DATA SCIENCE DOJO

2205 152ND AVE NE, REDMOND, WA 98052

The Library of Congress has catalogued this edition as follows:
Data Science Dojo LLC
Data Science and Data Engineering Handbook : 1.9
Registration Number: TX0008030133
2015-05-22

Printed in the United States of America

Contents

Appendices

A. Data Exploration Supplementary

A.1 Important R Commands

It is important to explore any dataset by using these common commands.

```
>head()
```

first 6 rows of data

```
>tail ()
```

last 6 rows of data

```
>colnames ()
```

identify column names

```
>dim()
```

provides dimensions, rows and columns of dataframe

```
>summary()
```

5-number summary, mean and missing values

```
>View()
```

view as a spreadsheet in RStudio

A.2 R Core Graphics

Iris Dataset
the Iris dataset is a classic dataset used for teaching data visualization and exploration, introductory classification, and machine learning. It is a very simple dataset that contains 3 classes of 50 instances

each, where each class refers to a type of species of iris plant - **setosa, versicolor, virginica**. There are 5 features/columns in the data set - **Sepal Length, Sepal Width, Petal Length, Petal Width and Species**.
Load the dataset by using the data R command

```
> data(iris)
```

A.2.1 Exercise

Questions

1. Look at the first 6 rows of the iris dataset.

```
>head(iris)
```

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Figure A.1: First 6 rows of iris dataset

2. What are the dimensions of this dataset?

```
>dim(iris)
[1] 150     5
```

The iris dataset has 150 rows and 5 columns

3. What is the maximum and mean Sepal Length?

```
>summary(iris)
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
       Species  
 setosa    :50  
 versicolor:50  
 virginica :50  
```

Figure A.2: Summary of iris

The summary provides the minimum, maximum, median, mean, Q1 and Q3 of all numerical features. The maximum Sepal Length is 7.900 and the mean Sepal Length is 5.843.

A.2.2 Graphical Parameters

There are a number of functions that can be used on graphs.

- **xlab** is used to label the x-axis
- **ylab** is used to label the y-axis
- **main** is used to title the graph
- **col** is used to color the graph
- Customize many features of the graph using the **par** function
- Text and symbol size controlled by **cex** function
- Plotting symbols controlled by **pch** function
- Line width controlled by **lwd** function
- Legend details are controlled by **auto.key** function.

A.3 BoxPlot

Definition: A boxplot or box-and-whiskers plot is a standardized way of displaying the distribution of data based on the five number summary: minimum, first quartile, median, third quartile, and maximum.

Create a boxplot of Sepal Length for all three species and color the graphs blue, green and red.

```
> boxplot(Sepal.Length ~ Species, data=iris, main="Sepal
   Length for various Species", xlab="Species", ylab="
   Sepal Length",col =c("blue", "green","red"))
```

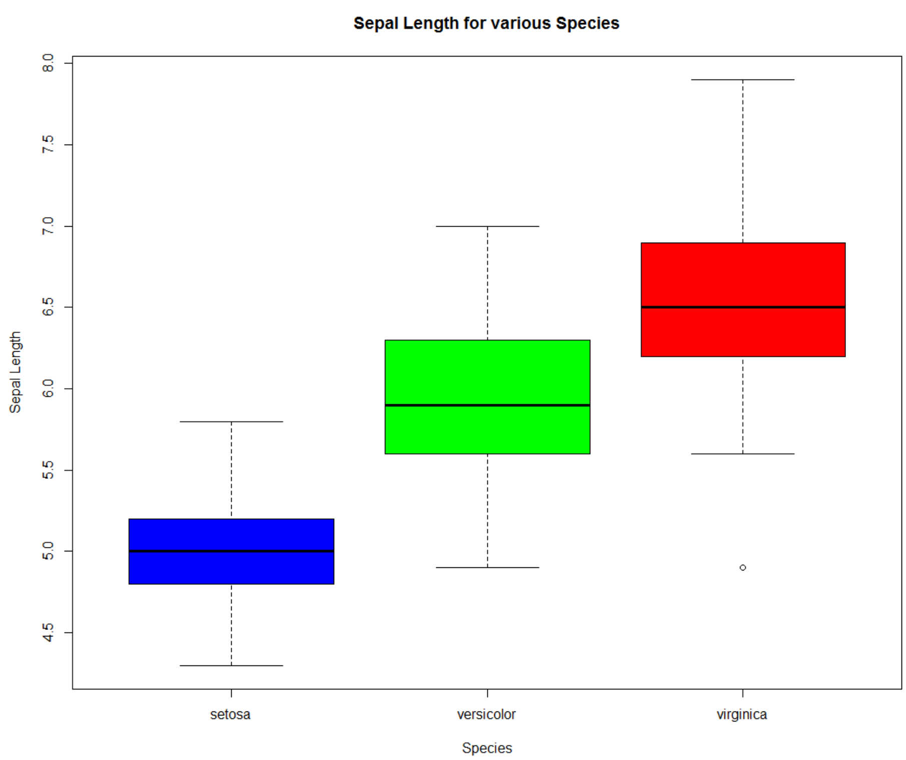

Figure A.3: Boxplot of Sepal Length for various Species

Create a notched boxplot of Petal Length for all three species.

```
> boxplot(Petal.Length ~ Species, data=iris, main="Petal
  Length for various Species", xlab="Species", ylab="
  Petal Length",notch= TRUE, col =c("blue", "green","red"
  ))
```

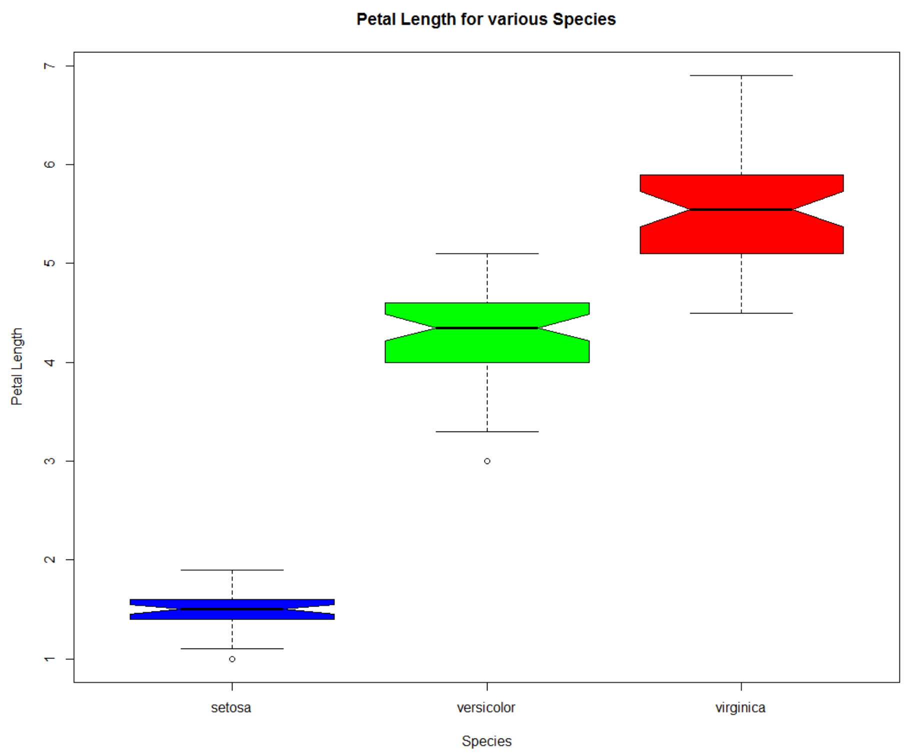

Figure A.4: Notched Boxplot of Petal Length for various Species

A.3.1 Saving Plots

It is important to understand how to save plots in R. Since R runs on so many different operating systems, and supports so many different graphics formats, it's not surprising that there are a variety of ways of saving your plots, depending on what operating system you are using and what you plan to do with the graph.

Save the boxplot you just created above as a pdf.

```
>pdf(''myplot.pdf'')
> boxplot(Petal.Length ~ Species, data=iris, main="Petal
  Length for various Species", xlab="Species", ylab="
  Petal Length",notch=TRUE, col =c("blue", "green","red")
  )
>dev.off() # Retunrs plot to IDE
```

A.4 R core plot function

Definition: The plot function is used to plot two numerical variables and helps in determining a relationship between them.

