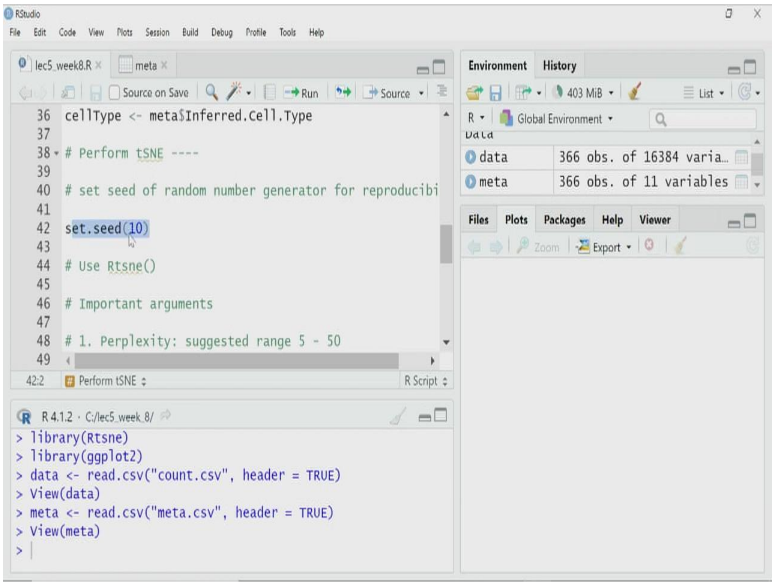
(Refer Slide Time: 7:08)



```
cellType <- meta$Inferred.Cell.Type

# Perform tSNE ----

# set seed of random number generator for reproducibi

set.seed(10)

# Use Rtsne()

# Important arguments

# 1. Perplexity: suggested range 5 - 50
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

```
Data
data    366 obs. of 16384 varia...
meta    366 obs. of 11 variables
```



```
cellType <- meta$Inferred.Cell.Type

# Perform tSNE ----

# set seed of random number generator for reproducibi

set.seed(10)

# Use Rtsne()

# Important arguments

# 1. Perplexity: suggested range 5 - 50
```

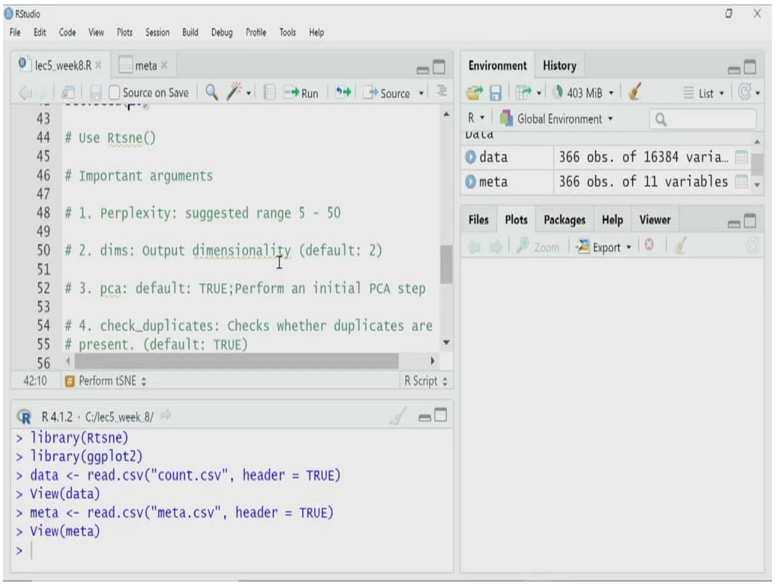
```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

```
Data
data    366 obs. of 16384 varia...
meta    366 obs. of 11 variables
```

set.seed(10)

So that is why what I am doing here I am setting the seed of random number generator as set dot seed and I am giving you a value 10. Here you could give us some other value. Now I am ready to run tSNE. Now tSNE function Rtsne the function in are has lots of arguments and I will strongly advise you to look into the documentation of that in the help file of that to understand each of these parameters or argument for this function.

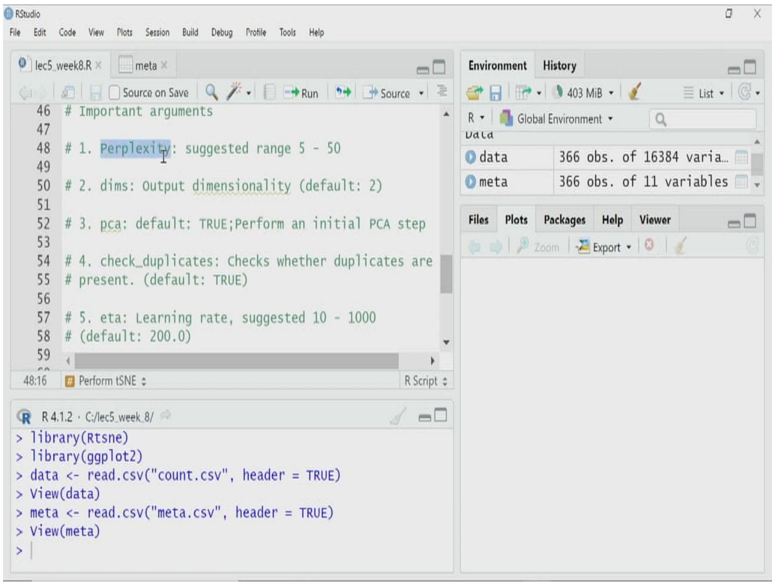
(Refer Slide Time: 7:38)



```
# Use Rtsne()

# Important arguments

# 1. Perplexity: suggested range 5 - 50

# 2. dims: Output dimensionality (default: 2)

# 3. pca: default: TRUE;Perform an initial PCA step

# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

Data
data 366 obs. of 16384 varia...
meta 366 obs. of 11 variables



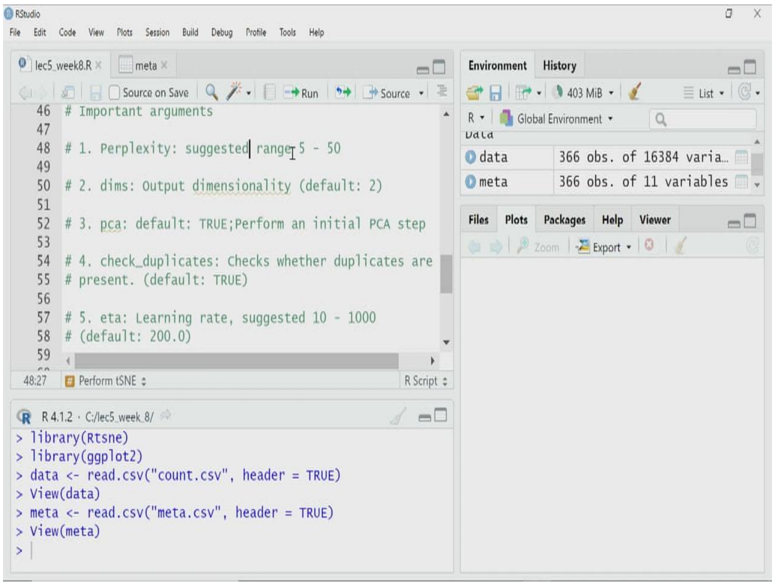
```
# Important arguments

# 1. Perplexity: suggested range 5 - 50

# 2. dims: Output dimensionality (default: 2)

# 3. pca: default: TRUE;Perform an initial PCA step

# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

Data
data 366 obs. of 16384 varia...
meta 366 obs. of 11 variables

I have listed a few of them here. Those are the important one because for this particular type of analysis, first one is obviously perplexity. If you look back into our lecture on tSNE, we have explained there how perplexity can affect the final outcome of the TSNE.

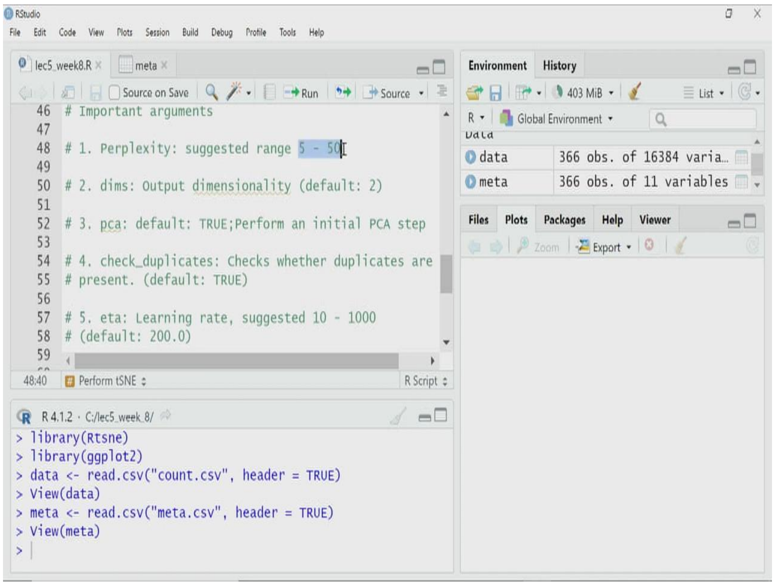
(Refer Slide Time: 7:57)



```
# Important arguments

# 1. Perplexity: suggested range 5 - 50

# 2. dims: Output dimensionality (default: 2)

# 3. pca: default: TRUE;Perform an initial PCA step

# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

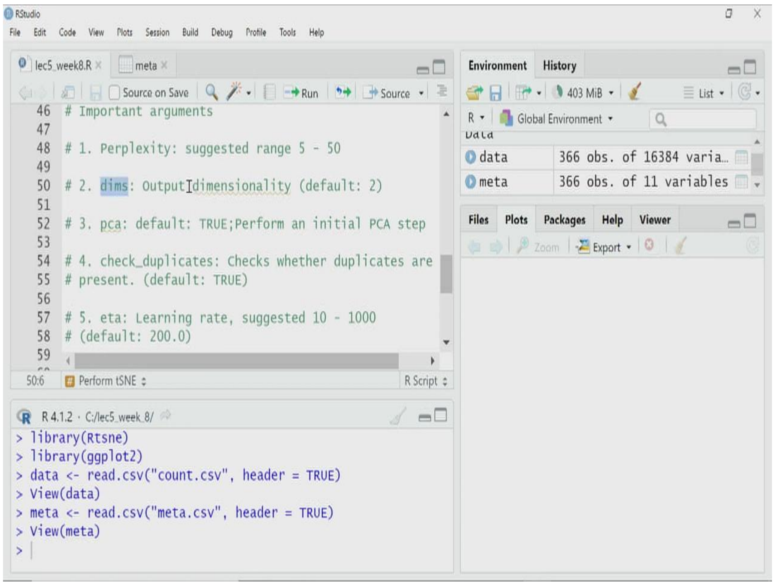
```
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```



```
# Important arguments

# 1. Perplexity: suggested range 5 - 50

# 2. dims: Output dimensionality (default: 2)

# 3. pca: default: TRUE;Perform an initial PCA step

# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

```
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```

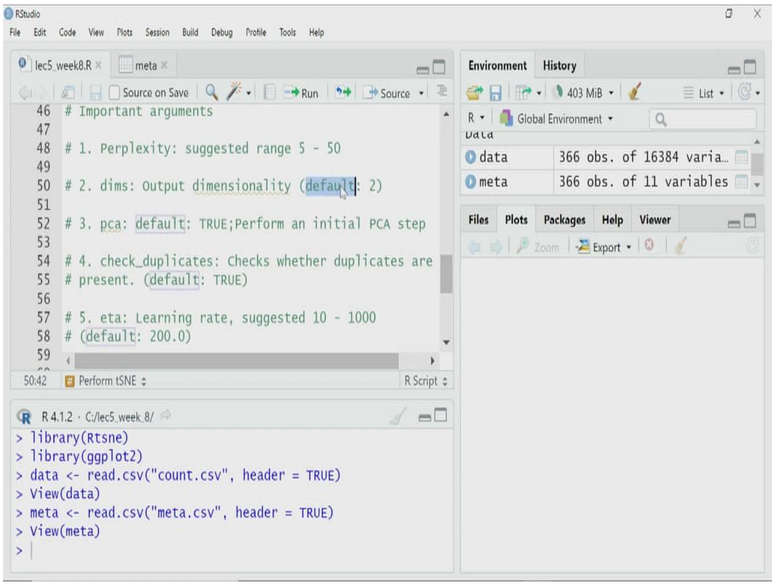


```
# Important arguments

# 1. Perplexity: suggested range 5 - 50

# 2. dims: Output dimensionality (default: 2)

# 3. pca: default: TRUE;Perform an initial PCA step

# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

Data

```
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```



```
# Important arguments

# 1. Perplexity: suggested range 5 - 50

# 2. dims: Output dimensionality (default: 2)

# 3. pca: default: TRUE;Perform an initial PCA step

# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)
```

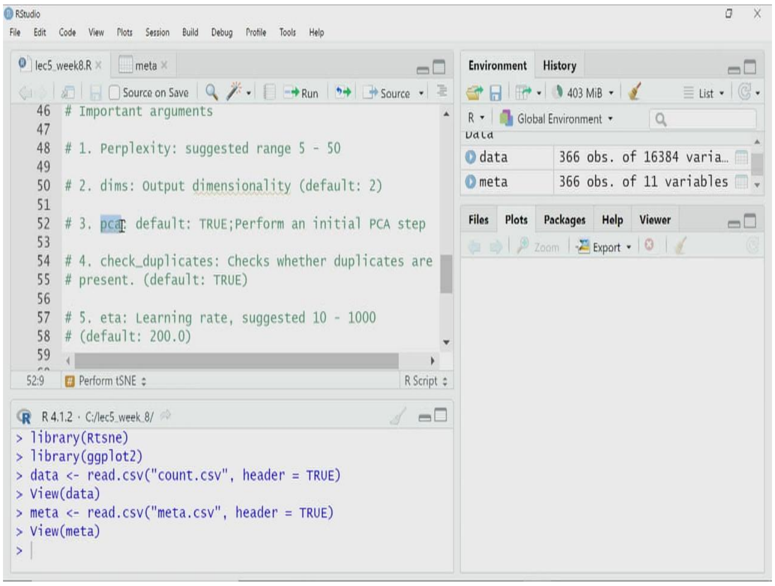
```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

Data

```
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```

So the suggestions range is 5 to 50, we will decide the particular value in between. dims stands for the output dimension. By default its value is two that means from higher dimension, I want to map the data in two dimension. And in this particular example, I will keep that a default value 2.

