



# Exploratory Data Analysis in R

S. Morteza Najibi, Shiraz University

April 22, 2019

# Graphics in R



- ▶ Exploratory Data Analysis (EDA)
- ▶ Powerful environment for visualizing scientific data
- ▶ Fully programmable
- ▶ An example of Exploratory Data Analysis will be performed with the Iris data set.

# Important high-level plotting functions



- ▶ `plot`: generic x-y plotting
  - ▶ `barplot`: bar plots
  - ▶ `boxplot`: box-and-whisker plot
  - ▶ `hist`: histograms
  - ▶ `pie`: pie charts
  - ▶ `dotchart`: cleveland dot plots
  - ▶ `image`, `heatmap`, `contour`, `persp`: functions to generate image-like plots
  - ▶ `qqnorm`, `qqline`, `qqplot`: distribution comparison plots
  - ▶ `pairs`, `coplot`, `matplot`: display of multivariant data
- Help on these functions

# Iris Data



The iris data set is available at the UCI Machine Learning Repository.

The data set was created by statistician Douglas Fisher. The data consists of 3 classes of flower types:

- ▶ *setosa*
- ▶ *virginica*
- ▶ *versicolor*

# The Iris Attributes



The data has 4 attributes:

- ▷ sepal width
- ▷ sepal length
- ▷ petal width
- ▷ petal length

# Load Data



The data needs to be loaded into R.

```
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
```

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

# Summary Statistics



The summary function can be used to assess the statistical properties of each attribute.

```
summary(iris[, -5])
```

| ## | 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   |

other statistics may be calculated as described in the introduction to R examples.

A decorative background pattern consisting of a grid of triangles and squares. The pattern is composed of light gray lines forming a series of interconnected geometric shapes. The triangles are of various sizes and are arranged in a way that creates a sense of depth and complexity. The squares are also of various sizes and are interspersed among the triangles. The overall effect is a subtle, textured background that adds visual interest to the slide.

# Visualizations

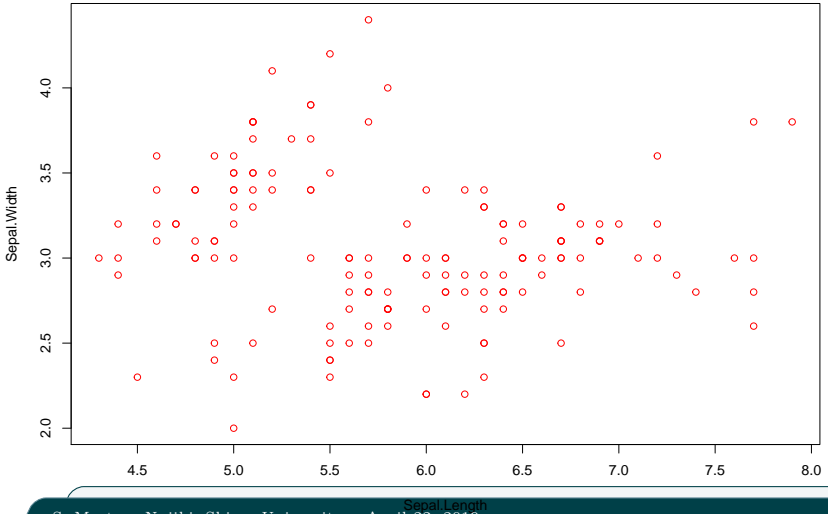




# Plot

► plot x versus y

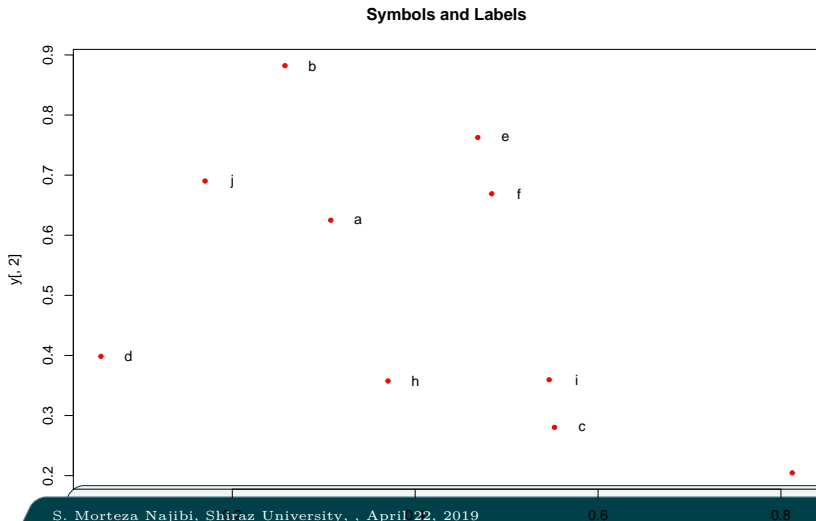
```
plot(iris[,1], iris[,2],xlab=names(iris)[1],ylab=names(iris)[2],col=2)
```



# Some more techniques in plot



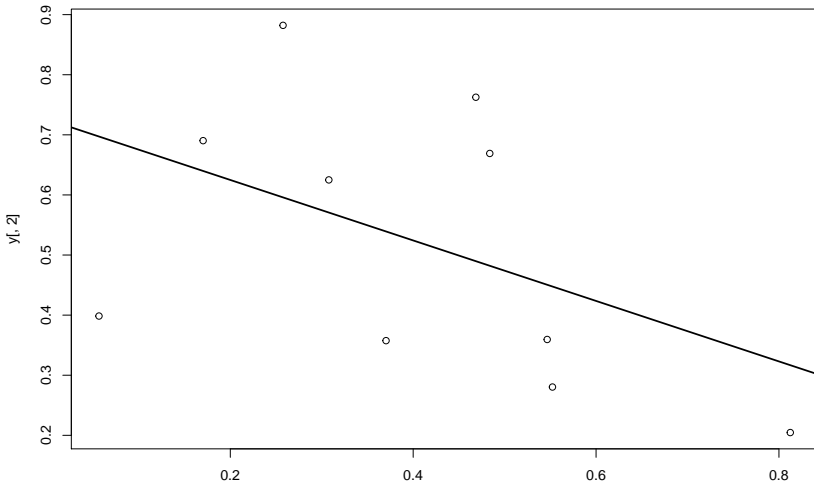
```
set.seed(100)
y <- matrix(runif(30), ncol=3, dimnames=list(letters[1:10], LETTERS[1:3]))
plot(y[,1], y[,2], pch=20, col="red", main="Symbols and Labels")
text(y[,1]+0.03, y[,2], rownames(y))
```





