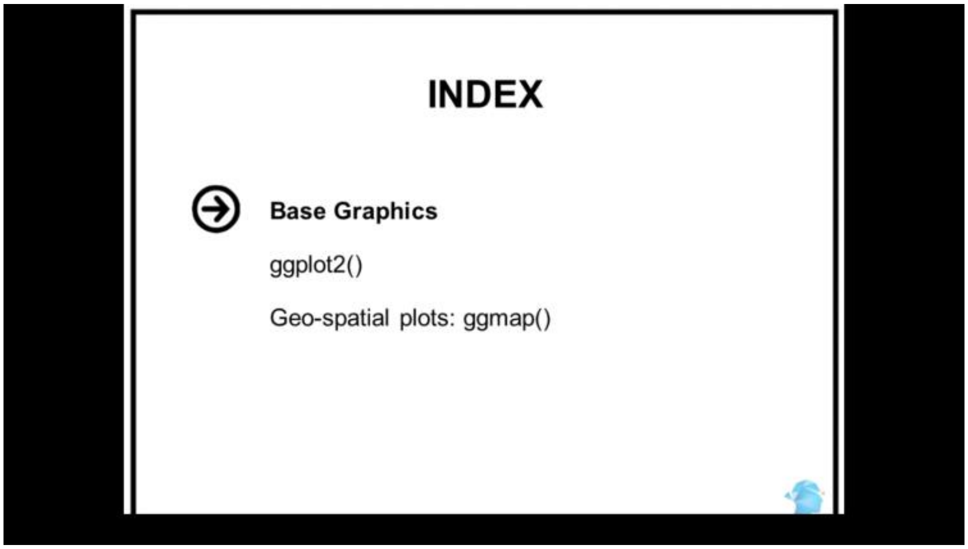
Data Visualization with R

Friday, September 09, 2016 3:21 PM

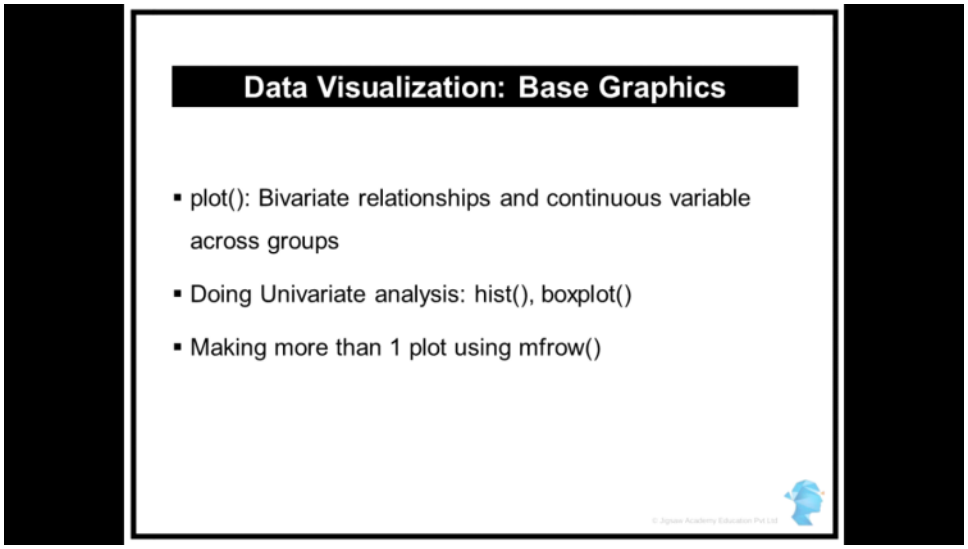


INDEX

Base Graphics

ggplot2()

Geo-spatial plots: ggmap()

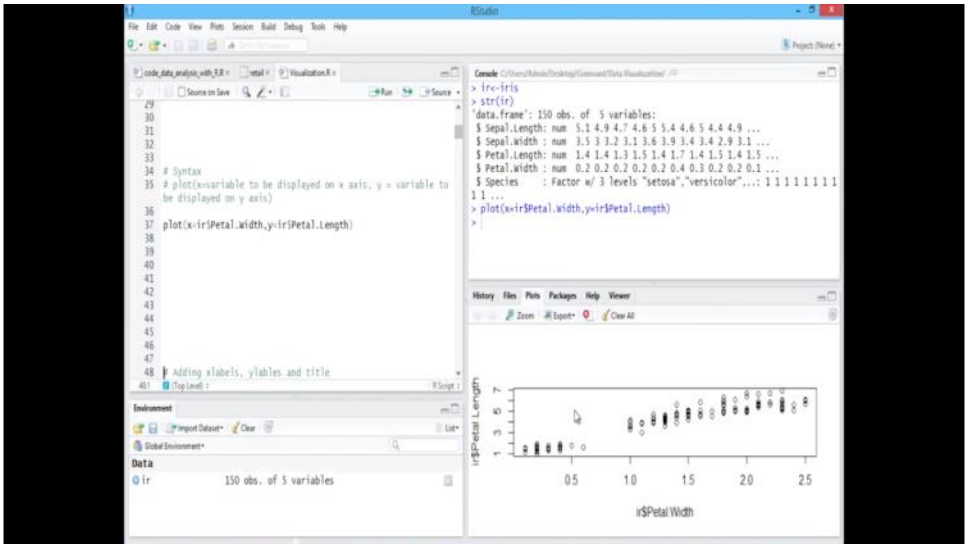


Data Visualization: Base Graphics

- plot(): Bivariate relationships and continuous variable across groups
- Doing Univariate analysis: hist(), boxplot()
- Making more than 1 plot using mfrow()

Lets look at the basic plot function under ggplot2

Plot(x=<variable that we want in the X axis>, y=<variable that we want in the Y axis>)



```
# Syntax
# plot(x=variable to be displayed on x axis, y = variable to be displayed on y axis)

plot(x=ir$Petal.Width,y=ir$Petal.Length)

# Adding xlabels, ylables and title
```

```
> ir<-iris
> str(ir)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 ...
> plot(x=ir$Petal.Width,y=ir$Petal.Length)
> 
```

```
Data
ir            150 obs. of 5 variables
```

This is a very basic plot. Now if we want to add title to the plot along with Labels for X and Y axis

Plot(x=iris$petal_length, y=iris$petal_width, main=c("Petal width Vs Length"), xlab=c("Length"),ylab=c("Width"))

To add color to this plot use this:
Plot(x=iris$petal_length, y=iris$petal_width, main=c("Petal width Vs Length"), xlab=c("Length"),ylab=c("Width"), col="red")

If we want to change the default markers from circles to another form, we need to add the PCH attribute to the plot function:

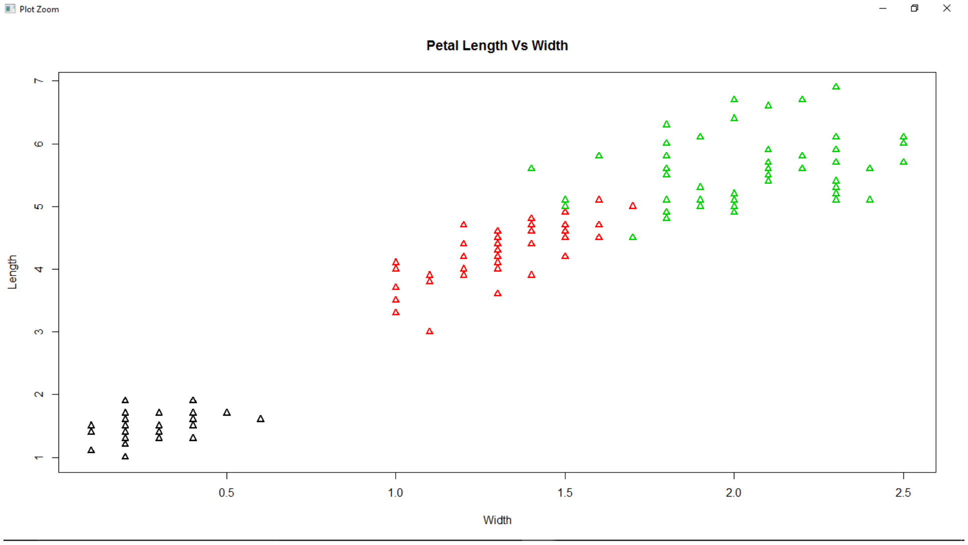
Plot(x=iris$petal_length, y=iris$petal_width, main=c("Petal width Vs Length"), xlab=c("Length"),ylab=c("Width"), col="red", pch=2)

Another optional attribute is the line width parameter:
Plot(x=iris$petal_length, y=iris$petal_width, main=c("Petal width Vs Length"), xlab=c("Length"),ylab=c("Width"), col="red", pch=2, lwd=2)

Conditional Bivariant Plots:

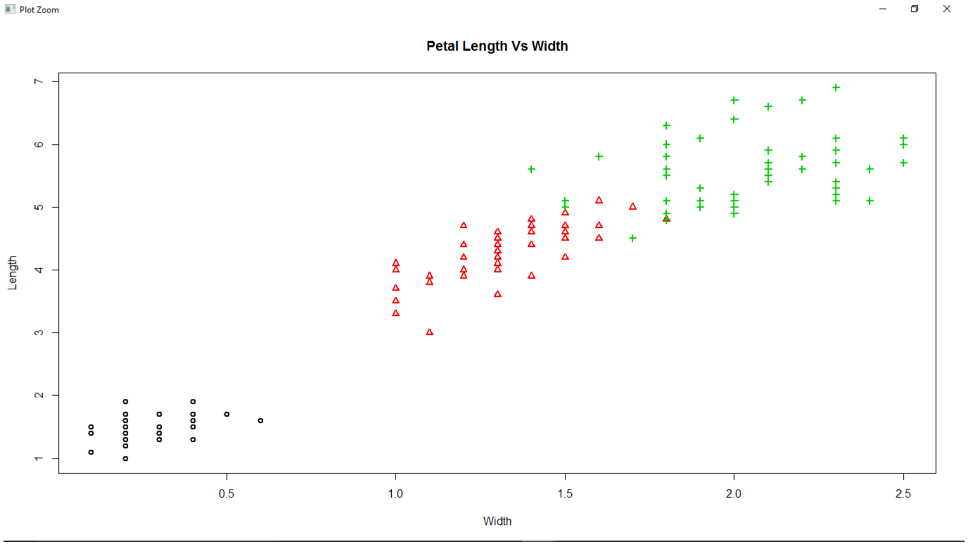
We can add conditions to the "col" attribute of the plot function.
This helps in grouping the data in the plot using color coding features that are generated by R.

Plot(x=iris$petal_length, y=iris$petal_width, main=c("Petal width Vs Length"), xlab=c("Length"),ylab=c("Width"), col=iris$species, pch=2)

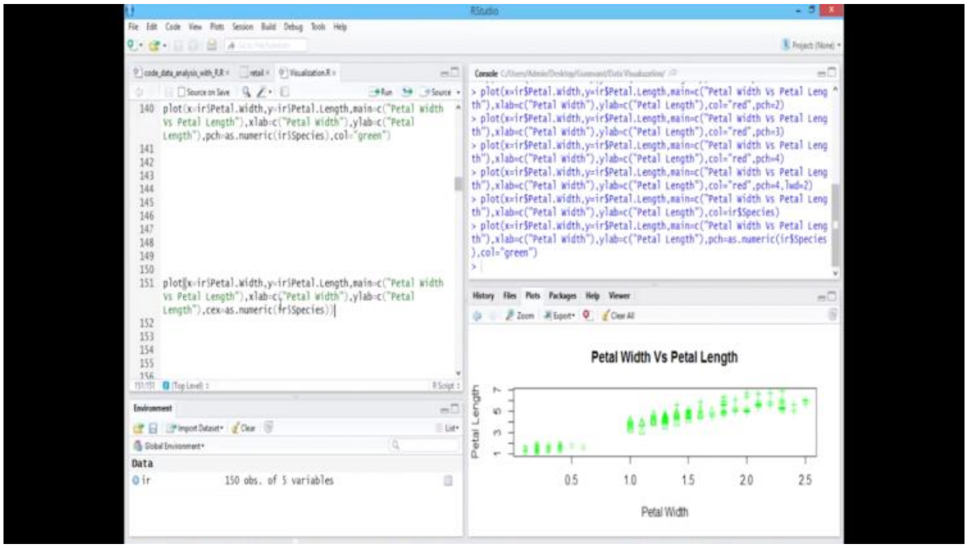


To do conditional plots based on different chracters , rather than color, we can use the PCH attribute to load the condition: Make sure to use the as.numeric attribute

```
Plot(x=iris$petal_length, y=iris$petal_width, main=c("Petal width Vs Length"),
xlab=c("Length"),ylab=c("Width"), pch=as.numeric(iris$species), col="red")
```



Another way to differentiate the species is to use the CEX command. This will differentiate based on the size :



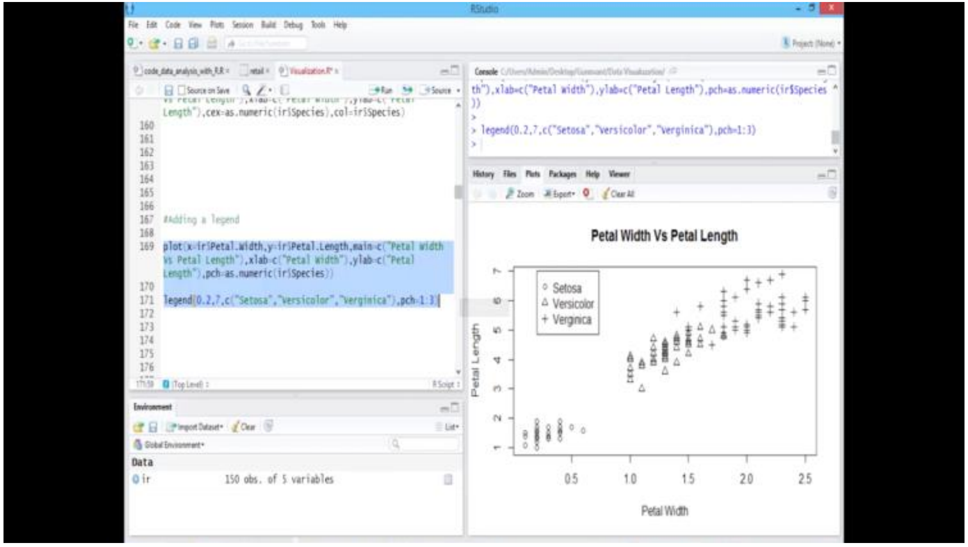
```
plot(x=ir$Petal.Width,y=ir$Petal.Length,main=c("Petal Width
Vs Petal Length"),xlab=c("Petal Width"),ylab=c("Petal
Length"),pch=as.numeric(ir$Species),col="green")

plot(x=ir$Petal.Width,y=ir$Petal.Length,main=c("Petal Width
Vs Petal Length"),xlab=c("Petal Width"),ylab=c("Petal
Length"),cex=as.numeric(ir$Species))
```

```
> plot(x=ir$Petal.Width,y=ir$Petal.Length,main=c("Petal Width Vs Petal Leng
th"),xlab=c("Petal Width"),ylab=c("Petal Length"),col="red",pch=2)
> plot(x=ir$Petal.Width,y=ir$Petal.Length,main=c("Petal Width Vs Petal Leng
th"),xlab=c("Petal Width"),ylab=c("Petal Length"),col="red",pch=3)
> plot(x=ir$Petal.Width,y=ir$Petal.Length,main=c("Petal Width Vs Petal Leng
th"),xlab=c("Petal Width"),ylab=c("Petal Length"),col="red",pch=4)
> plot(x=ir$Petal.Width,y=ir$Petal.Length,main=c("Petal Width Vs Petal Leng
th"),xlab=c("Petal Width"),ylab=c("Petal Length"),col="red",pch=4,lwd=2)
> plot(x=ir$Petal.Width,y=ir$Petal.Length,main=c("Petal Width Vs Petal Leng
th"),xlab=c("Petal Width"),ylab=c("Petal Length"),col=ir$Species)
> plot(x=ir$Petal.Width,y=ir$Petal.Length,main=c("Petal Width Vs Petal Leng
th"),xlab=c("Petal Width"),ylab=c("Petal Length"),pch=as.numeric(ir$Species
),col="green")
>
```

