Week 2: Visualization with R

CKMT 105

Data Science Certificate Program

Ryerson University

Bora Caglayan

15 Jan, 2015

Course Details Reminder

Required Text: - Course notes - Handouts - All announcements through Blackboard CMS.

Recommended Reading: - Weekly reading material

Grading: - Lab Attendance 15% - 3 homeworks - 30 (10%x3) - Homework deadlines (week 6, week 9, week 12) - Midterm - 20% - Final - 35% - You may skip 2 labs without penalty.

Weekly Schedule Overview

- Each week a lecture followed by a lab session.
- Lab sessions are scored based on attendance. Collaboration during the labs is encouraged.
- Individual work on homeworks.
- Expected course work by the students is one hour per week.
- Homework sets will be given in the end of week 3, week 6 and week 8.
- Please check Blackboard CMS regularly and follow the announcements.
- Course notes will be provided before the lectures in Blackboard.

Lecture Outline

- Standard plotting
- Plotting with ggplot package
- Showing data on maps
- Dealing with many data points
- Lab Session

Lecture Outline

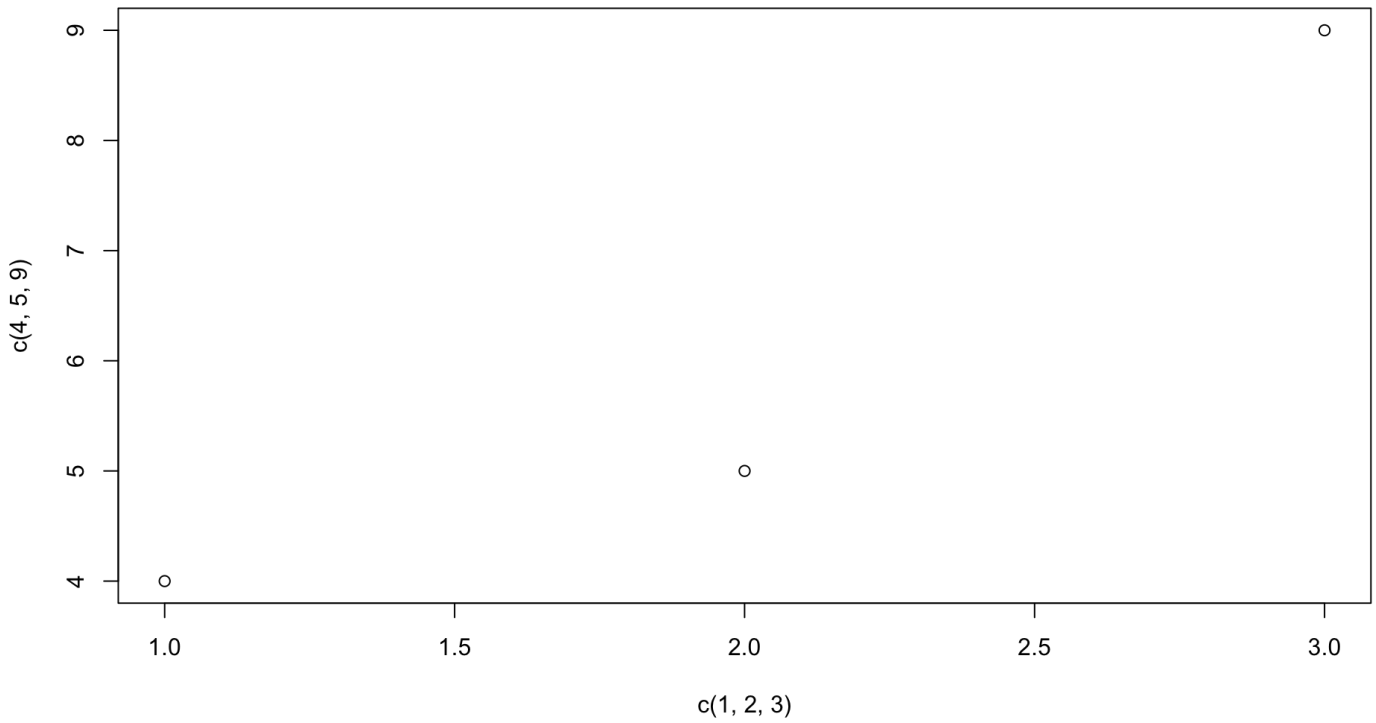
title: false

- R comes with a powerful standard visualization library. - More sophisticated plots can be created using libraries such as ggplot and ggmaps.

Plotting Basics

Here is a basic plot:

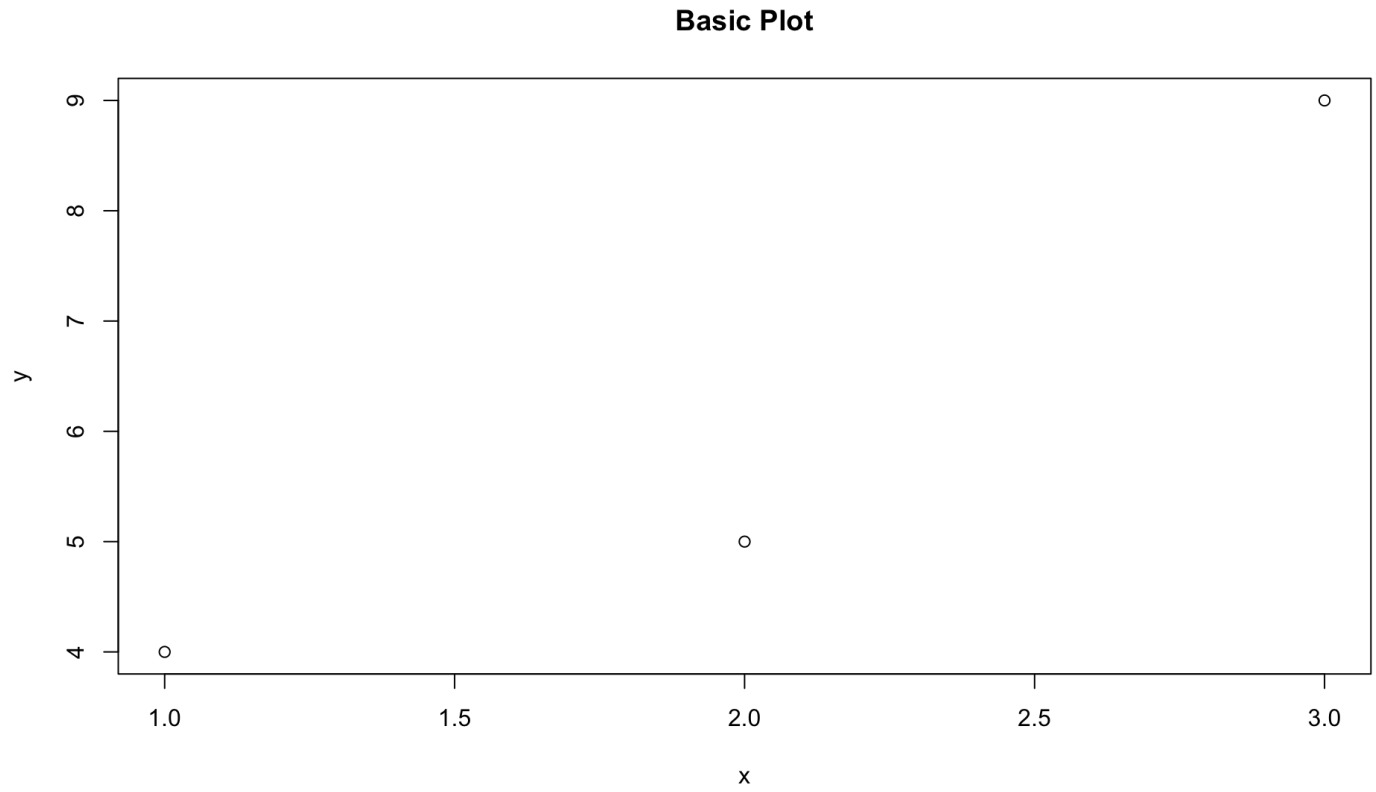
```
plot(c(1,2,3), c(4,5,9))
```



Plotting Basics

Adding labels:

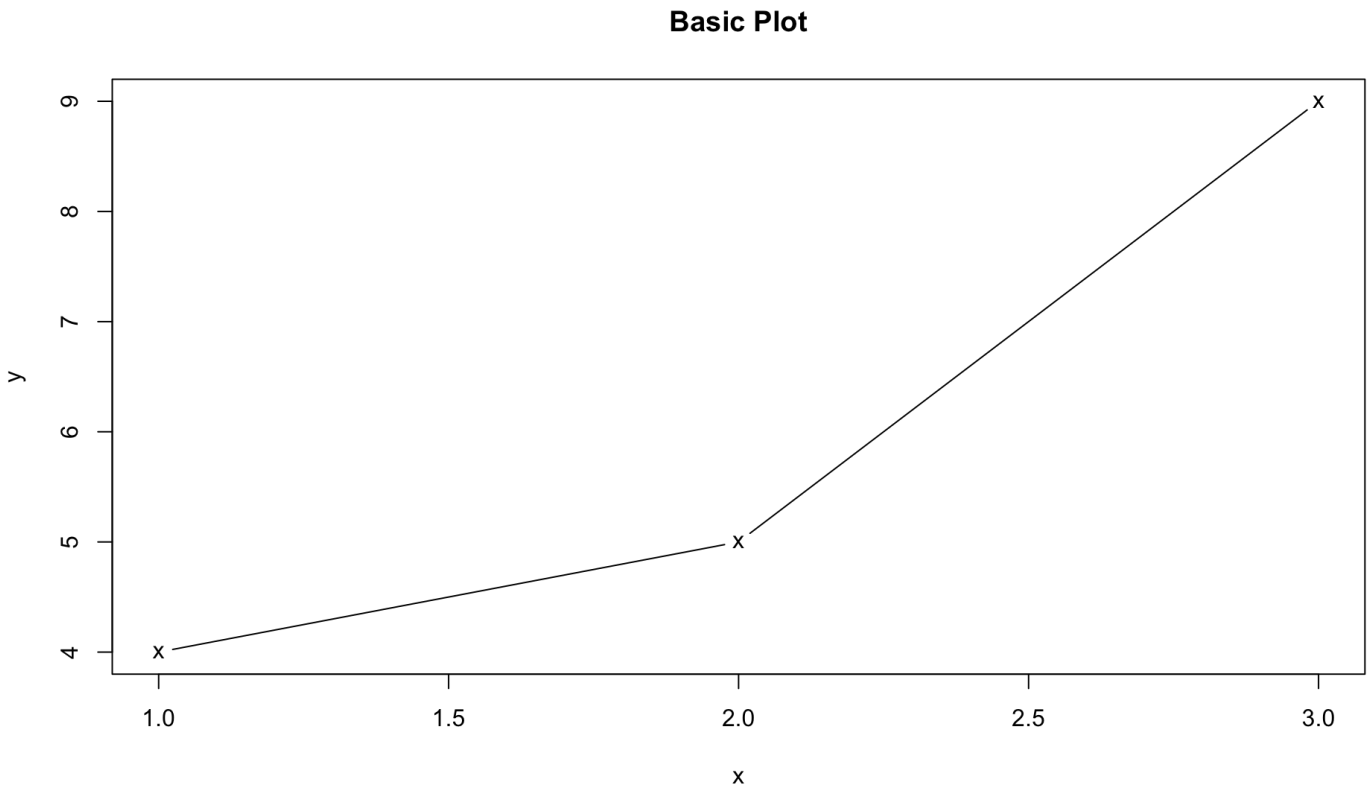
```
plot(c(1,2,3), c(4,5,9), xlab="x", ylab="y", main="Basic Plot")
```



Plotting Basics

Markers and lines:

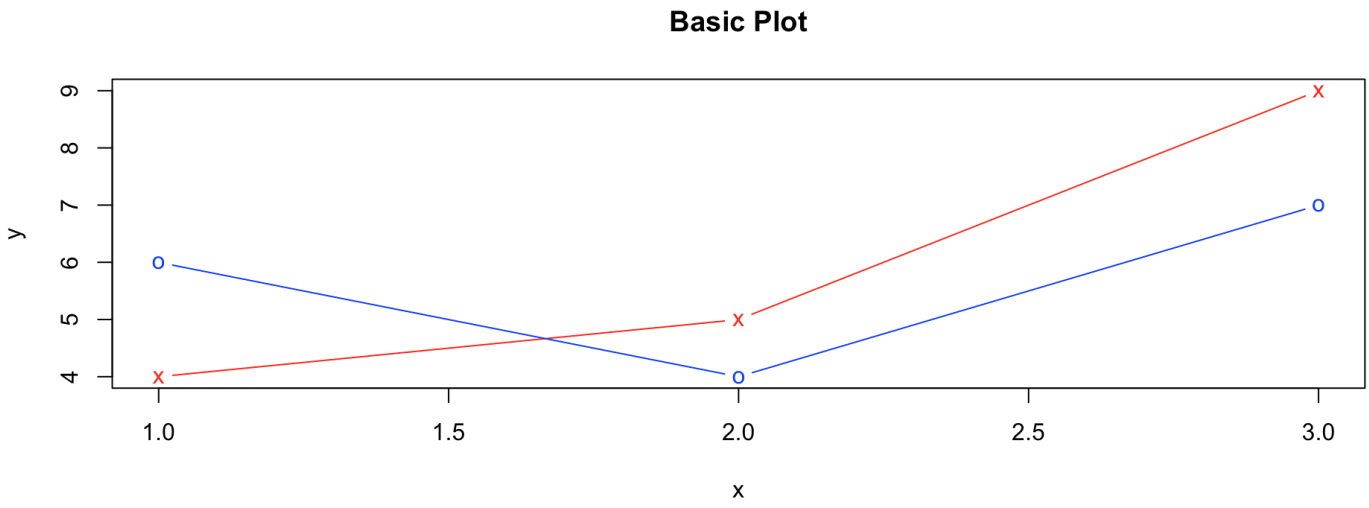
```
plot(c(1,2,3), c(4,5,9), xlab="x", ylab="y", main="Basic Plot", type="b", pch="x")
```



Plotting Basics

Multiple series:

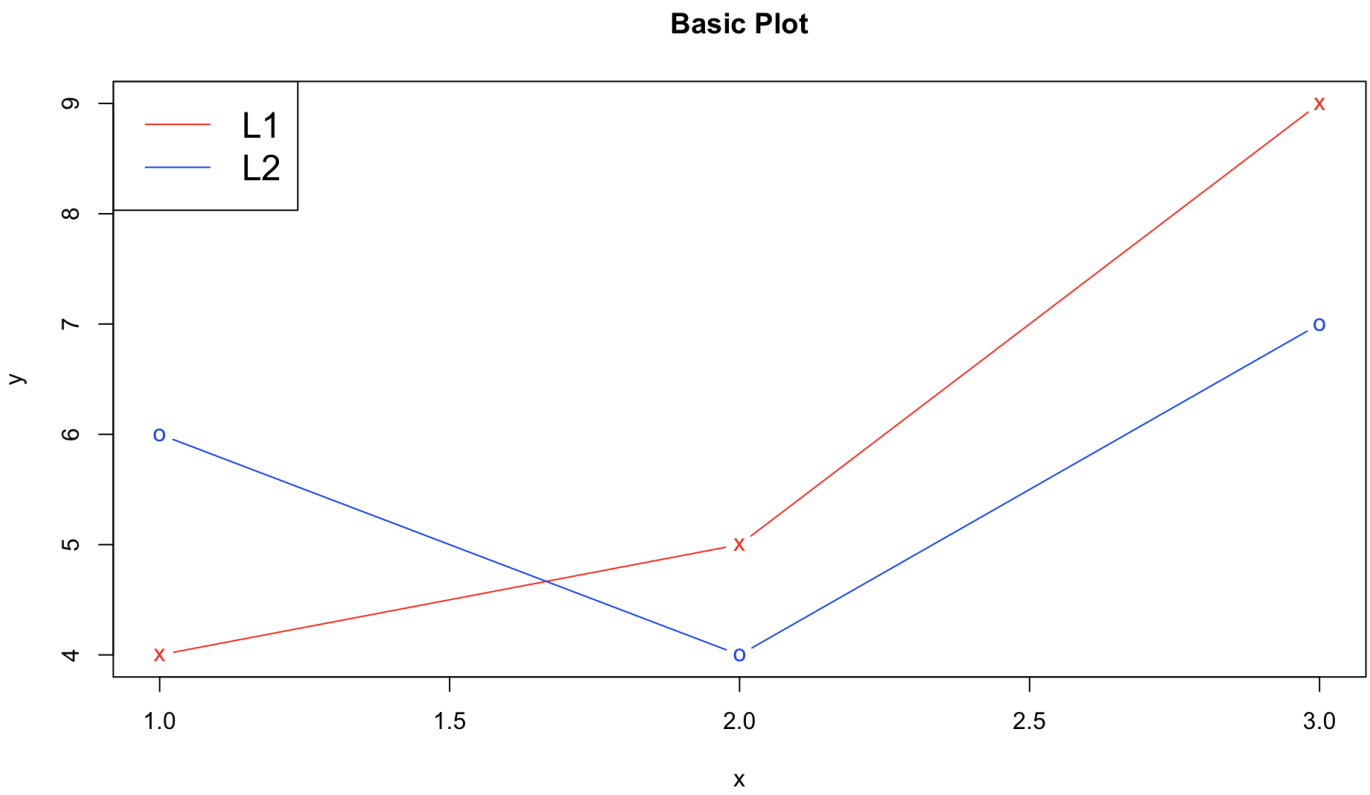
```
plot(c(1,2,3), c(4,5,9), xlab="x", ylab="y", main="Basic Plot", col="red", type="b", pch="
x")
lines(c(1,2,3), c(6, 4, 7), type="b", pch="o", col="blue")
```



Plotting Basics

Adding legends

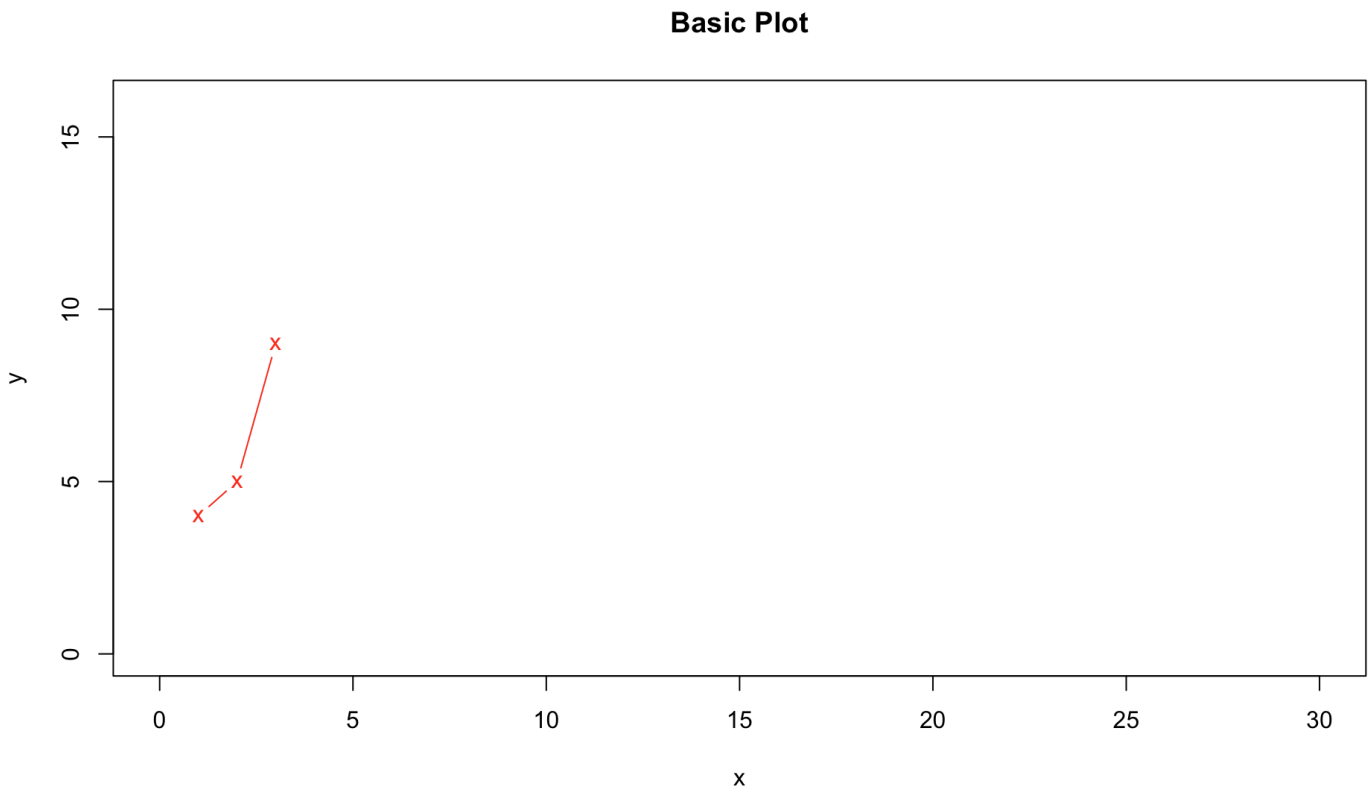
```
legend('topleft', c("L1", "L2") ,
   lty=1, col=c('red', 'blue'), cex=1.5)
```



Plotting Basics

Changing limits

```
plot(c(1,2,3), c(4,5,9), xlab="x", ylab="y", main="Basic Plot", col="red", type="b", pch="
x", xlim=c(0,30),ylim=c(0,16))
```



Plotting Histogram

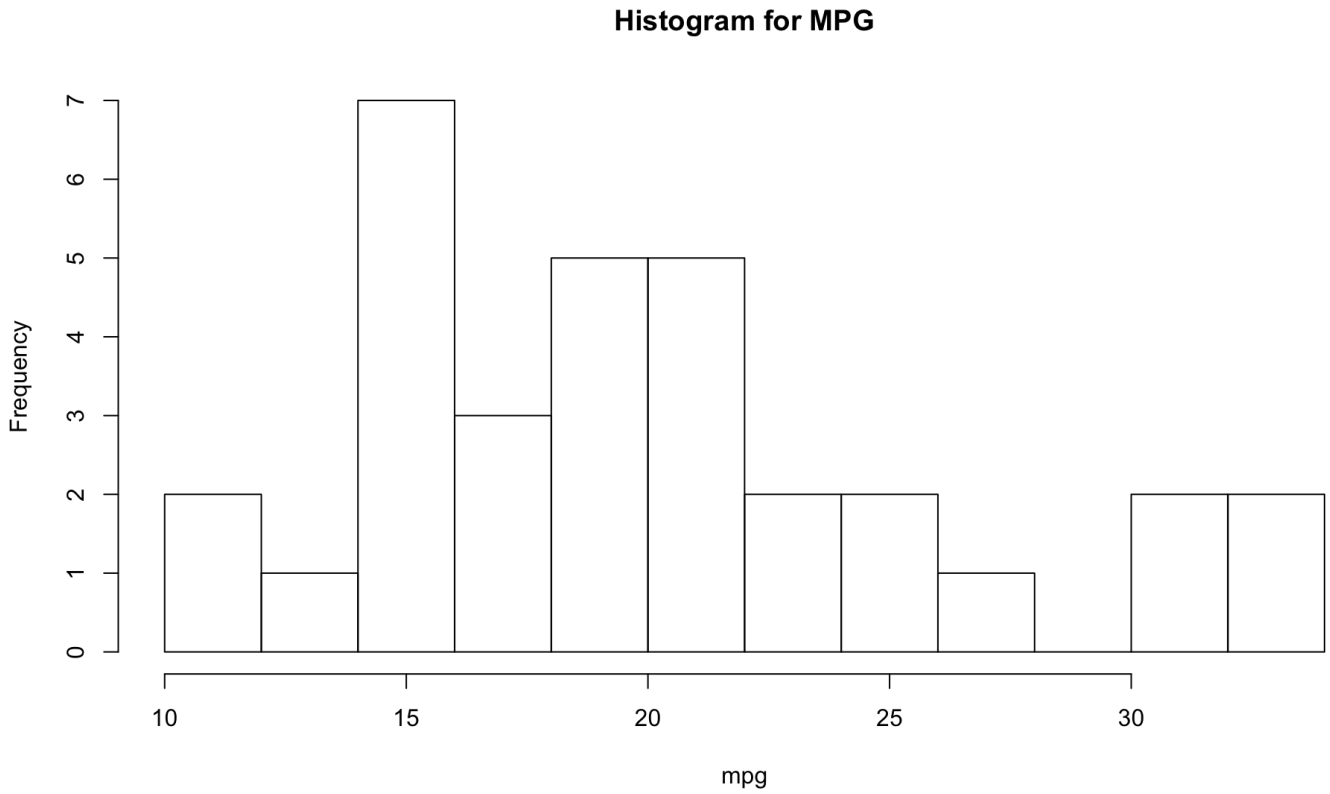
Saving a plot - R can direct the output to different targets. - png, bmp, tiff, jpeg are also valid targets. - pdf function can direct output to different pages and stores the plot in vector graphics.

```
getwd() # get working directory

pdf("my_report.pdf") # select pdf file as the output target
plot(mtcars$mpg, main="Kernel Density of Miles Per Gallon", type="l") # create plot
dev.off() # save pdf file
```