## Add a regression line

```
plot(y[,1], y[,2])
myline <- lm(y[,2]~y[,1]);
abline(myline, lwd=2)
```

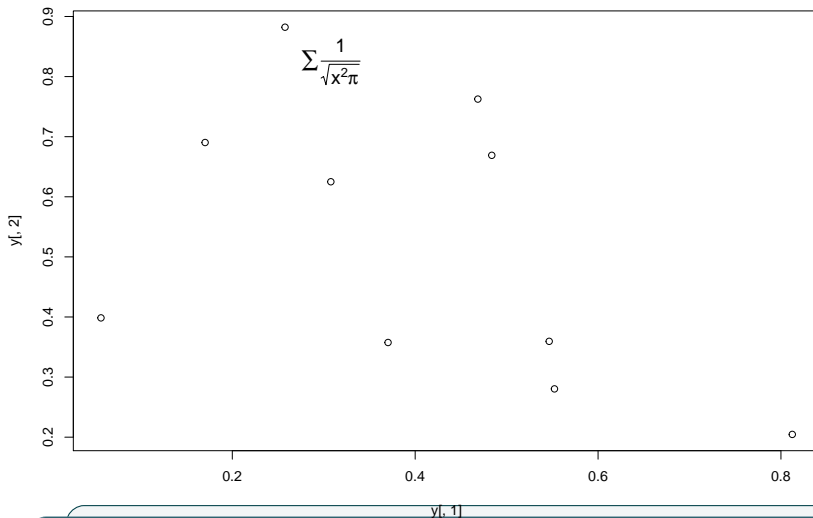




# Add a mathematical expression to a plot



```
plot(y[,1], y[,2])  
text(y[1,1], y[1,2]+.2,expression(sum(frac(1,sqrt(x^2*pi)))), cex=1.3)
```

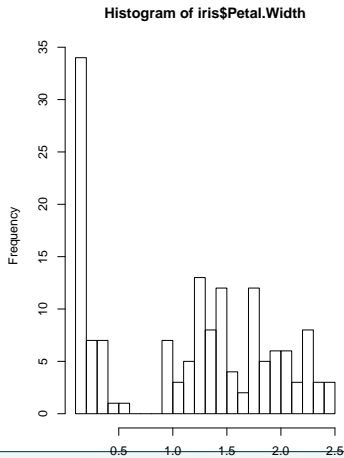
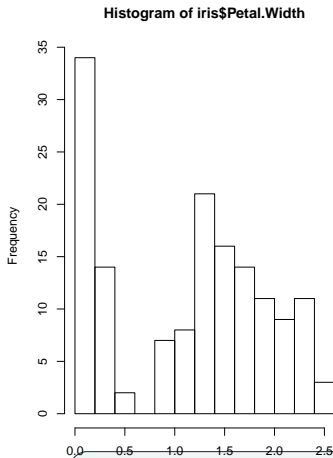


# Histograms



► Set the number of breaks to 9, therefore, there will be 10 equal sized bins.

```
par(mfrow=c(1,2))  
hist(iris$Petal.Width, breaks=9)  
hist(iris$Petal.Width, breaks=19)
```



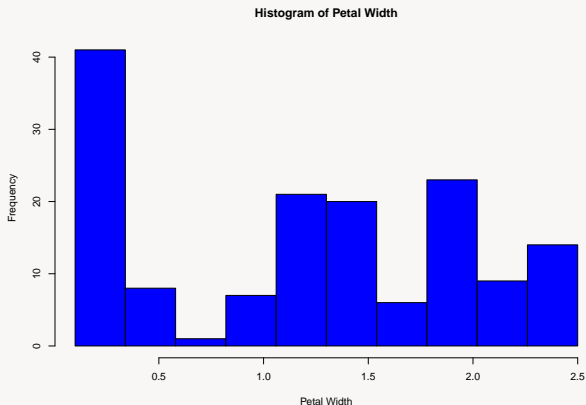


## some notes



Note, the number of breaks can be a single number, vector, function to evaluate points or character string.

```
b <- seq(min(iris$Petal.Width), max(iris$Petal.Width),  
         length=11)  
hist(iris$Petal.Width, breaks=b, col="blue", xlab="Petal Width",  
     main="Histogram of Petal Width")
```

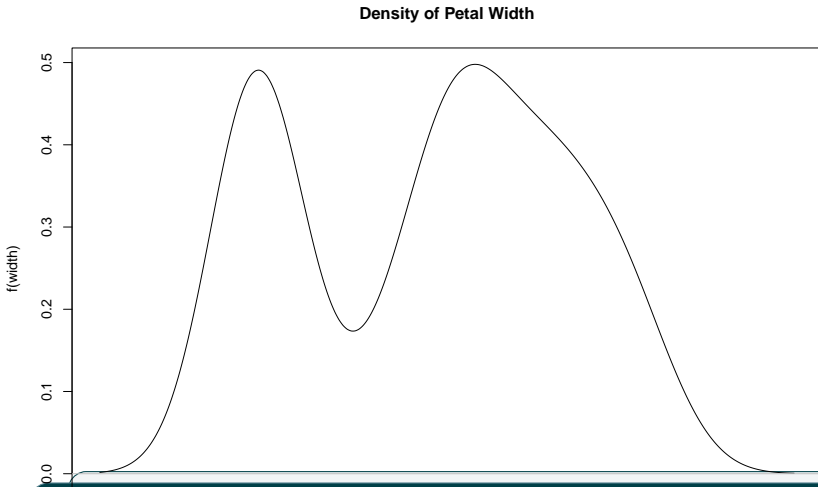




# Density Plots

Density plots can be viewed as smoothed versions of a histogram. We can estimate the density using R's **density** function

```
dens.pw = density(iris$Petal.Width)
plot(dens.pw, ylab = "f(width)", xlab = "width", main= "Density of Petal Width")
```





# Density plot cont'd



Let's also look at the density function of `petal.length` for each of the three classes of irises.

```
density.setosa = density(iris$Petal.Length[iris$Species == "setosa"])  
density.versicolor = density(iris$Petal.Length[iris$Species == "versicolor"])  
density.virginica = density(iris$Petal.Length[iris$Species == "virginica"])
```