Data

ir 150 obs. of 5 variables

To add a legend to the plot this IS WHAT WE USE:



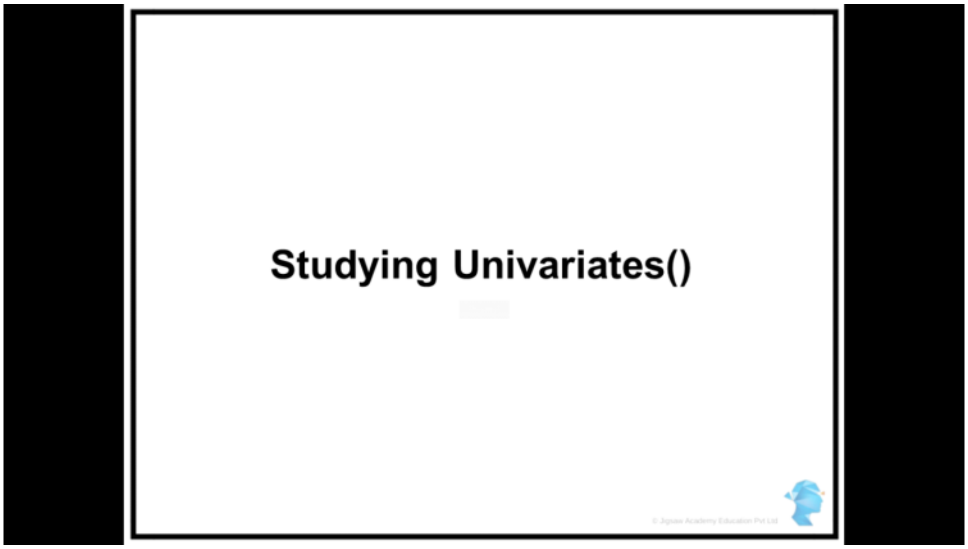
```
Length"),cex=as.numeric(ir$Species),col=ir$Species)

#Adding a legend

plot(x=ir$Petal.Width,y=ir$Petal.Length,main=c("Petal Width
Vs Petal Length"),xlab=c("Petal Width"),ylab=c("Petal
Length"),pch=as.numeric(ir$Species))

legend(0.2,7,c("Setosa","Versicolor","Verginica"),pch=1:3)
```

```
th"),xlab=c("Petal Width"),ylab=c("Petal Length"),pch=as.numeric(ir$Species
))
>
> legend(0.2,7,c("Setosa","Versicolor","Verginica"),pch=1:3)
>
```

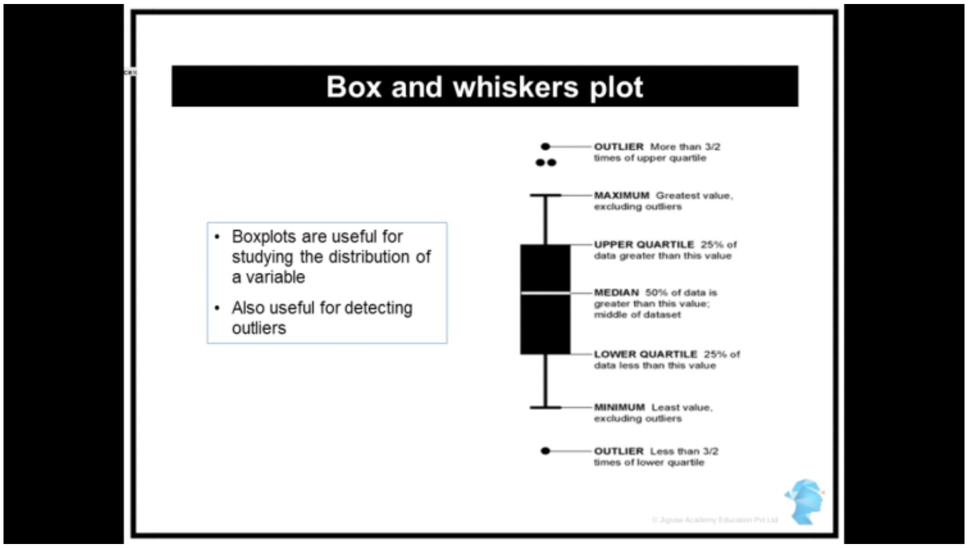
Data

ir 150 obs. of 5 variables

RStudio
File Edit Code View Plots Session Build Debug Tools Help
code_data_analysis_with_R.R
retail
Visualization.R*
Source on Save
Run
Source
Console
History Files Plots Packages Help Viewer
Zoom
Export
Clear All
Environment
Import Dataset
Clear
Global Environment
Petal Width Vs Petal Length
○ Setosa
△ Versicolor
+ Verginica
Petal Length
1
2
3
4
5
6
7
0.5
1.0
1.5
2.0
2.5
Petal Width



Studying Univariates()



Box and whiskers plot

- Boxplots are useful for studying the distribution of a variable
- Also useful for detecting outliers

OUTLIER More than 3/2 times of upper quartile

MAXIMUM Greatest value, excluding outliers

UPPER QUARTILE 25% of data greater than this value

MEDIAN 50% of data is greater than this value; middle of dataset

LOWER QUARTILE 25% of data less than this value

MINIMUM Least value, excluding outliers

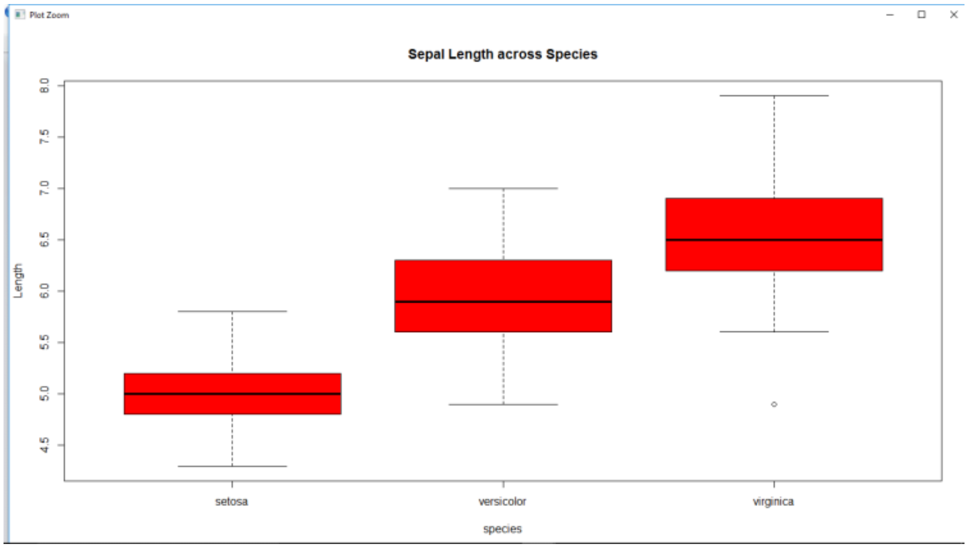
OUTLIER Less than 3/2 times of lower quartile

© Jigsaw Academy Education Pvt Ltd

Inorder to run a boxplot we will use the boxplot function:
Boxplot(variable)

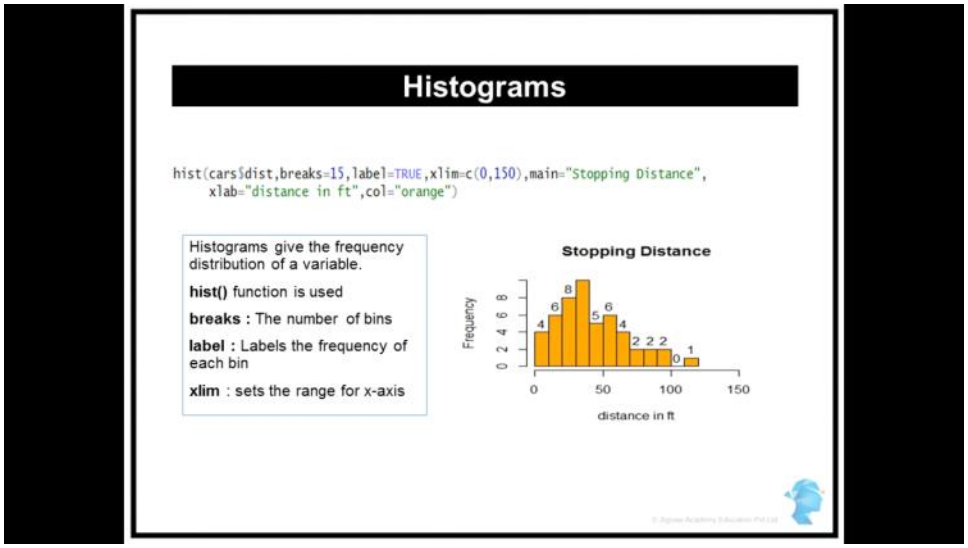
Here is how we can use the PLOT function to plot boxplots:
plot(x=iris_test$Species, y=iris_test$Sepal.Length, main = c("Sepal Length across Species"), xlab = "species", ylab = "Length", col="red")

On the X attribute add the factor based on which we want to perform the box plot and Y axis has the attribute for which
We want to perform the analysis.



Histograms:

To get a histogram plot we can use the HIST function.



Histograms

```
hist(cars$dist,breaks=15,label=TRUE,xlim=c(0,150),main="Stopping Distance",
     xlab="distance in ft",col="orange")
```

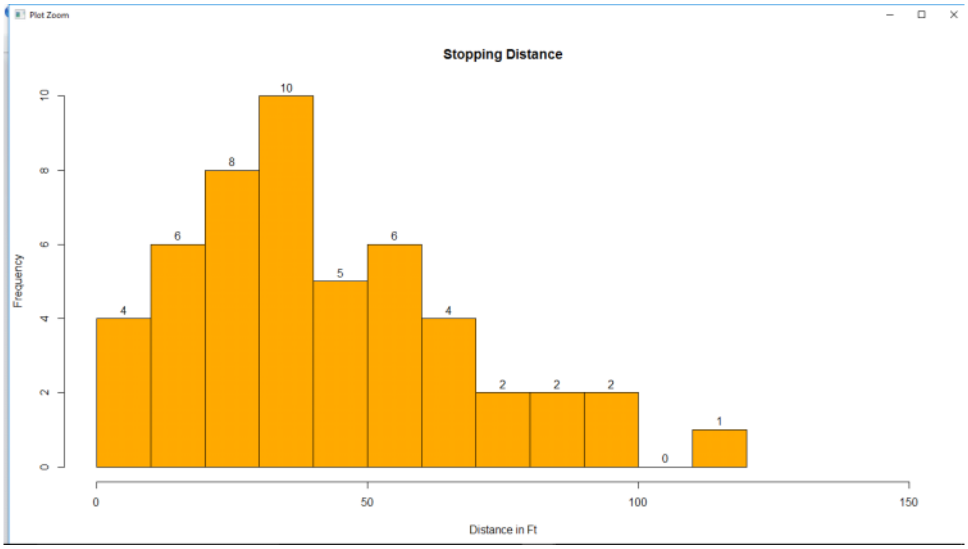
Histograms give the frequency distribution of a variable.

hist() function is used

breaks : The number of bins

label : Labels the frequency of each bin

xlim : sets the range for x-axis



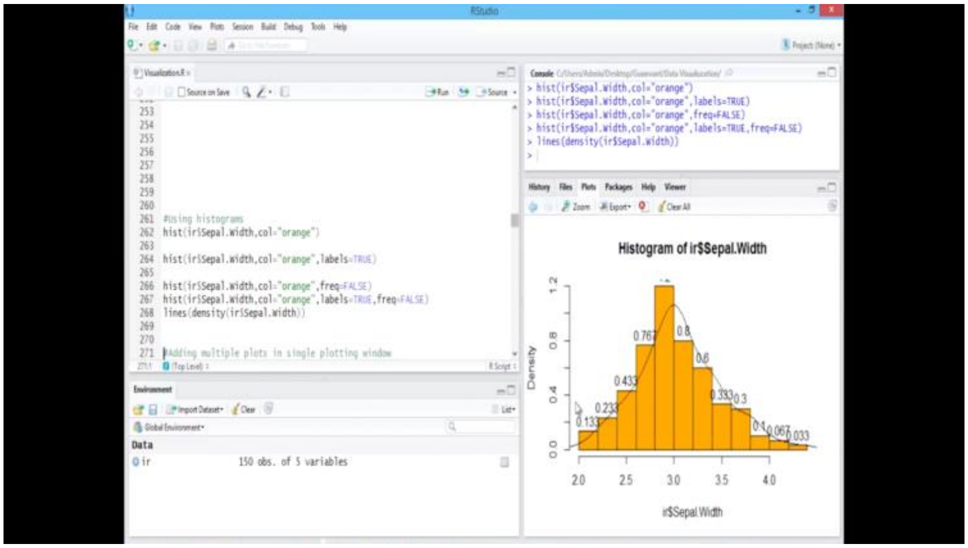
hist(cars$dist, labels = TRUE, breaks = 15, xlim = c(0,150), main=c("Stopping Distance"), xlab="Distance in Ft", col="orange")

Here lables=TRUE will show the counts of frequency in the data points.

If we supply the value: freq=FALSE.
This will plot a density function.