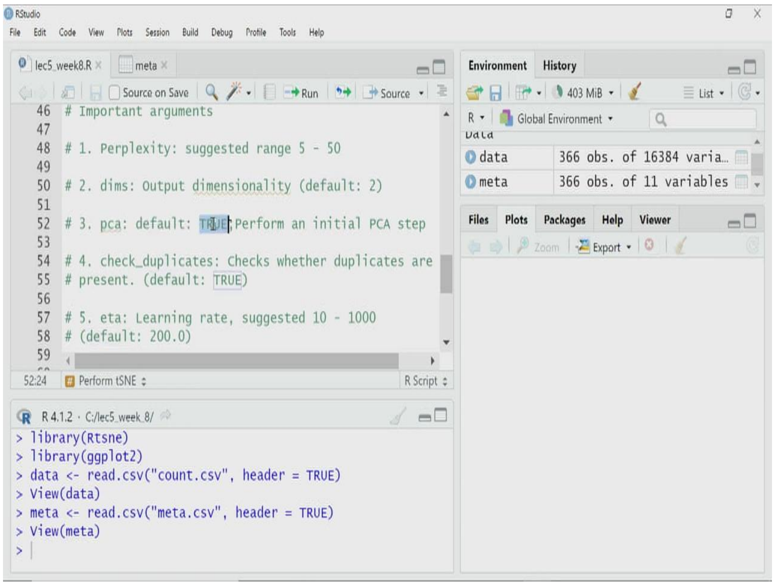
(Refer Slide Time: 8:16)



```
# Important arguments

# 1. Perplexity: suggested range 5 - 50

# 2. dims: Output dimensionality (default: 2)

# 3. pca: default: TRUE;Perform an initial PCA step

# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> 
```

data	366 obs. of 16384 varia...
meta	366 obs. of 11 variables



```
# Important arguments

# 1. Perplexity: suggested range 5 - 50

# 2. dims: Output dimensionality (default: 2)

# 3. pca: default: TRUE;Perform an initial PCA step

# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)
```

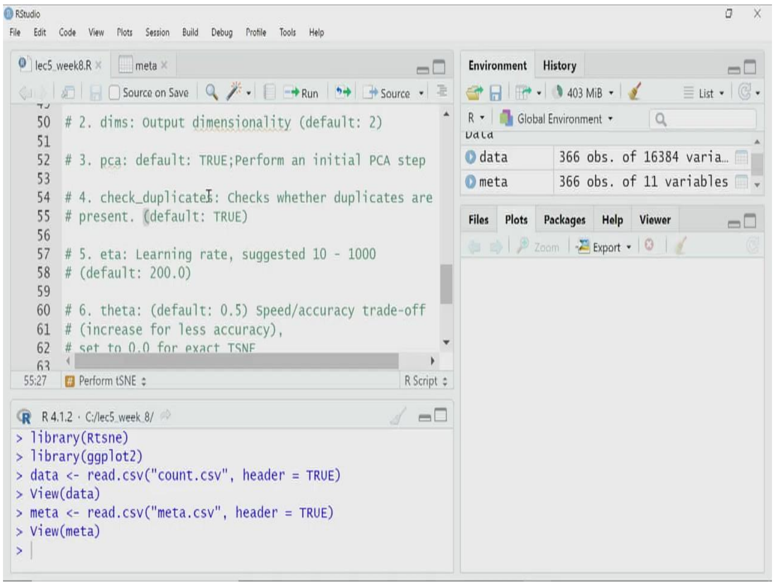
```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> 
```

data	366 obs. of 16384 varia...
meta	366 obs. of 11 variables

PCA is another argument by default it is true, it is usually recommended that if you are performing tSNE with a huge dimensional original data for example, here are 16,000 Something dimension, it is better to perform PCA, first principal component analysis first to reduce the dimension and then use tSNE on that reduced dimension data.

So that is why are tSNE function by default has put PCA as a first step initial state and I will keep that here in this particular example, if you want you can make it false. Then one important thing in using the tSNE algorithm, that is the algorithm will be in trouble if there are duplicate data.

(Refer Slide Time: 9:10)

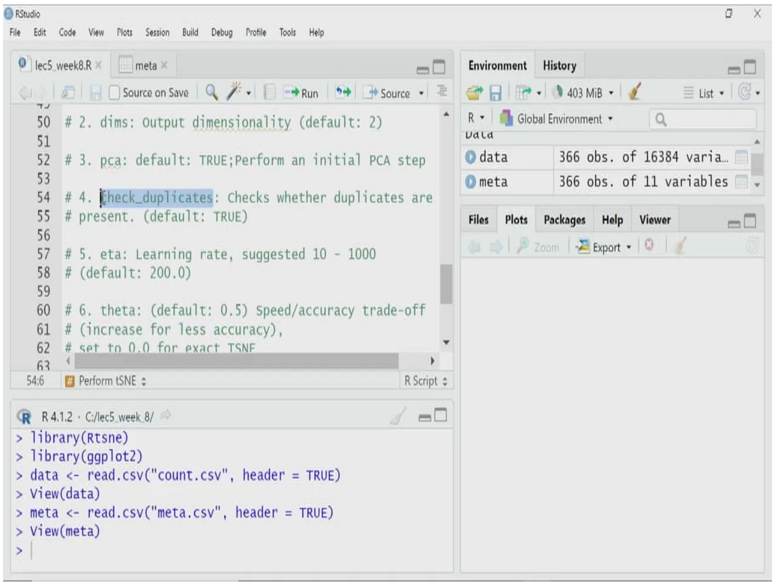


```
# 2. dims: Output dimensionality (default: 2)

# 3. pca: default: TRUE;Perform an initial PCA step

# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

Data

data	366 obs. of 16384 varia…
meta	366 obs. of 11 variables

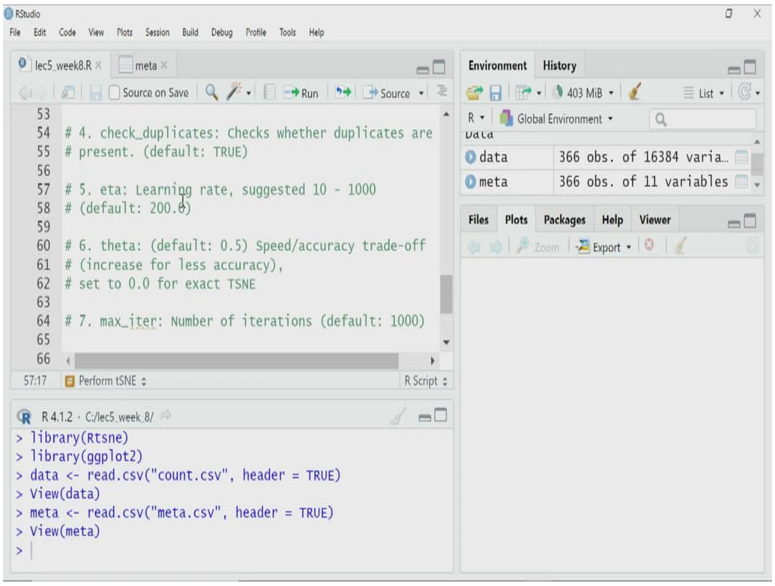


```
# 2. dims: Output dimensionality (default: 2)

# 3. pca: default: TRUE;Perform an initial PCA step

# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

Data

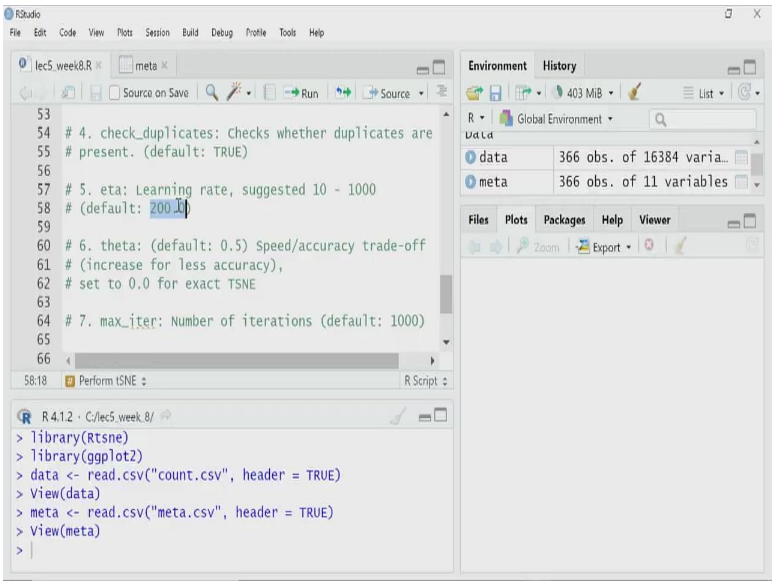
data	366 obs. of 16384 varia…
meta	366 obs. of 11 variables



```
# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)
```

```
Data
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> 
```



```
# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)
```

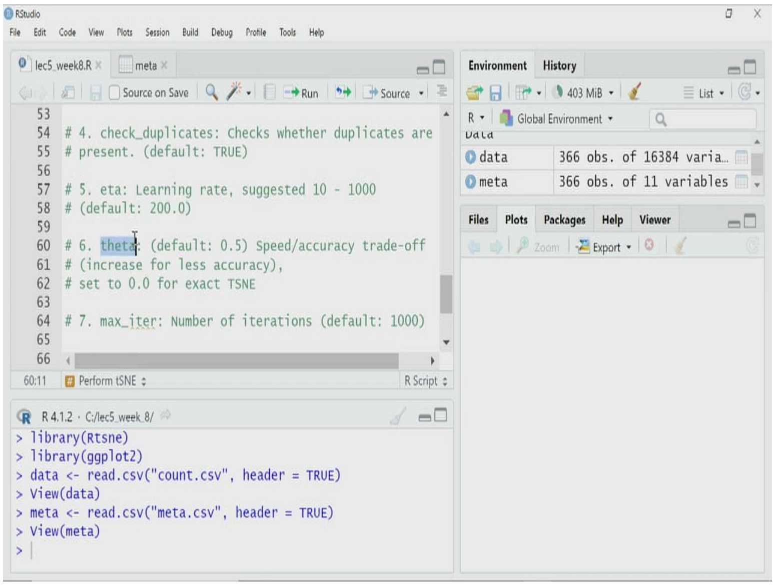
```
Data
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> 
```

So either you yourself check if there is any duplicate or not, or otherwise, there is a argument in this function called Check underscore duplicates, and by default is true, you should keep it default value at true. The fifth important argument for tSNE function here is the learning rate defined by ETA. And it is usually suggested that the learning rate should vary from 10 to 1000 and the default value in case of Rtsne is 200 learning rate is one parameter which decides the step size when the optimization algorithm works.

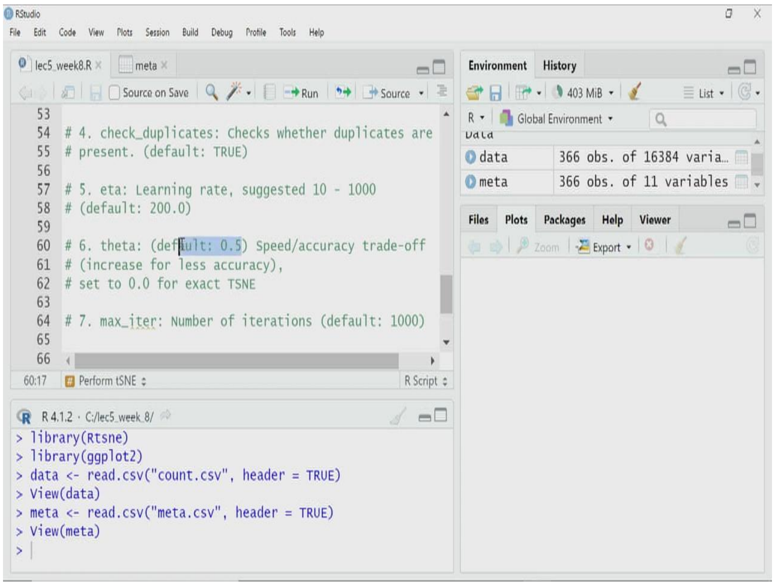
And if you remember our lecture on tSNE, you can understand that it is trying to minimize the difference between two probability distribution. So in that case, it is trying to do an optimization. And in this ETA the learning rate parameter is deciding the step size for that optimization algorithm.

(Refer Slide Time: 9:59)



```
# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```



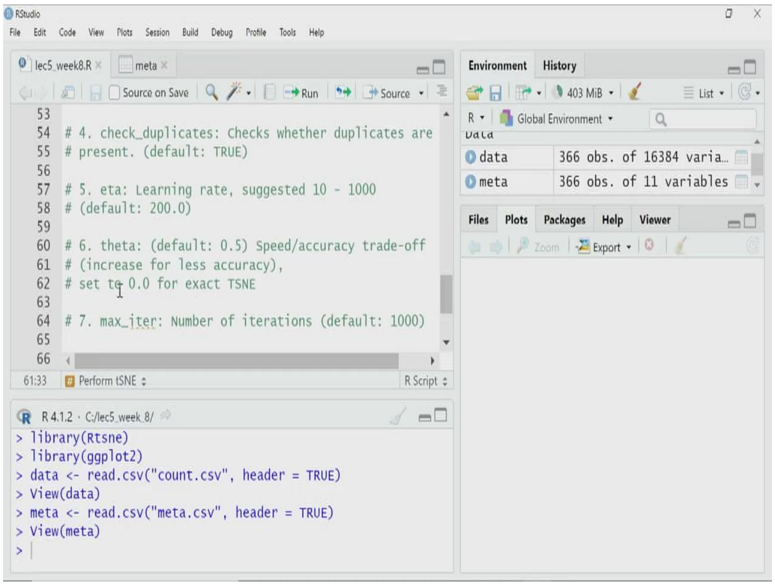
```
# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

So, if you read If required, you have to change this value, then there is a parameter called theta the default value is 0.5. And it decides the ratio between speed and accuracy it decide the tradeoff between speed and accuracy. Remember, in tSNE, you have an exact method to minimize the difference between two probability distributions, but that could be very time consuming.

Now, that is why these particular tSNE they function has a he has used a algorithm which is which is doing this optimization by approximation. So, that makes it faster.