Using the iris dataset, create a scatterplot of Sepal Length versus Sepal Width. Make sure to label the x and y axes and title the graph.

```
>plot(Sepal.Length ~ Sepal.Width, data=iris, xlab= "Sepal
  Length", ylab="Sepal Width", main="Scatterplot of Sepal
   Width verus Sepal Length")
```

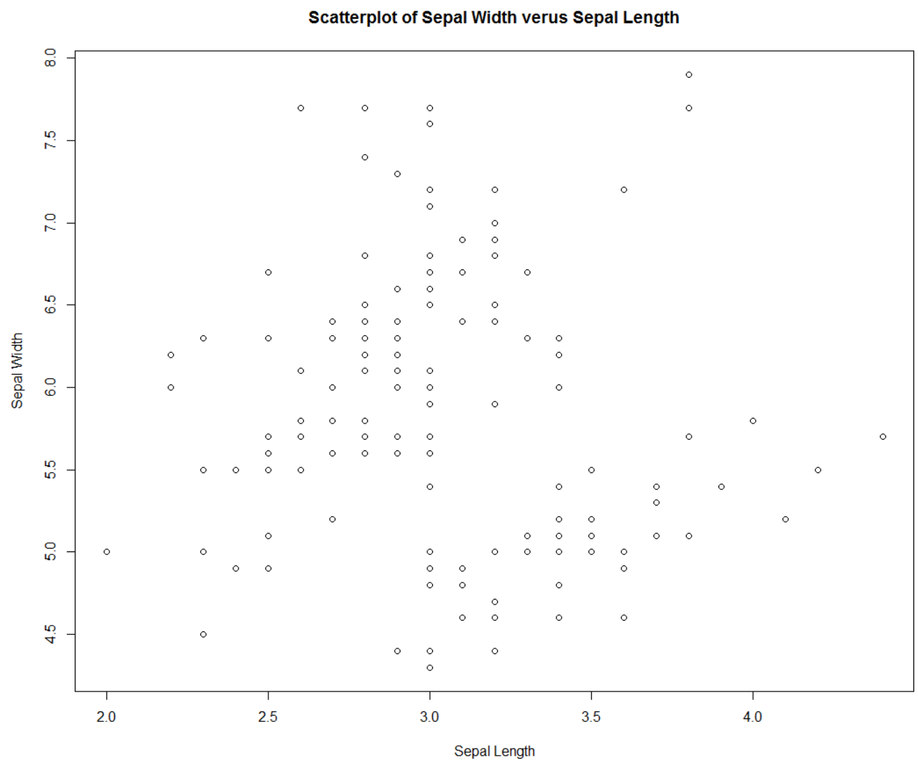

Figure A.5: Scatterplot

Questions

1. What do you notice about the scatterplot? Is there a relationship between the two variables?
2. What can we do to the graph to better understand the relationship?

A.5 Lattice Graphics

Some of the problems we encountered with the previous scatterplot was the inability to distinguish between the different species. We will introduce another type of scatterplot that will allow us to segment using a categorical variable, in this case species of iris plants. In order to do this, we will however need to install a graphics package called lattice.

A.5.1 Lattice xyplot function

Definition: The **xyplot** function is part of the lattice graphics and will need the lattice package to be installed. The xyplot produces bivariate scatterplots of numeric quantities. The *auto.key* argument is used to automatically produce a legend.

```
#Lattice Graphics
>install.packages(''lattice'')
>library(lattice)
>xyplot(Sepal.Width ~ Sepal.Length, data=iris, groups=
   Species, auto.key=list(corner=c(0,0), x=0, y=0.85, cex
   =1.5), cex=1.5, scales=list(cex=1.5))
```

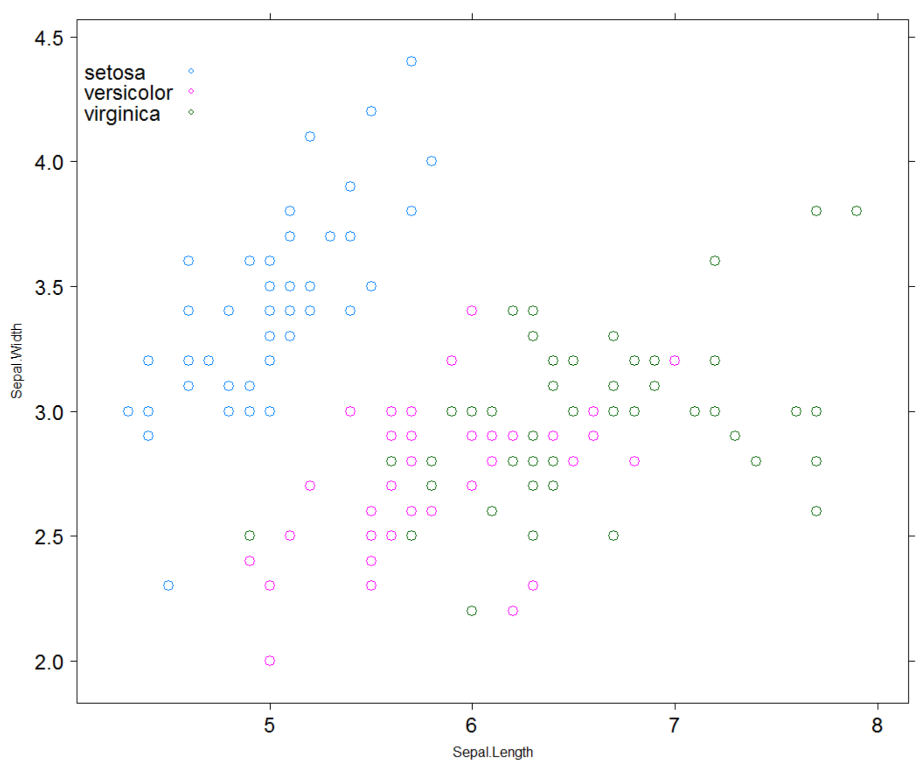

Figure A.6: Lattice Scatterplot

As can be seen form the graph above, the distinction in the species due to colors, provides insight into the correlation and relationships between Sepal Width and Length segmented by Species. We did not have this clarity in prior graph. **Segmentation** is very important and adds another layer of understanding to the data.

A.6 In-class Exercise 1

Compare and contrast 2D scatterplots for the iris dataset. Summarize your findings. Create scatterplots of Petal Length versus Petal Width using the *plot* and *xyplot* function.

Sample Solutions:

```
#Core Graphics
#plot Petal Length versus Petal Width
>plot(Petal.Length ~ Petal.Width, data=iris, main="Petal
   Width versus Petal Length")
```

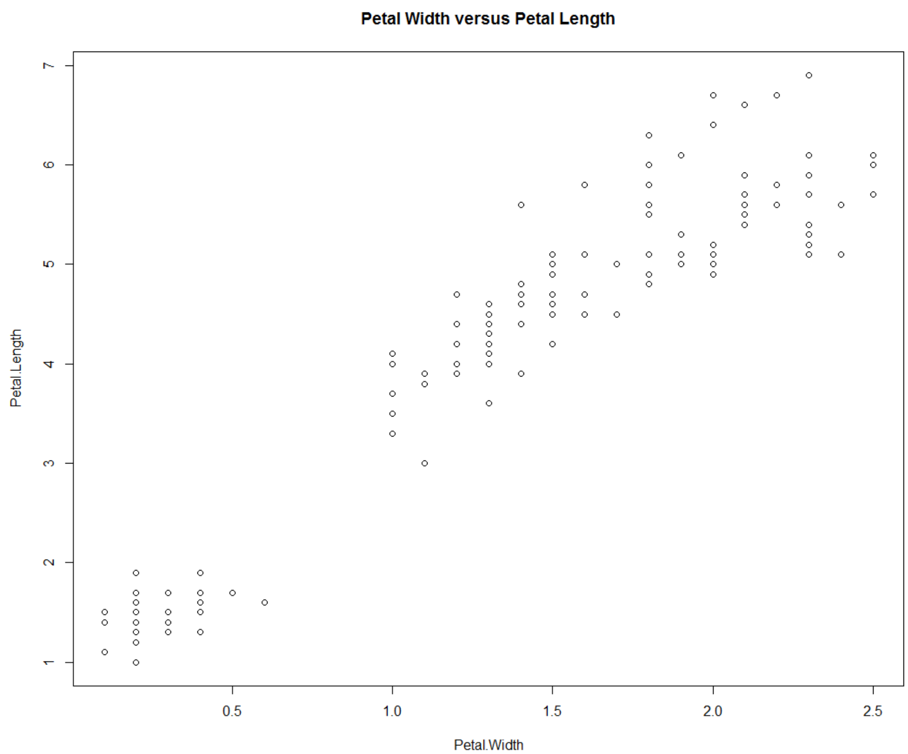

Figure A.7: plot

```
#Core Graphics
#Regression line on plot
>plot(Petal.Length ~ Petal.Width, data=iris, main="Petal
   Width versus Petal Length")
>abline(lm(Petal.Length ~ Petal.Width,data=iris),col="red"
   )
```

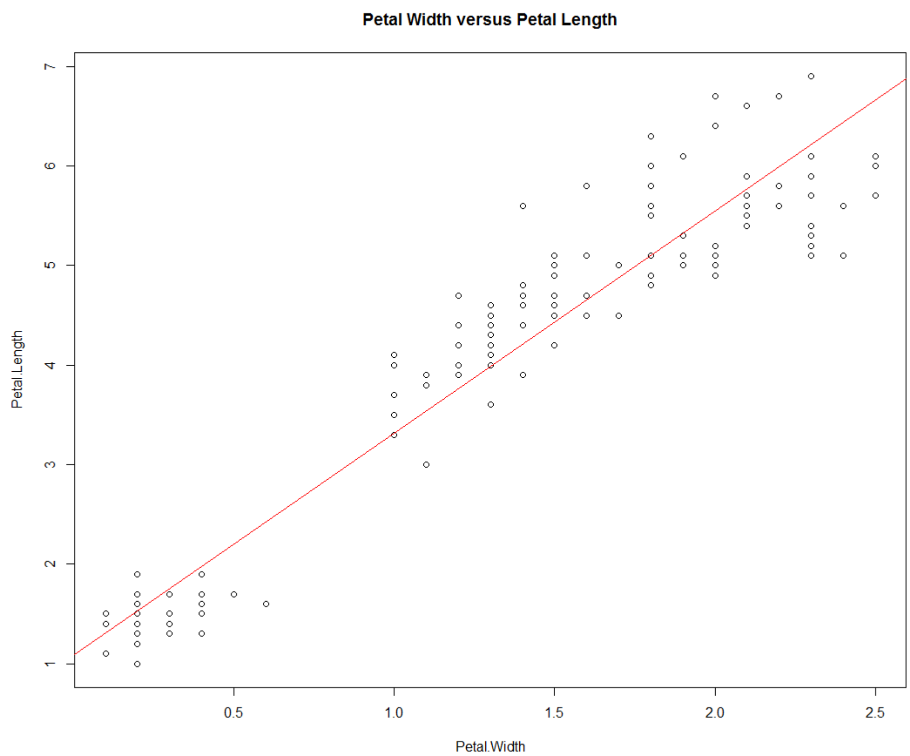

Figure A.8: plot with regression line

```
#Find correlation coefficient between variables
>cor(iris$Petal.Width,iris$Petal.Length)
[1] 0.96
# The correlation is close to 1. This indicates a strong
   positive linear relationship.
```

```
#Lattice Graphics
>xyplot(Petal.Width ~ Petal.Length, data=iris, groups=
   Species, auto.key=list(corner=c(0,0), x=0, y=0.85, cex
   =1.5), cex=1.5, scales=list(cex=1.5))
>xyplot(Petal.Width ~ Petal.Length, data=iris, groups=
   Species, auto.key=TRUE)
```