If we want to plot a line on this histogram, use the lines command

Lines(density(iris$sepal.width))

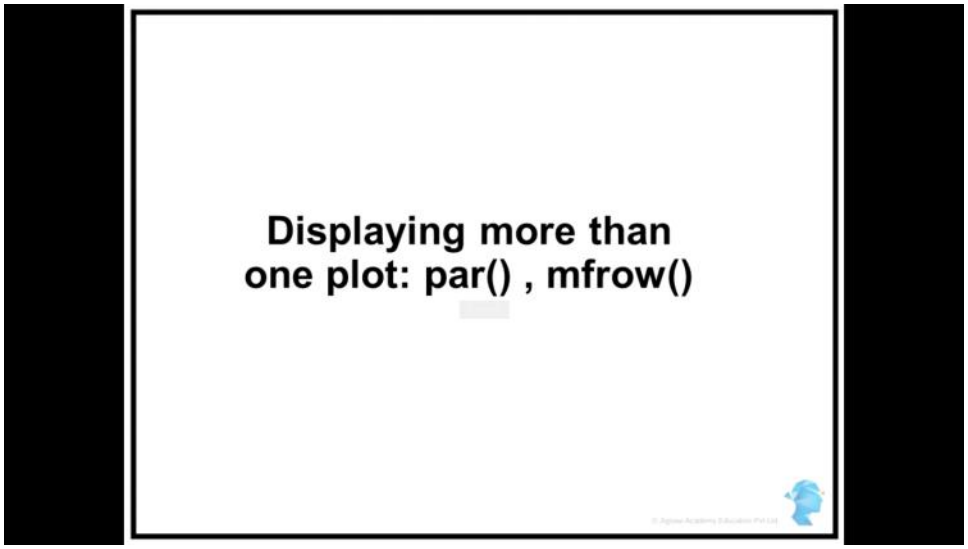


```
#Using histograms
hist(ir$Sepal.Width,col="orange")

hist(ir$Sepal.Width,col="orange",labels=TRUE)

hist(ir$Sepal.Width,col="orange",freq=FALSE)
hist(ir$Sepal.Width,col="orange",labels=TRUE,freq=FALSE)
lines(density(ir$Sepal.Width))


#Adding multiple plots in single plotting window
```

```
> hist(ir$Sepal.Width,col="orange")
> hist(ir$Sepal.Width,col="orange",labels=TRUE)
> hist(ir$Sepal.Width,col="orange",freq=FALSE)
> hist(ir$Sepal.Width,col="orange",labels=TRUE,freq=FALSE)
> lines(density(ir$Sepal.Width))
>
```

Data
ir 150 obs. of 5 variables

RStudio
File Edit Code View Plots Session Build Debug Tools Help
Project (None)
Visualization.R
Source on Save
Run
Source
Console
Environment
Import Dataset
Clear
List
Global Environment
History Files Plots Packages Help Viewer
Zoom
Export
Clear All
Histogram of ir$Sepal.Width
Density
0.0
0.4
0.8
1.2
0.133
0.233
0.433
0.767
0.8
0.6
0.333
0.3
0.1
0.067
0.033
2.0
2.5
3.0
3.5
4.0
ir$Sepal.Width

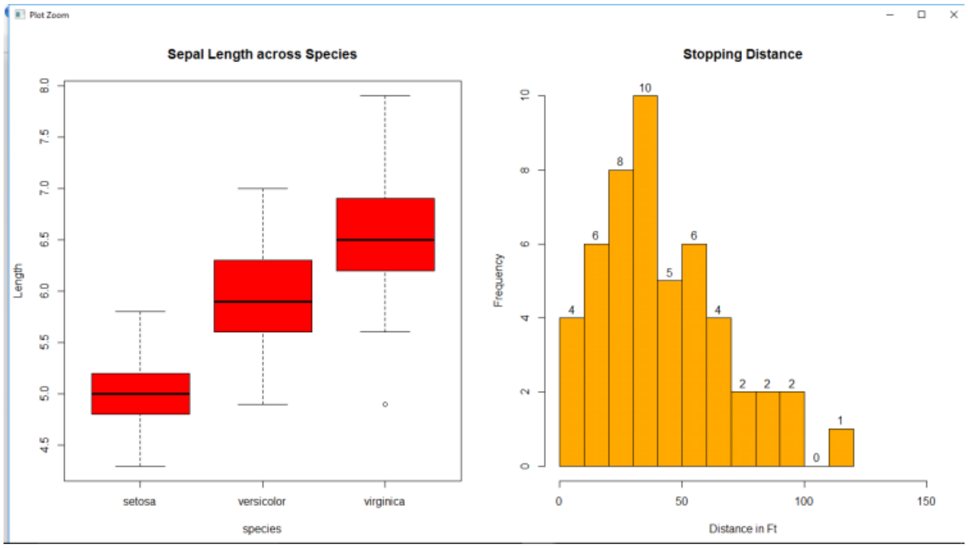


Displaying more than one plot: par() , mfrow()

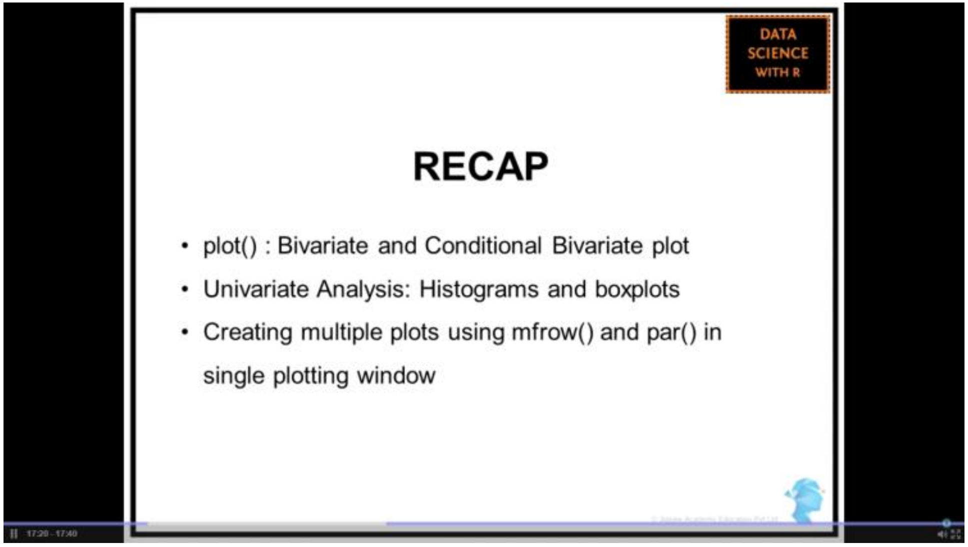
In order to display multiple plots on the same area we will use the above function.
PAR --> plotting area function
Mfrow--> number of rows and columns that we want the plotting area to be broken into.

Par(mfrow=c(1,2))

Then run the two distinct plot ffunctions to get the following results:

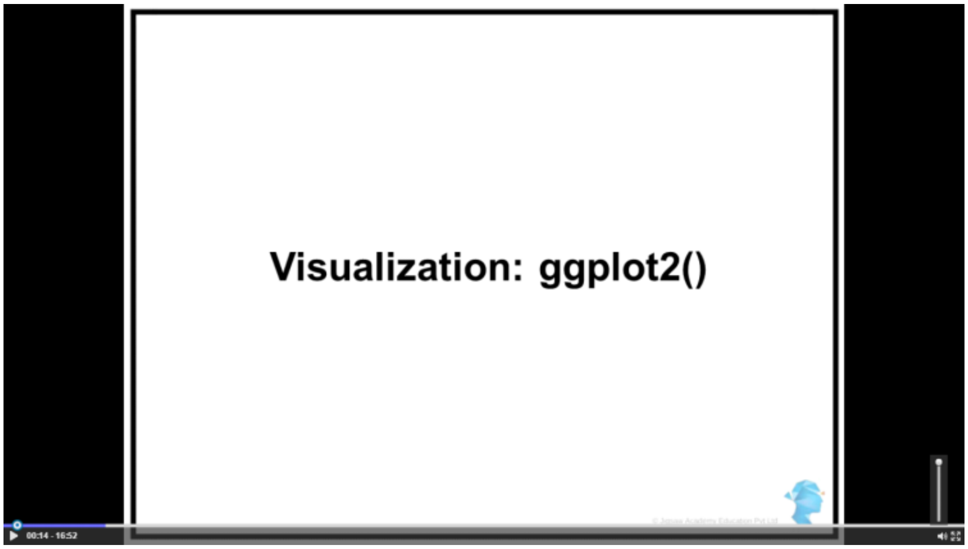


If we want to return to the default plotting area of just having a single plot, use: Dev.off()

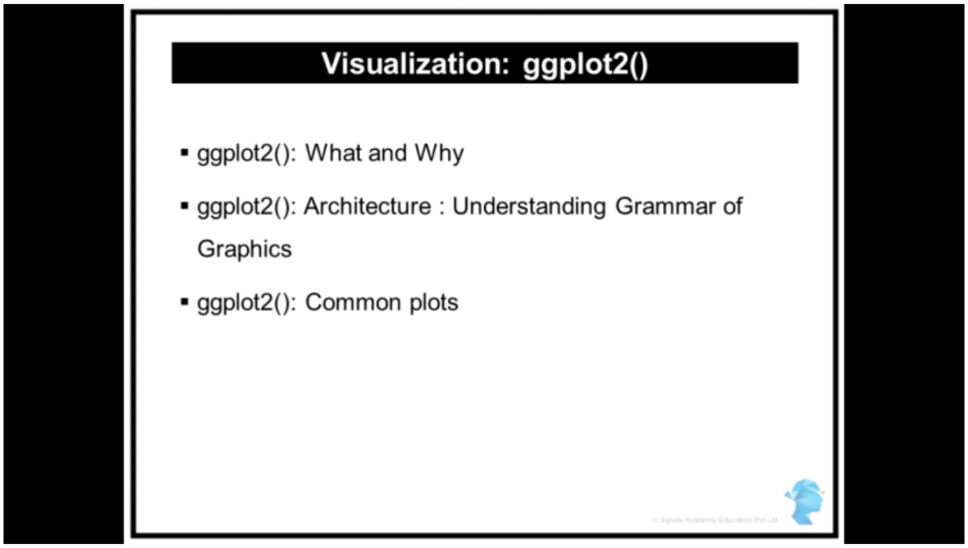


RECAP

- plot() : Bivariate and Conditional Bivariate plot
- Univariate Analysis: Histograms and boxplots
- Creating multiple plots using mfrow() and par() in single plotting window

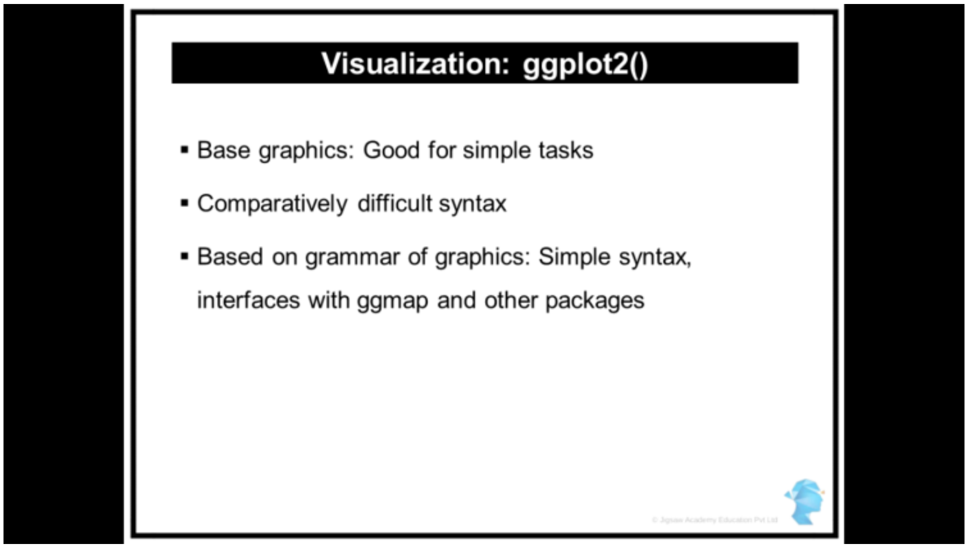


Visualization: ggplot2()



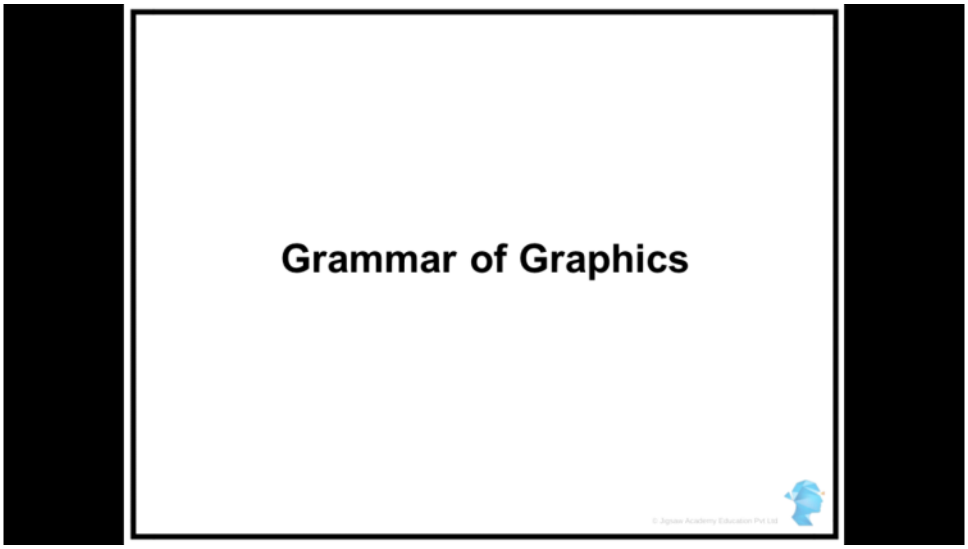
Visualization: ggplot2()

- ggplot2(): What and Why
- ggplot2(): Architecture : Understanding Grammar of Graphics
- ggplot2(): Common plots



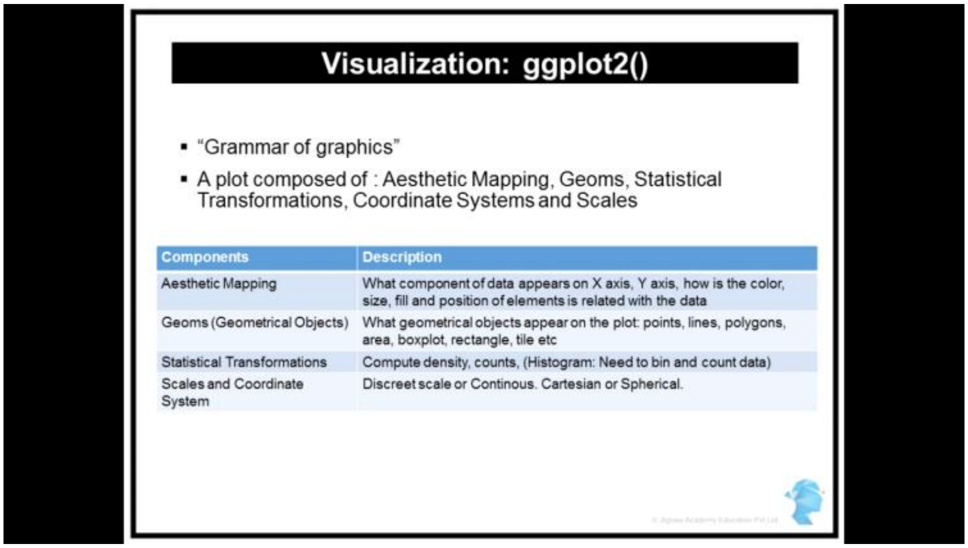
Visualization: ggplot2()

- Base graphics: Good for simple tasks
- Comparatively difficult syntax
- Based on grammar of graphics: Simple syntax, interfaces with ggmap and other packages



Grammar of Graphics

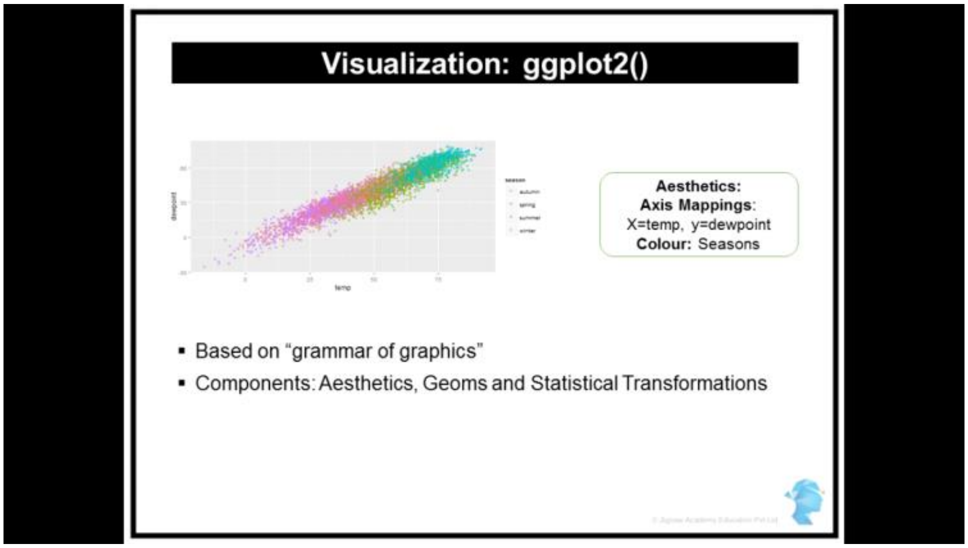
This is the architecture on which GGPLOT2 is built.