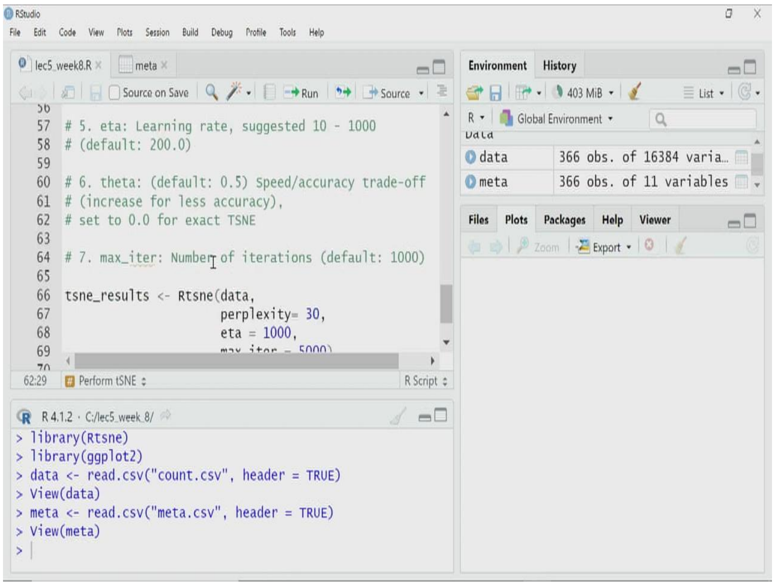
(Refer Slide Time: 10:43)



```
# 4. check_duplicates: Checks whether duplicates are
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

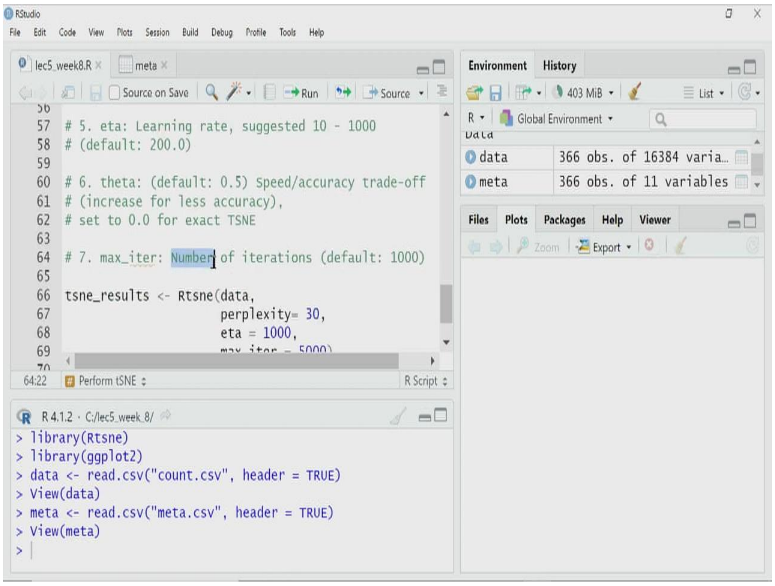
```
data    366 obs. of 16384 varia...
meta    366 obs. of 11 variables
```



```
# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)

tsne_results <- Rtsne(data,
                      perplexity= 30,
                      eta = 1000,
                      max_iter = 5000)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

```
data    366 obs. of 16384 varia...
meta    366 obs. of 11 variables
```



```
# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)

tsne_results <- Rtsne(data,
                      perplexity= 30,
                      eta = 1000,
                      max_iter = 5000)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

```
Data
data   366 obs. of 16384 varia…
meta   366 obs. of 11 variables
```

tsne_results ← Rtsne(data,

perplexity = 30,

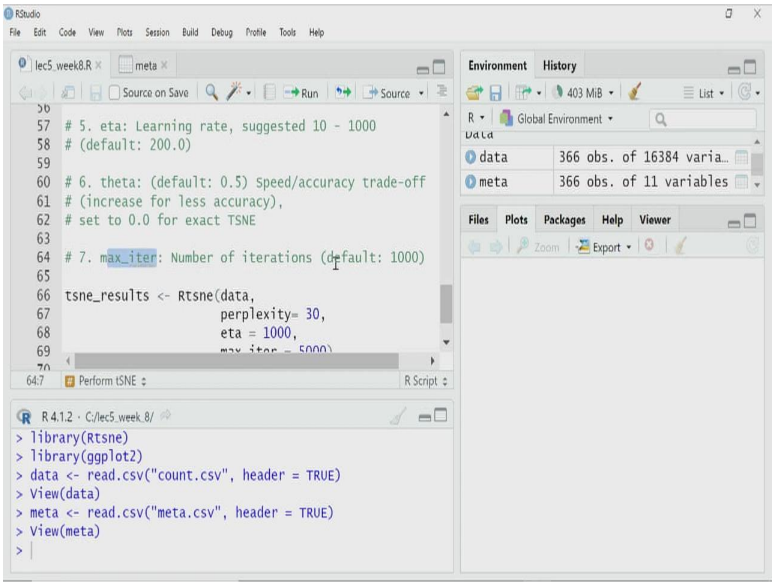
ets = 1000,

max_iter = 5000)

But if you want that, no, I do not want the approximation or you want the exact tSNE then you have to set this theta at zero. Otherwise, you can leave it at this particular value 0.5 for this particular approximate algorithm. Another important parameter or argument for using this function is max ETA.

There is a number of iterations it should do remember, it is an optimization technique, if you run this algorithm for a short time a number of states you may not have reached the optimum solution, and you really do not know when you will reach the optimum solution. So, you should try to balance between the time taken for optimal solution to solving the problem and the good solution. So, this number of iteration that you want the maximum number of iterations.

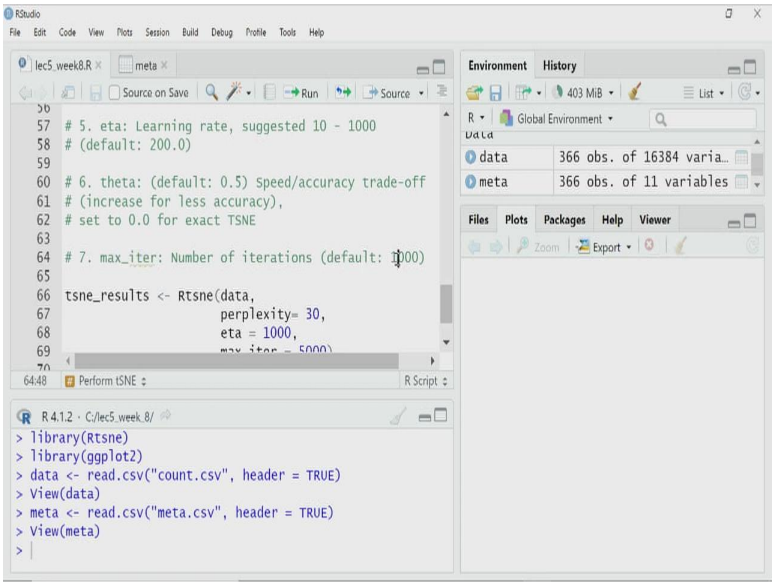
(Refer Slide Time: 11:31)



```
# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)

tsne_results <- Rtsne(data,
                      perplexity= 30,
                      eta = 1000,
                      max_iter = 5000)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

```
data   366 obs. of 16384 varia…
meta   366 obs. of 11 variables
```



```
# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)

tsne_results <- Rtsne(data,
                      perplexity= 30,
                      eta = 1000,
                      max_iter = 5000)
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
>
```

```
data   366 obs. of 16384 varia…
meta   366 obs. of 11 variables
```

tsne_results ← Rtsne(data,

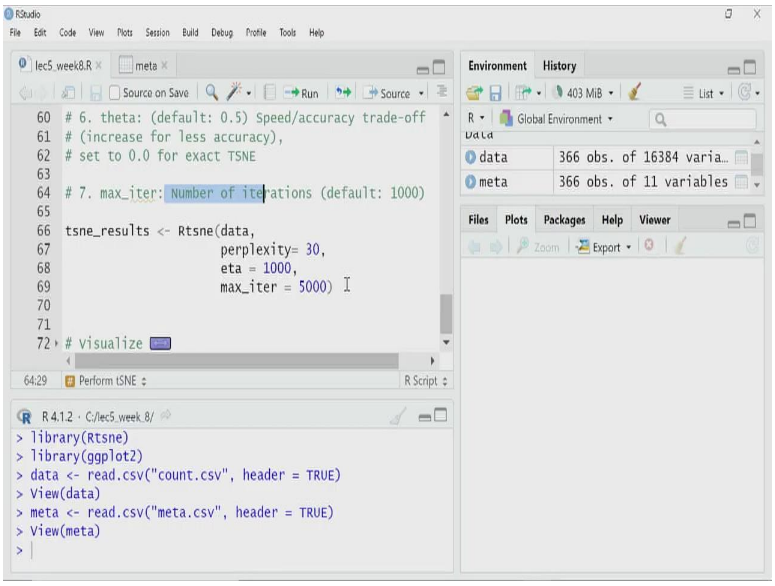
perplexity = 30,

ets = 1000,

max_iter = 5000)

That you want this algorithm should use is decided by this max eater obviously, I cannot keep it to one crore or something then it will keep on working for a very long time, the default value is 1000 You can make a change to 2000 5000 something like that. So, these are few of the important argument as I say please look into the documentation file and you will see there are many other para arguments which could be very useful.

(Refer Slide Time: 11:59)



```
# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)

tsne_results <- Rtsne(data,
                      perplexity= 30,
                      eta = 1000,
                      max_iter = 5000)


# visualize
```

```
> library(Rtsne)
> library(ggplot2)
> data <- read.csv("count.csv", header = TRUE)
> View(data)
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> |
```

```
data    366 obs. of 16384 varia…
meta    366 obs. of 11 variables
```

tsne_results ← Rtsne(data,

perplexity = 30,

ets = 1000,

max_iter = 5000)

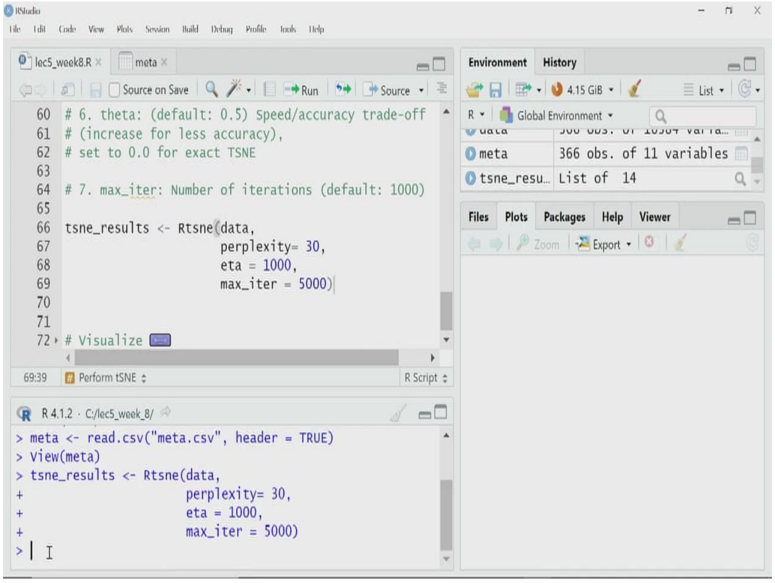
So, let us go into our data and the Rtsne function and I will be using. So here I am using the Rtsne function the first argument is the data the matrix that I have with rows for sample and the columns for the genes, I am setting perplexity equal to 30. I have set ETA equal to 1000 learning that is called 1000 and I am keeping maximum at value high value 5000.

Now remember, the main function of tSNE doing tSNE is to visualize the data bring down that higher dimensional data to a lower dimension like to dimension for visualization purpose, I am not using tSNE for clustering of my data, they are maybe some cluster may appear in my visualization during when I do use this need to visualize, but we have to remember in the last class also I discussed that this cluster size cluster distance may not be meaningful when I am doing tSNE.

So, the whole purpose is visualization. So, I want to see the heterogeneity in distribution of gene expression in among these 366 samples. That is what I want to visualize in this particular example, I do not want all data to clump in one place, I do not also want them to hold distribute the whole space without giving any pattern. So, I want some amount of pattern so that I can make a meaning out of these heterogeneity in gene expression.

So, that is the purpose and when you are trying to do that you have to play with these different arguments or parameters for this particular algorithm. As I have discussed in this case, I have used this particular parameter values perplexity 30 at 1000 maximum of 5000. Because the final outcome, that visualization that I get the plot I get is something similar to what the original authors in their paper have created. So, that also give a good visualization of distribution of heterogeneity depending upon cell type in this particular data set.

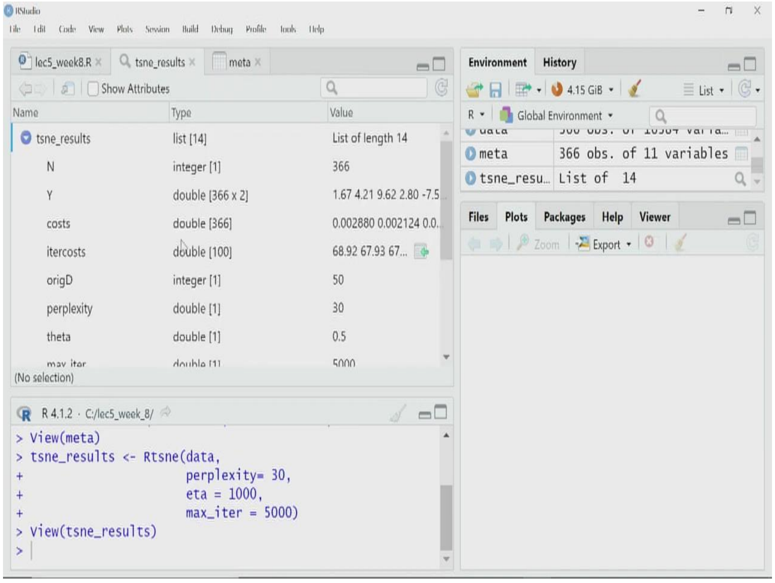
(Refer Slide Time: 14:04)



```
# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)

tsne_results <- Rtsne(data,
                      perplexity= 30,
                      eta = 1000,
                      max_iter = 5000)


# Visualize
```

```
meta     366 obs. of 11 variables
tsne_resu… List of  14
```

```
> meta <- read.csv("meta.csv", header = TRUE)
> View(meta)
> tsne_results <- Rtsne(data,
+                       perplexity= 30,
+                       eta = 1000,
+                       max_iter = 5000)
>
```



Name	Type	Value
tsne_results	list [14]	List of length 14
N	integer [1]	366
Y	double [366 x 2]	1.67 4.21 9.62 2.80 -7.5…
costs	double [366]	0.002880 0.002124 0.0…
itercosts	double [100]	68.92 67.93 67…
origD	integer [1]	50
perplexity	double [1]	30
theta	double [1]	0.5
max_iter	double [1]	5000

```
meta     366 obs. of 11 variables
tsne_resu… List of  14
```

```
> View(meta)
> tsne_results <- Rtsne(data,
+                       perplexity= 30,
+                       eta = 1000,
+                       max_iter = 5000)
> View(tsne_results)
>
```

tsne_results ← Rtsne(data,

perplexity = 30,

ets = 1000,

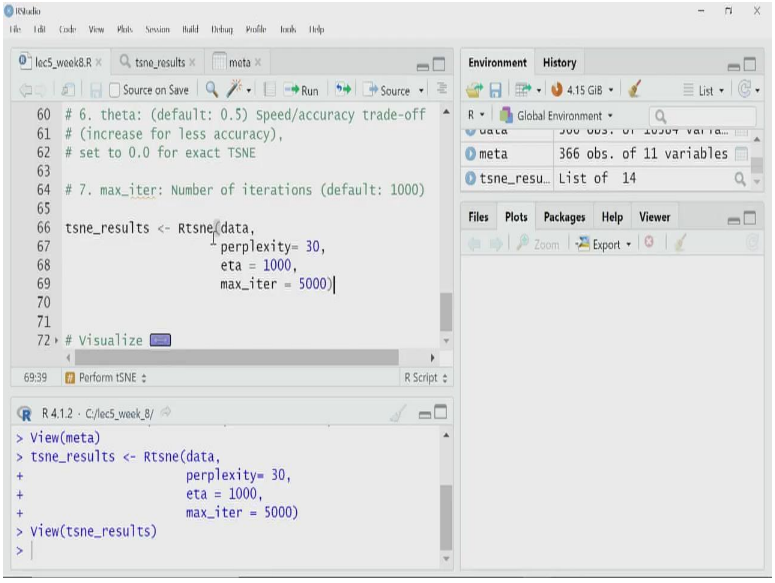
max_iter = 5000)

So let us learn it may take some time depending upon your computer, so you have to wait patiently, at last the tSNE has worked and output you can see here in the environment tab, there is a new variable has come tSNE underscore result where I have assigned all the result of the tSNE.