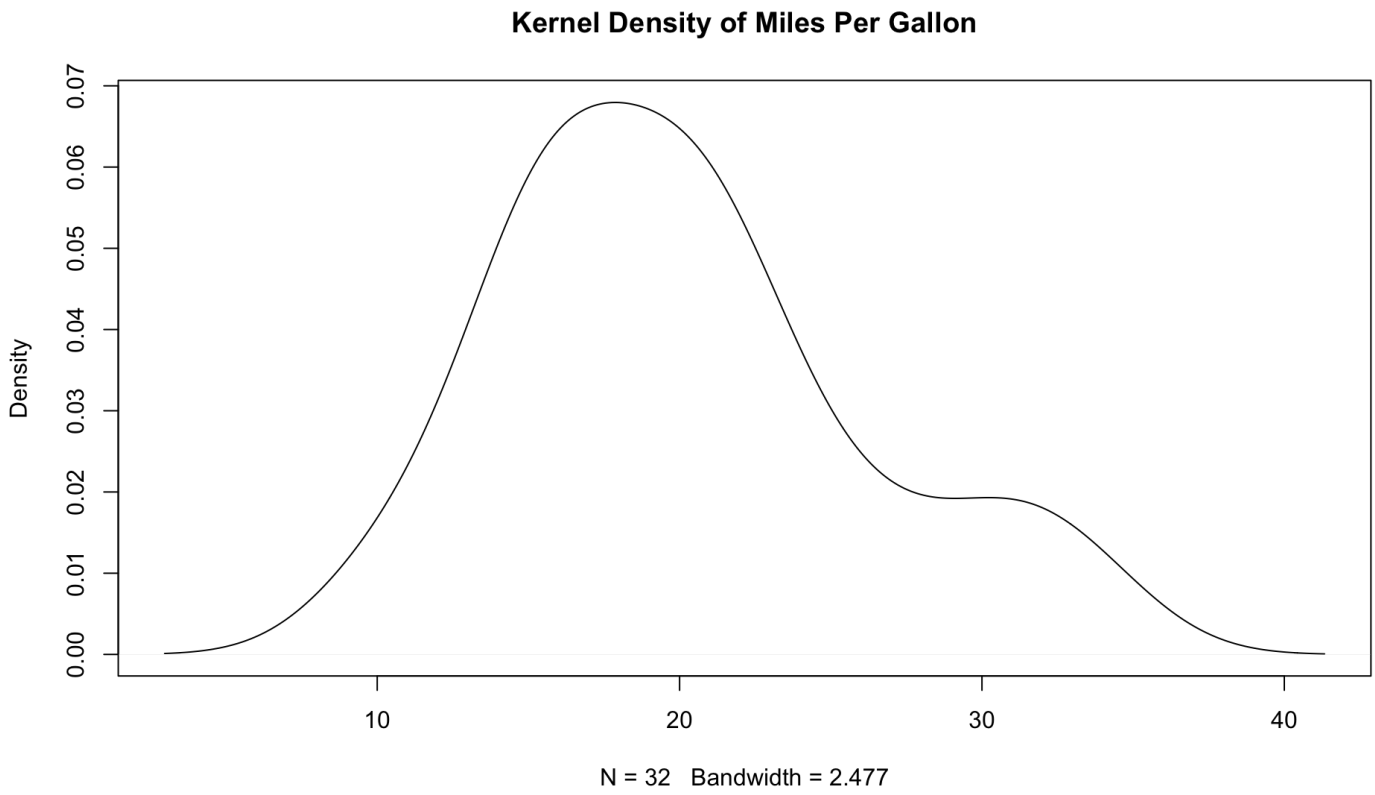
Plotting Histogram

```
hist(mtcars$mpg, xlab= "mpg", main="Histogram for MPG", breaks=10)
```



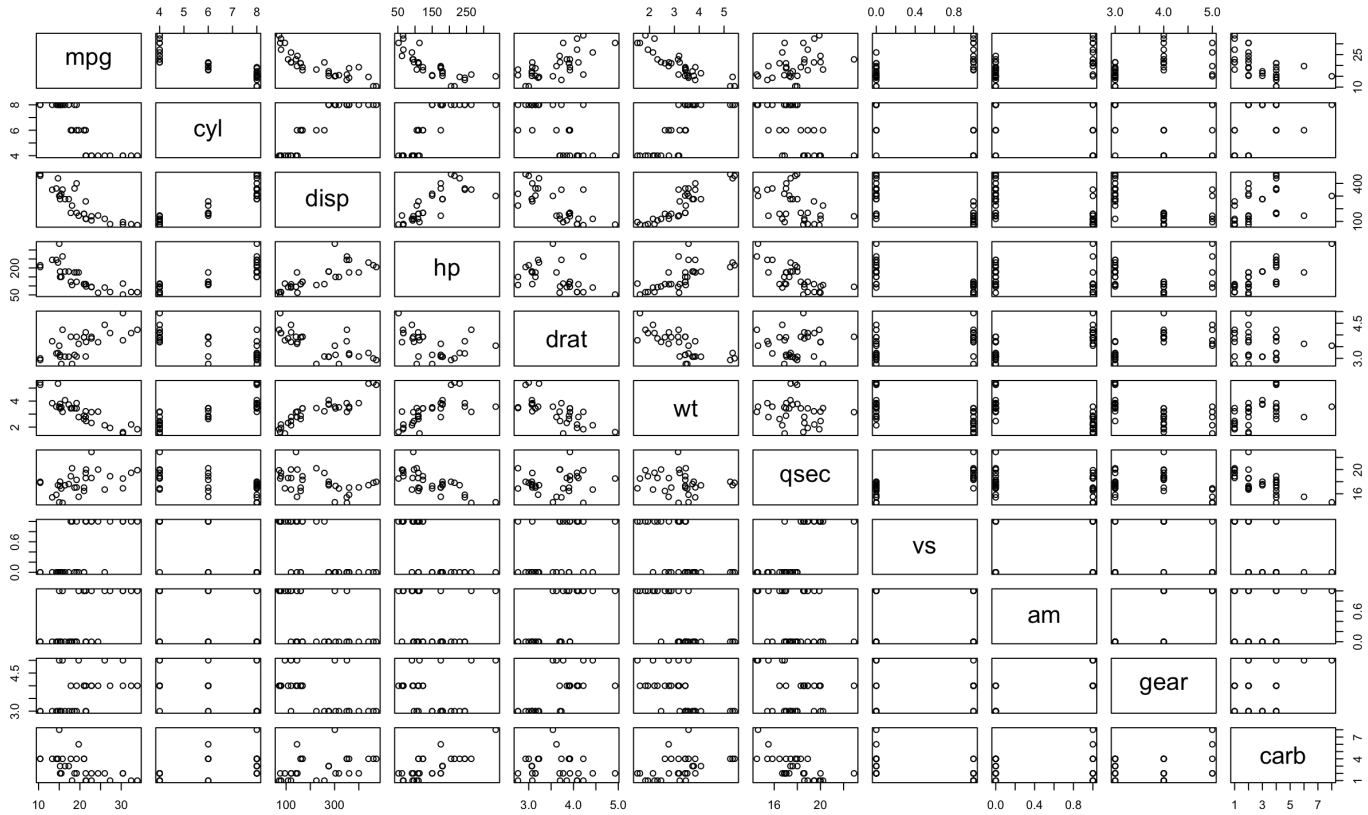
Plotting Probability Density Function

```
plot(density(mtcars$mpg), main="Kernel Density of Miles Per Gallon")
```