Visualization: ggplot2()

- "Grammar of graphics"
- A plot composed of : Aesthetic Mapping, Geoms, Statistical Transformations, Coordinate Systems and Scales

Components	Description
Aesthetic Mapping	What component of data appears on X axis, Y axis, how is the color, size, fill and position of elements is related with the data
Geoms (Geometrical Objects)	What geometrical objects appear on the plot: points, lines, polygons, area, boxplot, rectangle, tile etc
Statistical Transformations	Compute density, counts, (Histogram: Need to bin and count data)
Scales and Coordinate System	Discreet scale or Continous. Cartesian or Spherical.



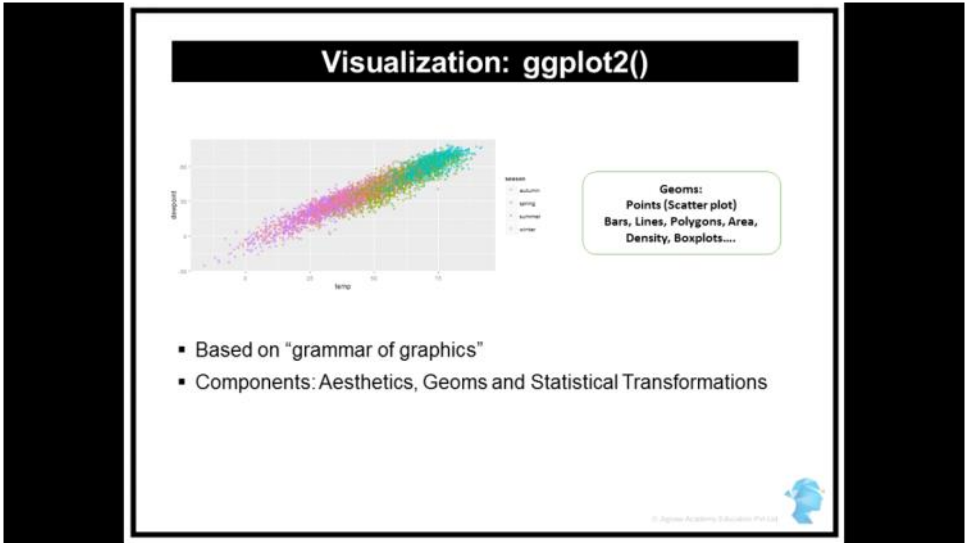
Visualization: ggplot2()

dewpoint
temp
season
autumn
spring
summer
winter

Aesthetics:
Axis Mappings:
X=temp, y=dewpoint
Colour: Seasons

- Based on "grammar of graphics"
- Components: Aesthetics, Geoms and Statistical Transformations

This is based on a sample data for seasons, temperature and dewpoints.

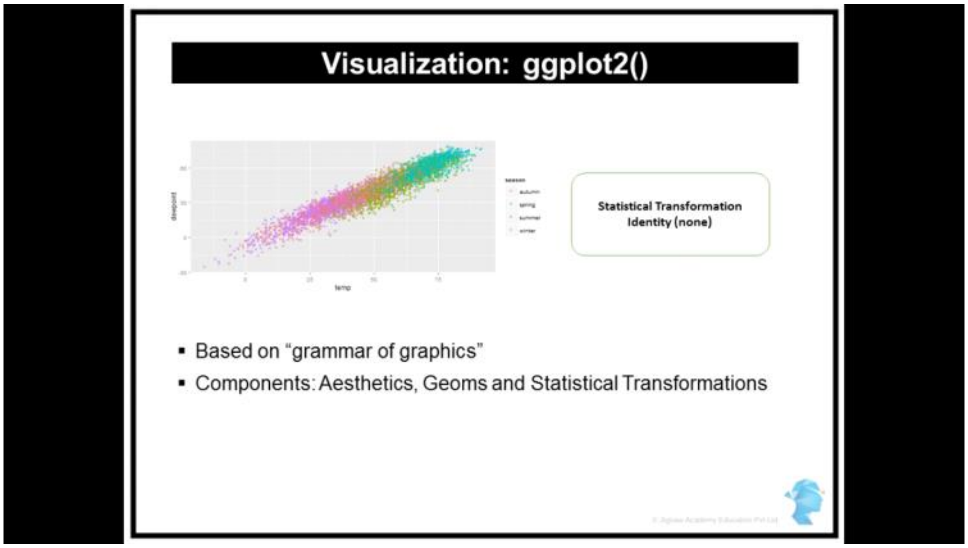


Visualization: ggplot2()

dewpoint
temp
season
autumn
spring
summer
winter

Geoms:
Points (Scatter plot)
Bars, Lines, Polygons, Area,
Density, Boxplots....

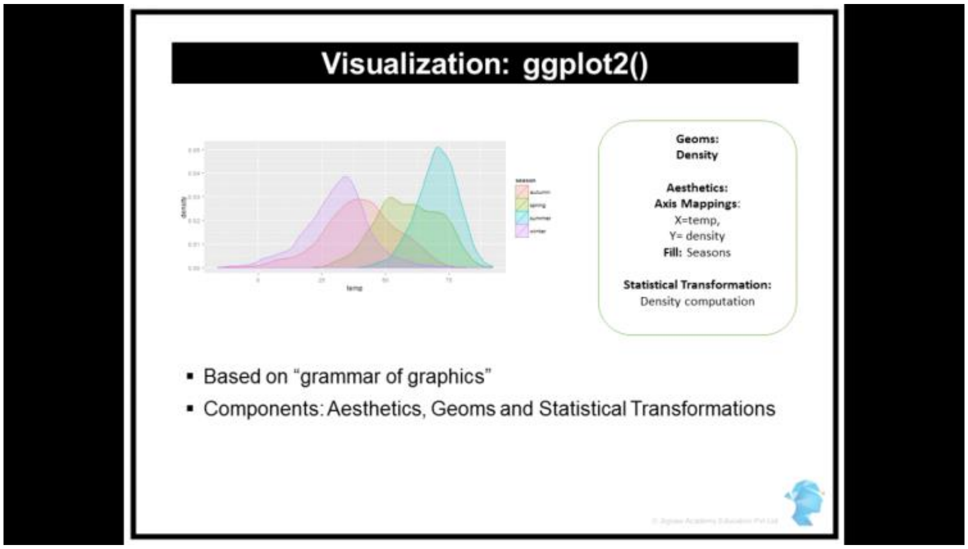
- Based on "grammar of graphics"
- Components: Aesthetics, Geoms and Statistical Transformations



Visualization: ggplot2()

Statistical Transformation
Identity (none)

- Based on “grammar of graphics”
- Components: Aesthetics, Geoms and Statistical Transformations



Visualization: ggplot2()

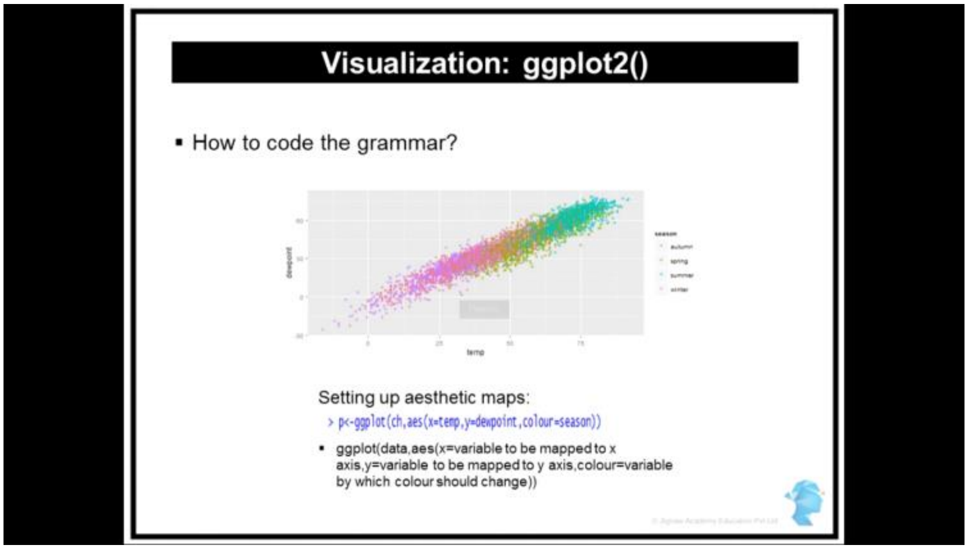
Geoms:
Density

Aesthetics:
Axis Mappings:
X=temp,
Y= density
Fill: Seasons

Statistical Transformation:
Density computation

- Based on “grammar of graphics”
- Components: Aesthetics, Geoms and Statistical Transformations

Learning to write code for GGPLOT2



Visualization: ggplot2()

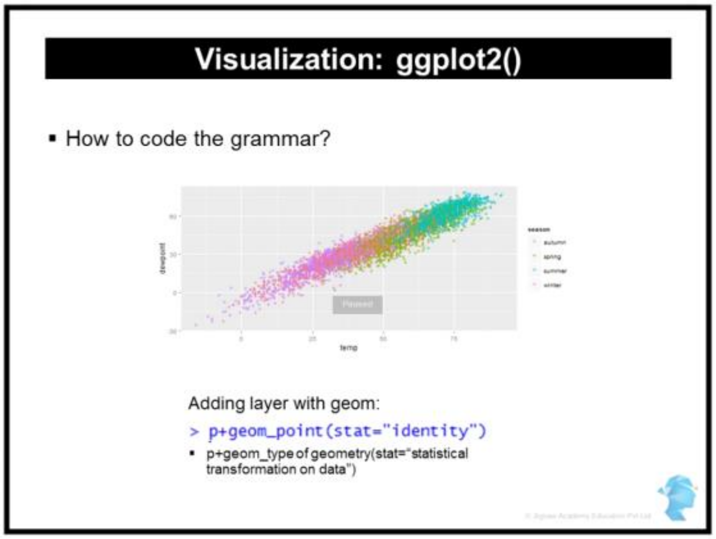
- How to code the grammar?

Setting up aesthetic maps:

```
> p<-ggplot(ch,aes(x=temp,y=dewpoint,colour=season))
```

- ggplot(data,aes(x=variable to be mapped to x axis,y=variable to be mapped to y axis,colour=variable by which colour should change))

```
p<-ggplot(dataset, aes(x=temp,y=dewpoint, colour=season))
p+geom_point(stat="identity")
```



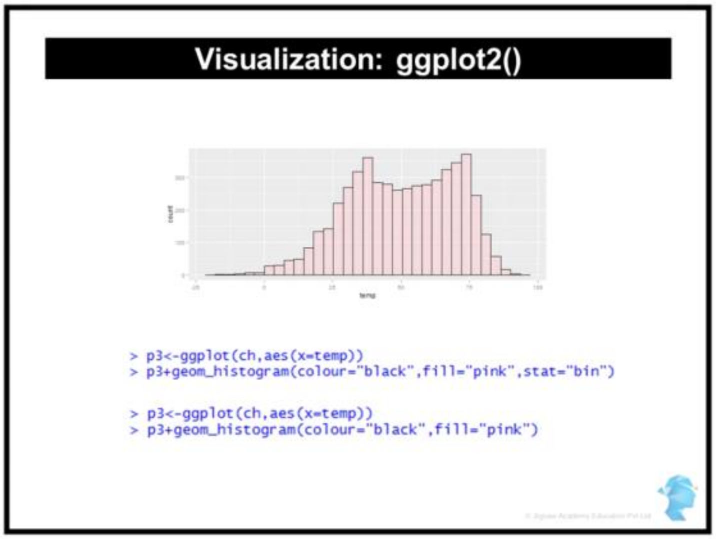
Visualization: ggplot2()

- How to code the grammar?

Adding layer with geom:

```
> p+geom_point(stat="identity")
```

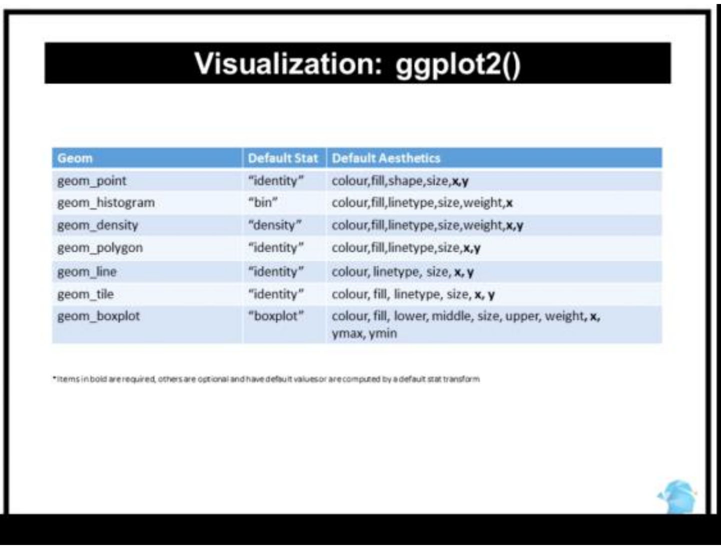
- p+geom_type of geometry(stat="statistical transformation on data")



Visualization: ggplot2()

```
> p3<-ggplot(ch,aes(x=temp))
> p3+geom_histogram(colour="black",fill="pink",stat="bin")
```

```
> p3<-ggplot(ch,aes(x=temp))
> p3+geom_histogram(colour="black",fill="pink")
```



Visualization: ggplot2()

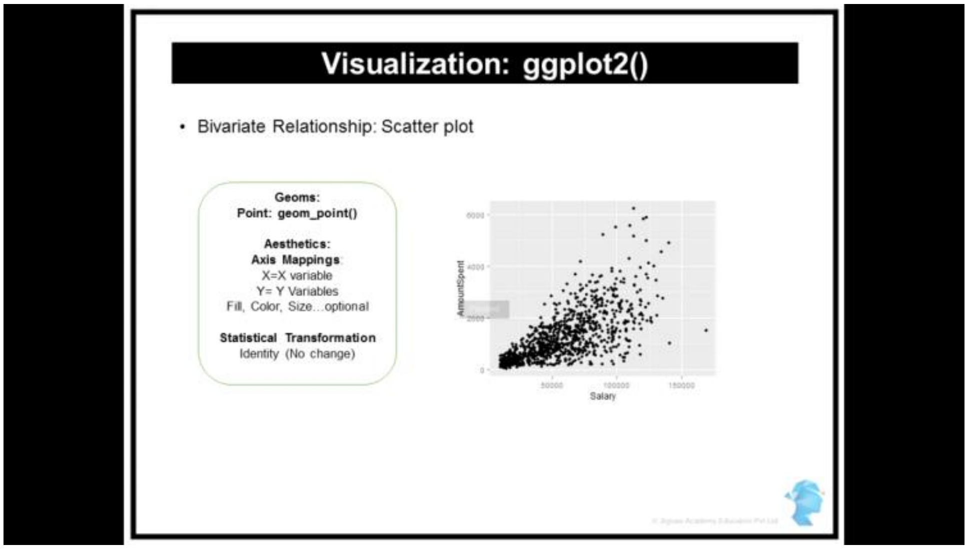
Geom	Default Stat	Default Aesthetics
geom_point	"identity"	colour,fill,shape,size,**x,y**
geom_histogram	"bin"	colour,fill,linetype,size,weight,**x**
geom_density	"density"	colour,fill,linetype,size,weight,**x,y**
geom_polygon	"identity"	colour,fill,linetype,size,**x,y**
geom_line	"identity"	colour, linetype, size, **x, y**
geom_tile	"identity"	colour, fill, linetype, size, **x, y**
geom_boxplot	"boxplot"	colour, fill, lower, middle, size, upper, weight, **x**, ymax, ymin