So it is a list so, let me click it to see what we have, we have lots of information here and all these things you can dig into and use for your visualization or analysis purpose, the most important thing is this Y variable and you can see it has a dimension of 366 into 2 that means it is two dimensional data.

So I have 366 samples which are mapped into these two dimensions. So now I have 366 into 2 a variable with 266 rows and 2 columns. So this is why I will use to visualize because in this way I have all these 366 examples mapped in two dimension that information is there.

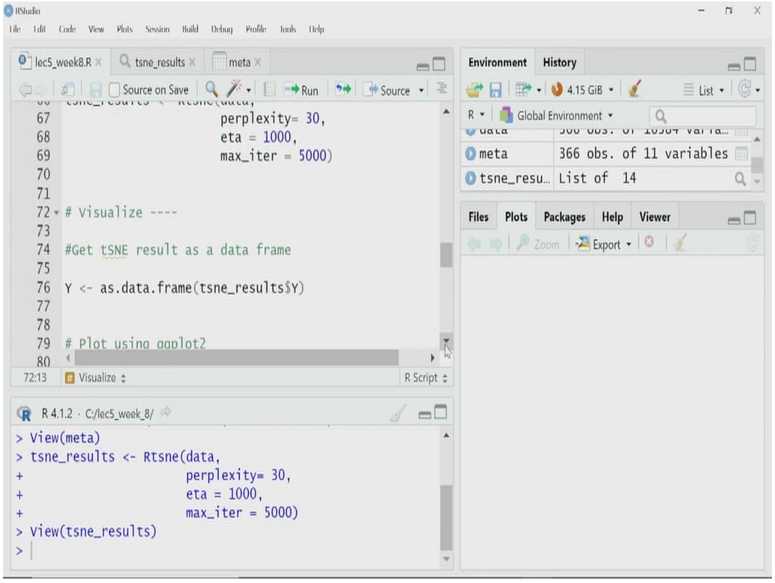
(Refer Slide Time: 15:07)



```
# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)

tsne_results <- Rtsne(data,
                      perplexity= 30,
                      eta = 1000,
                      max_iter = 5000)


# Visualize
```

meta	366 obs. of 11 variables
tsne_resu...	List of 14

```
> View(meta)
> tsne_results <- Rtsne(data,
+                       perplexity= 30,
+                       eta = 1000,
+                       max_iter = 5000)
> View(tsne_results)
>
```



```
tsne_results <- Rtsne(data,
                      perplexity= 30,
                      eta = 1000,
                      max_iter = 5000)


# Visualize ----

#Get tSNE result as a data frame

Y <- as.data.frame(tsne_results$Y)


# Plot using ggplot2
```

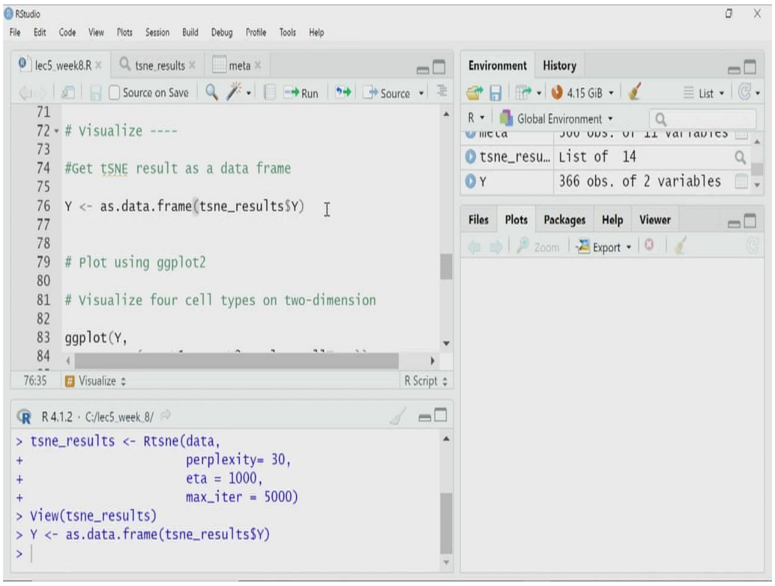
```
data        366 obs. of 10304 varia...
meta        366 obs. of 11 variables
tsne_resu…  List of 14
```

```
> View(meta)
> tsne_results <- Rtsne(data,
+                       perplexity= 30,
+                       eta = 1000,
+                       max_iter = 5000)
> View(tsne_results)
> 
```

tsne_results ← as.data.frame(tsne_results$Y)

So, I want to visualize it. So, what I will do you can use any other visualization option in R, but I will use here in this case I will use gg plot 2. So, I have already installed and loaded gg plot 2. Now, this is in least the data is in least format and in that there is a particular variable Y what I will do I will convert this into a data frame so, that my gg plot 2 can use that.

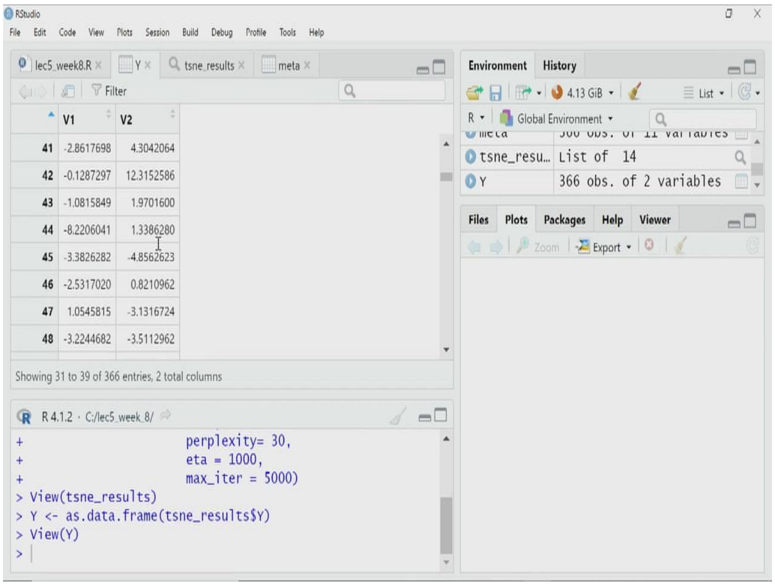
(Refer Slide Time: 15:33)



```
# Visualize ----

#Get tSNE result as a data frame

Y <- as.data.frame(tsne_results$Y)


# Plot using ggplot2

# Visualize four cell types on two-dimension

ggplot(Y,
```

```
> tsne_results <- Rtsne(data,
+                       perplexity= 30,
+                       eta = 1000,
+                       max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> |
```

O	tsne_resu...	List of 14
O	Y	366 obs. of 2 variables

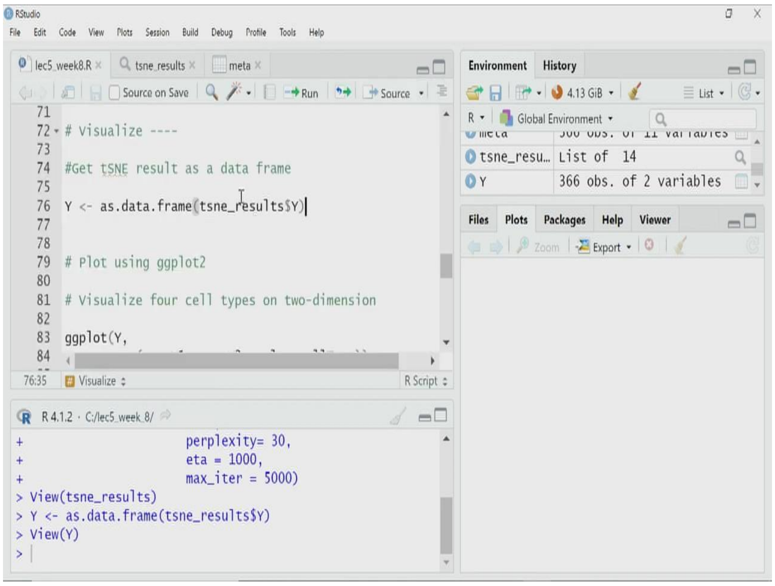


	V1	V2
41	-2.8617698	4.3042064
42	-0.1287297	12.3152586
43	-1.0815849	1.9701600
44	-8.2206041	1.3386280
45	-3.3826282	-4.8562623
46	-2.5317020	0.8210962
47	1.0545815	-3.1316724
48	-3.2244682	-3.5112962

Showing 31 to 39 of 366 entries, 2 total columns

```
+                       perplexity= 30,
+                       eta = 1000,
+                       max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> |
```

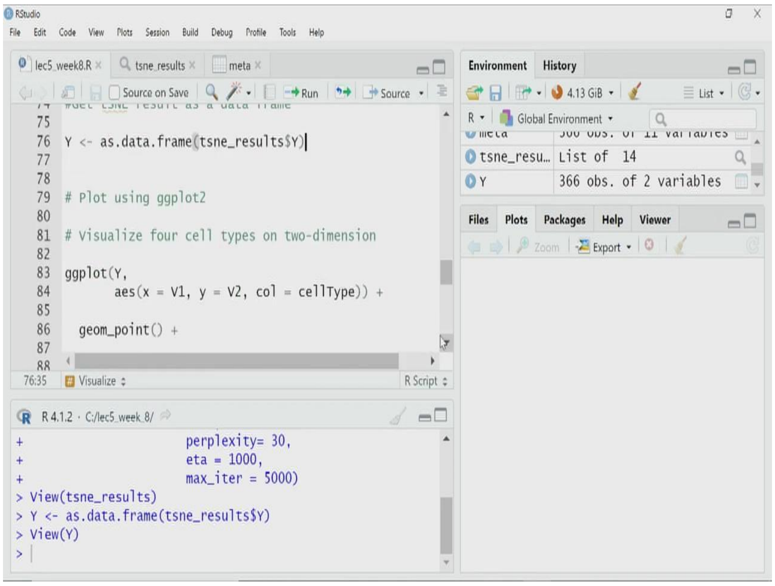
O	tsne_resu...	List of 14
O	Y	366 obs. of 2 variables



```
# Visualize ----

#Get tSNE result as a data frame

Y <- as.data.frame(tsne_results$Y)


# Plot using ggplot2

# Visualize four cell types on two-dimension

ggplot(Y,
```

```
meta        300 obs. of 11 variables
tsne_resu…  List of 14
Y           366 obs. of 2 variables
```

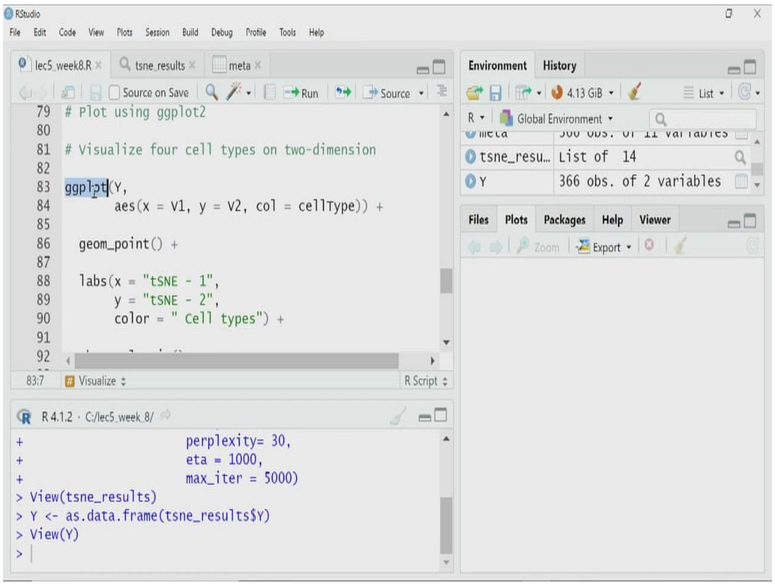
```
+                       perplexity= 30,
+                       eta = 1000,
+                       max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
>
```



```
Y <- as.data.frame(tsne_results$Y)


# Plot using ggplot2

# Visualize four cell types on two-dimension

ggplot(Y,
       aes(x = V1, y = V2, col = cellType)) +

  geom_point() +
```

```
meta        300 obs. of 11 variables
tsne_resu…  List of 14
Y           366 obs. of 2 variables
```

```
+                       perplexity= 30,
+                       eta = 1000,
+                       max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
>
```



```
# Plot using ggplot2

# Visualize four cell types on two-dimension

ggplot(Y,
       aes(x = V1, y = V2, col = cellType)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2",
       color = " Cell types") +
```

```
meta        300 obs. of 11 variables
tsne_resu…  List of 14
Y           366 obs. of 2 variables
```

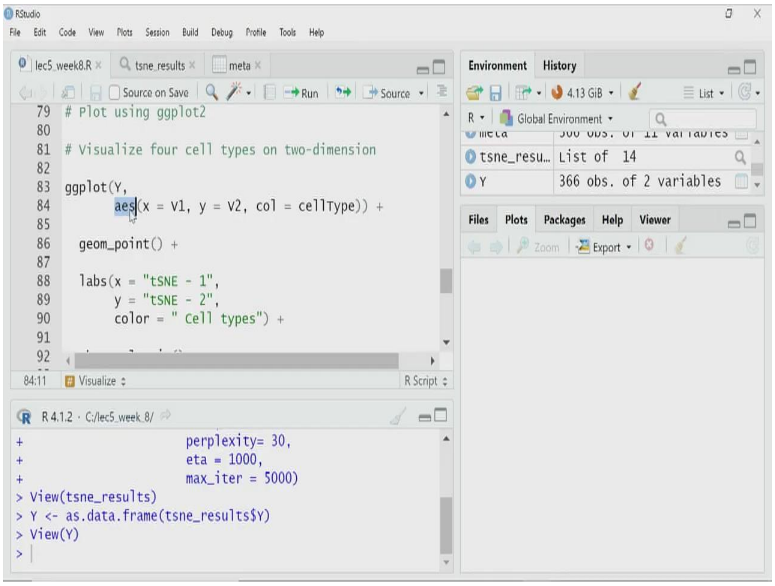
```
+                       perplexity= 30,
+                       eta = 1000,
+                       max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
>
```

```
ggplot(Y,
          aes(x = v1, y = v2, col = cellType)) +
  geom_point() +
  labs(x = “tSNE - 1”,
          y = “tSNE - 2:,
           color = “ Cell types”) +
  theme_classic()
```

So, what I am doing I am using a function called as data frame so, it will convert these data into data frame and which data I wonder Y variable from the list of tSNE underscore results. So, I am writing tSNE underscore result then the dollar sign and y this is the argument to the function as dot data dot frame. So, it will convert these into a data frame and I assigned that data frame into Y variable.