Plotting A Data Frame

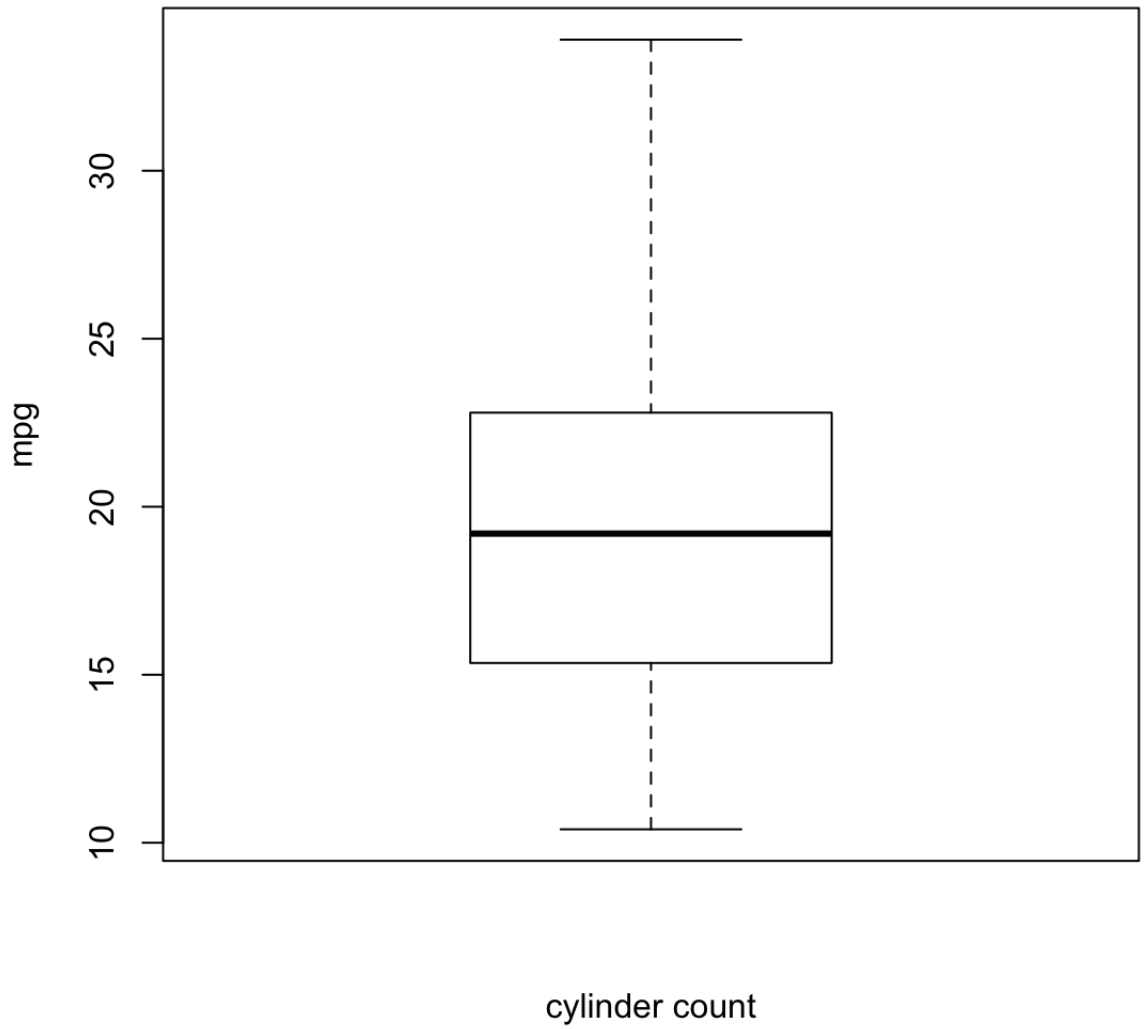
```
plot(mtcars)
```



Boxplots

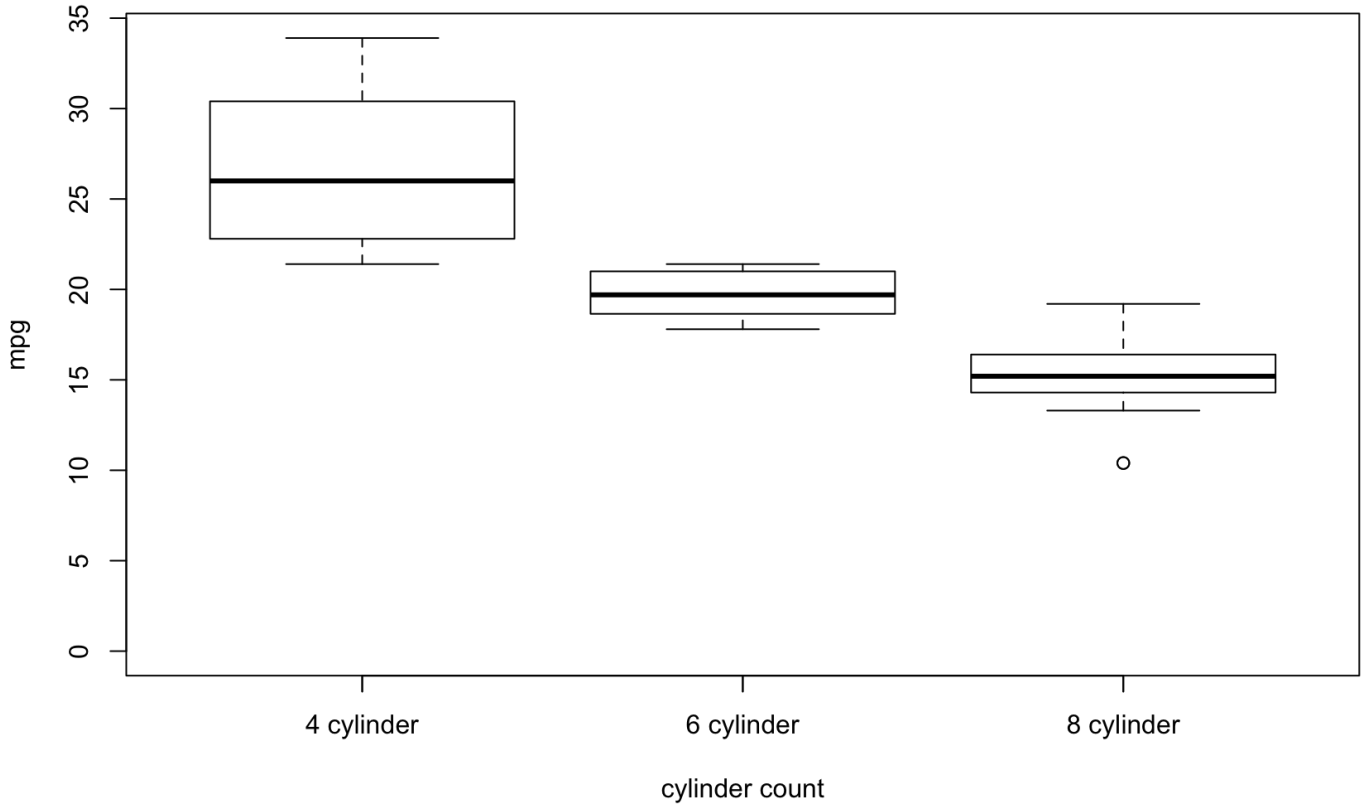
Boxplots are used to visualize and compare distributions. Shows the following: - Median: Horizontal line within rectangle - Lower quartile, Upper quartile: Lower and upper bounds of the rectangle - Max, minimum: Lower and upper whiskers - Outlier points: More than 1.5 times of upper quartile or less than lower quartile divided by 1.5

```
boxplot(mtcars$mpg, xlab="cylinder count", ylab="mpg")
```