*Items in bold are required, others are optional and have default valuesor are computed by a default stat transform

Creating Common Plots

© Jigsaw Academy Education Pvt Ltd

Lets look at customer demography data:



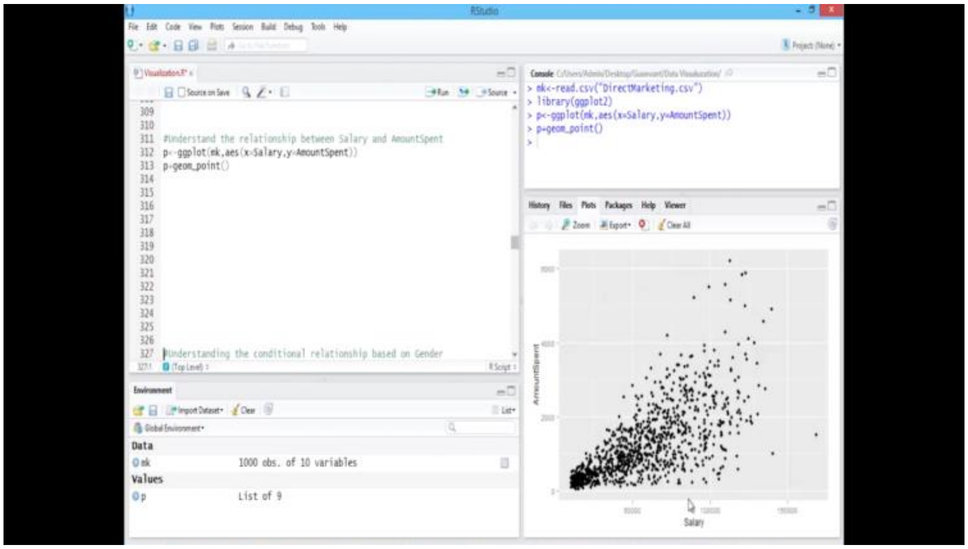
Visualization: ggplot2()

- Bivariate Relationship: Scatter plot

Geoms:
Point: geom_point()

Aesthetics:
Axis Mappings:
X=X variable
Y= Y Variables
Fill, Color, Size…optional

Statistical Transformation
Identity (No change)



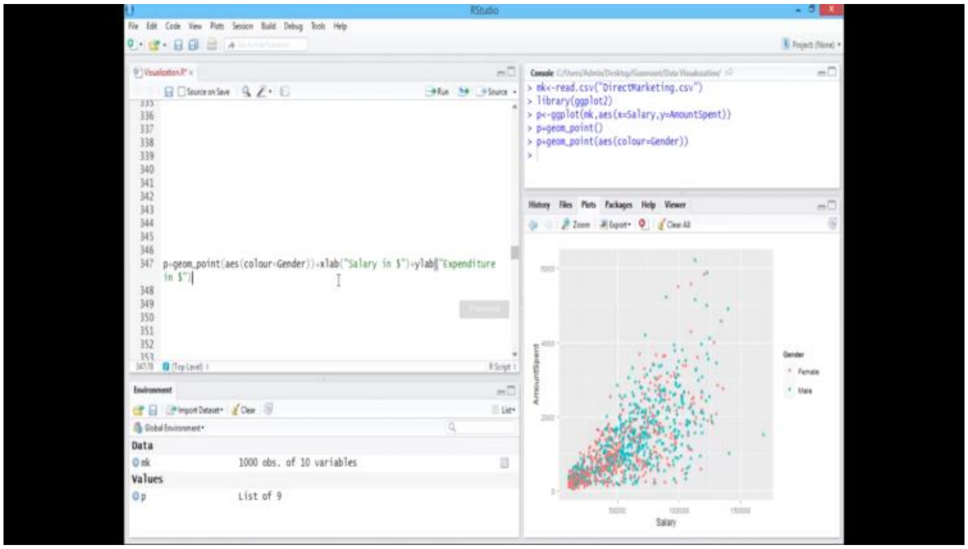
```
#Understand the relationship between Salary and AmountSpent
p<-ggplot(mk,aes(x=Salary,y=AmountSpent))
p+geom_point()



#Understanding the conditional relationship based on Gender
```

Console C:/Users/Admin/Desktop/Data Visualization/

```
> mk<-read.csv("DirectMarketing.csv")
> library(ggplot2)
> p<-ggplot(mk,aes(x=Salary,y=AmountSpent))
> p+geom_point()
>
```

```
Data
mk          1000 obs. of 10 variables
Values
p           List of 9
```

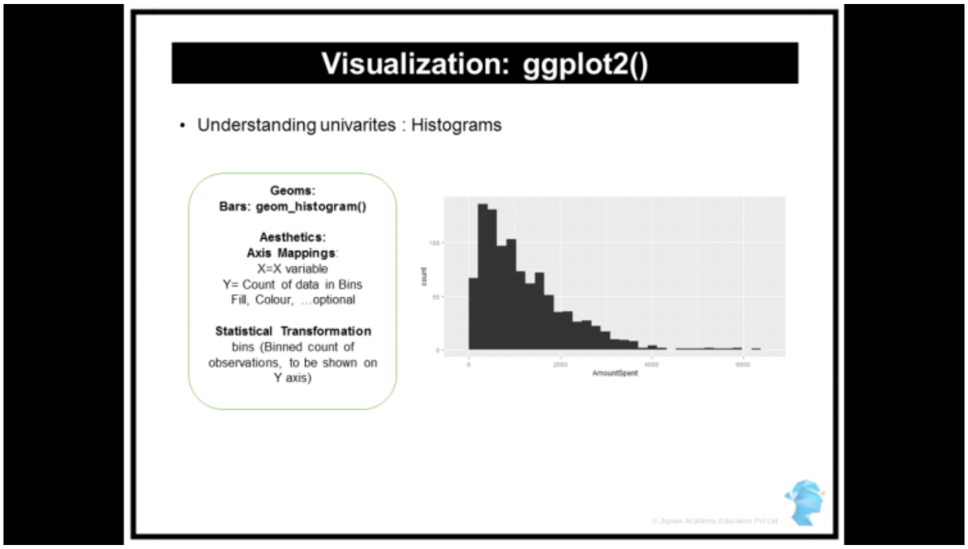
To further segregate the data based on gender and to add lables to the graph:
We can add the AES function to the geom_point function
i.e. p+geom_point(aes(colour="Gender"))+xlab("salary")+ylab("expenditure")



```
p+geom_point(aes(colour=Gender))+xlab("Salary in $")+ylab("Expenditure in $")
```

```
> mk<-read.csv("DirectMarketing.csv")
> library(ggplot2)
> p<-ggplot(mk,aes(x=Salary,y=AmountSpent))
> p+geom_point()
> p+geom_point(aes(colour=Gender))
>
```

Data
mk 1000 obs. of 10 variables
Values
p List of 9



Visualization: ggplot2()

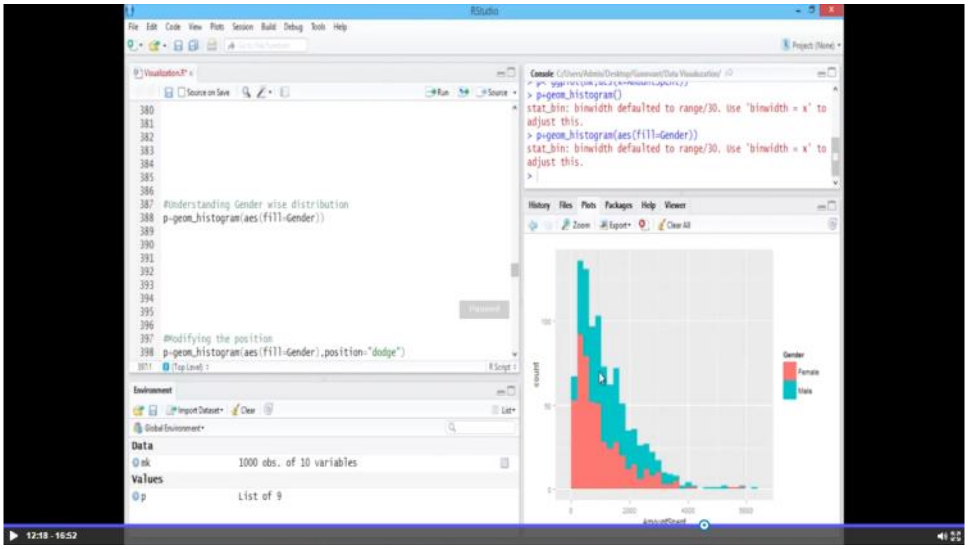
- Understanding univarites : Histograms

Geoms:
Bars: geom_histogram()

Aesthetics:
Axis Mappings:
X=X variable
Y= Count of data in Bins
Fill, Colour, …optional

Statistical Transformation
bins (Binned count of observations, to be shown on Y axis)

p+geom_histogram(aes(fill=Species))

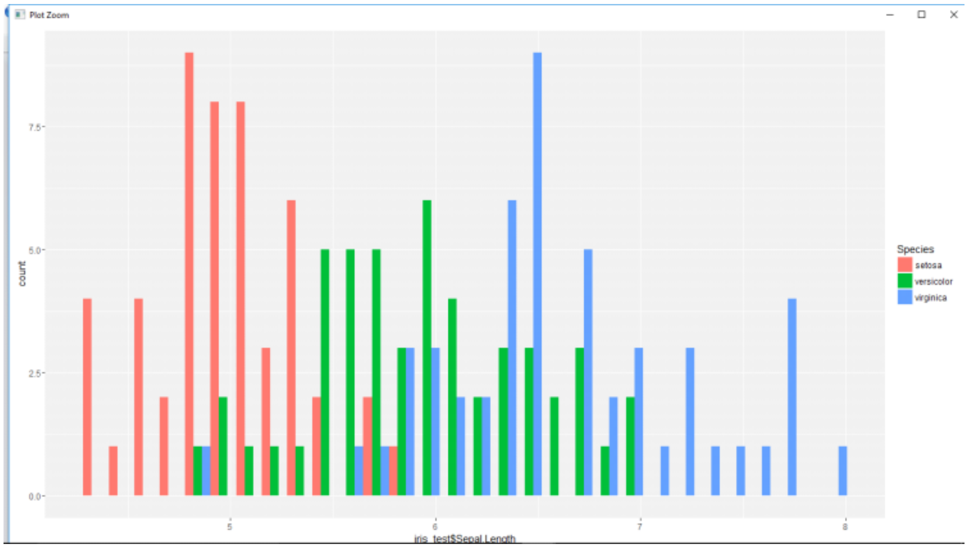


```
#Understanding Gender wise distribution
p+geom_histogram(aes(fill=Gender))


#Modifying the position
p+geom_histogram(aes(fill=Gender),position="dodge")
```

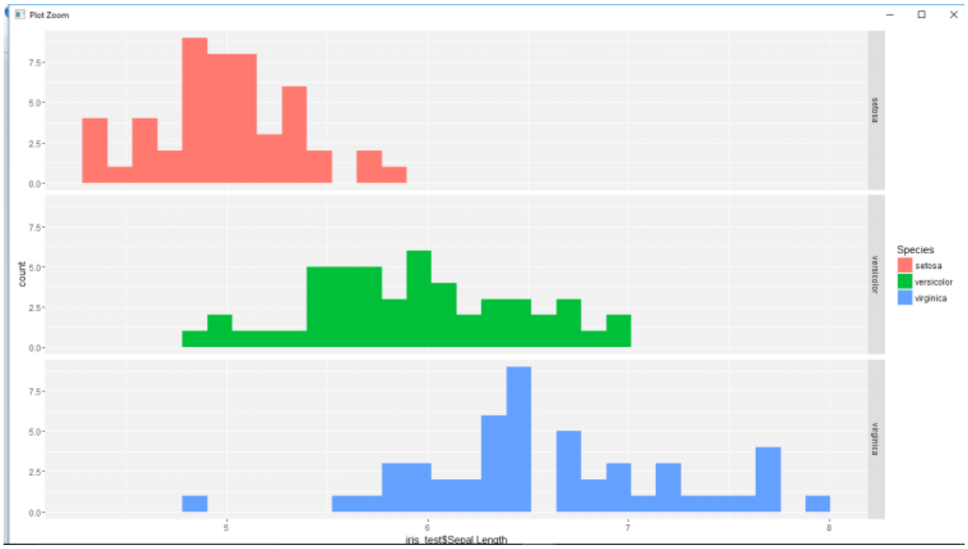
```
> p+geom_histogram()
stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to
adjust this.
> p+geom_histogram(aes(fill=Gender))
stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to
adjust this.
>
```

Data
mk 1000 obs. of 10 variables
Values
p List of 9

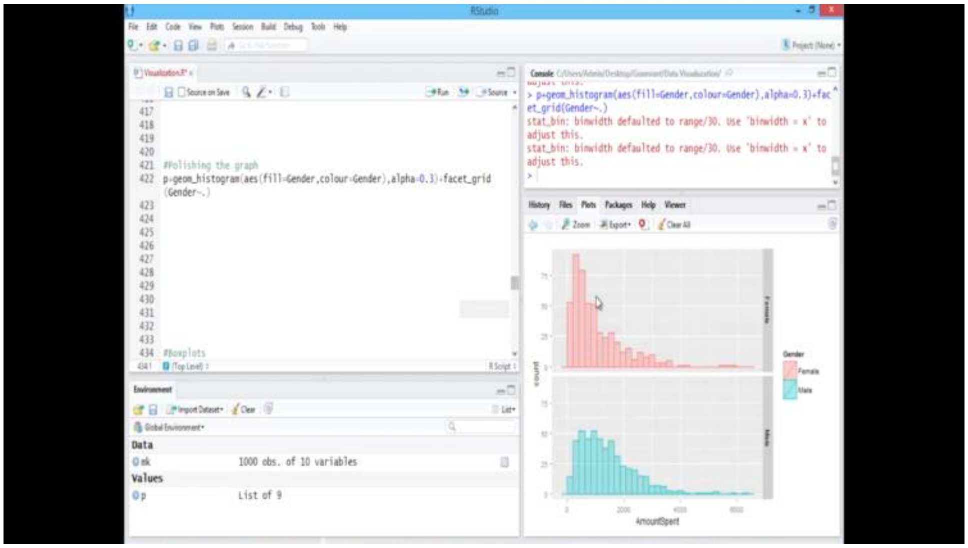
p+geom_histogram(aes(fill=Species),position = "dodge")



p+geom_histogram(aes(fill=Species))+facet_grid(Species~.)