So, let us do that I have Y you can see Y has 366 observations and 2 variables if you click you can see you have 366 samples now, they are values from the 16,000 dimension is now mapped to these two dimensional values like V 1 and V 2. So, I will plot this data to visualize as I said I will use gg plot 2. So, I am calling gg plot function here giving the data frame Y which has 366 samples two dimension two columns AES the data

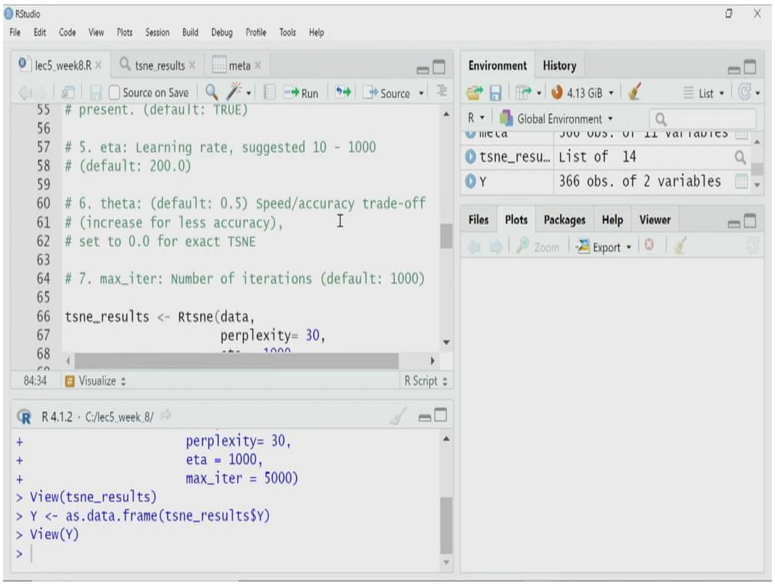
(Refer Slide Time: 16:35)



```
# Plot using ggplot2

# Visualize four cell types on two-dimension

ggplot(Y,
        aes(x = V1, y = V2, col = cellType)) +

  geom_point() +

  labs(x = "tSNE - 1",
        y = "tSNE - 2",
        color = " Cell types") +
```

```
+                     perplexity= 30,
+                     eta = 1000,
+                     max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
>
```

tsne_resu...	List of 14
Y	366 obs. of 2 variables

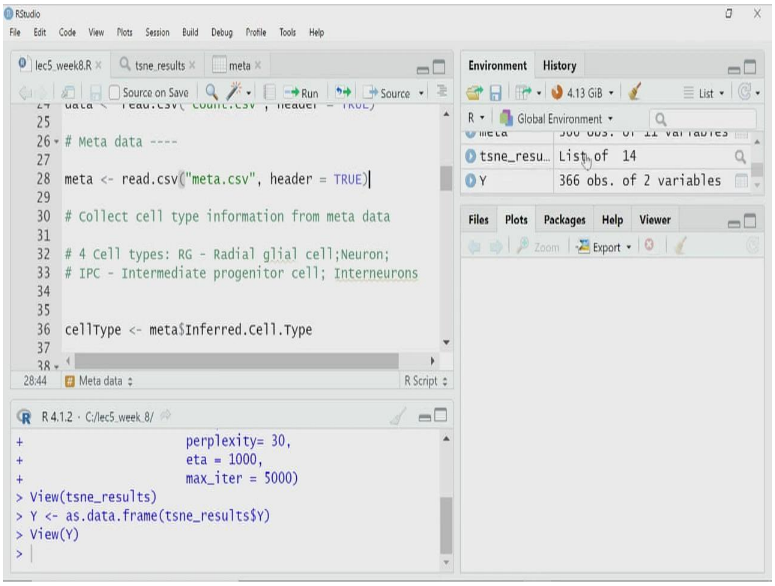


```
# present. (default: TRUE)

# 5. eta: Learning rate, suggested 10 - 1000
# (default: 200.0)

# 6. theta: (default: 0.5) Speed/accuracy trade-off
# (increase for less accuracy),
# set to 0.0 for exact TSNE

# 7. max_iter: Number of iterations (default: 1000)

tsne_results <- Rtsne(data,
                      perplexity= 30,
```

```
+                    perplexity= 30,
+                    eta = 1000,
+                    max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
>
```

tsne_resu... List of 14

Y 366 obs. of 2 variables

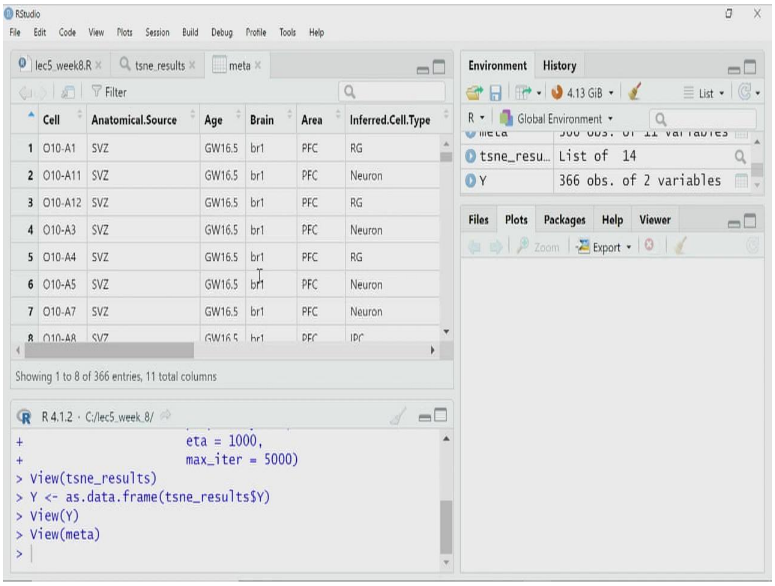


```
# Meta data ----

meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons


cellType <- meta$Inferred.Cell.Type
```

```
+                    perplexity= 30,
+                    eta = 1000,
+                    max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
>
```

tsne_resu... List of 14

Y 366 obs. of 2 variables



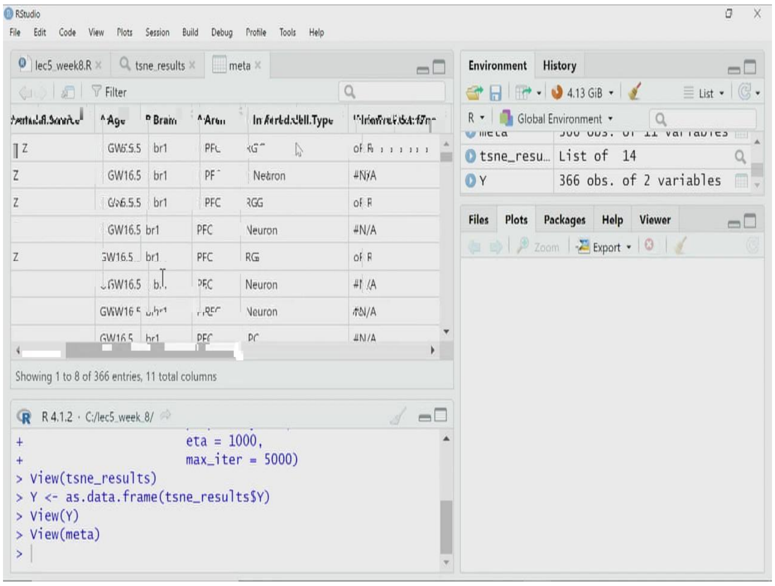
	Cell	Anatomical.Source	Age	Brain	Area	Inferred.Cell.Type
1	010-A1	SVZ	GW16.5	br1	PFC	RG
2	010-A11	SVZ	GW16.5	br1	PFC	Neuron
3	010-A12	SVZ	GW16.5	br1	PFC	RG
4	010-A3	SVZ	GW16.5	br1	PFC	Neuron
5	010-A4	SVZ	GW16.5	br1	PFC	RG
6	010-A5	SVZ	GW16.5	br1	PFC	Neuron
7	010-A7	SVZ	GW16.5	br1	PFC	Neuron
8	010-A8	SVZ	GW16.5	br1	PFC	IPC

Showing 1 to 8 of 366 entries, 11 total columns

```
+                    eta = 1000,
+                    max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> View(meta)
>
```

tsne_resu... List of 14

Y 366 obs. of 2 variables



[illegible]	Age	Brain	Area	[illegible].Cell.Type	[illegible]
Z	GW6.5.5	br1	PFC	RG	of R
Z	GW16.5	br1	PF	Neuron	#N/A
Z	GW6.5.5	br1	PFC	RGG	of R
	GW16.5	br1	PFC	Neuron	#N/A
Z	GW16.5	br1	PFC	RG	of R
	GW16.5	b1	PFC	Neuron	#N/A
	GWW16.5	br1	PFC	Neuron	#N/A
	GW16.5	br1	PFC	PC	#N/A

Showing 1 to 8 of 366 entries, 11 total columns

meta	366 obs. of 11 variables
tsne_resu...	List of 14
Y	366 obs. of 2 variables

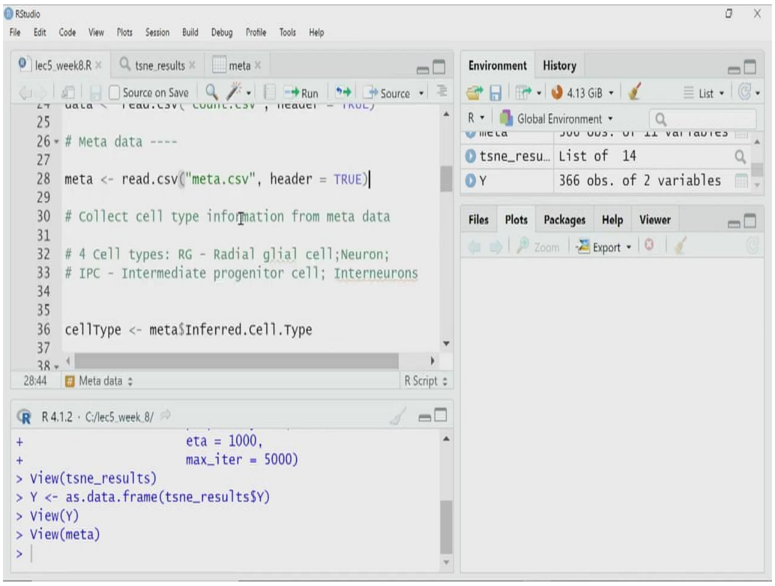
R 4.1.2 · C:/lec5_week_8/

```
+                      eta = 1000,
+                      max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> View(meta)
> 
```

```
ggplot(Y,
          aes(x = v1, y = v2, col = cellType)) +
  geom_point() +
  labs(x = “tSNE - 1”,
          y = “tSNE - 2:,
          color = “ Cell types”) +
  theme_classic()
```

I am calling the aesthetic in aesthetic, I am saying x equal to V1 the first column Y equal to V2 the second column and I have said color it in categorize it depending upon Cell type, what is cell type? let me go back and check my original data file. So, here in meta I have loaded the metadata and if you can see, I have different information including this cell type, So what I will do, I will create a variable where the cell type information will be stored.

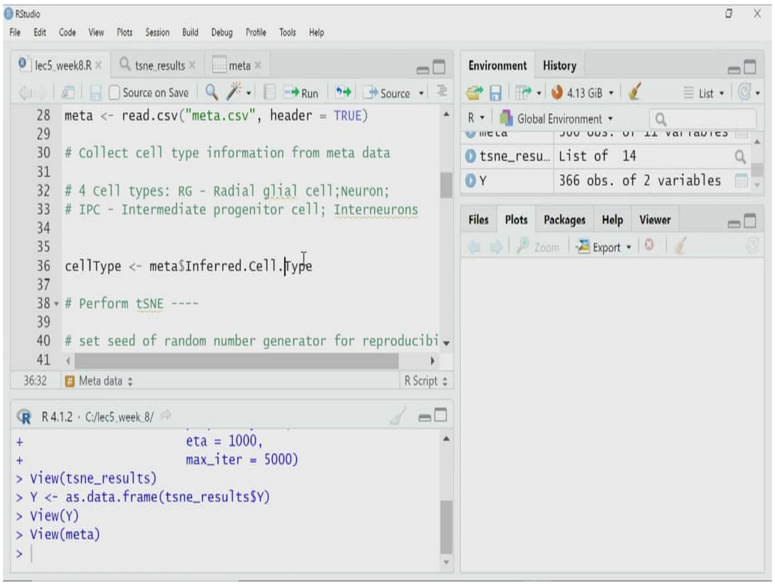
(Refer Slide Time: 17:14)



```
data <- read.csv("count.csv", header = TRUE)

# Meta data ----

meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons


cellType <- meta$Inferred.Cell.Type

```

```
+                     eta = 1000,
+                     max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> View(meta)
>
```

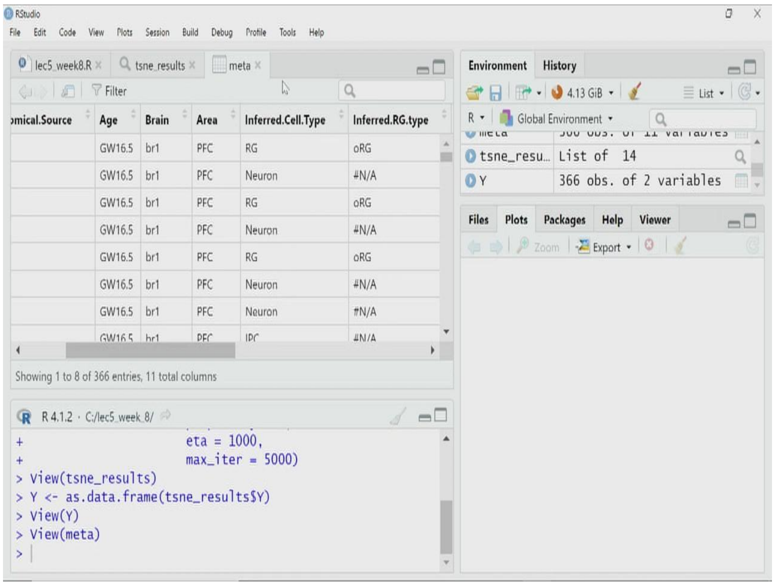
```
meta          366 obs. of 11 variables
tsne_resu…    List of 14
Y             366 obs. of 2 variables
```



```
meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons


cellType <- meta$Inferred.Cell.Type

# Perform tSNE ----

# set seed of random number generator for reproducibi
```

```
+                     eta = 1000,
+                     max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> View(meta)
>
```

```
meta          366 obs. of 11 variables
tsne_resu…    List of 14
Y             366 obs. of 2 variables
```