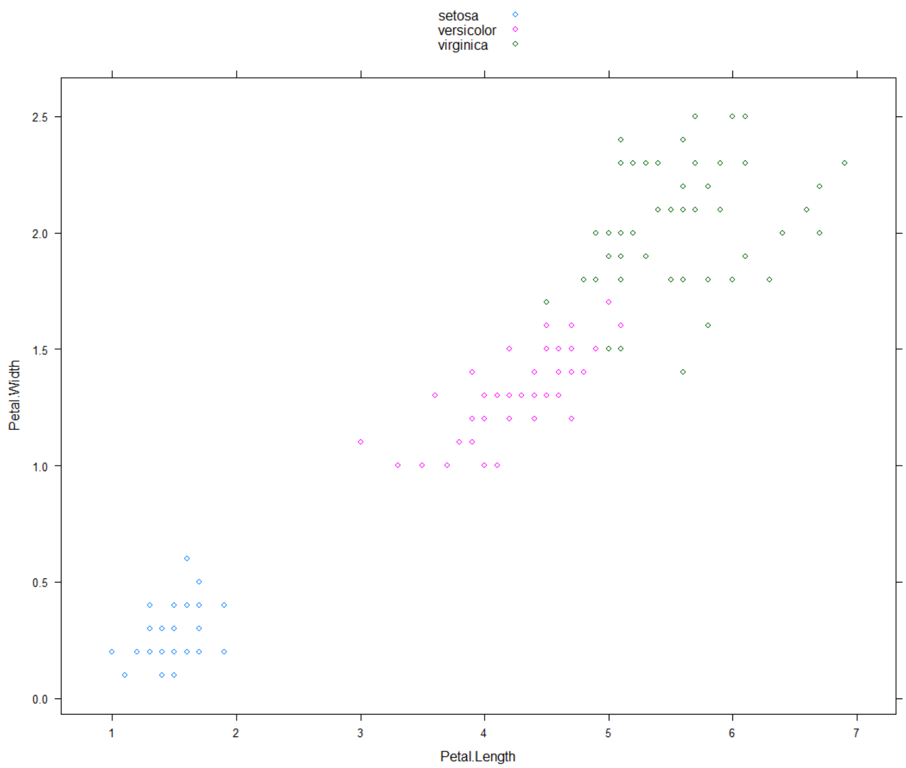

Figure A.9: plot with regression line

Again segmentation by secies shows us strong correlation between Petal Width and Petal Length.

A.7 R core Histogram function

Definition: A**histogram** is a graphical display of numerical data using bars of different heights to represent the frequency of that variable. It shows the frequency distribution of a quantitative variable. R has a built-in, core histogram function, hist().

- Create a histogram of Petal Width
- Color the graph blue
- Breaks =12

```
#Core Graphics
>hist(iris$Petal.Length)
>hist(iris$Petal.Length, breaks=12,col=''blue'')
```

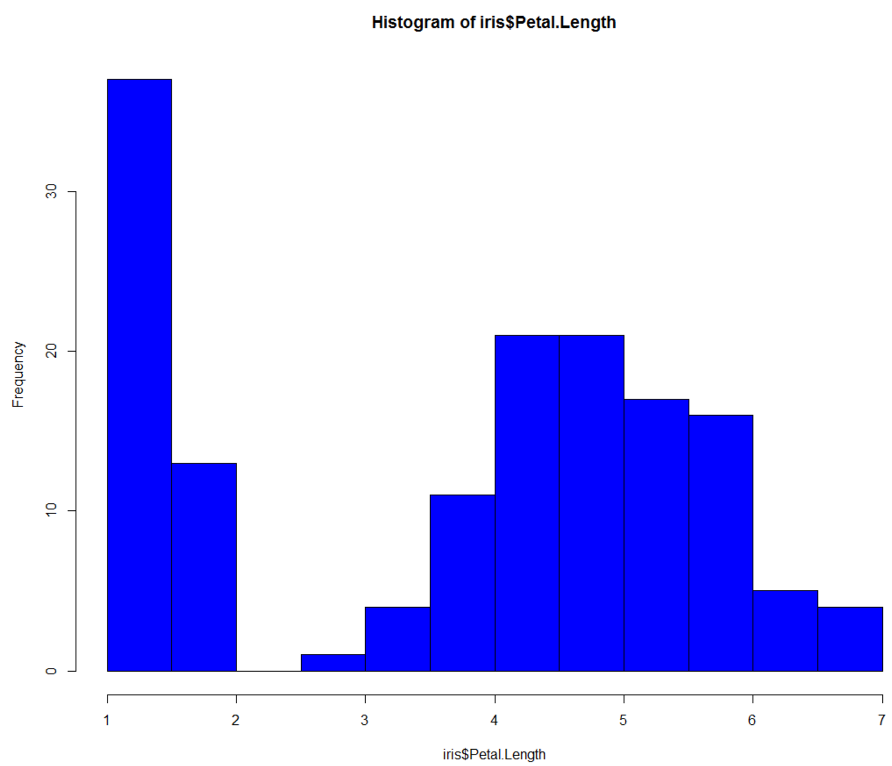

Figure A.10: Histogram

Questions Try changing the number of bins/breaks to 12, 15, 20. What do you notice?

A.8 Lattice Histogram function

We can also create better histograms using the histogram function in lattice package.

```
#Lattice Graphics
>histogram(iris$Petal.Length, breaks=12, type="count",
   main="Histogram")
```

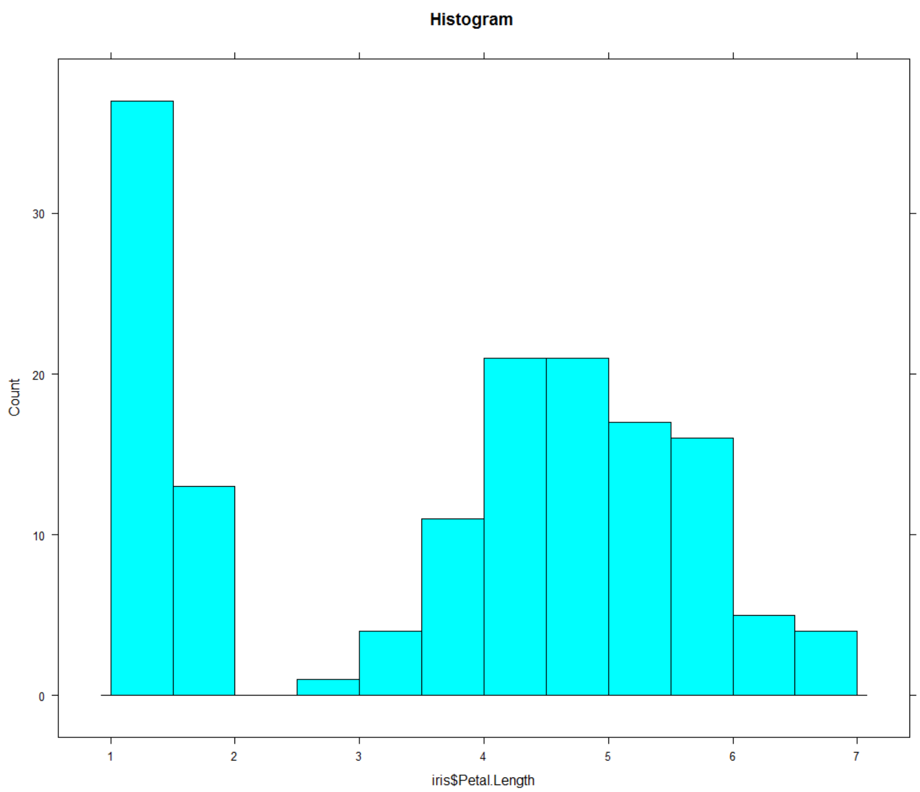

Figure A.11: Lattice Histogram

A.9 Density Plot Fucntion

Definition: A density plot is a function that describes the relative likelihood for this random variable to take on a given value for a numeric variable. The probability of the random variable falling within a particular range of values is given by the integral of this variable's density over that range i.e. it is given by the area under the density function. The probability density function is nonnegative everywhere, and its integral over the entire space is equal to one. It can be thought of as histogram with an infinite number of bins. It is a variation on the histogram and estimates densities for data. The idea of the total area under the curve is useful when dealing with density plots. The total area under the curve of a density plot should equal 1. Note: A density plot cannot be created is there are missing values.

```
#Lattice Graphics
>densityplot(iris$Petal.Length)
```