For further polishing the graph:



```
#Polishing the graph
p+geom_histogram(aes(fill=Gender,colour=Gender),alpha=0.3)+facet_grid
(Gender~.)



#Boxplots
```

```
> p+geom_histogram(aes(fill=Gender,colour=Gender),alpha=0.3)+fac
et_grid(Gender~.)
stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to
adjust this.
stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to
adjust this.
>
```

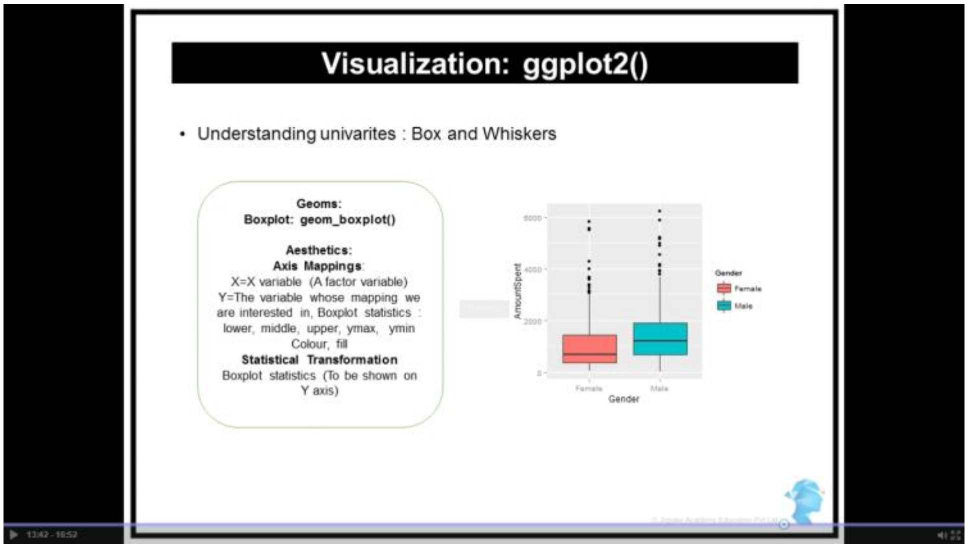
Data

mk 1000 obs. of 10 variables

Values

p List of 9

BOX PLOTS:

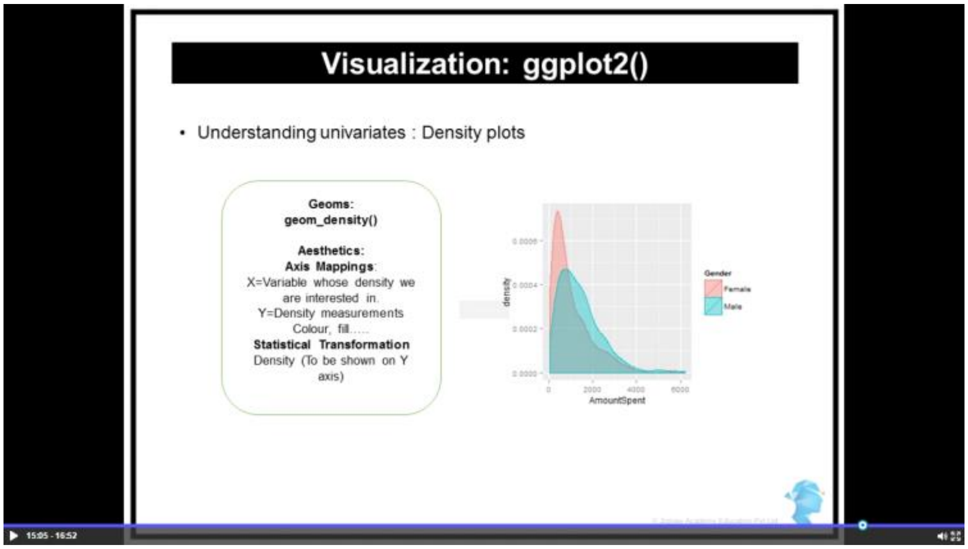


Visualization: ggplot2()

- Understanding univarites : Box and Whiskers

Geoms:
Boxplot: geom_boxplot()

Aesthetics:
Axis Mappings:
X=X variable (A factor variable)
Y=The variable whose mapping we are interested in, Boxplot statistics : lower, middle, upper, ymax, ymin
Colour, fill
Statistical Transformation
Boxplot statistics (To be shown on Y axis)

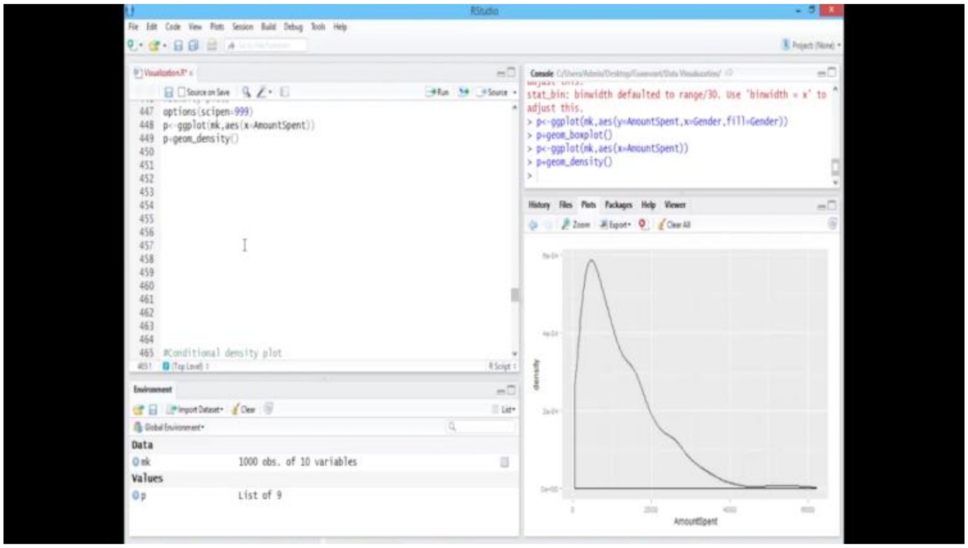


Visualization: ggplot2()

- Understanding univariates : Density plots

Geoms:
geom_density()

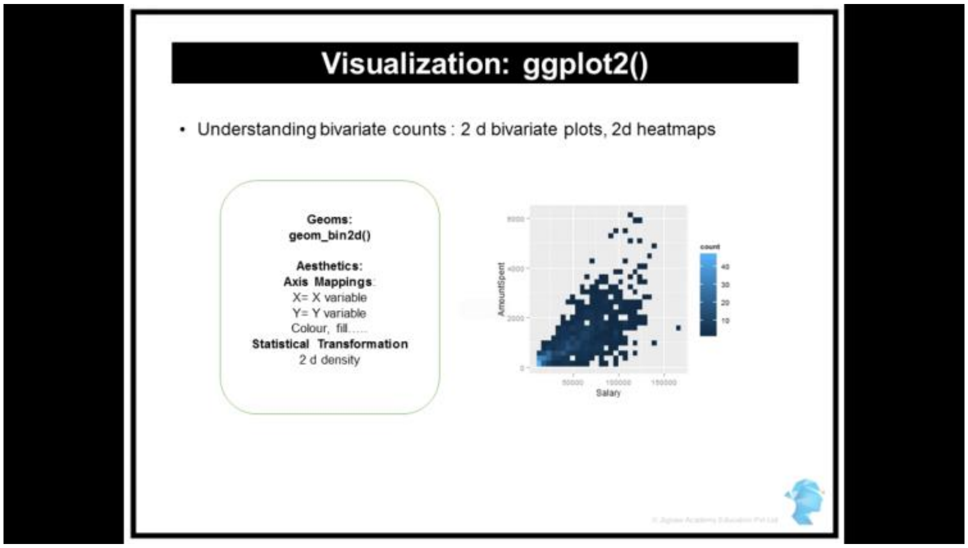
Aesthetics:
Axis Mappings:
X=Variable whose density we are interested in.
Y=Density measurements
Colour, fill.....
Statistical Transformation
Density (To be shown on Y axis)



```
options(scipen=999)
p<-ggplot(mk,aes(x=AmountSpent))
p+geom_density()







#Conditional density plot
```

```
stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to
adjust this.
> p<-ggplot(mk,aes(y=AmountSpent,x=Gender,fill=Gender))
> p+geom_boxplot()
> p<-ggplot(mk,aes(x=AmountSpent))
> p+geom_density()
>
```

Data
mk 1000 obs. of 10 variables
Values
p List of 9

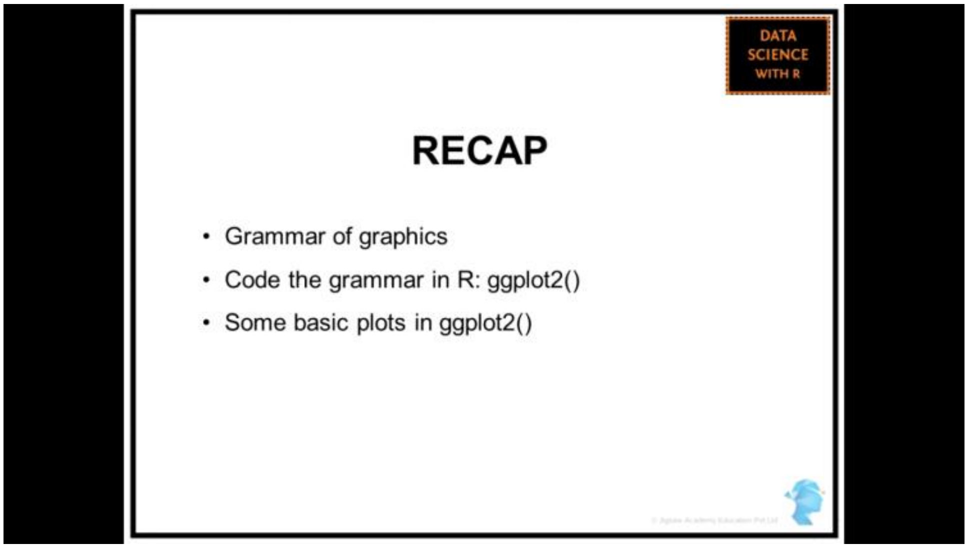


Visualization: ggplot2()

- Understanding bivariate counts : 2 d bivariate plots, 2d heatmaps

Geoms:
geom_bin2d()

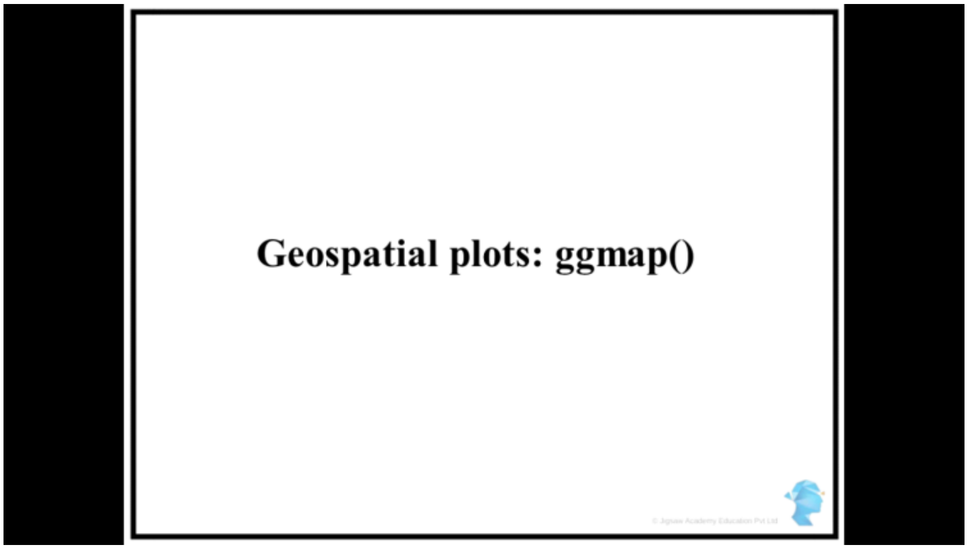
Aesthetics:
Axis Mappings:
X= X variable
Y= Y variable
Colour, fill.....
Statistical Transformation
2 d density



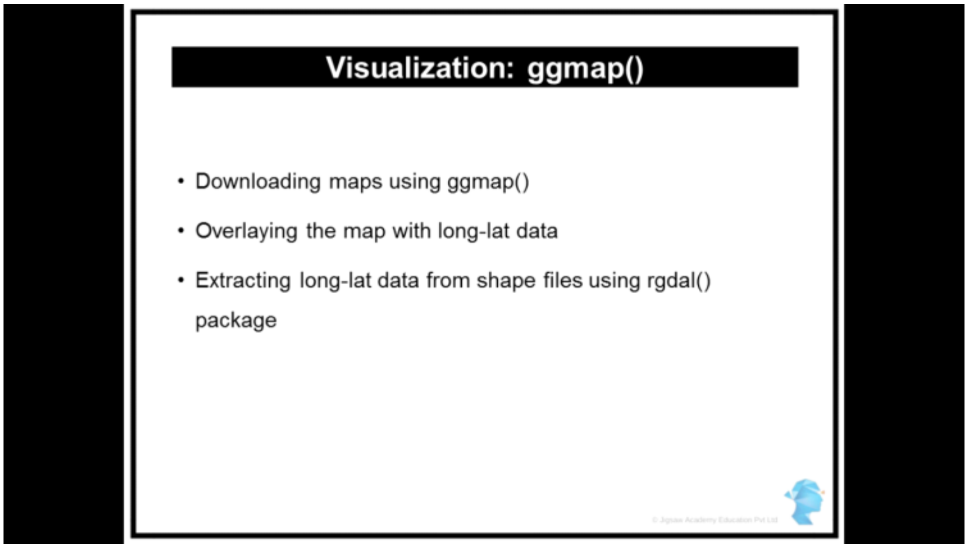
DATA
SCIENCE
WITH R

RECAP

- Grammar of graphics
- Code the grammar in R: ggplot2()
- Some basic plots in ggplot2()

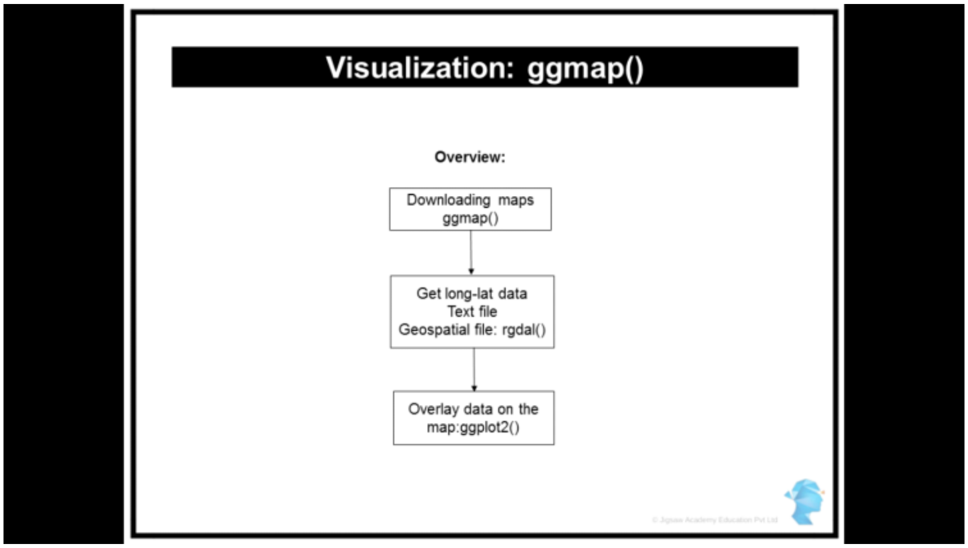


Geospatial plots: ggmap()



Visualization: ggmap()

- Downloading maps using ggmap()
- Overlaying the map with long-lat data
- Extracting long-lat data from shape files using rgdal() package



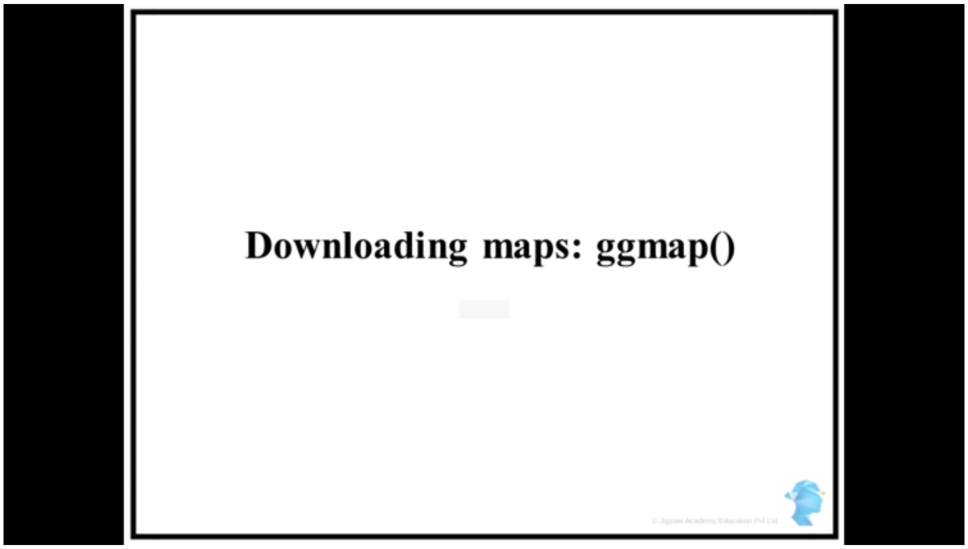
Visualization: ggmap()