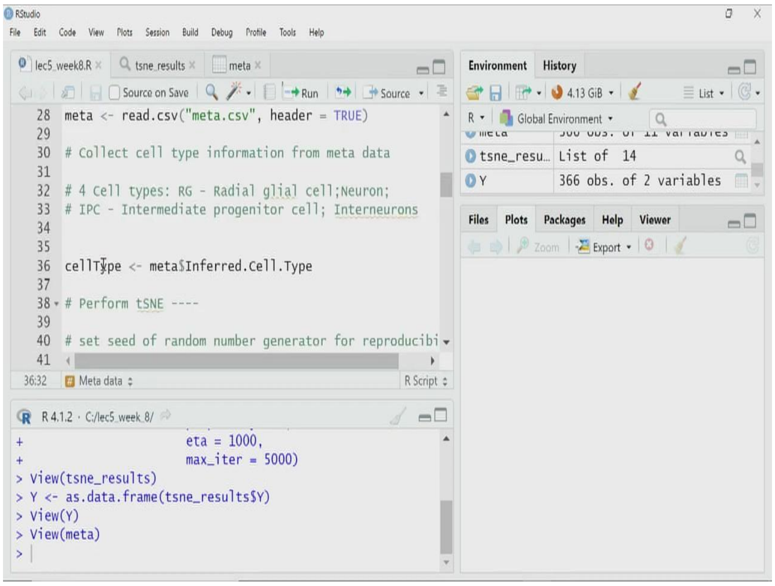


omical.Source	Age	Brain	Area	Inferred.Cell.Type	Inferred.RG.type
	GW16.5	br1	PFC	RG	oRG
	GW16.5	br1	PFC	Neuron	#N/A
	GW16.5	br1	PFC	RG	oRG
	GW16.5	br1	PFC	Neuron	#N/A
	GW16.5	br1	PFC	RG	oRG
	GW16.5	br1	PFC	Neuron	#N/A
	GW16.5	br1	PFC	Neuron	#N/A
	GW16.5	br1	PFC	IPC	#N/A

Showing 1 to 8 of 366 entries, 11 total columns

```
+                     eta = 1000,
+                     max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> View(meta)
>
```

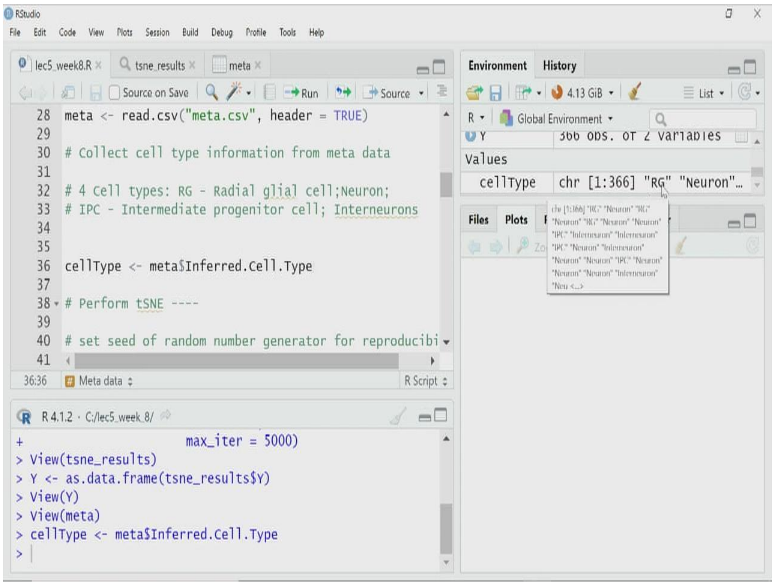
```
meta          366 obs. of 11 variables
tsne_resu…    List of 14
Y             366 obs. of 2 variables
```



```r
meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons


cellType <- meta$Inferred.Cell.Type

# Perform tSNE ----

# set seed of random number generator for reproducibi
```

```
tsne_resu…   List of  14
Y            366 obs. of 2 variables
```

```
+                       eta = 1000,
+                       max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> View(meta)
> 
```



```r
meta <- read.csv("meta.csv", header = TRUE)

# Collect cell type information from meta data

# 4 Cell types: RG - Radial glial cell;Neuron;
# IPC - Intermediate progenitor cell; Interneurons


cellType <- meta$Inferred.Cell.Type

# Perform tSNE ----

# set seed of random number generator for reproducibi
```

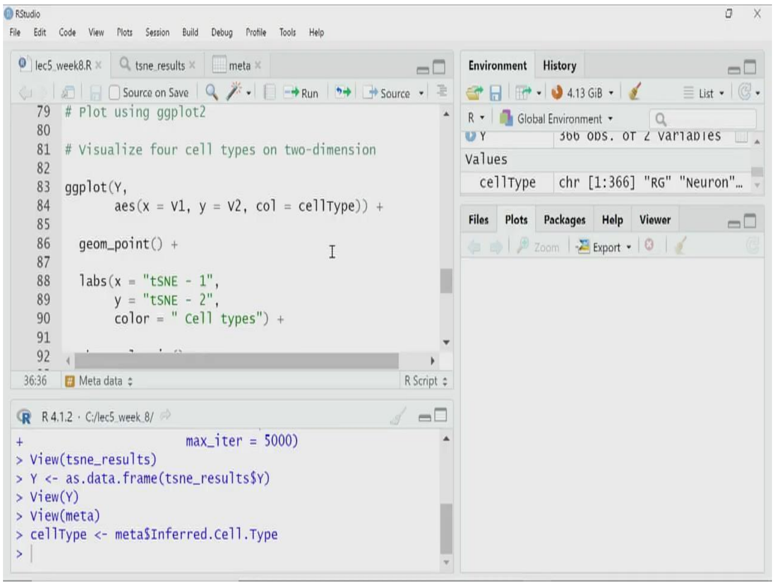
```
Y            366 obs. of 2 variables
Values
  cellType   chr [1:366] "RG" "Neuron"…
```

```
chr [1:366] "RG" "Neuron" "RG"
"Neuron" "RG" "Neuron" "Neuron"
"IPC" "Interneuron" "Interneuron"
"IPC" "Neuron" "Interneuron"
"Neuron" "Neuron" "IPC" "Neuron"
"Neuron" "Neuron" "Interneuron"
"Neu <...>
```

```
+                       max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> View(meta)
> cellType <- meta$Inferred.Cell.Type
> 
```

So, to do that, I have written here that meta dollar inferred dot sale dot type right so I am extracting this column data from meta and I am assigning that to cell type. So, let me do that. So, now I have a list of cell type right RG neuron inter neuron something like that for all those 366 samples. So, now, what I am doing here during plotting that information is not used during tSNE now during plotting whatever you want, that when you create the plot, color code each sample depending upon the cell type information that I have.

(Refer Slide Time: 17:56)



```
# Plot using ggplot2

# Visualize four cell types on two-dimension

ggplot(Y,
       aes(x = V1, y = V2, col = cellType)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2",
       color = " Cell types") +
```

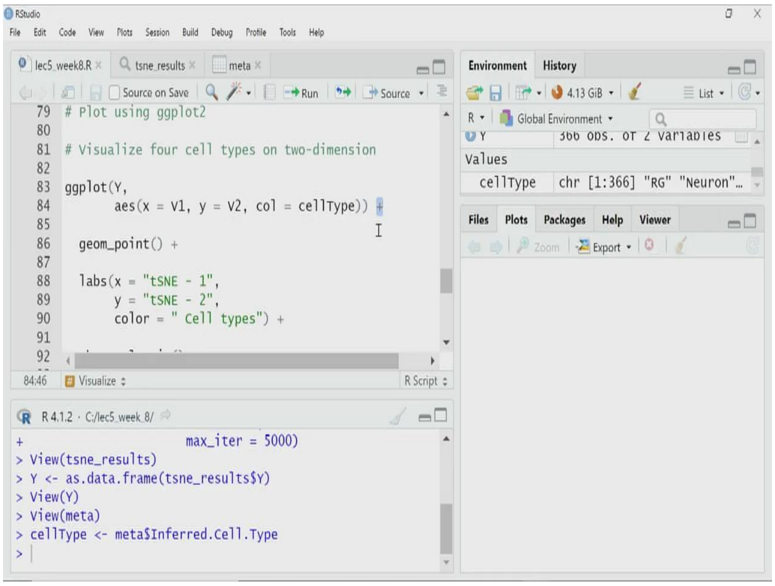
```
Y          366 obs. of 2 variables
Values
  cellType chr [1:366] "RG" "Neuron"…
```

```
+                         max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> View(meta)
> cellType <- meta$Inferred.Cell.Type
>
```

```
ggplot(Y,
         aes(x = v1, y = v2, col = cellType)) +
  geom_point() +
  labs(x = “tSNE - 1”,
         y = “tSNE - 2:,
         color = “ Cell types”) +
  theme_classic()
```

So what I am writing color equal to cell type and then this is the main thing where I have given the data and aesthetic if you remember, now, I have to give the third important thing I have to give the geometry.

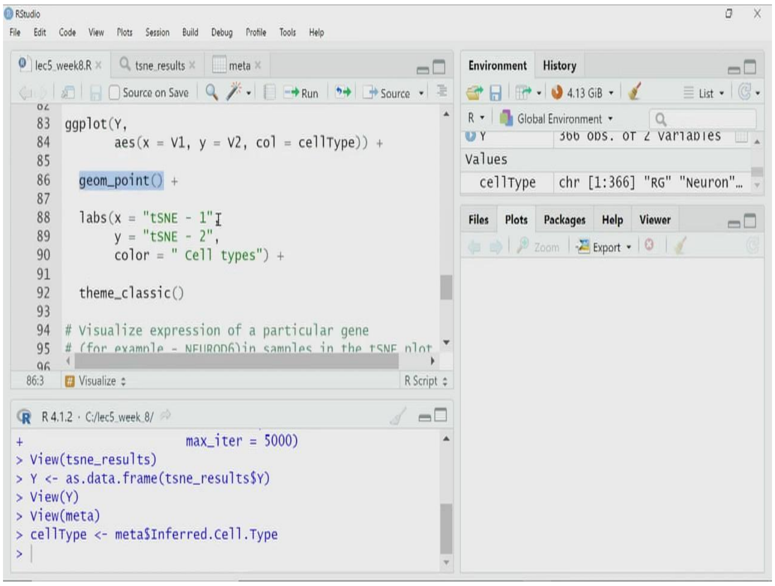
(Refer Slide Time: 18:10)



```
# Plot using ggplot2

# Visualize four cell types on two-dimension

ggplot(Y,
       aes(x = V1, y = V2, col = cellType)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2",
       color = " Cell types") +
```

```
Y        366 obs. of 2 variables
Values
cellType  chr [1:366] "RG" "Neuron"…
```

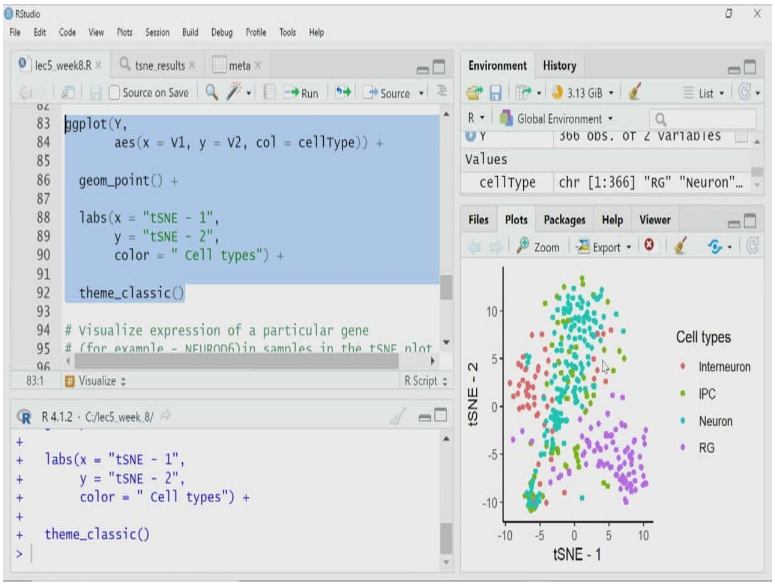
```
+                       max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> View(meta)
> cellType <- meta$Inferred.Cell.Type
> 
```



```
ggplot(Y,
       aes(x = V1, y = V2, col = cellType)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2",
       color = " Cell types") +

  theme_classic()

# Visualize expression of a particular gene
# (for example - NEUROD6)in samples in the tSNE plot
```

```
Y        366 obs. of 2 variables
Values
cellType  chr [1:366] "RG" "Neuron"…
```

```
+                       max_iter = 5000)
> View(tsne_results)
> Y <- as.data.frame(tsne_results$Y)
> View(Y)
> View(meta)
> cellType <- meta$Inferred.Cell.Type
> 
```



```
ggplot(Y,
       aes(x = V1, y = V2, col = cellType)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2",
       color = " Cell types") +

  theme_classic()

# Visualize expression of a particular gene
# (for example - NEUROD6)in samples in the tSNE plot
```

```
Y        366 obs. of 2 variables
Values
cellType    chr [1:366] "RG" "Neuron"…
```

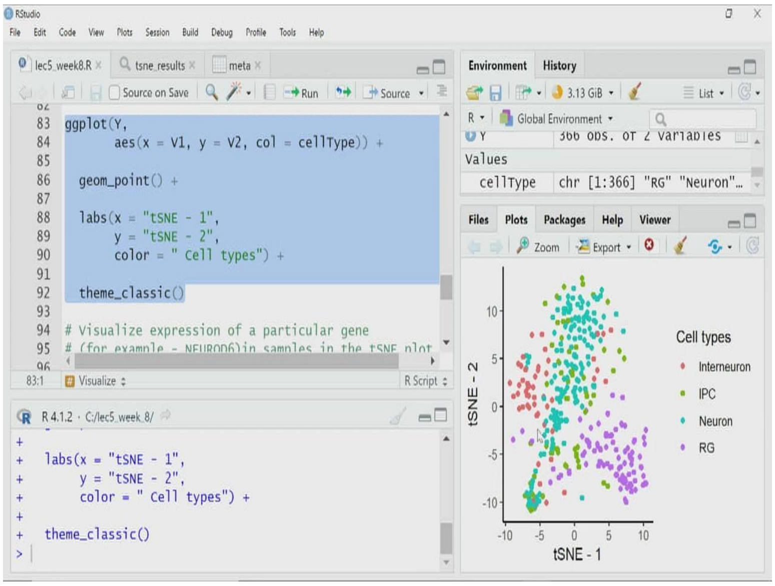
```
+
+     labs(x = "tSNE - 1",
+          y = "tSNE - 2",
+          color = " Cell types") +
+
+     theme_classic()
>
```

```
ggplot(Y,
          aes(x = v1, y = v2, col = cellType)) +
  geom_point() +
  labs(x = “tSNE - 1”,
         y = “tSNE - 2:,
          color = “ Cell types”) +
  theme_classic()
```

So, I have putting plus sign and I am calling geom underscored point function that means I want a scuttle there are some other things also for example, I have added labels I am writing x axis should be labeled as tSNE 1 and Y axis should be labeled as tSNE 2 and the legend heading where the color code will be explained is named as cell types. And eventually I am using my favorite theme theme classic.