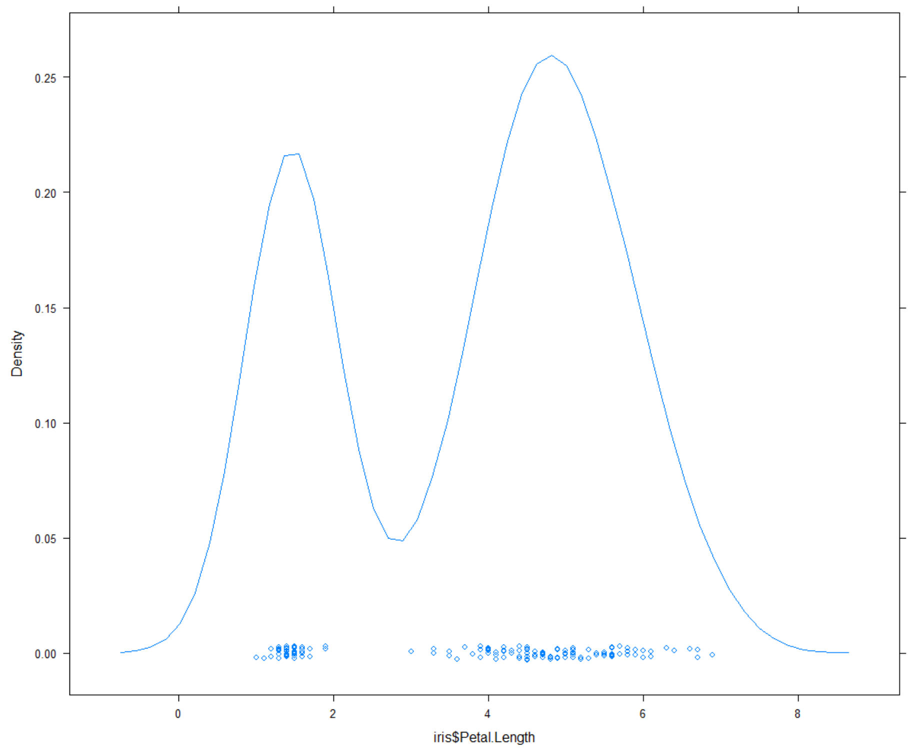

Figure A.12: Density Plot of Petal Length

Notice there are points at the bottom of the graph. These represent the actual data points. The density of the points are related to the peaks in the graph. A higher density of points leads to a peak. To remove the points from the graph, add an argument called *plot.point=F* in the code.

A.9.1 Multiple Density Plot

Definition A multiple density plot is used when we would like to overlay multiple graphs.
Create a multiple density plot of Petal Width segmented by Species.

```
#Lattice Graphics
>densityplot(~ Petal.Width, data=iris, groups=Species,
   plot.points=F, xlab=list(label="Kernel Density of Petal
    Width", fontsize=20), ylab="", main=list(label="
   Density of Petal Width by Species", fontsize=24), auto.
   key=list(corner=c(0,0), x=0.4, y=0.8, cex=2), scales=
   list(cex=1.5))
```

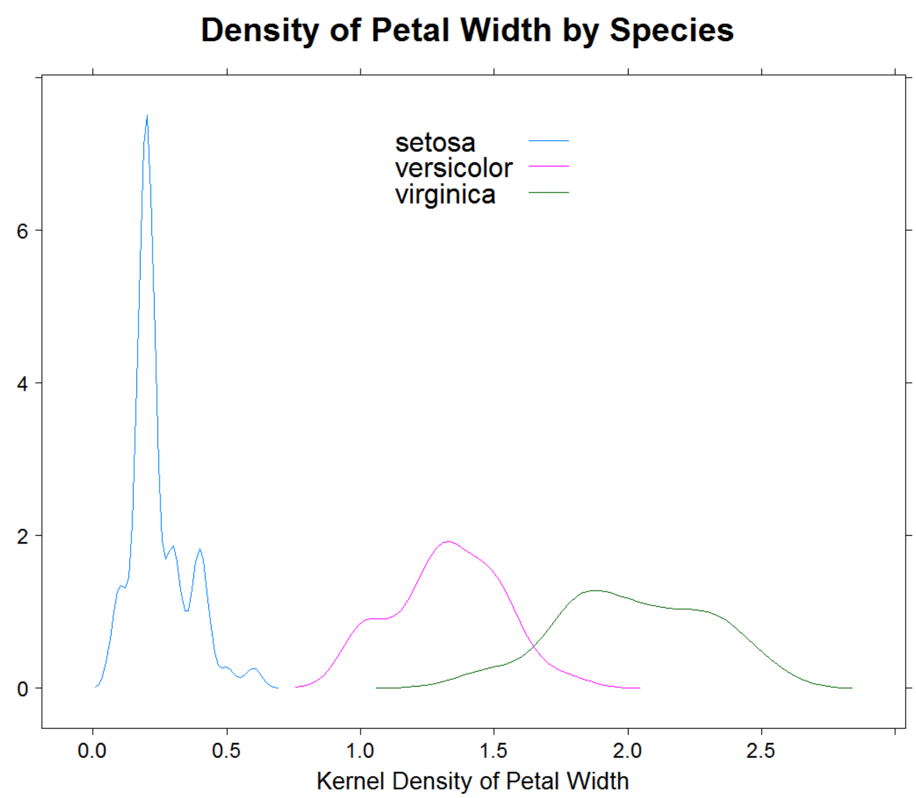

Figure A.13: Multiple Density Plot

Questions

1. Is the area under the curve equal to 1 for each individual species or the whole graph?
2. Which has a higher probability of having a petal width of 1.2? Versicolor or Virginica? Why?

A.10 In-class Exercise 2

Using the **mtcars** dataset, predict mpg based on other columns. Create at least 2 different plots illustrating useful relationships in data and summarize your findings. Goal: Predict mpg based on other columns.

The mtcars dataset was extracted from the 1947 Motor Trend US magazine and comprises fuel consumption and ten aspects of automobile design and performance for 32 automobiles. In other words the dimensions of this dataset are 32 by 10. This is common dataset used to perform regression analysis on.

Features of this dataset are:

- mpg= miles per gallon
- cyl = number fo cylinders
- disp= displacement
- drat = rear axle ratio
- hp= gross horse power
- wt = weight (1000lbs)
- qsec = 1/4mile time
- vs= V/S
- am = Transmission(0=automatic, 1= manual)
- gear = number of forward gears
- carb = number of carburetors

Load the *mtcars* dataset

```
>data(mtcars)
```

Look at the first 6 rows of this dataset

```
>head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Figure A.14: Mtcars Dataset

Create a boxplot of mpg versus cyl.

```
>boxplot(mpg ~ cyl, data= mtcars, main="Car Mileage Data",
    xlab="Number of Cylinders" ,ylab= "mpg", col = c("blue
  ", "yellow", "red"))
```

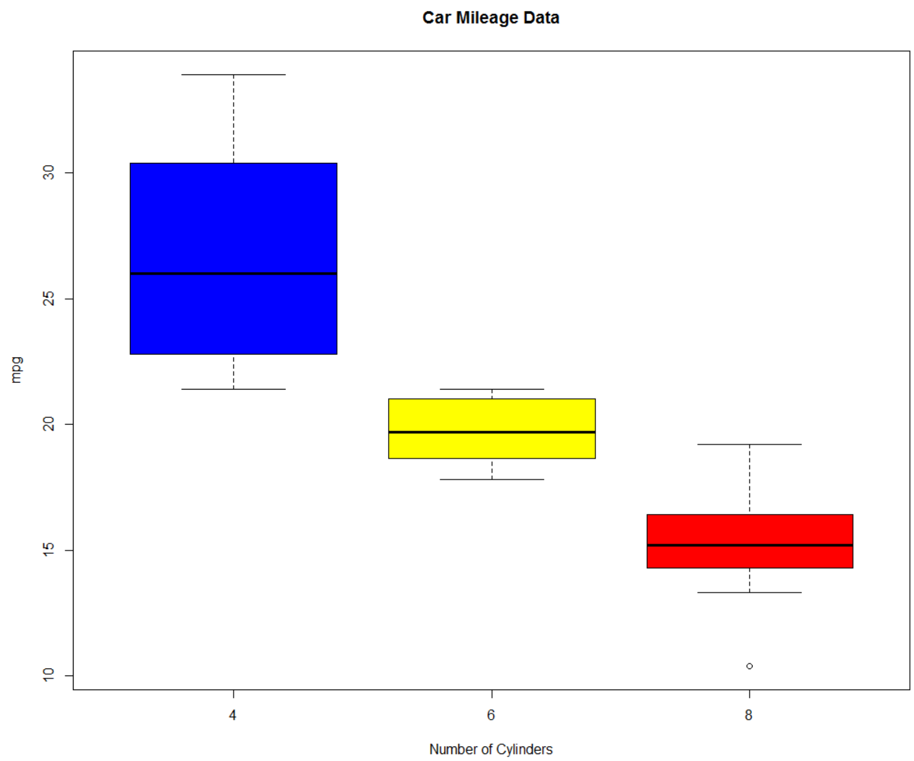

Figure A.15: Boxplot of mpg vs cyl

Sample solutions:

```
#Lattice Graphics
>densityplot(~ mpg, data=mtcars, groups=cyl, plot.points=F, auto.key=list(columns=3, title="Cylinders"))
```

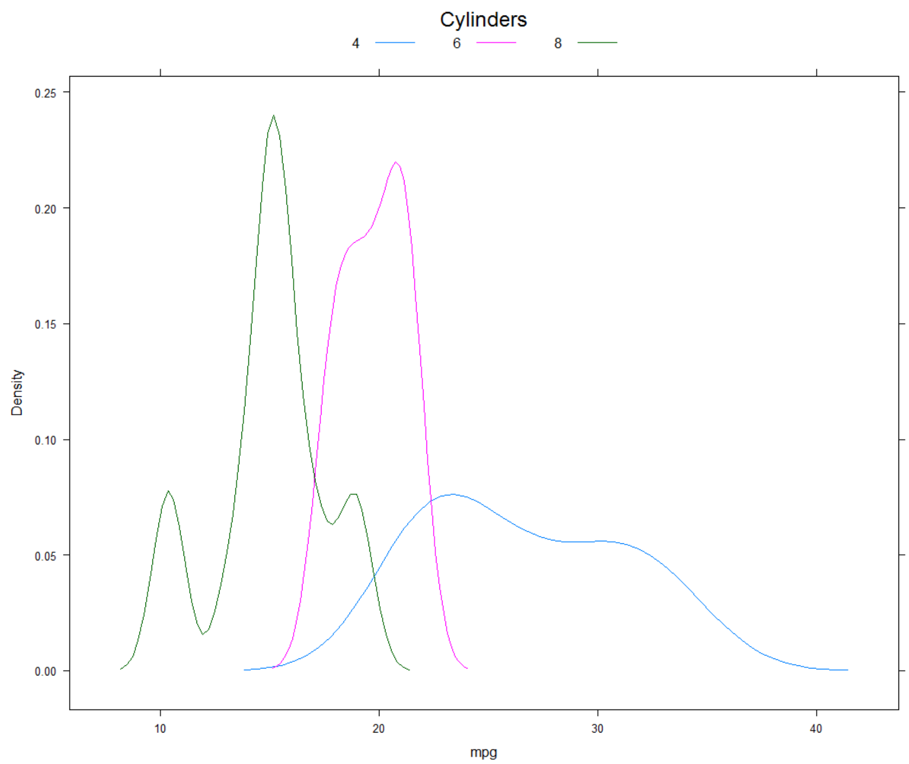

Figure A.16: Density plot of mpg vs cyl

```
#Core Graphics
>plot(mpg ~ disp, data=mtcars)
>abline(lm(mpg ~ disp, data=mtcars), col="red")
```

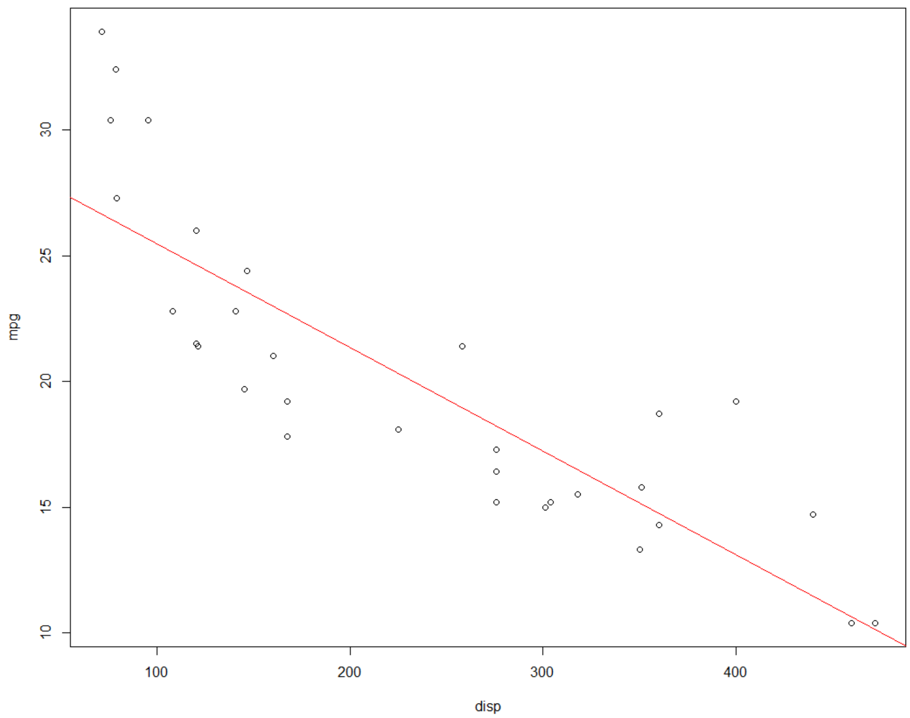

Figure A.17: Plot of mpg vs displ

```
#Correlation between mpg and disp
>cor(mtcars$mpg,mtcars$disp)
[1] -0.8475514
#Strong negative correlation between mpg and disp
```

Futher questions Do you think cars with manual transmission have more efficient fuel management compared to automatic transmission? How can you test this theory? Use your knowledge of the plot, xyplot, boxplot, density plot, histograms, density plot, scatterplot matrix functions (with and without segmentation) to create graphs for this dataset.

A.11 R Core Scatterplot Matrix

Definition: A Scatterplot matrix is a scatterplot on all the numeric variable of a dataset.

Create a scatterplot matrix on iris dataset

```
#Core Graphics
>pairs(~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data=iris, main="Simple Scatter Matrix")
```

Figure A.18: Scatterplot Matrix

A.12 Lattice Scatterplot Matrix

Lattice scatterplot matrix is a better scatterplot.

Create a lattice scatterplot matrix on iris dataset

```
#Lattics Graphics
>splom(iris[1:4], groups=iris$Species, panel=panel.superpose,
   key=list(title="Three Flower Types", columns=3,
      points=list(pch=super.sym$pch[1:3], col=super.sym$col[1:3]),       text=list(c("Setosa","Versicolor","Verginica"))))
# Cleaner version
>splom(iris[1:4], groups=iris$Species, auto.key=TRUE)
```

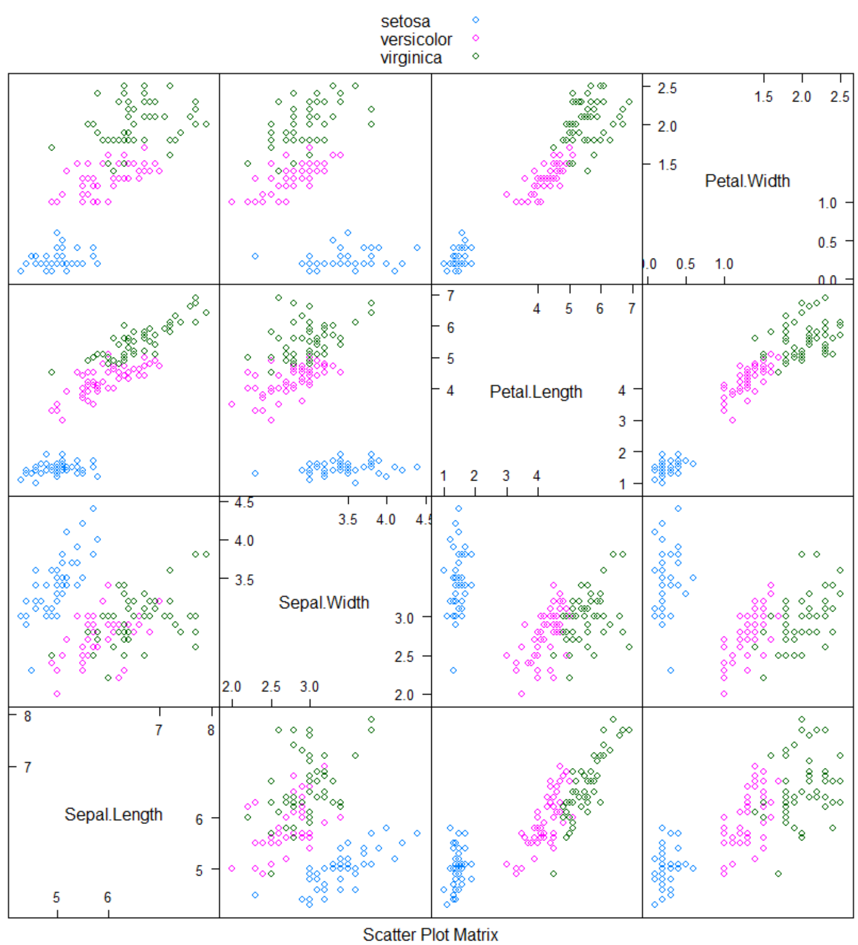

Figure A.19: Lattice scatterplot matrix

A.12.1 Global Graphical Settings

You can modify global setting when generating your /images reports using:

```
>my.theme = trellis.par.get()
>names(my.theme)