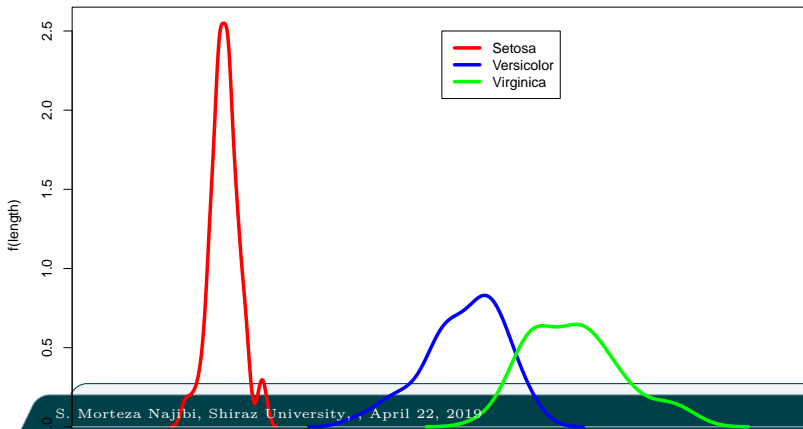


## Density plot cont'd



```
plot(density.setosa, ylab="f(length)", xlab="length (cm)", main="Density plot of  
  Legnths", xlim = c(0,8), lwd=4, col="red")  
lines(density.versicolor, col="blue", lwd=4)  
lines(density.virginica, col="green", lwd=4)  
legend(4, 2.5, c("Setosa", "Versicolor", "Virginica"), lwd=rep(4,3),  
  col=c("red", "blue", "green"))
```

Density plot of Petal  
Legnths



# Empirical Cumulative Distribution Functions



The cumulative area under the density function is the cumulative distribution function. Looking at this function using a sample of points gives use the empirical cumulative distribution function (ECDF). Let's start by calculating the cumulative areas

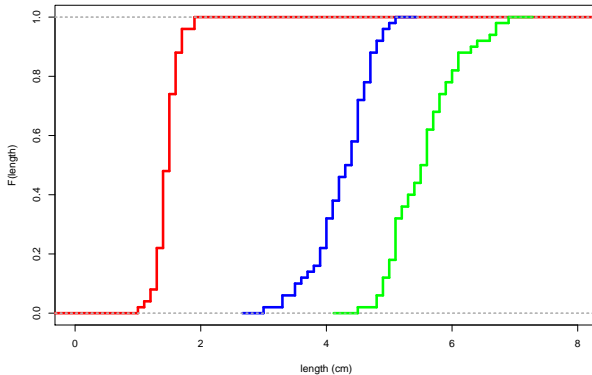
```
ecdf.setosa = ecdf(iris$Petal.Length[iris$Species == "setosa"])
ecdf.versicolor = ecdf(iris$Petal.Length[iris$Species == "versicolor"])
ecdf.virginica = ecdf(iris$Petal.Length[iris$Species == "virginica"])
```

The functions may also be plotted (notice the step function patterns)

```
plot(ecdf.setosa, ylab = "F(length)", xlab = "length (cm)", col.h = "red",
     main = "ECDF for Petal Lengths",
     verticals = T, col.v = "red", do.p = F, lwd = 4, xlim = c(0, 8))
lines(ecdf.versicolor, col.h = "blue", col.v = "blue", verticals = T, lwd = 4,
     do.p = F)
lines(ecdf.virginica, col.h = "green", col.v = "green", verticals = T, lwd = 4,
     do.p = F)
```



ECDF for Petal Lengths



# Quantile Plots



The inverse of the ECDF is the quantile function.

```
ps = seq(0,1,length=25)
quantile.setosa = quantile(iris$Petal.Length[iris$Species == "setosa"], probs=ps)
quantile.setosa
```

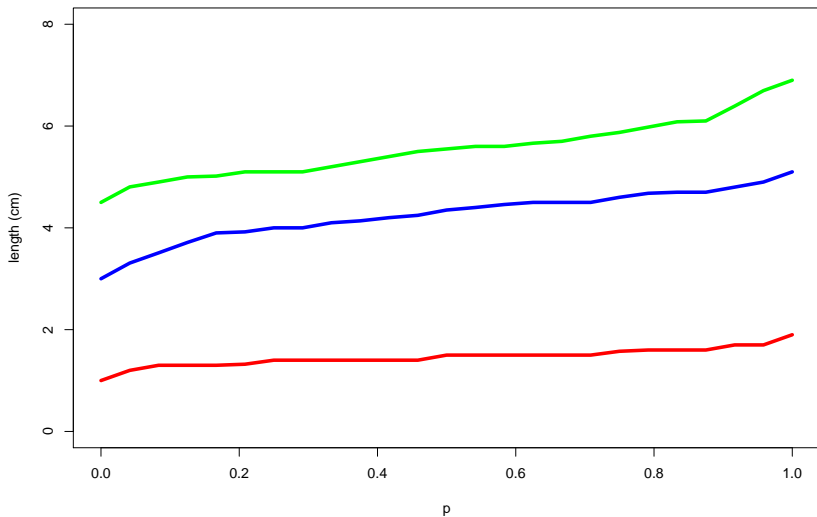
```
##          0% 4.166667% 8.333333%      12.5% 16.66667% 20.83333%      25%
## 1.000000 1.200000 1.300000 1.300000 1.300000 1.320833 1.400000
## 29.16667% 33.33333%      37.5% 41.66667% 45.83333%      50% 54.16667%
## 1.400000 1.400000 1.400000 1.400000 1.400000 1.500000 1.500000
## 58.33333%      62.5% 66.66667% 70.83333%      75% 79.16667% 83.33333%
## 1.500000 1.500000 1.500000 1.500000 1.575000 1.600000 1.600000
##      87.5% 91.66667% 95.83333%      100%
## 1.600000 1.700000 1.700000 1.900000
```



```
quantile.versicolor = quantile(iris$Petal.Length  
                             [iris$Species == "versicolor"],probs=ps)  
quantile.virginica = quantile(iris$Petal.Length  
                             [iris$Species == "virginica"], probs=ps)  
plot(ps, quantile.setosa, xlab = "p", ylab = "length (cm)",  
     main = "Quantiles for Petal Length",  
     type="l", lwd = 4, col="red", ylim=c(0,8))  
lines(ps, quantile.versicolor, col = "blue", lwd = 4)  
lines(ps, quantile.virginica, col = "green", lwd = 4)
```