Overview:

Downloading maps ggmap()

Get long-lat data
Text file
Geospatial file: rgdal()

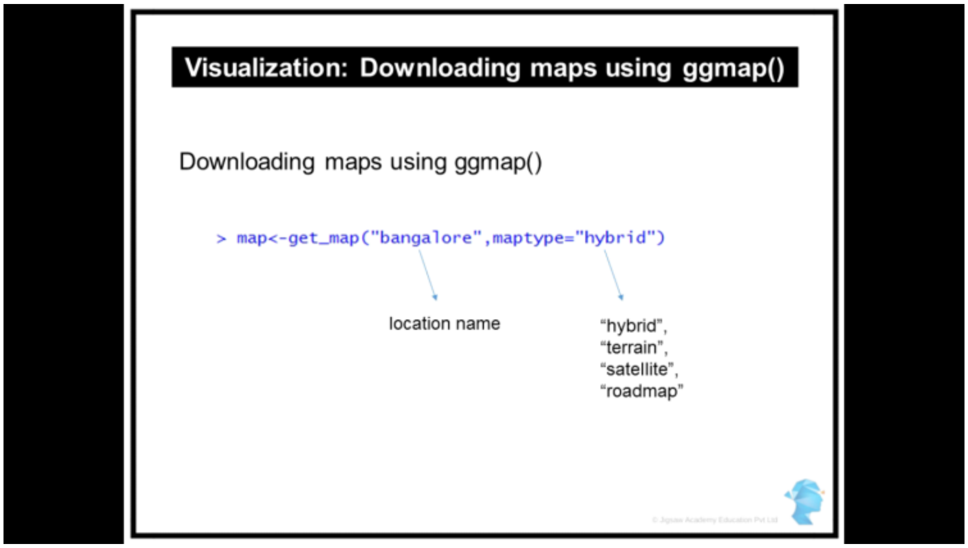
Overlay data on the map:ggplot2()



Downloading maps: ggmap()

Here is the command to get the map of a location that we want to plot:

```
Map<- get_map("North Carolina", maptype="hybrid")
```



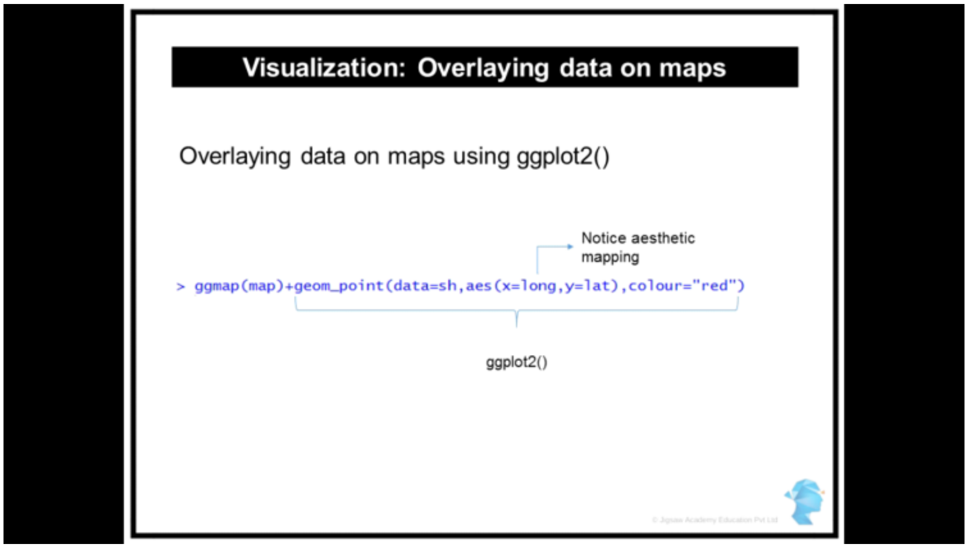
Visualization: Downloading maps using ggmap()

Downloading maps using ggmap()

```
> map<-get_map("bangalore",maptype="hybrid")
```

location name

"hybrid",
"terrain",
"satellite",
"roadmap"



Visualization: Overlaying data on maps

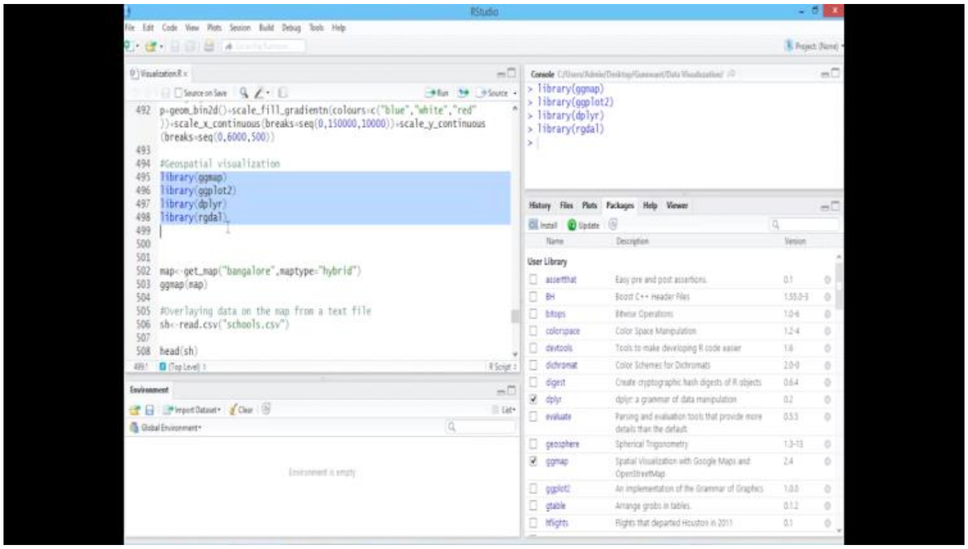
Overlaying data on maps using ggplot2()

Notice aesthetic mapping

```
> ggmap(map)+geom_point(data=sh,aes(x=long,y=lat),colour="red")
```

ggplot2()

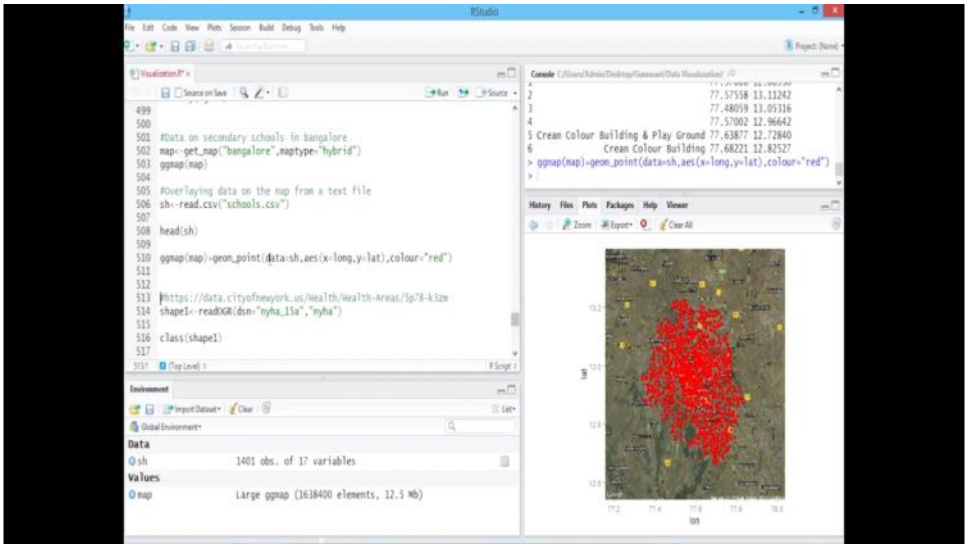
Example:



```
p+geom_bin2d()+scale_fill_gradientn(colours=c("blue","white","red"
))+scale_x_continuous(breaks=seq(0,150000,10000))+scale_y_continuous
(breaks=seq(0,6000,500))

#Geospatial visualization
library(ggmap)
library(ggplot2)
library(dplyr)
library(rgdal)



map<-get_map("bangalore",maptype="hybrid")
ggmap(map)

#Overlaying data on the map from a text file
sh<-read.csv("schools.csv")

head(sh)
```

```
> library(ggmap)
> library(ggplot2)
> library(dplyr)
> library(rgdal)
>
```

	Name	Description	Version
	User Library		
□	assertthat	Easy pre and post assertions.	0.1
□	BH	Boost C++ Header Files	1.55.0-3
□	bitops	Bitwise Operations	1.0-6
□	colorspace	Color Space Manipulation	1.2-4
□	devtools	Tools to make developing R code easier	1.6
□	dichromat	Color Schemes for Dichromats	2.0-0
□	digest	Create cryptographic hash digests of R objects	0.6.4
☑	dplyr	dplyr: a grammar of data manipulation	0.2
□	evaluate	Parsing and evaluation tools that provide more details than the default.	0.5.5
□	geosphere	Spherical Trigonometry	1.3-13
☑	ggmap	Spatial Visualization with Google Maps and OpenStreetMap	2.4
□	ggplot2	An implementation of the Grammar of Graphics	1.0.0
□	gtable	Arrange grobs in tables.	0.1.2
□	hflights	Flights that departed Houston in 2011	0.1

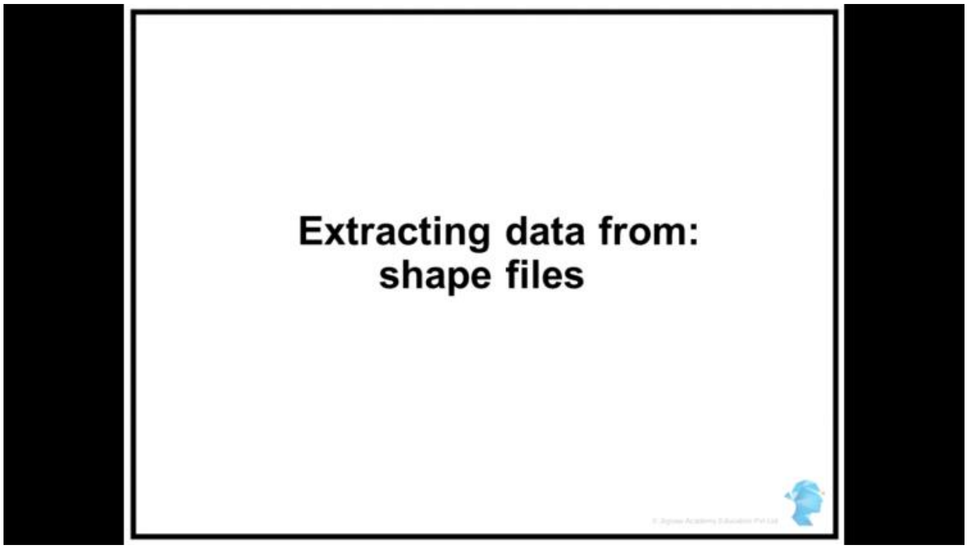


```
#Data on secondary schools in bangalore
map<-get_map("bangalore",maptype="hybrid")
ggmap(map)

#Overlaying data on the map from a text file
sh<-read.csv("schools.csv")

head(sh)

ggmap(map)+geom_point(data=sh,aes(x=long,y=lat),colour="red")


#https://data.cityofnewyork.us/Health/Health-Areas/5p78-k3zm
shape1<-readOGR(dsn="nyha_15a","nyha")

class(shape1)
```

```
2                                       77.57558 13.11242
3                                       77.48059 13.05316
4                                       77.57002 12.96642
5 Crean Colour Building & Play Ground 77.63877 12.72840
6                  Crean Colour Building 77.68221 12.82527
> ggmap(map)+geom_point(data=sh,aes(x=long,y=lat),colour="red")
>
```

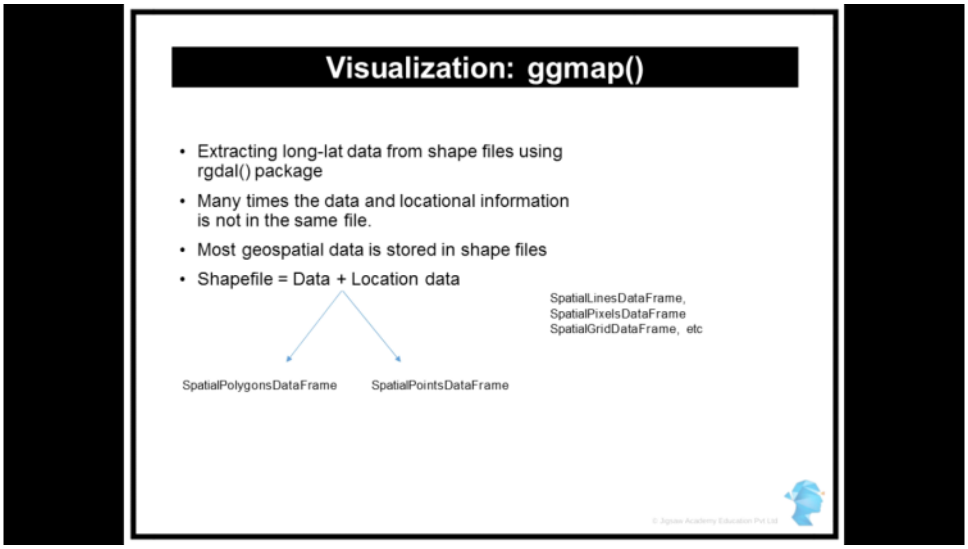
```
Data
sh          1401 obs. of 17 variables
Values
map         Large ggmap (1638400 elements, 12.5 Mb)
```

Ggmap(map)+geom_point(data=sh, aes(x=long, y=lat), colour="red")



Extracting data from: shape files

© Jigsaw Academy Education Pvt Ltd



Visualization: ggmap()

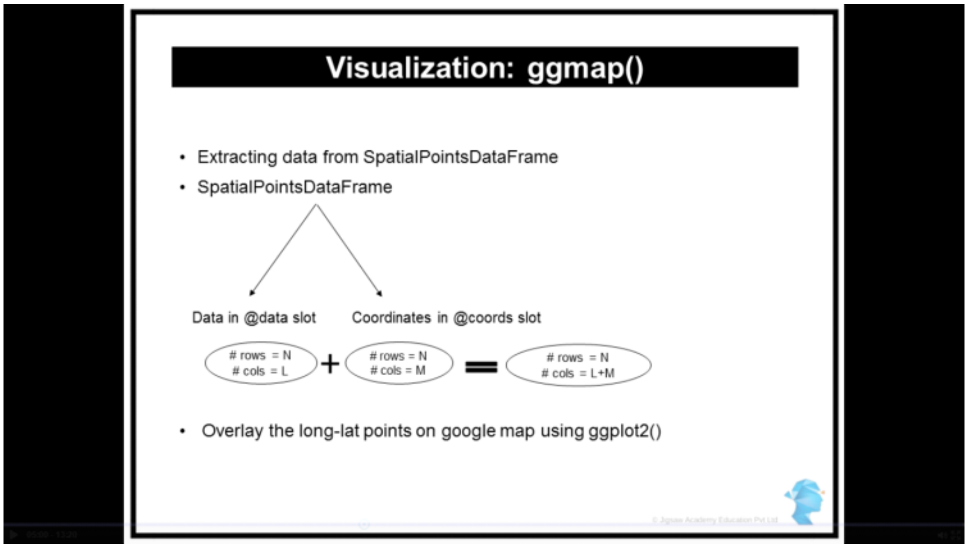
- Extracting long-lat data from shape files using rgdal() package
- Many times the data and locational information is not in the same file.
- Most geospatial data is stored in shape files
- Shapefile = Data + Location data