>show.settings()
>my.theme$fontsize$text=20
```

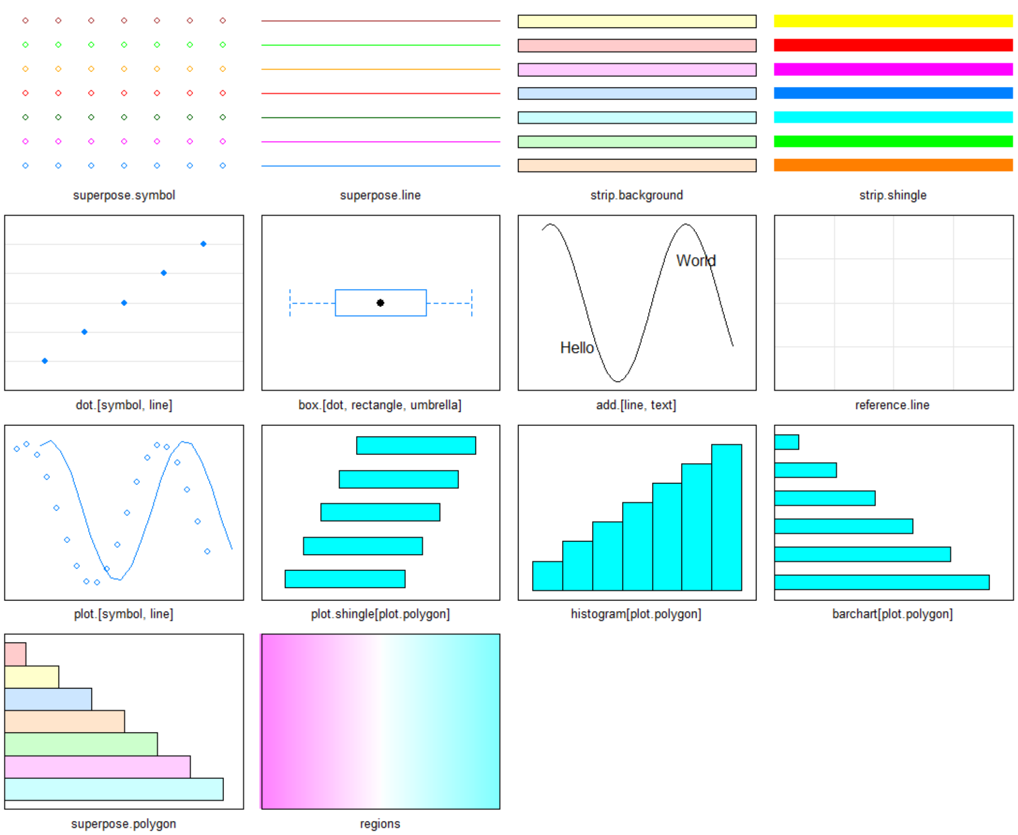

Figure A.20: Theme Settings

A.12.2 Scatterplot Matrix with segmentation

```
# Getting settings for legend
super.sym <- trellis.par.get("superpose.symbol")
splom(iris[1:4], groups=iris$Species, panel=panel.
   superpose, key=list(title="Three Flower Types", columns
   =3, points=list(pch=super.sym$pch[1:3],  col=super.sym$
   col[1:3]),
text=list(c("Setosa","Versicolor","Verginica"))))
```

A.13 GGally Graphics package

You can also enhance your graphs by using the GGally package. This package allows us to see all the graphs we studied - density, boxplot, histogram with correlation coefficients all in one. It is an extrmeely powerful package, but time consuming for larger datasets.

```
>install.packages(''GGally'')
>library(GGally)
>ggpairs(iris, ggplot2::aes(color=Species))
```

Figure A.21: ggpairs

A.14 GGplot2 Graphics

The ggplots2 package is the most recommended package for creating graphs. ggplot2 is a plotting system for R, based on grammar of graphics, which tries to combine lattice and core R graphs and give you the best of both worlds. It takes care of the hassle involved in adding arguments for creating legends, etc. It is an extremely powerful tool that helps with creating multi-layered graphs. For more resources
http://ggplot2.org/ More information about diamonds(cut, clarity, carat, color, dept, table) can be found at':
http://www.diamondse.info/

Load the *diamonds* dataset, found within the ggplot2 package.

```
#GGplot2
>install.packages(''ggplot2'')
>library(ggplot2)
>data(diamonds)
```

Look at the first 6 rows of the diamonds dataset.

```
>head(diamonds)
```

```
  carat       cut color clarity depth table price    x    y    z
1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
```

Figure A.22: Diamonds dataset

Look at structure of this dataset

```
>str(diamonds)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame':      53940 obs. of  10 variables:
 $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
 $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
 $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

Figure A.23: Diamonds dataset

The ggplot function is built on this idea of layering. You have a base function that acts as a canvas, laying the groundwork to draw graphs on. As you add more layers, the graphs build on top of each other. The **aes** represents the **aesthetics** of the graph.

First Layer

```
ggplot(diamonds, aes())
```

Add a scatterplot layer to orginal layer

```
ggplot(diamonds, aes()) + geom_point()
```

Aesthetics supplied to ggplot() are used as defaults for every layer i.e. global setting, whereas aesthetics applied to a specific layer provide settings for only that layer i.e. local setting.

```
#Histogram of diamonds with x=carat, global aes
>ggplot(diamonds, aes(x=carat))+ geom_histogram()
```

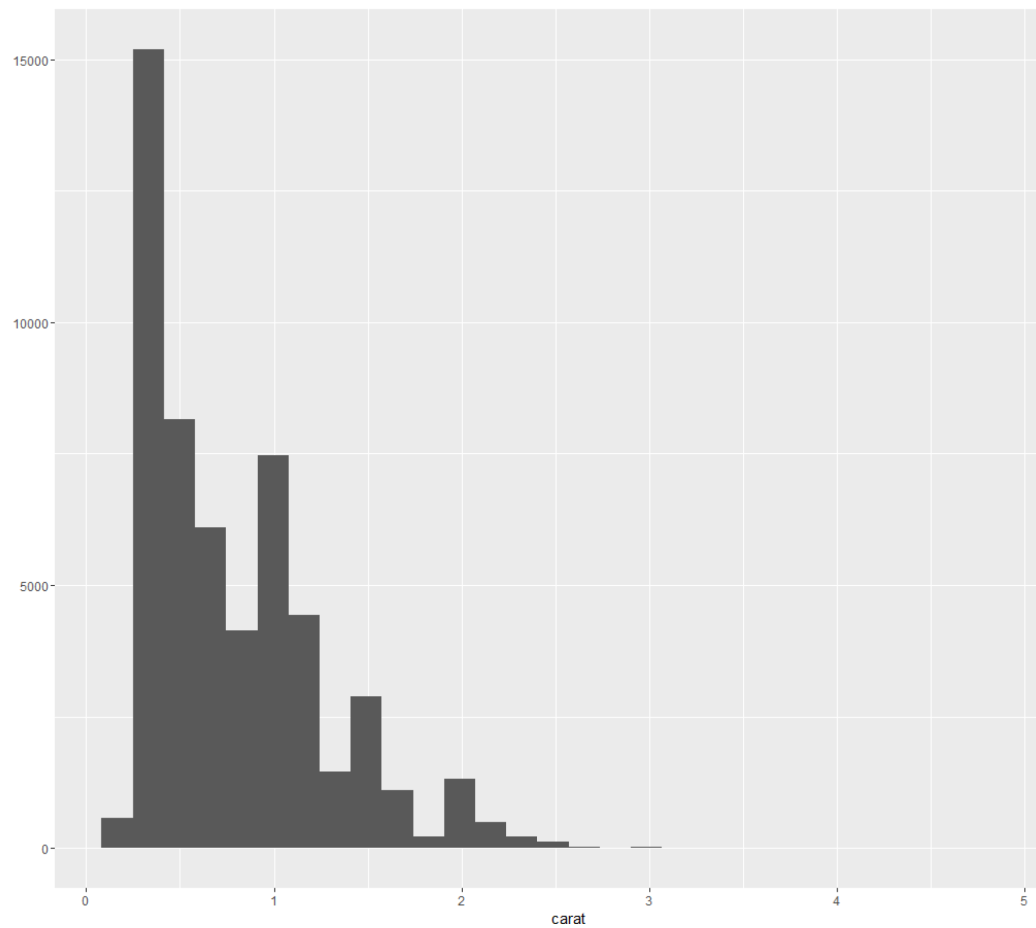

Figure A.24: histogram

```
#Histogram of diamonds with x=carat, local aes
>ggplot(diamonds) + geom_density(aes(x=carat),fill="gray50
  ")
```