## Quantiles for Petal Length

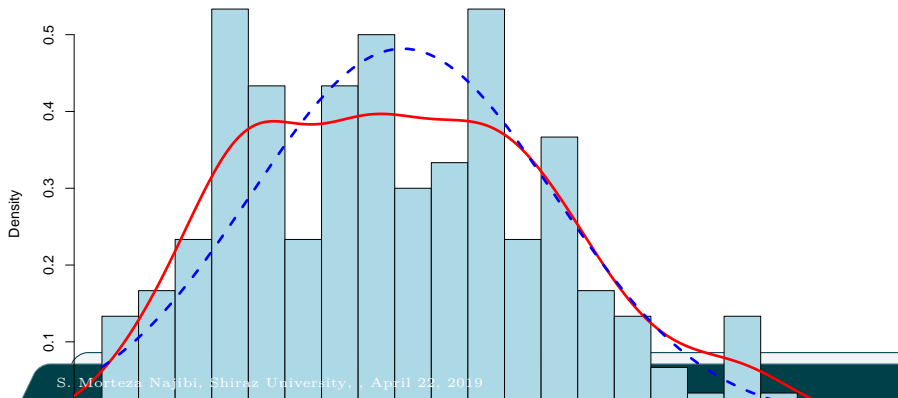


# check Normality plot



```
hist(iris$Sepal.Length, probability=TRUE, breaks=15, col="light blue")
rug(jitter(iris$Sepal.Length, 5))
points(density(iris$Sepal.Length), type='l', lwd=3, col='red')
f <- function(t) {
  dnorm(t, mean=mean(iris$Sepal.Length), sd=sd(iris$Sepal.Length) )
}
curve(f, add=T, col="blue", lwd=3, lty=2)
```

Histogram of iris\$Sepal.Length



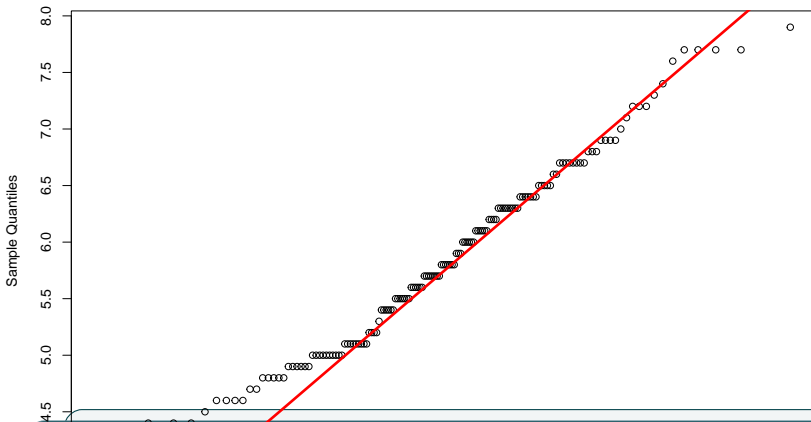
# Q-Q plot



A q-q plot is a plot of the quantiles of the first data set against the quantiles of the second data set.

```
x <- iris$Sepal.Length  
qqnorm(x)  
qqline(x, col="red", lwd=3)
```

Normal Q-Q Plot

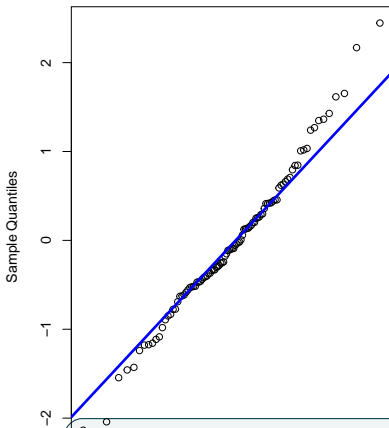




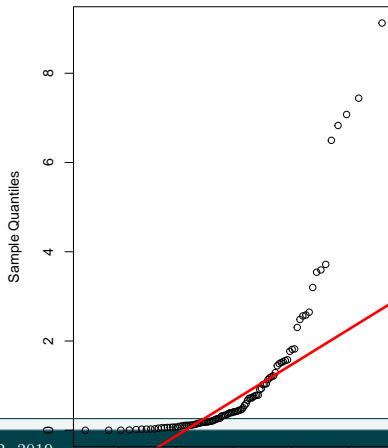
## Some more examples

```
par(mfrow=c(1,2))
x <- rnorm(100)
y <- rnorm(100)^2
qqnorm(x, main="Gaussian variable");qqline(x, col="blue", lwd=3)
qqnorm(y, main="Non gaussian variable");qqline(y, col="red", lwd=3)
```

Gaussian variable



Non gaussian variable



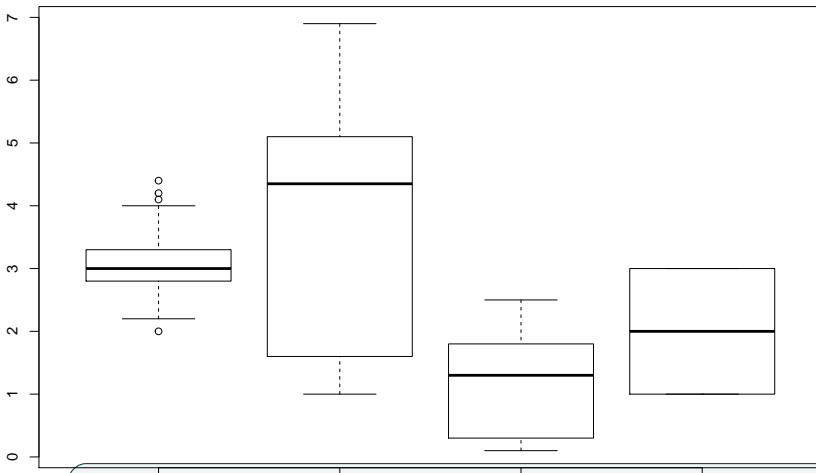
# Box Plots



Box plots are used to compactly show many pieces of information about a variables distribution including some summary statistics.