So, let us plot it using gg plot 2. So, here is my plot. As you can see, nicely there is a will a nice type of pattern in the data. All these 366 data point is now mapped from the 16,000 dimension to two dimension of tSNE 1 and tSNE 2 and they are now color coded depending upon the cell type information that I have in a metadata file.

(Refer Slide Time: 19:10)



```
ggplot(Y,
       aes(x = V1, y = V2, col = cellType)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2",
       color = " Cell types") +

  theme_classic()

# Visualize expression of a particular gene
# (for example - NEUROD6)in samples in the tSNE plot
```

```
Y          366 obs. of 2 variables
Values
  cellType  chr [1:366] "RG" "Neuron"…
```

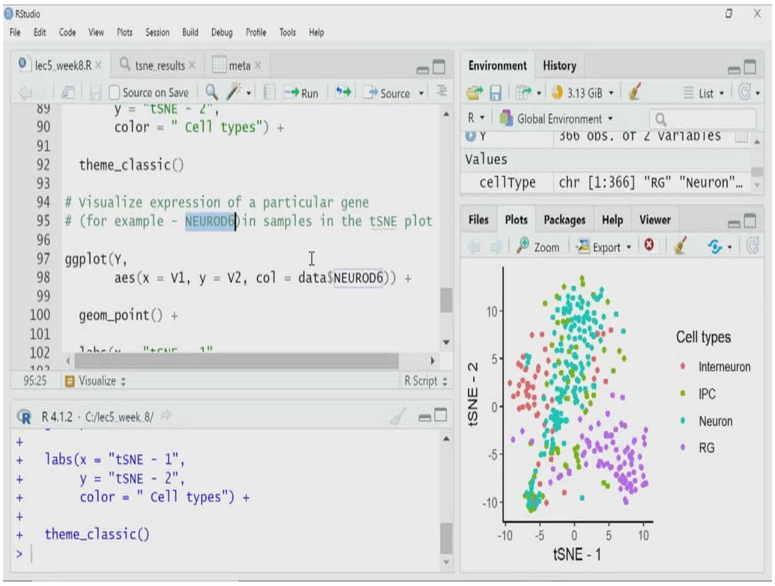
```
+
+   labs(x = "tSNE - 1",
+        y = "tSNE - 2",
+        color = " Cell types") +
+
+   theme_classic()
> 
```

```
ggplot(Y,
           aes(x = v1, y = v2, col = cellType)) +
  geom_point() +
  labs(x = “tSNE - 1”,
         y = “tSNE - 2:,
          color = “ Cell types”) +
  theme_classic()
```

For example, this RG the radial glial cells, they are forming some sort of separate clusters, other types of cells originate from them, whereas the neurons have a broad distribution and inter neuron and IPC they are claw, they have some amount of mixing with these neuron types of cell, but still the inter neurons are slightly separated.

So, this is how I could visualize the heterogeneity and the pattern as a whole as a whole in the 16,000 dimensional data for these four cell types. Now in this case, I have plotted these tSNE data E and color coded it depending upon the cell type. Suppose you want to use the same plot but you want to color code each Short these data points based upon the level of expression of a particular type of gene, that can be also easily done.

(Refer Slide Time: 20:20)



```
        y = "tSNE - 2",
        color = " Cell types") +

  theme_classic()

# Visualize expression of a particular gene
# (for example - NEUROD6)in samples in the tSNE plot

ggplot(Y,
        aes(x = V1, y = V2, col = data$NEUROD6)) +

  geom_point() +

  labs(x = "tSNE - 1"
```

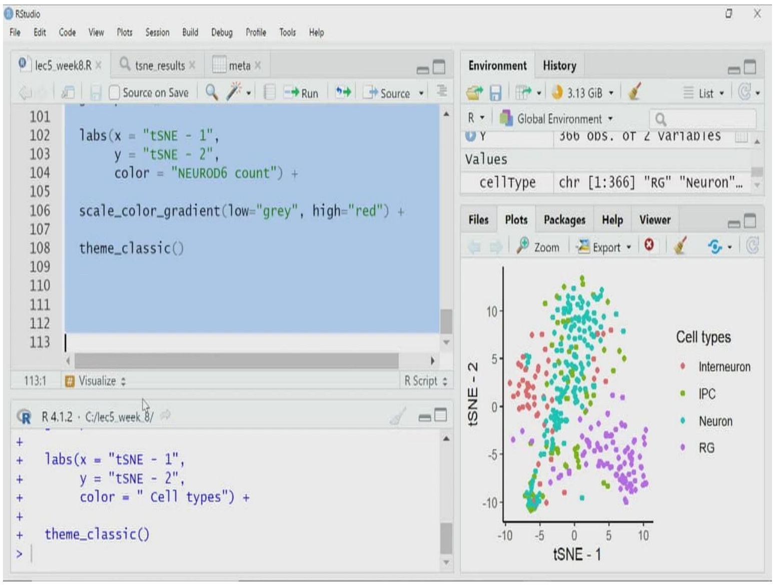
```
Y          366 obs. of 2 variables
Values
  cellType   chr [1:366] "RG" "Neuron"…
```

```
+
+   labs(x = "tSNE - 1",
+        y = "tSNE - 2",
+        color = " Cell types") +
+
+   theme_classic()
>
```

```
ggplot(Y,

        aes(x = v1, y = v2, col = data$NEUROD6)) +

  geom_point() +

  labs(x = “tSNE - 1”,

         y = “tSNE - 2:,

          color = “ NEUROD6”) +

  scaler_color_gradient(low = “grey”, high = “red”) +

  theme_classic()
```

So that is what I do in the next diagram, where the plot overall picture will remain same, but that the color of the dots will get changed depending upon the level of count expression or count for a particular gene. So, what I am doing here, for example, I am taking a new gene called NEURO D6 is the transcription factor, and I, the expression account for this gene is there in the original data file. So, what I want to do now, I want to color code this is the scatterplot point, depending upon the level of expression of this gene in those samples.

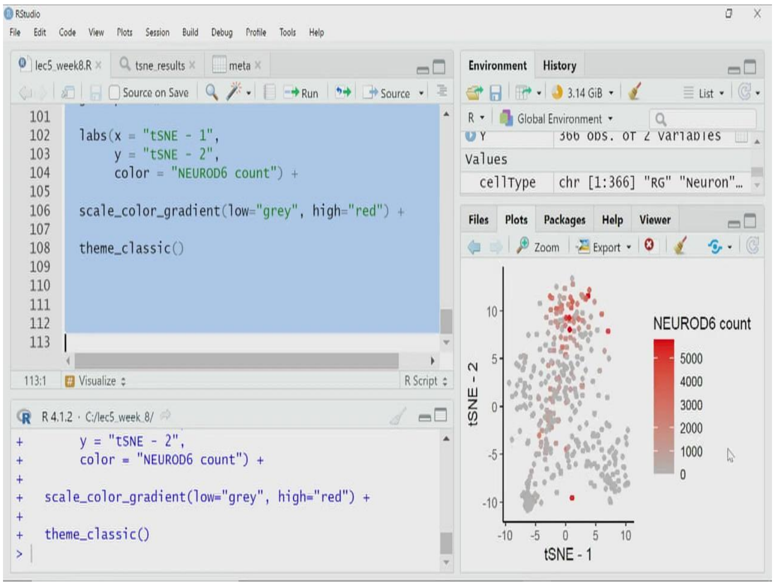
(Refer Slide Time: 20:44)



```
labs(x = "tSNE - 1",
     y = "tSNE - 2",
     color = "NEUROD6 count") +

scale_color_gradient(low="grey", high="red") +

theme_classic()
```

```
+
+   labs(x = "tSNE - 1",
+        y = "tSNE - 2",
+        color = " Cell types") +
+
+   theme_classic()
> 
```

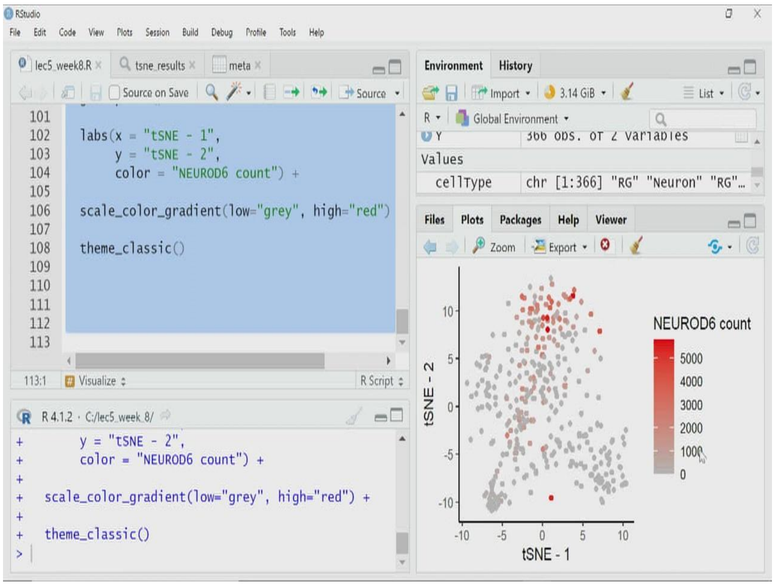
```
Y         366 obs. of 2 variables
Values
  cellType   chr [1:366] "RG" "Neuron"…
```



```
labs(x = "tSNE - 1",
     y = "tSNE - 2",
     color = "NEUROD6 count") +

scale_color_gradient(low="grey", high="red") +

theme_classic()
```

```
+        y = "tSNE - 2",
+        color = "NEUROD6 count") +
+
+   scale_color_gradient(low="grey", high="red") +
+
+   theme_classic()
> 
```

```
Y         366 obs. of 2 variables
Values
  cellType   chr [1:366] "RG" "Neuron"…
```



```
labs(x = "tSNE - 1",
     y = "tSNE - 2",
     color = "NEUROD6 count") +

scale_color_gradient(low="grey", high="red")

theme_classic()
```

```
+        y = "tSNE - 2",
+        color = "NEUROD6 count") +
+
+   scale_color_gradient(low="grey", high="red") +
+
+   theme_classic()
> 
```

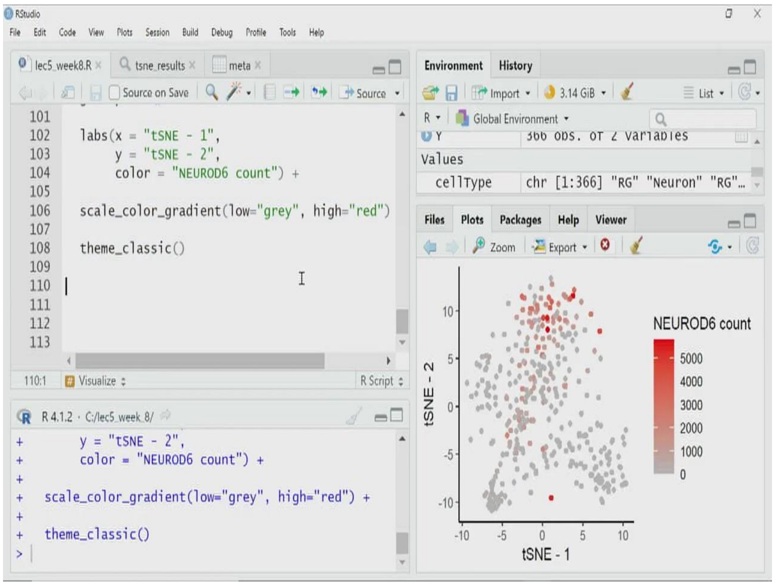
```
Y         366 obs. of 2 variables
Values
  cellType   chr [1:366] "RG" "Neuron" "RG"…
```

```
ggplot(Y,
        aes(x = v1, y = v2, col = data$NEUROD6)) +
  geom_point() +
  labs(x = “tSNE - 1”,
       y = “tSNE - 2:,
        color = “ NEUROD6”) +
  scaler_color_gradient(low = “grey”, high = “red”) +
  theme_classic()
```

Let me do that let me run this code, code snippet and then I will explain and what are the options I have using gg plot 2. So, here I have the plot the same diagram and I have used a scale the gray is zero and the red is the highest value 5000 The Count and from the legend you can see this is the count for NEURO D6 and there are some silly here, which are very high count where are the most of the other cells, they have very low count.

So, in this way, you can decide which particular genes expression you want to show you can actually easily overlay multiple color and color code multiple genes also using gg plot 2 using the same type of code.