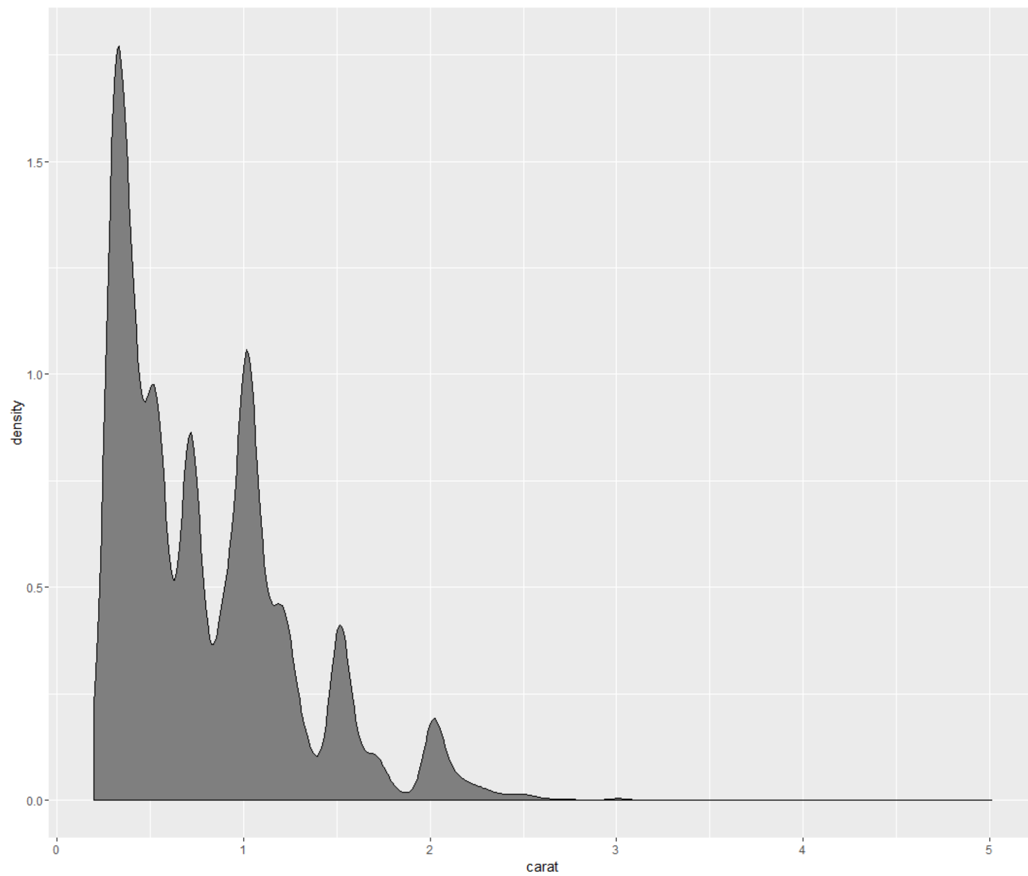

Figure A.25: density plot

A.14.1 Create a scatterplot of diamonds

Assign the ggplot base canvas to a variable **g**. Then add layers.

```
>g <- ggplot(diamonds, aes(x=carat, y=price))
>g + geom_point(aes(color=color))
```

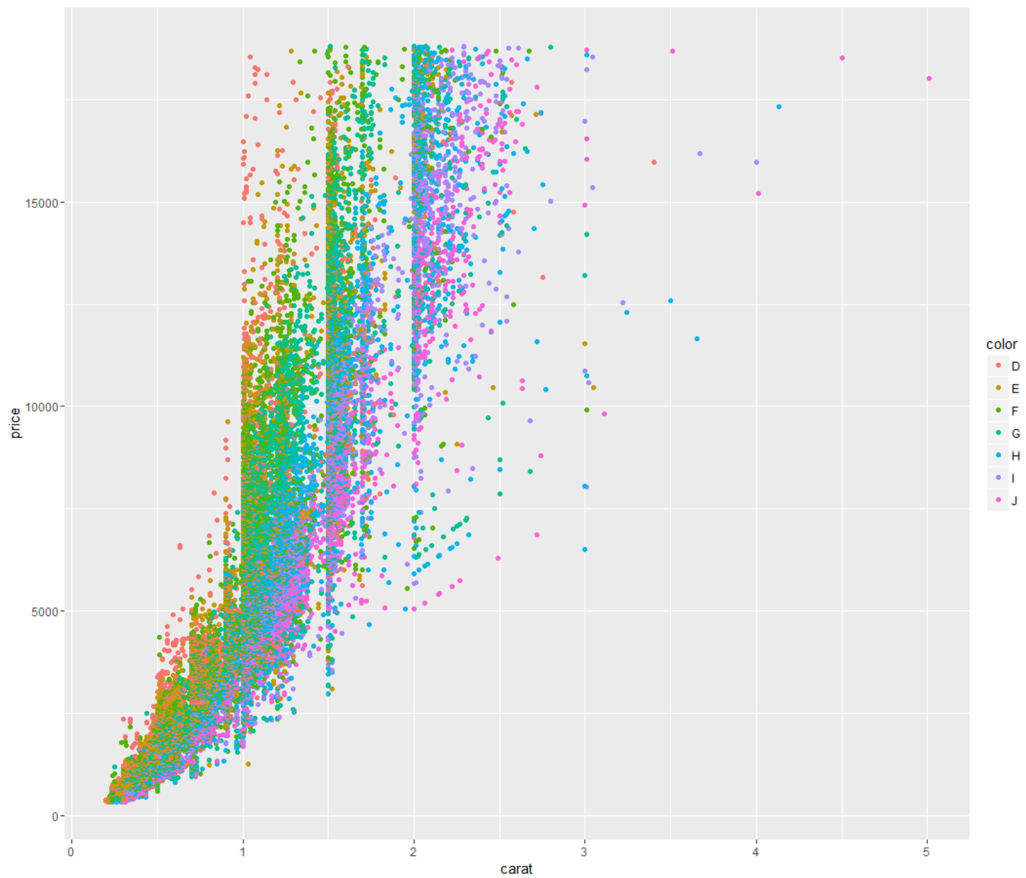

Figure A.26: scatterplot with ggplot2

Questions

1. What do you notice about this graph?
2. How does the clarity change as the carat increases? What effect does this have on the price?

Note: You can see some sort of banding or segmenting occurring around specific carat values.

A.14.2 Separate segments with facet wrap

Graphs can be placed next to each other, wrapping with a certain number of columns or rows. The label for each plot will be at the top of the plot.

```
>g <- ggplot(diamonds, aes(x=carat, y=price))
>g + geom_point(aes(color=color)) + facet_wrap(~ color)
```

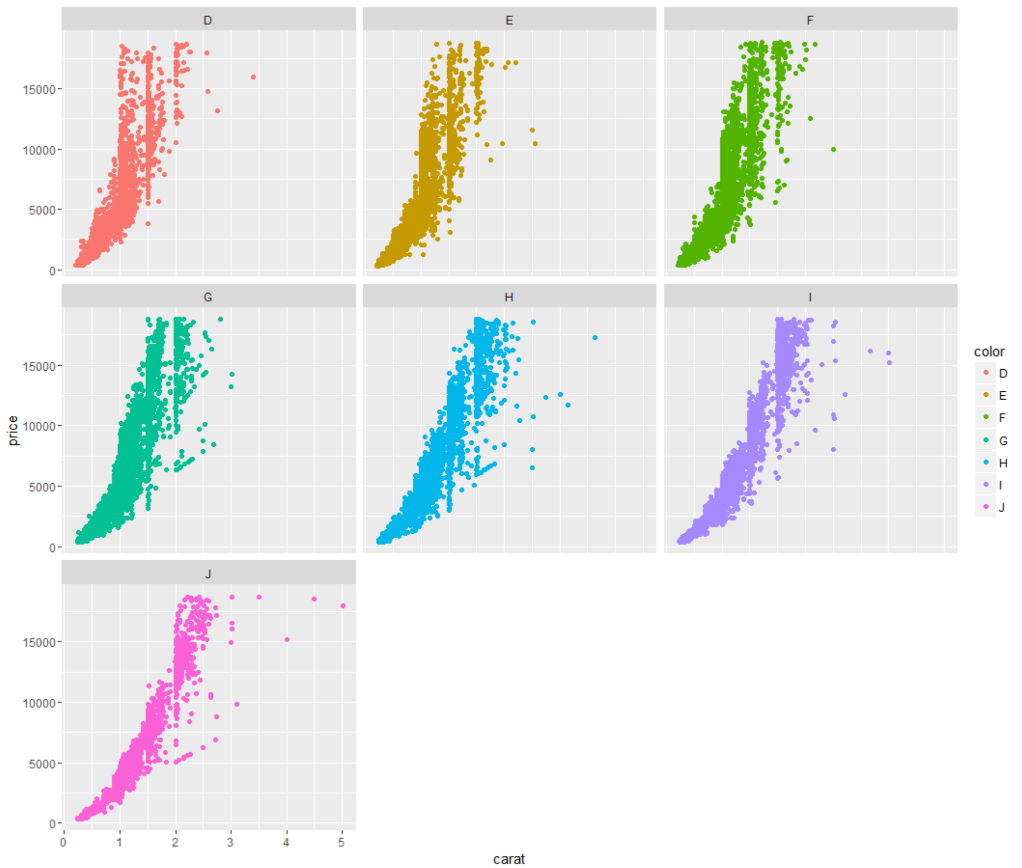

Figure A.27: facet wrap

A.14.3 Segmenting segments with facet grid

The data can be split up by one or two variables that vary on the horizontal and/or vertical direction.

```
>g <- ggplot(diamonds, aes(x=carat, y=price))
>g + geom_point(aes(color=clarity)) + facet_grid(cut ~
   clarity)
```