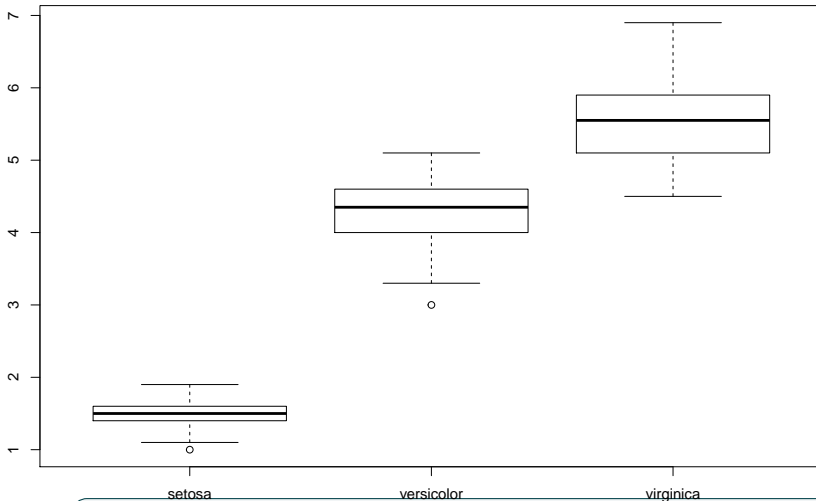
A box plot of each attribute of the iris data set

```
boxplot(iris[, -1])
```



A box plot may also be used for one attribute `petal.length` and show how it varies with the class `Species`

```
boxplot(iris$Petal.Length ~ iris$Species)
```

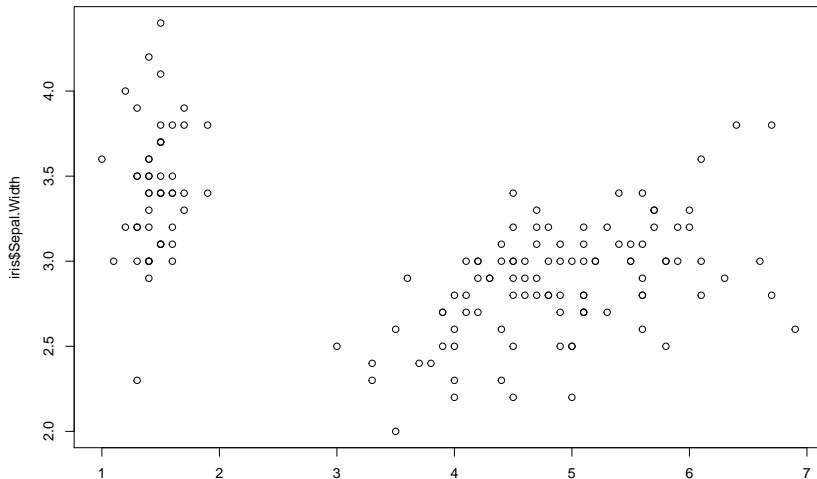


# Scatter Plots



Scatter plots are used to plot two variables against each other (or 3 in the case of 3D plots).

```
plot(iris$Petal.Length, iris$Sepal.Width)
```

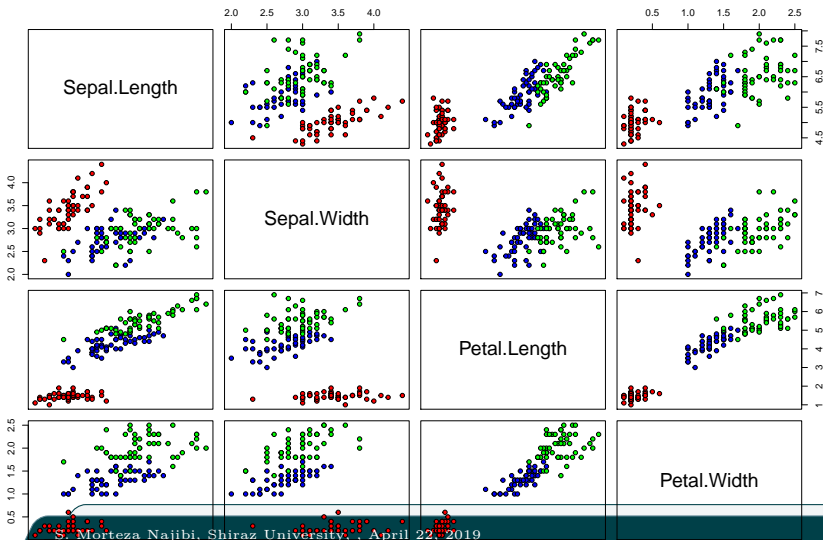


# Pairwise Plots



For data sets with only a few attributes, all the pairwise scatter plots may be constructed.

```
pairs(as.matrix(iris[, -5]), pch=21, bg=c("red", "blue", "green")[iris$Species])
```





# Color Selection Utilities



Default color palette and how to change it

```
palette()
```

```
## [1] "black"    "red"      "green3"   "blue"     "cyan"     "magenta"  "yellow"
## [8] "gray"
```

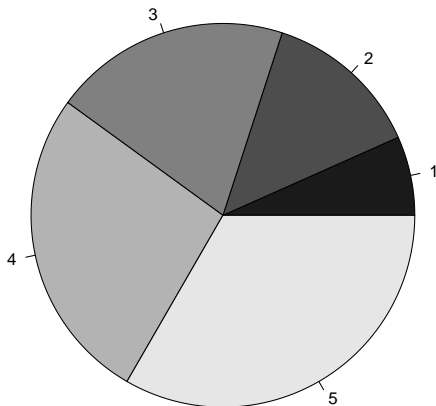
```
palette(rainbow(5, start=0.1, end=0.2))
palette()
```

```
## [1] "#FF9900" "#FFBF00" "#FFE600" "#F2FF00" "#CCFF00"
```



The gray function allows to select any type of gray shades by providing values from 0 to 1

```
pie(1:5,col=gray(seq(0.1, 1, by= 0.2)))
```



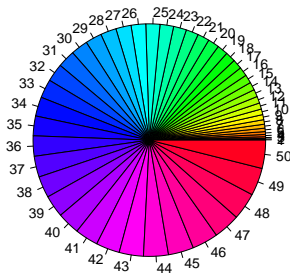
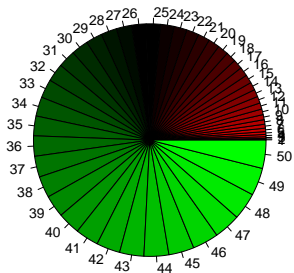
# Color gradients with colorpanel function from gplots library



```
library(gplots, warn.conflicts = F)
```

```
## Warning: package 'gplots' was built under R version 3.5.2
```

```
par(mfrow=c(1,2))
pie(1:50, col=colorpanel(50,"red","black","green"))
pie(1:50, col=rainbow(50))
```