SpatialLinesDataFrame,
SpatialPixelsDataFrame
SpatialGridDataFrame, etc

SpatialPolygonsDataFrame

SpatialPointsDataFrame

© Jigsaw Academy Education Pvt Ltd



Visualization: ggmap()

- Extracting data from SpatialPointsDataFrame
- SpatialPointsDataFrame

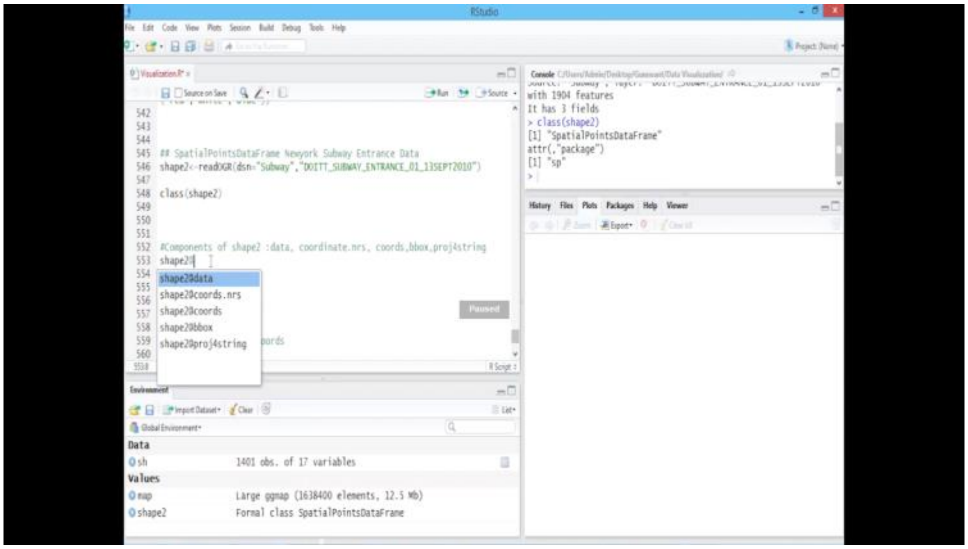
- Overlay the long-lat points on google map using ggplot2()

SpatialPointsDataframe:

Shape2<- readOGR(dsn="foldername", "filename")
Class(shape2)

This will be of class SpatialPointsDataFrame

To get to know the components of this dataframe we will use the @ symbol and not $
i.e. shapes2@



```
## SpatialPointsDataFrame Newyork Subway Entrance Data
shape2<-readOGR(dsn="Subway","DOITT_SUBWAY_ENTRANCE_01_13SEPT2010")

class(shape2)



#Components of shape2 :data, coordinate.nrs, coords,bbox,proj4string
shape2@
```

```
shape2@data
shape2@coords.nrs
shape2@coords
shape2@bbox
shape2@proj4string
```

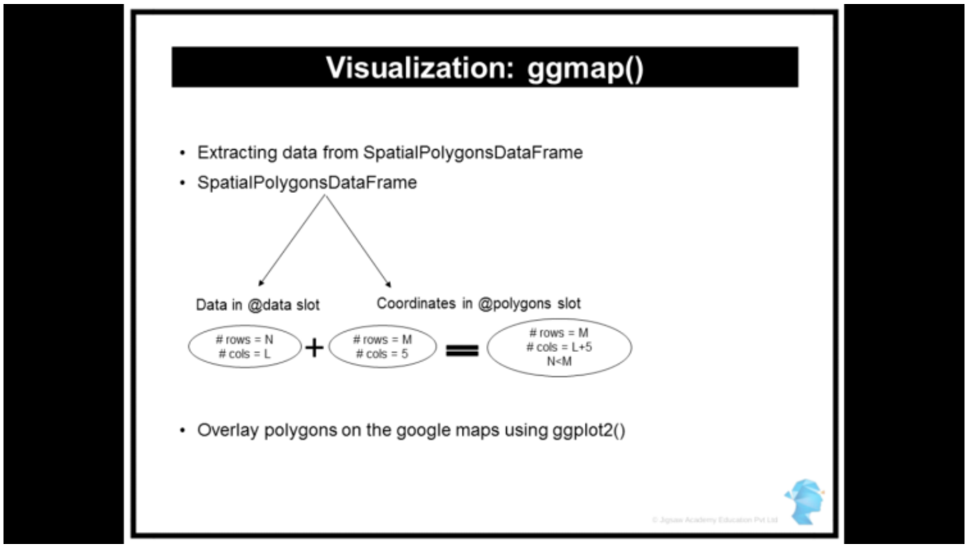
```
with 1904 features
It has 3 fields
> class(shape2)
[1] "SpatialPointsDataFrame"
attr(,"package")
[1] "sp"
>
```

```
Data
sh        1401 obs. of 17 variables
Values
map       Large ggmap (1638400 elements, 12.5 Mb)
shape2    Formal class SpatialPointsDataFrame
```

If the data is in northings and eastings, we will need to convert the coordinates to lat-long format

i.e sptransform(shape2, CRS("+init=epsg:4326"))
Once this transformation is done, we will combine the two dataframes

Datac<-data.frame(shape2@data, shape2@coords)

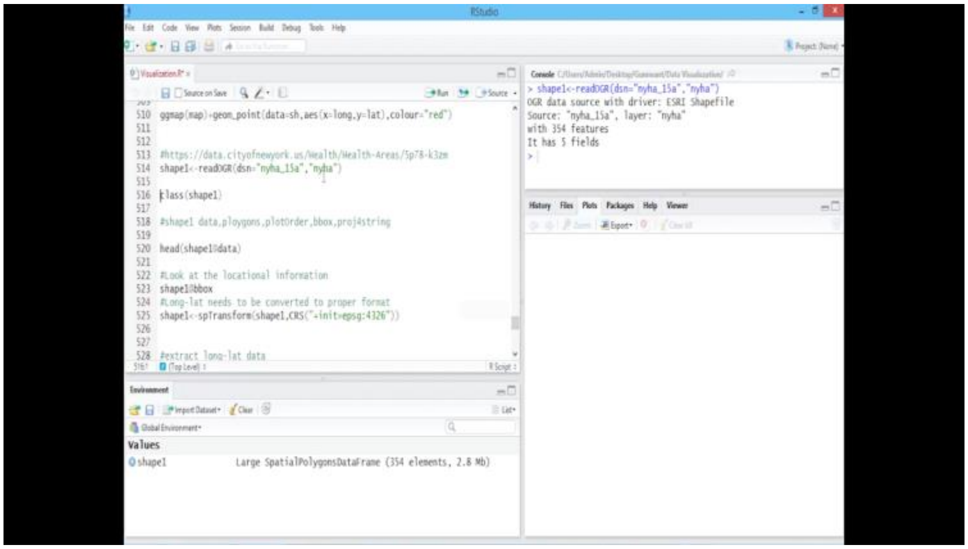


Visualization: ggmap()

- Extracting data from SpatialPolygonsDataFrame
- SpatialPolygonsDataFrame

- Overlay polygons on the google maps using ggplot2()

These are polygon dataframes

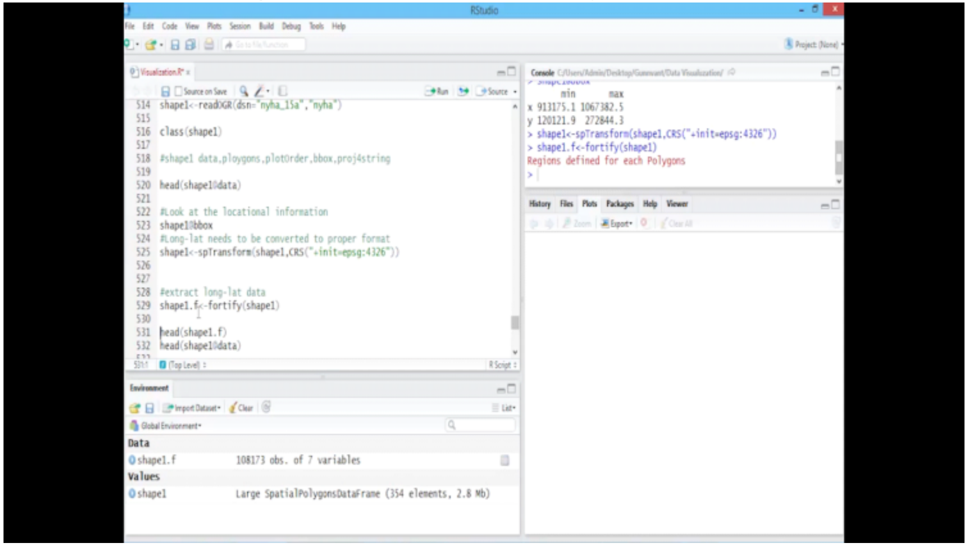


```
ggmap(map)+geom_point(data=sh,aes(x=long,y=lat),colour="red")


#https://data.cityofnewyork.us/Health/Health-Areas/5p78-k3zm
shape1<-readOGR(dsn="nyha_15a","nyha")

class(shape1)

#shape1 data,ploygons,plotOrder,bbox,proj4string

head(shape1@data)

#Look at the locational information
shape1@bbox
#Long-lat needs to be converted to proper format
shape1<-spTransform(shape1,CRS("+init=epsg:4326"))


#extract long-lat data
```

```
> shape1<-readOGR(dsn="nyha_15a","nyha")
OGR data source with driver: ESRI Shapefile
Source: "nyha_15a", layer: "nyha"
with 354 features
It has 5 fields
>
```

```
Values
shape1          Large SpatialPolygonsDataFrame (354 elements, 2.8 Mb)
```

To extract the lat-long data, we will use the fortify function:

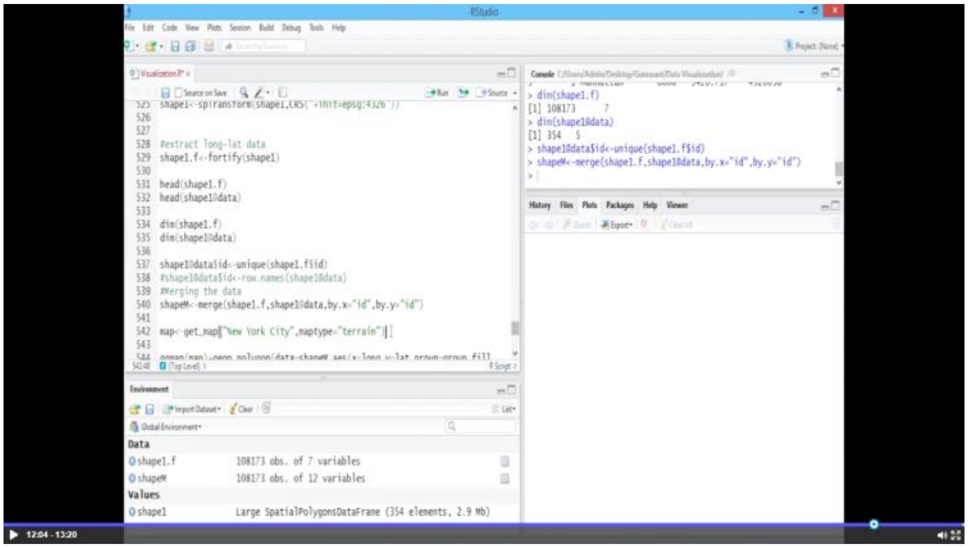


```
shape1<-readOGR(dsn="nyha_15a","nyha")

class(shape1)

#shape1 data,ploygons,plotOrder,bbox,proj4string

head(shape1@data)

#Look at the locational information
shape1@bbox
#Long-lat needs to be converted to proper format
shape1<-spTransform(shape1,CRS("+init=epsg:4326"))


#extract long-lat data
shape1.f<-fortify(shape1)

head(shape1.f)
head(shape1@data)
```

```
       min       max
x 913175.1 1067382.5
y 120121.9  272844.3
> shape1<-spTransform(shape1,CRS("+init=epsg:4326"))
> shape1.f<-fortify(shape1)
Regions defined for each Polygons
>
```

```
Data
shape1.f        108173 obs. of 7 variables
Values
shape1          Large SpatialPolygonsDataFrame (354 elements, 2.8 Mb)
```

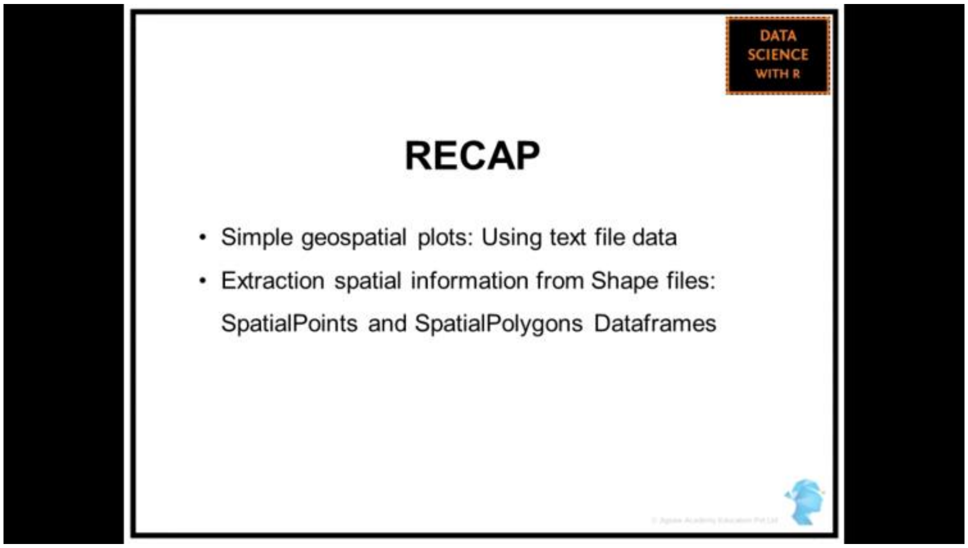
Merge the two datasets based on ID



```r
shape1<-spTransform(shape1,CRS("+init=epsg:4326"))


#extract long-lat data
shape1.f<-fortify(shape1)

head(shape1.f)
head(shape1@data)

dim(shape1.f)
dim(shape1@data)

shape1@data$id<-unique(shape1.f$id)
#shape1@data$id<-row.names(shape1@data)
#Merging the data
shapeM<-merge(shape1.f,shape1@data,by.x="id",by.y="id")

map<-get_map("New York City",maptype="terrain")

ggmap(map)+geom_polygon(data=shapeM,aes(x=long,y=lat,group=group,fill
```

```
> dim(shape1.f)
[1] 108173      7
> dim(shape1@data)
[1] 354   5
> shape1@data$id<-unique(shape1.f$id)
> shapeM<-merge(shape1.f,shape1@data,by.x="id",by.y="id")
>
```

```
Data
shape1.f        108173 obs. of 7 variables
shapeM          108173 obs. of 12 variables
Values
shape1          Large SpatialPolygonsDataFrame (354 elements, 2.9 Mb)
```


DATA
SCIENCE
WITH R

RECAP

- Simple geospatial plots: Using text file data
- Extraction spatial information from Shape files: SpatialPoints and SpatialPolygons Dataframes