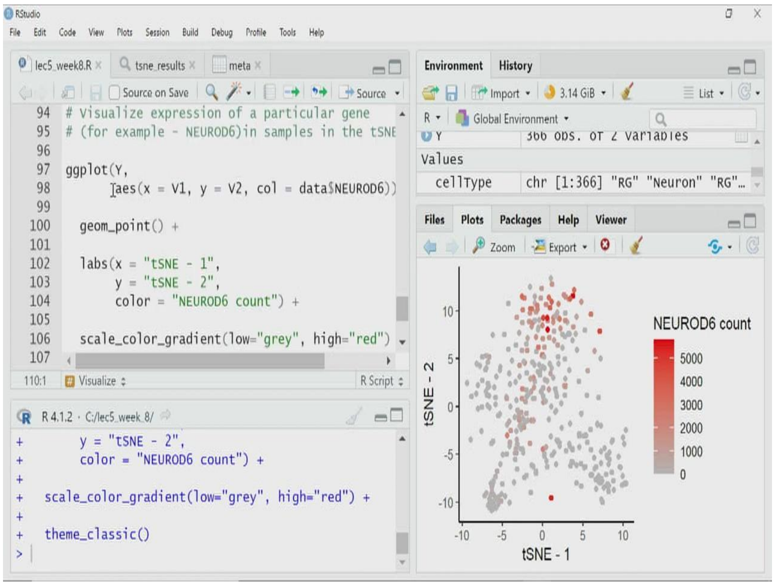
(Refer Slide Time: 21:32)



```
    labs(x = "tSNE - 1",
         y = "tSNE - 2",
         color = "NEUROD6 count") +

    scale_color_gradient(low="grey", high="red")

    theme_classic()

```

```
+       y = "tSNE - 2",
+       color = "NEUROD6 count") +
+
+   scale_color_gradient(low="grey", high="red") +
+
+   theme_classic()
>
```

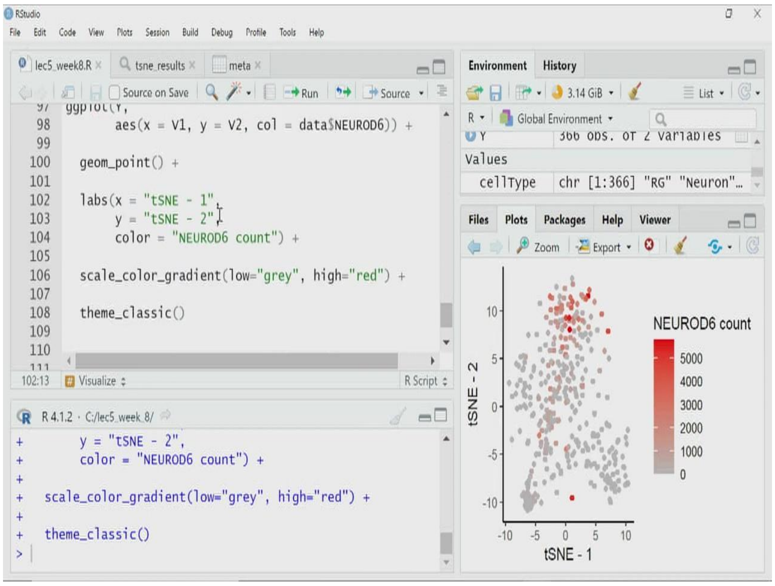
```
Y          366 obs. of 2 variables
Values
  cellType  chr [1:366] "RG" "Neuron" "RG"…
```



```
# Visualize expression of a particular gene
# (for example - NEUROD6)in samples in the tSNE

ggplot(Y,
    aes(x = V1, y = V2, col = data$NEUROD6))

    geom_point() +

    labs(x = "tSNE - 1",
         y = "tSNE - 2",
         color = "NEUROD6 count") +

    scale_color_gradient(low="grey", high="red")

```

```
+       y = "tSNE - 2",
+       color = "NEUROD6 count") +
+
+   scale_color_gradient(low="grey", high="red") +
+
+   theme_classic()
>
```

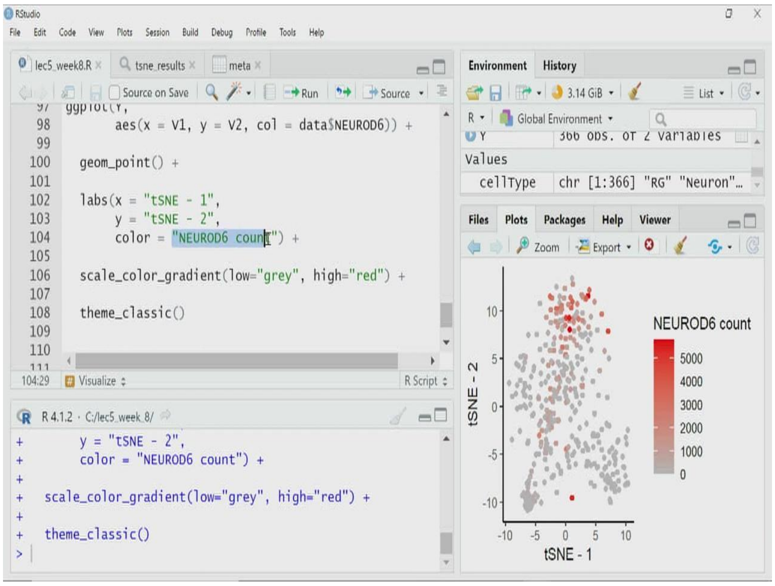
```
Y          366 obs. of 2 variables
Values
  cellType  chr [1:366] "RG" "Neuron" "RG"…
```



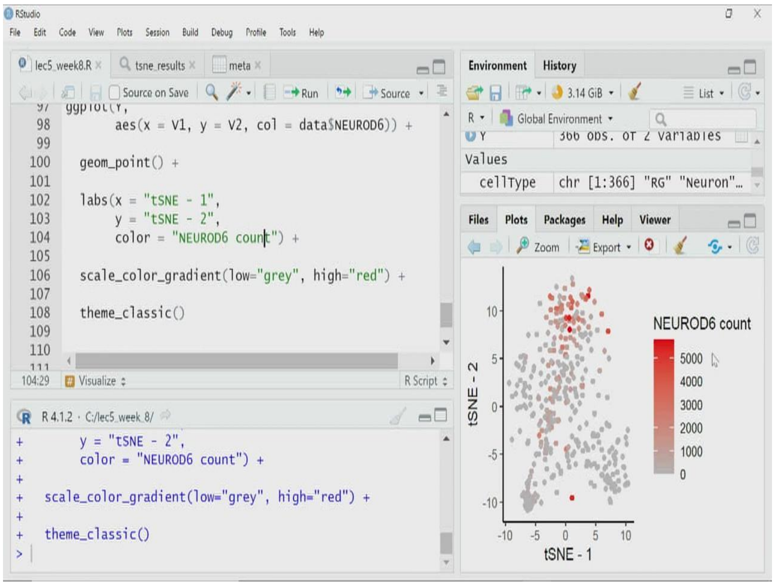
```
ggplot(Y,
       aes(x = V1, y = V2, col = data$NEUROD6)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2"
       color = "NEUROD6 count") +

  scale_color_gradient(low="grey", high="red") +

  theme_classic()
```

```
+        y = "tSNE - 2",
+        color = "NEUROD6 count") +
+
+   scale_color_gradient(low="grey", high="red") +
+
+   theme_classic()
>
```

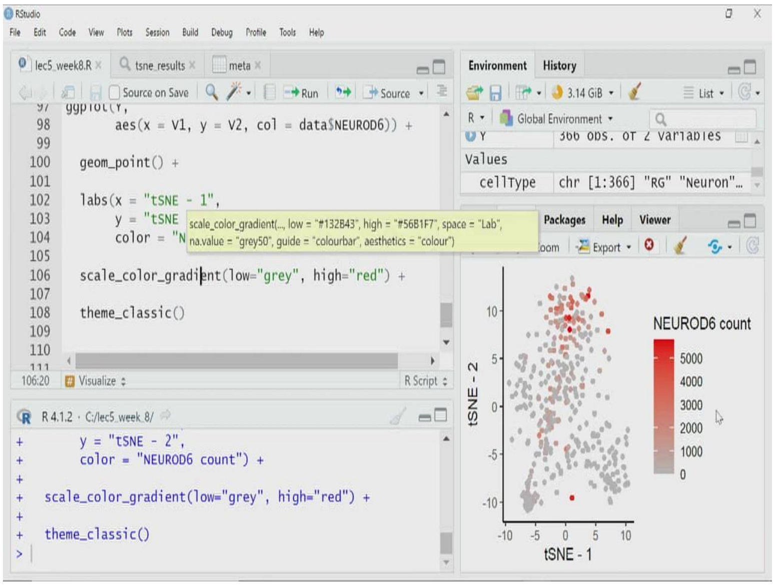
```
Y          366 obs. of 2 variables
Values
  cellType chr [1:366] "RG" "Neuron"…
```



```
ggplot(Y,
       aes(x = V1, y = V2, col = data$NEUROD6)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2",
       color = "NEUROD6 count") +

  scale_color_gradient(low="grey", high="red") +

  theme_classic()
```



```
ggplot(Y,
       aes(x = V1, y = V2, col = data$NEUROD6)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2",
       color = "NEUROD6 count") +

  scale_color_gradient(low="grey", high="red") +

  theme_classic()
```



```
ggplot(Y,
       aes(x = V1, y = V2, col = data$NEUROD6)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE
       color = "N

  scale_color_gradient(low="grey", high="red") +

  theme_classic()
```

scale_color_gradient(..., low = "#132B43", high = "#56B1F7", space = "Lab", na.value = "grey50", guide = "colourbar", aesthetics = "colour")

```
Y         366 obs. of 2 variables
Values
cellType  chr [1:366] "RG" "Neuron"…
```

```
+       y = "tSNE - 2",
+       color = "NEUROD6 count") +
+
+   scale_color_gradient(low="grey", high="red") +
+
+   theme_classic()
>
```

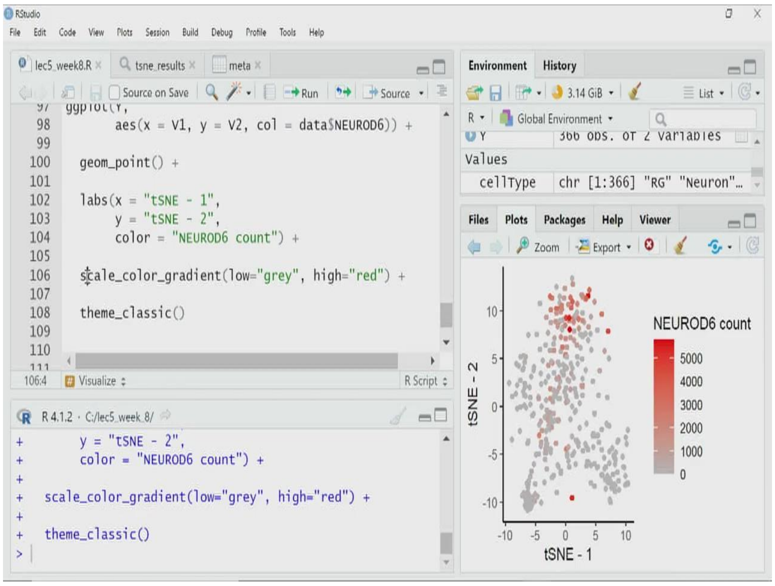
```
ggplot(Y,

          aes(x = v1, y = v2, col = data$NEUROD6)) +

  geom_point() +

  labs(x = “tSNE - 1”,

          y = “tSNE - 2:,

          color = “ NEUROD6”) +

  scaler_color_gradient(low = “grey”, high = “red”) +

  theme_classic()
```

So, let me explain what I have done in the gg plot 2 code here. Again, I am calling the gg plot function, giving the tSNE result Y as the data if the data frame in aesthetic when I am calling aesthetic, I am saying X is V1 obviously, it will remain same Y is V2 is remain same exactly the same one. Now, I have changed the color code.

So I am asking color coded or categorize the data based upon data Dollar NEURO D6. So that means I am saying take the data for NEURO D6 column in that data variable and use that information to color code the scatterplot. And then I am calling these geometry geom underscore point as usual, because I want the scatterplot labels remains mostly same, except the legend of the region title is getting changed here I am writing color equal to NEURO D6 count.

So that is why the legend here title has changed to NEURO D6 count. And interesting thing that I have used a gray in a gradient of color from gray to red to which represent the scale of expression.

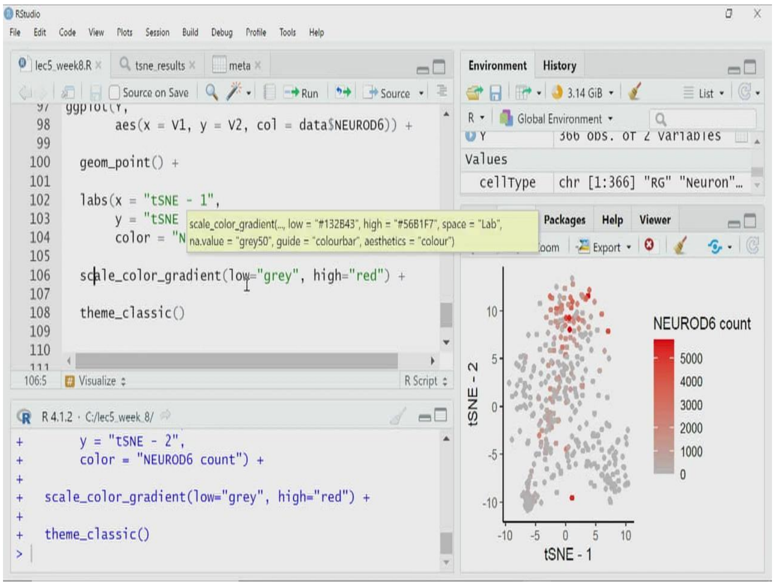
(Refer Slide Time: 22:44)



```
ggplot(Y,
       aes(x = V1, y = V2, col = data$NEUROD6)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2",
       color = "NEUROD6 count") +

  scale_color_gradient(low="grey", high="red") +

  theme_classic()
```

```
Y          366 obs. of 2 variables
Values
  cellType   chr [1:366] "RG" "Neuron"…
```

```
+          y = "tSNE - 2",
+          color = "NEUROD6 count") +
+
+     scale_color_gradient(low="grey", high="red") +
+
+     theme_classic()
>
```



```
ggplot(Y,
       aes(x = V1, y = V2, col = data$NEUROD6)) +

  geom_point() +

  labs(x = "tSNE - 1",
       y = "tSNE - 2",
       color = "NEUROD6 count") +

  scale_color_gradient(low="grey", high="red") +

  theme_classic()
```

```
scale_color_gradient(..., low = "#132B43", high = "#56B1F7", space = "Lab",
na.value = "grey50", guide = "colourbar", aesthetics = "colour")
```

```
Y          366 obs. of 2 variables
Values
  cellType   chr [1:366] "RG" "Neuron"…
```

```
+          y = "tSNE - 2",
+          color = "NEUROD6 count") +
+
+     scale_color_gradient(low="grey", high="red") +
+
+     theme_classic()
>
```

```
ggplot(Y,
          aes(x = v1, y = v2, col = data$NEUROD6)) +
  geom_point() +
  labs(x = “tSNE - 1”,
         y = “tSNE - 2:,
          color = “ NEUROD6”) +
 scaler_color_gradient(low = “grey”, high = “red”) +
 theme_classic()
```

So what I have done here I have added a function called scale underscore color underscore gradient. And I have said the lowest value should be gray, and the highest value should be red. So it has considered zero count as gray and the highest reading in the data set as red and rest up nothing in between R scale within this color range. And then I have used the theme Classic. So that is how I have created this diagram.

That is how it is so simple to perform tSNE if you have nicely clear your data origin initially and then you will perform the analysis step by step. The crucial thing before I end I will mention once again, remember, the purpose of using tSNE is to visualize the data. And there are lots of arguments starting from perplexity to ETA to maximum iteration, all these things can affect the final visualization. There is no gold standard to decide upon these type of values, values of these arguments.

You may have to play around use your common sense you have to understand what how these arguments are affecting your visualization. And then you have to choose that proper value for each of these arguments and use them. The best point of start is always to keep these things as the default one and then play around some critical argument and changing their values systematically to see where you are getting the better visualization. That is all for this video. Thank you for learning with me today.