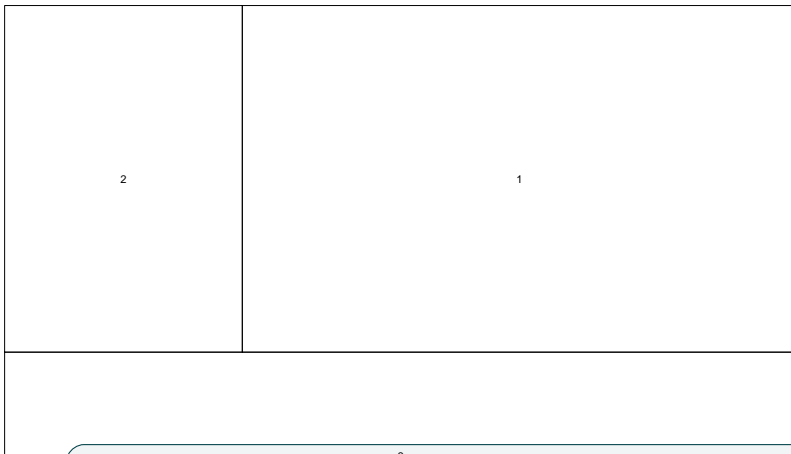




# Arranging Plots with Variable Width

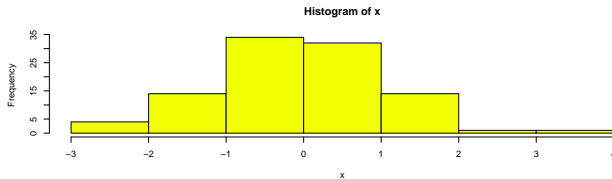
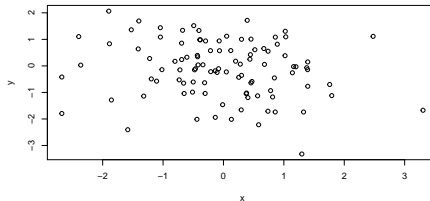
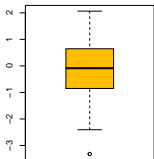
The layout function allows to divide the plotting device into variable numbers of rows and columns with the column-widths and the row-heights specified in the respective arguments.

```
nf <- layout(matrix(c(2,1,3,3), 2, 2, byrow=TRUE), c(3,7), c(5,3))
layout.show(nf)
```





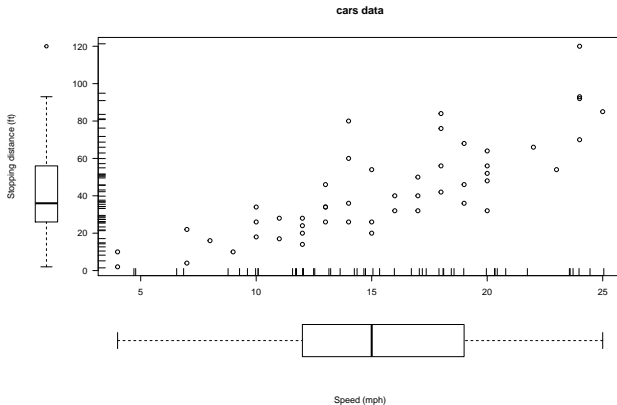
```
x <- rnorm(100) ; y <- rnorm(100)
layout(matrix(c(2,1,3,3), 2, 2, byrow=TRUE), c(3,7), c(5,4))
plot(x,y)
boxplot(y,horizontal = FALSE,col=2)
hist(x,col=4)
```





# Margine plots

```
data(cars)
layout( matrix( c(2,1,0,3), 2, 2, byrow=T ),
c(1,6), c(4,1))
par(mar=c(1,1,5,2)); plot(cars$dist ~ cars$speed,xlab='', ylab='',las = 1)
rug(side=1, jitter(cars$speed, 5) );rug(side=2, jitter(cars$dist, 20) );
title(main = "cars data")
par(mar=c(1,2,5,1))
boxplot(cars$dist, axes=F); title(ylab='Stopping distance (ft)', line=0)
par(mar=c(5,1,1,2))
boxplot(cars$speed, horizontal=T, axes=F); title(xlab='Speed (mph)', line=1)
```



## Exercise: Bar Plots, Pie plot,



- Consider the same approach in this slides for `barplot`



# Saving Graphics to Files



After the `pdf()` command all graphs are redirected to file `test.pdf`. Works for all common formats similarly: `jpeg`, `png`, `ps`, `tiff`, ...

```
pdf("test.pdf"); plot(1:10, 1:10); dev.off()
```

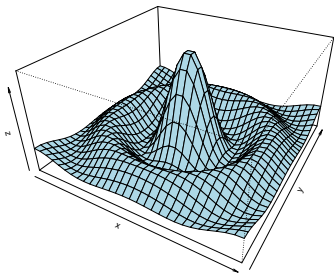
# 3 Dimesional functions and their plots



# persp 3d plot



```
x <- seq(-10, 10, length= 30)
y <- x
f <- function(x, y) { r <- sqrt(x^2+y^2); 10 * sin(r)/r }
z <- outer(x, y, f)
persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
```





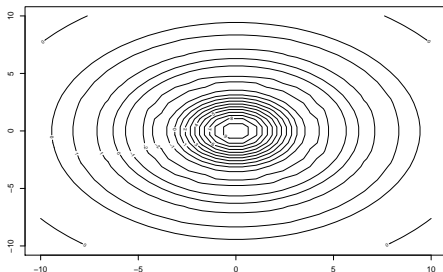


# Countour Plots



A contour plot is a graphical technique for representing a 3-dimensional surface by plotting constant  $z$  slices, called contours, on a 2-dimensional format.

```
contour(x,y,z) # or contour(z)
```