Boxplots

```
boxplot(mtcars[mtcars$cyl==4,]$mpg, mtcars[mtcars$cyl==6,]$mpg, mtcars[mtcars$cyl==8,]$mpg
, xlab="cylinder count", ylab="mpg", ylim=c(0,max(mtcars$mpg)))
axis(1, 1:3, c("4 cylinder", "6 cylinder", "8 cylinder"))
```

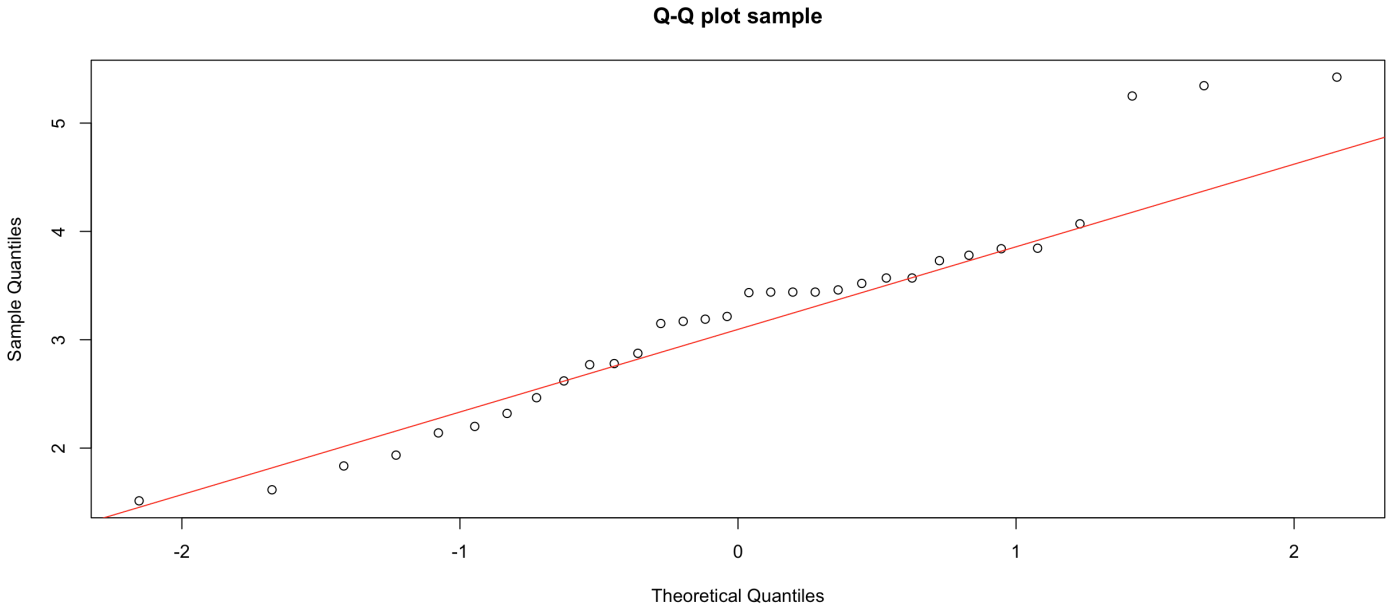


QQ Plots

- Quantile-Quantile plot evaluates the fit of sample data to the normal distribution.
- Might be important to check before applying certain statistical methods since some methods assume normal distribution.
- The quantiles of the standard normal distribution is represented by a straight line.
- The normality of the data can be evaluated by observing the extent in which the points appear on the line.
- Might also give ideas about the distribution: heavy tailed, right skew etc.

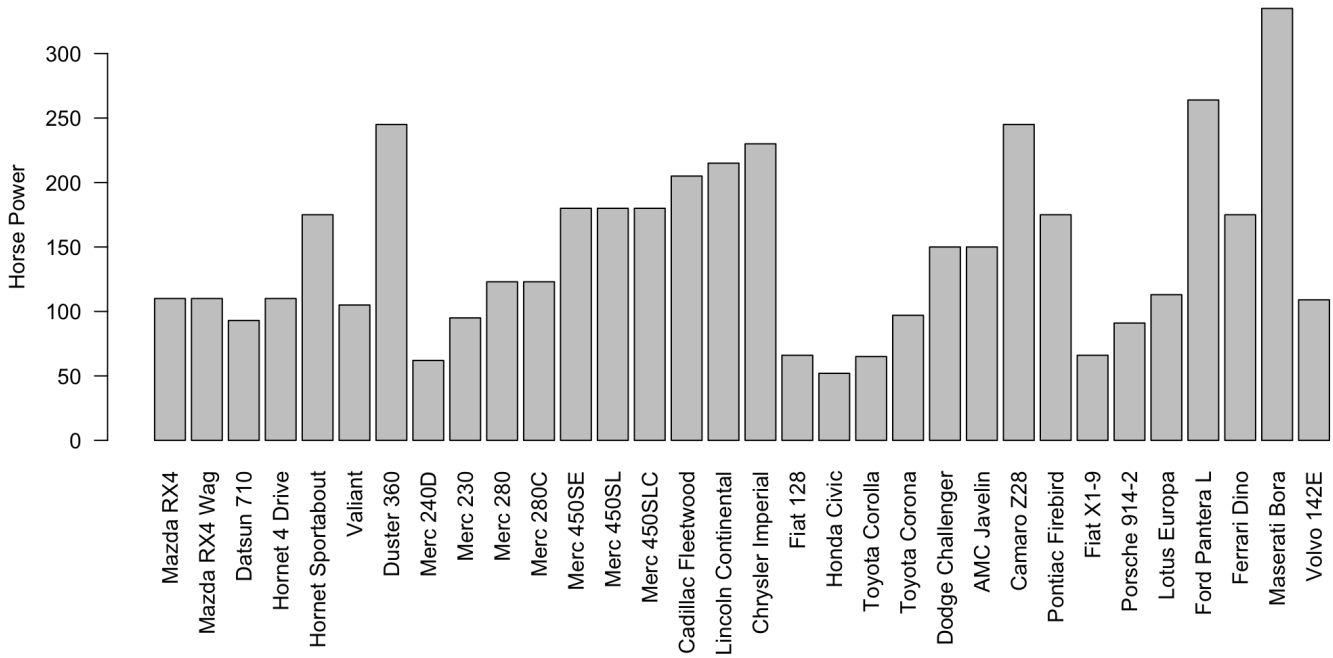
QQ Plots

```
qqnorm(mtcars$wt, main="Q-Q plot sample")
qqline(mtcars$wt, col="red")
```



Bar Plot

```
par(oma = c(4, 0, 0, 0)) # oma changes margin
barplot(mtcars$hp, names.arg=row.names(mtcars), las=2, ylab="Horse Power")
```



Interlude: Tables

1 way tables counts instances along changes in single dimension.

```
table(mtcars$gear)
```

```
##
##  3  4  5
## 15 12  5
```