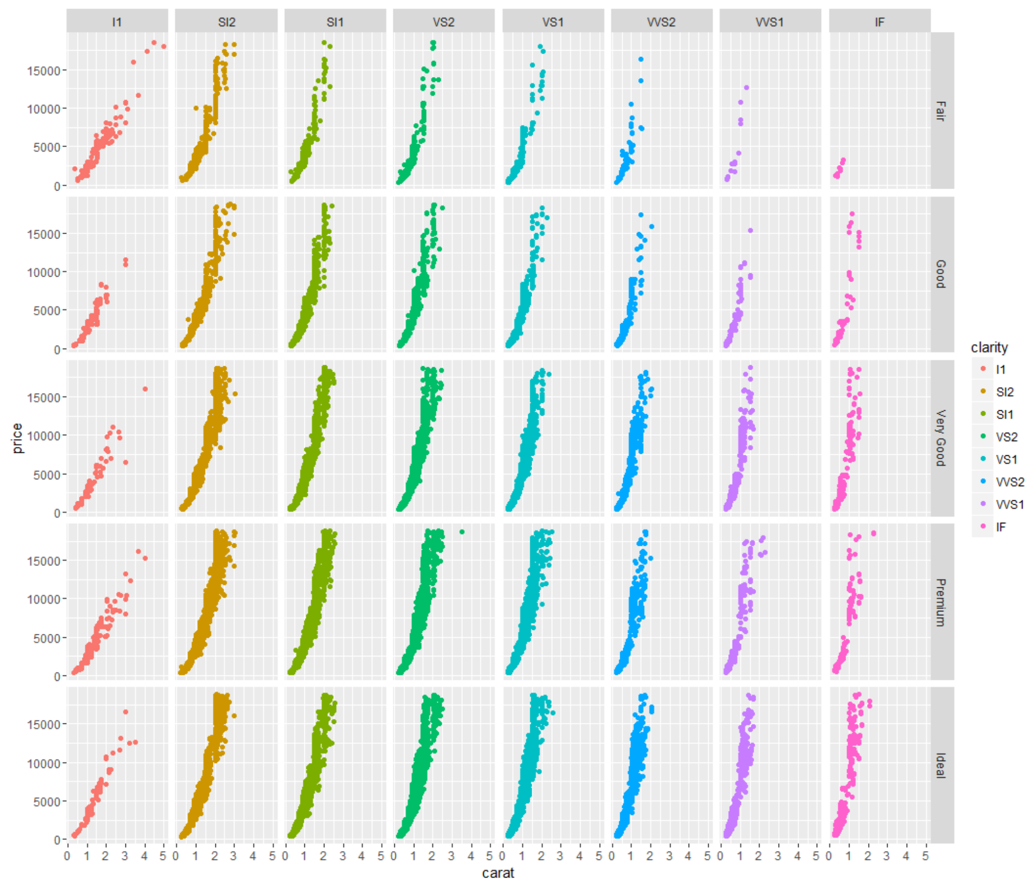

Figure A.28: facet grid

A.14.4 Different graphs with different variables

- When you are comparing two **numeric** variables, use a **scatterplot**
- When you are comparing a **categorical** and **numerical variable**, use a **boxplot**
- When you are comparing two **categorical** variables, use a **table** or bf pie chart

A.15 Storytelling with titanic

Refer to Chapter 3 to load the titanic dataset and to view the summary.

A.16 In-class Exercise 3

1. Create 2 boxplots of Age segmented by Gender
2. Create 2 boxplots of Age segmented by Survived
3. Create a density plot of Age, and make sure to omit missing values by using na.omit

Sample Solutions:

```
#Comparing age (numeric), with gender (categorical)
>boxplot(Age ~ Sex, data=titanic, col=c("red","green"),
  main="Age Distribution by Gender")
```

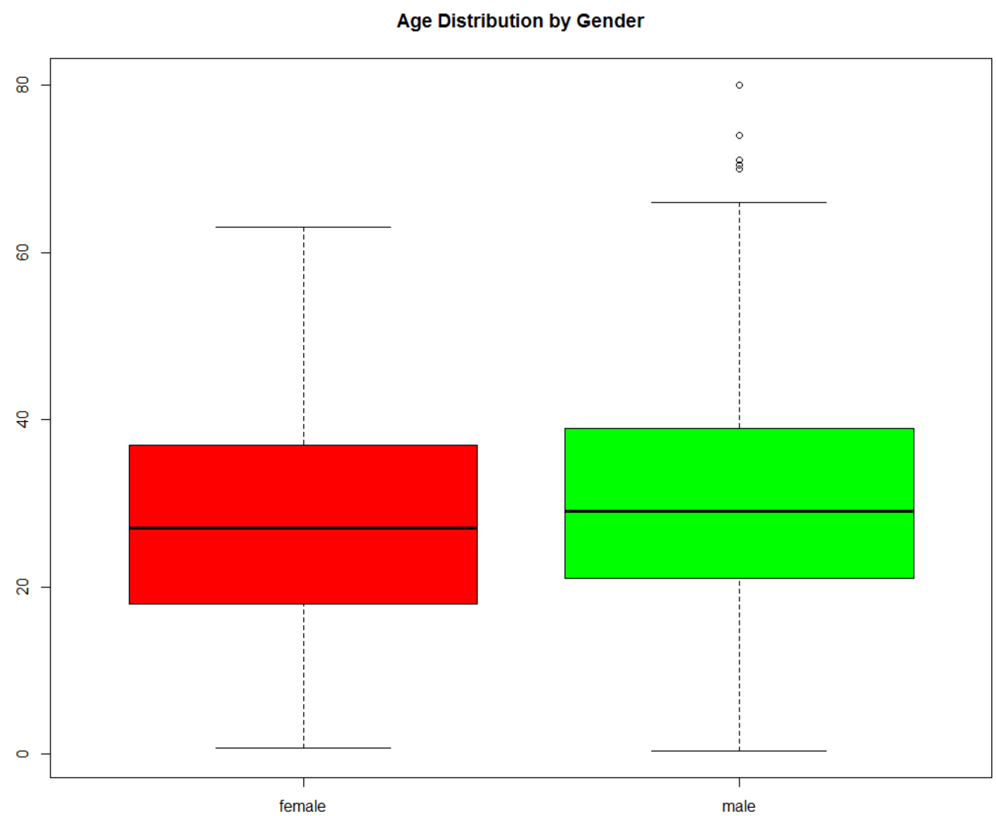

Figure A.29: Age segmented by Gender

```
#Comparing age (numeric), with gender (categorical)
>boxplot(Age ~ Survived, data=titanic, col=c("red","green"
   ),main="Age Distribution by Survived")
```

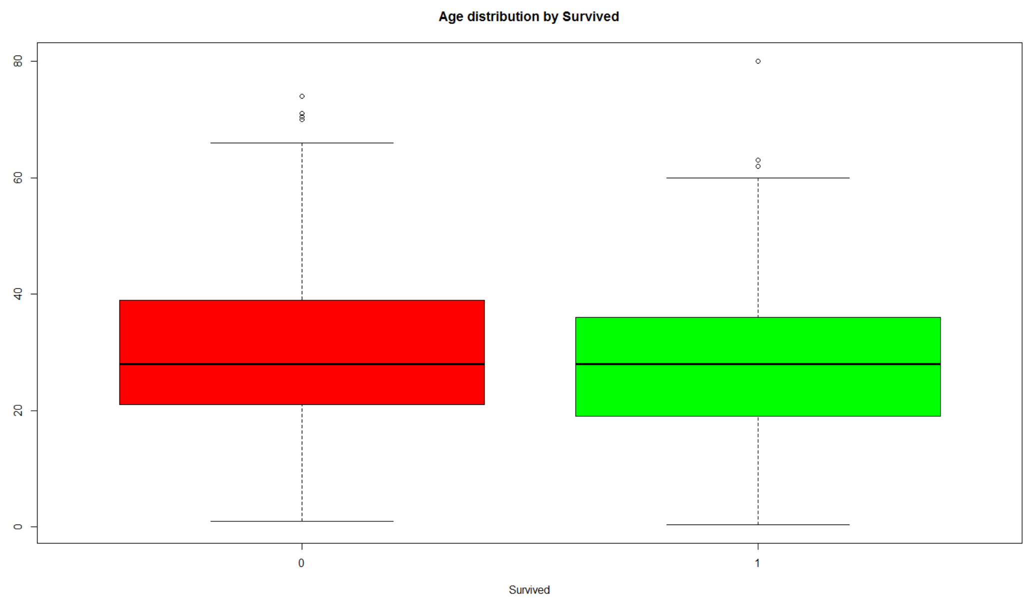

Figure A.30: Age segmented by Survived

```
#Density plot of age
#density(titanic$Age)    #NAs prevent this
d <- density(na.omit(titanic$Age))
plot(d)
polygon(d,border="green")
```

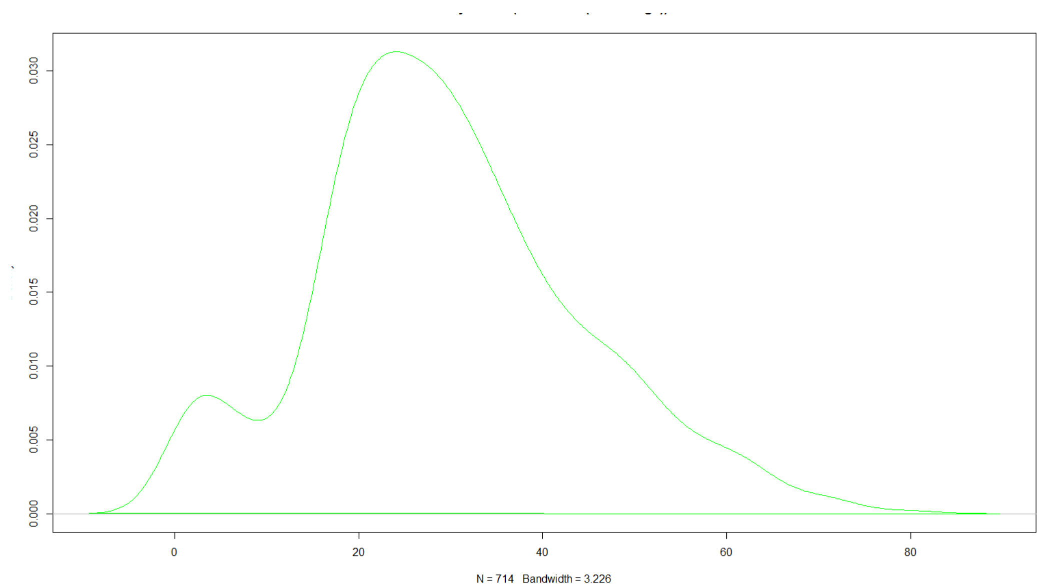

Figure A.31: Density plot of Ages