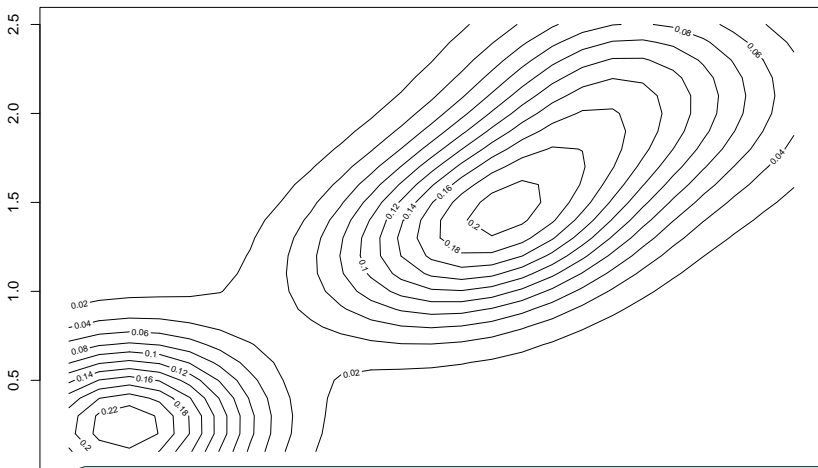


## 2D Density



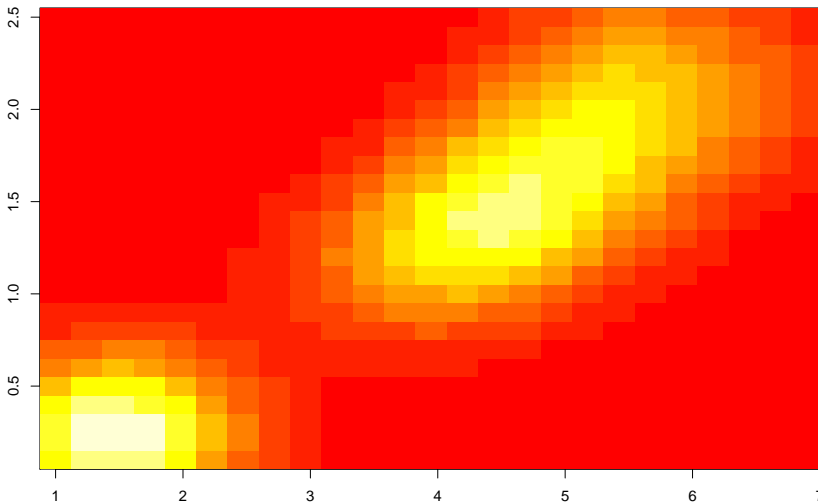
Density estimation is available for higher dimensional data. Use the **MASS** package.

```
library(MASS)
petal.dens = kde2d(iris$Petal.Length, iris$Petal.Width)
contour(petal.dens)
```



The plot may aslo be viewed as an image

```
image(petal.dens)
```



The background of the slide is a dark blue color. It features a large, faint, light blue geometric pattern. This pattern is composed of a grid of squares, each of which is further divided into four triangles by diagonal lines. The triangles and squares are arranged in a staggered, overlapping fashion, creating a complex, crystalline or lattice-like structure that covers the entire background.

# lattice Grapphics

# What is lattice?

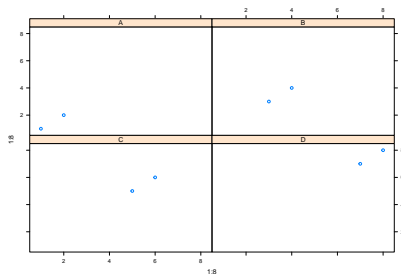


- ▶ High-level graphics system
- ▶ Implements Trellis graphics system from S-Plus Simplifies high-level plotting tasks: arranging complex graphical features
- ▶ Syntax similar to R's base graphics

# Scatter Plot Sample



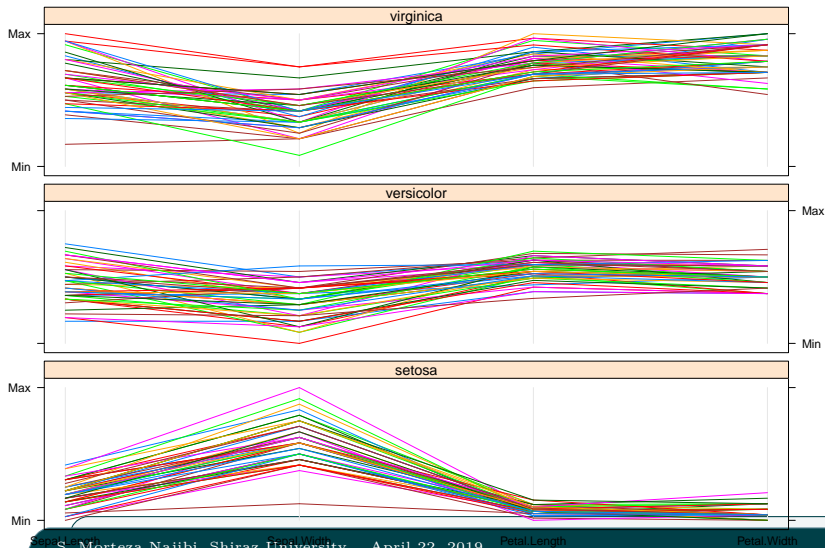
```
library(lattice)  
xyplot(1:8 ~ 1:8 | rep(LETTERS[1:4], each=2), as.table=TRUE)
```



# Line Plot Sample



```
parallelplot(~iris[1:4] | Species, iris, horizontal.axis = FALSE,  
            layout = c(1, 3, 1))
```





**ggplot2 package (a superfull package in graphic )**



- ▶ What is ggplot2?
  - ▶ High-level graphics system
  - ▶ Streamlines many graphics workflows for complex plots
  - ▶ Simpler qplot function provides many shortcuts
- ▶ ggplot function accepts two arguments
  - ▶ Data set to be plotted
  - ▶ **Aesthetic** mappings provided by **aes** function
  - ▶ Additional parameters such as geometric objects (e.g. points, lines, bars) are passed on by appending them with **+** as separator.

# qplot function

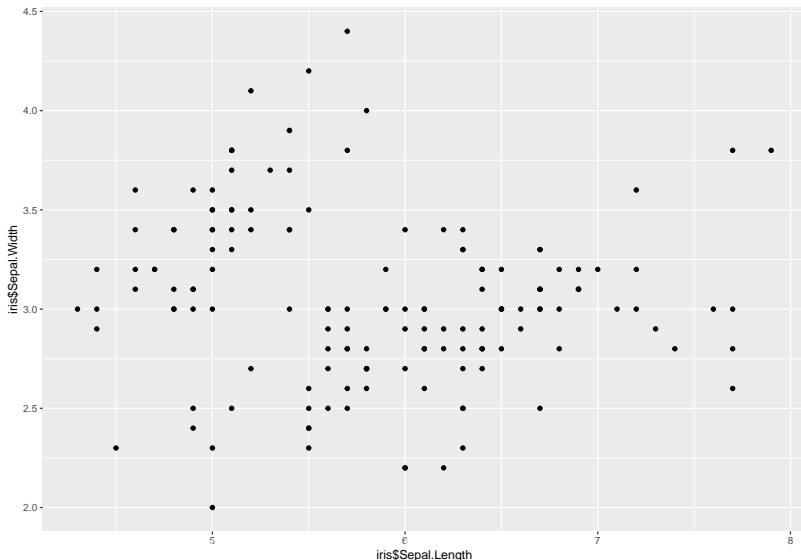


qplot syntax is similar to R's basic plot function Arguments:

- ▶ x: x-coordinates (e.g. col1)
- ▶ y: y-coordinates (e.g. col2)
- ▶ data: data frame with corresponding column names xlim, ylim:  
e.g. xlim=c(0,10)
- ▶ log: e.g. log="x" or log="xy"
- ▶ main: main title; see ?plotmath for mathematical formula xlab, ylab: labels for  
the x- and y-axes
- ▶ color, shape, size
- ▶ ...: many arguments accepted by plot function

## Create sample data

```
library(ggplot2,quietly = T)
#cat <- rep(c("A", "B"), 5)
qplot(iris$Sepal.Length, iris$Sepal.Width, geom="point")
```



# Prints dots with different sizes and colors

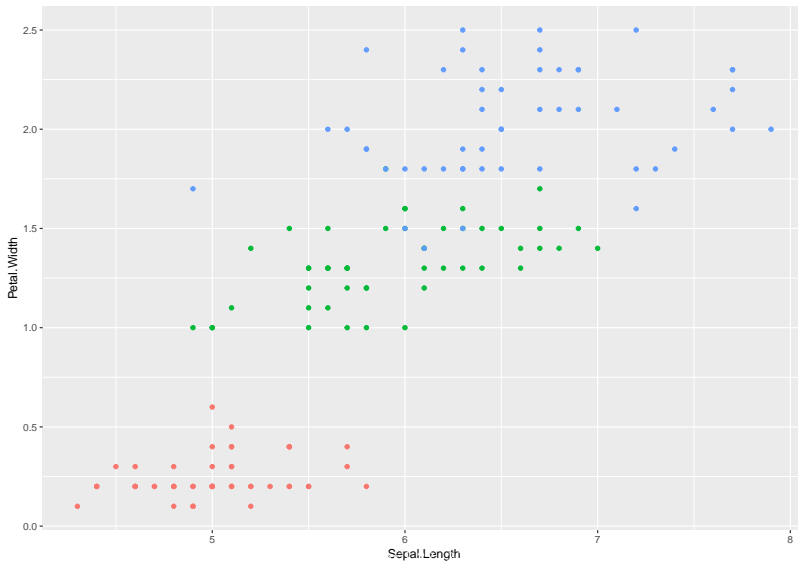
```
qplot(Sepal.Length, Petal.Width, data = iris, geom="point",  
      size=Petal.Length,col=Species,  
      main="Dot Size and Color Relative to Some Values")
```



# Drops legend



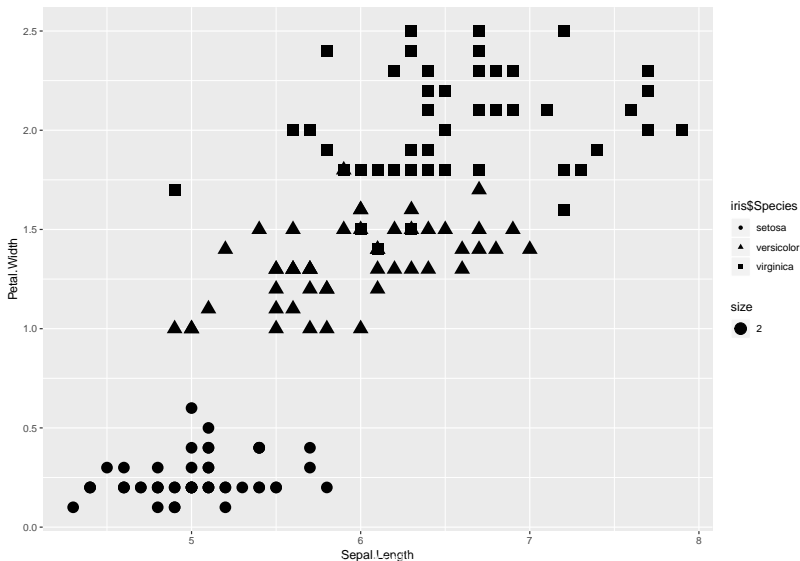
```
qplot(Sepal.Length, Petal.Width, data = iris, geom="point", col=Species)+  
  theme(legend.position = "none")
```



# Plot different shapes



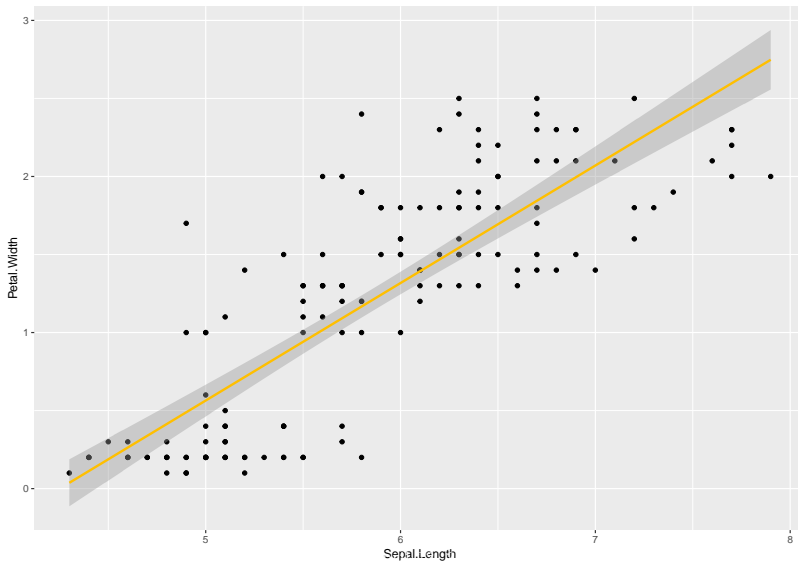
```
qplot(Sepal.Length, Petal.Width, data = iris, geom="point",  
      shape=iris$Species, size=2)
```



# Scatter Plot with Regression Line



```
qplot(Sepal.Length, Petal.Width, data = iris, geom = c("point"))+  
  geom_smooth(method=lm,col=2)
```



# More application of qplot



```
qplot(Sepal.Length, Petal.Width, data = iris, facets = Species ~ Petal.Length)
```