Interlude: Tables

2 way tables counts instances along changes two different dimensions

```
table(mtcars$gear, mtcars$cyl)
```

```
##
##      4  6  8
##   3  1  2 12
##   4  8  4  0
##   5  2  1  2
```

Interlude: Tables

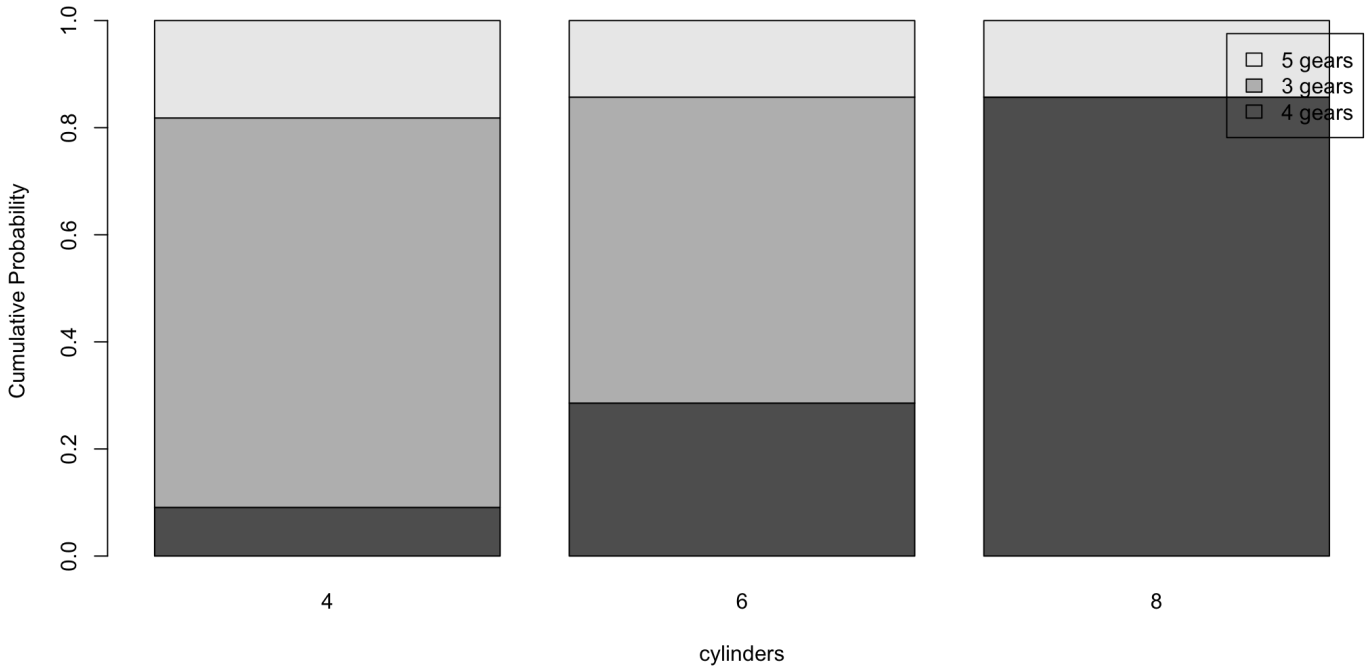
Prop tables show the probabilities along an axis

```
t <- table(mtcars$gear, mtcars$cyl)
prop.table(t,1)
```

```
##
##              4          6          8
##   3 0.06666667 0.13333333 0.80000000
##   4 0.66666667 0.33333333 0.00000000
##   5 0.40000000 0.20000000 0.40000000
```

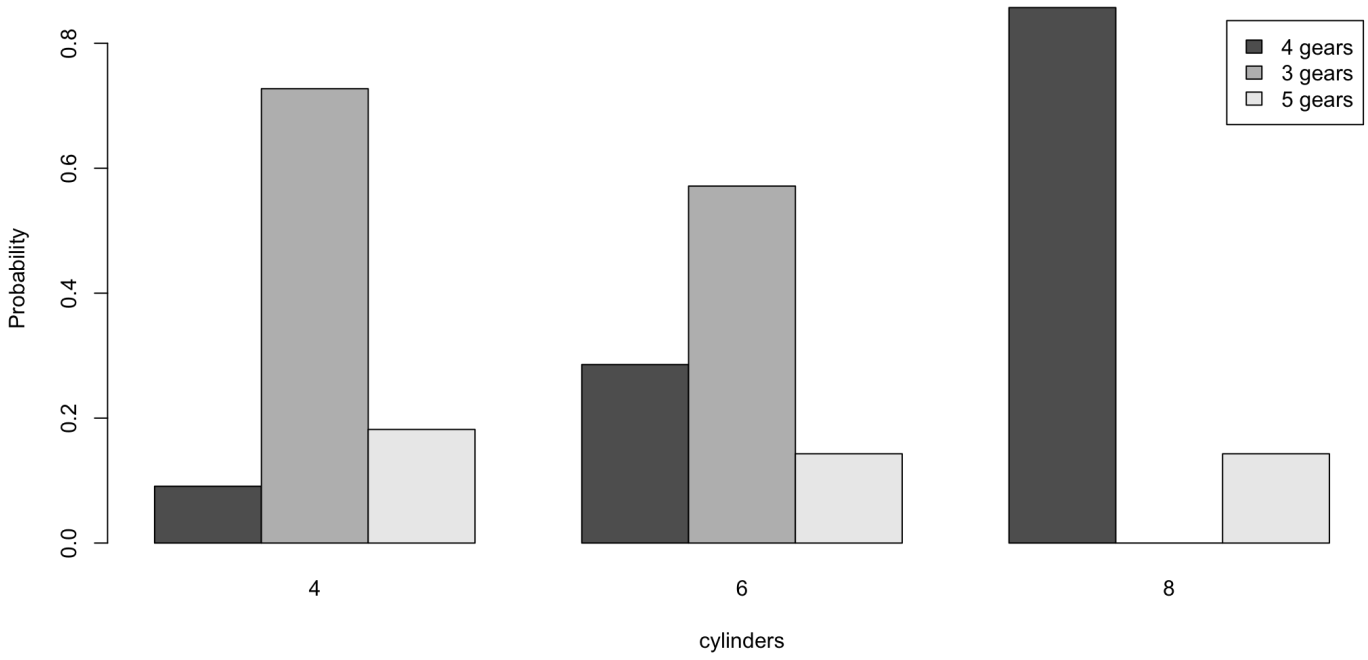
Stacked Bar Plot

```
t <- table(mtcars$gear, mtcars$cyl)
barplot(prop.table(t,2), legend=paste(unique(mtcars$gear), "gears"), ylab="Cumulative Prob
ability", xlab="cylinders")
```



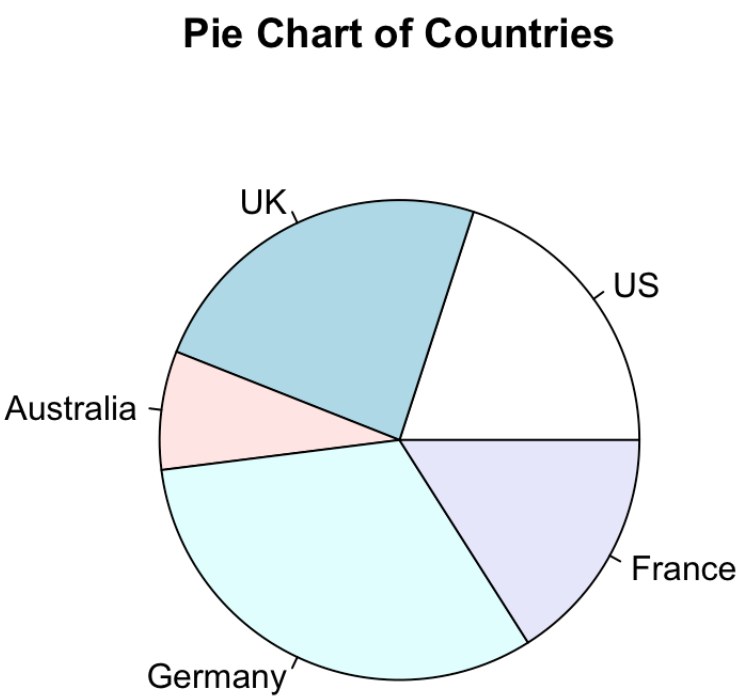
Grouped Bar Plot ==========

```
t <- table(mtcars$gear, mtcars$cyl)
barplot(prop.table(t,2), legend=paste(unique(mtcars$gear), "gears"), ylab="Probability", x
lab="cylinders", beside=TRUE)
```



Pie Charts ==========

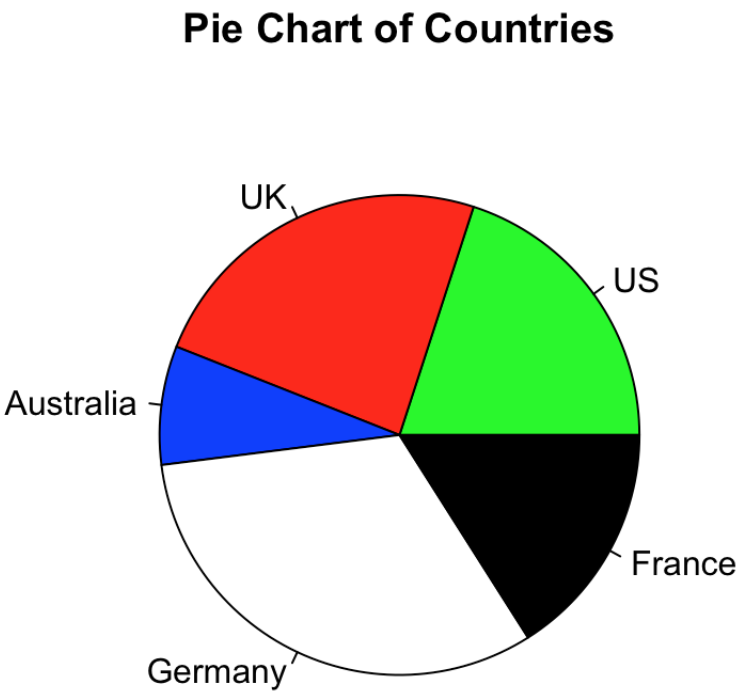
```
slices <- c(10, 12,4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
pie(slices, labels = lbls, main="Pie Chart of Countries")
```



Pie Charts

Custom Colours:

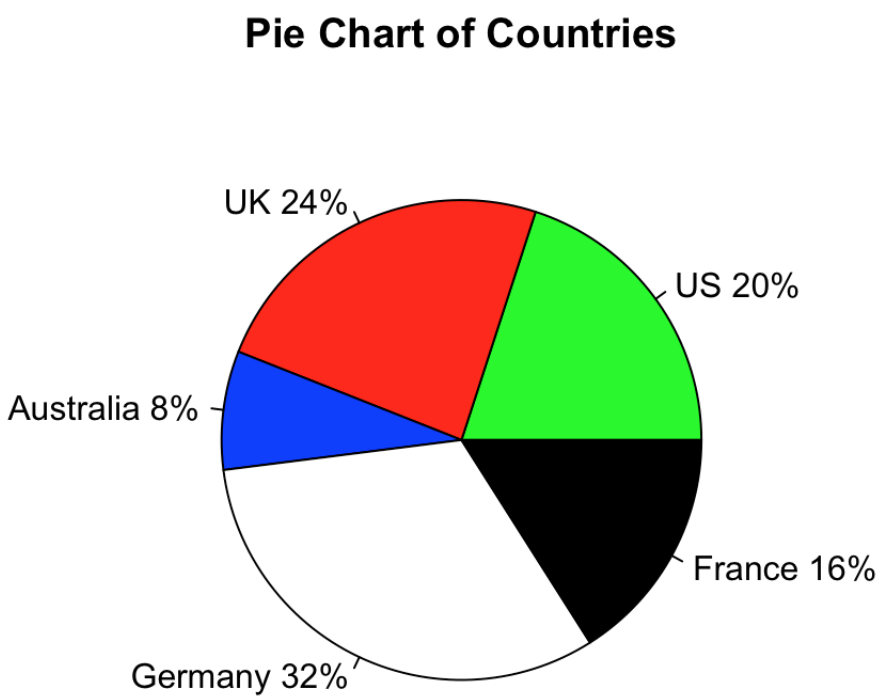
```
slices <- c(10, 12,4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
pie(slices, labels = lbls, main="Pie Chart of Countries", col=c("green", "red","blue","whi
te", "black"))
```



Pie Charts

Custom Labels:

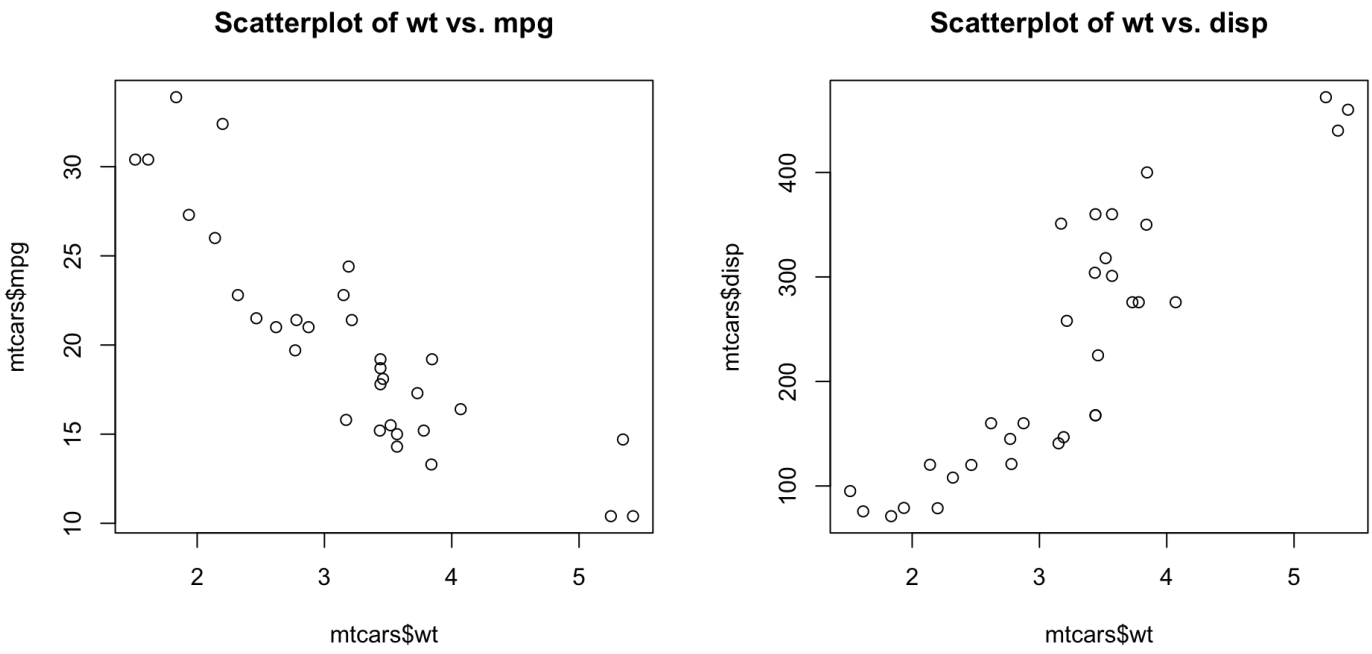
```
slices <- c(10, 12,4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
adv_lbls <- paste(paste(lbls, slices/sum(slices)*100), "%", sep="")
```



Subplots

Multiple plots shown on same figure:

```
par(mfrow=c(1,2))
plot(mtcars$wt,mtcars$mpg, main="Scatterplot of wt vs. mpg")
plot(mtcars$wt,mtcars$disp, main="Scatterplot of wt vs. disp")
```



Under the Hood

Graphical parameters

```
help(par)
```

Close a plot

```
graphics.off()
```

Custom Package: GGPLOT

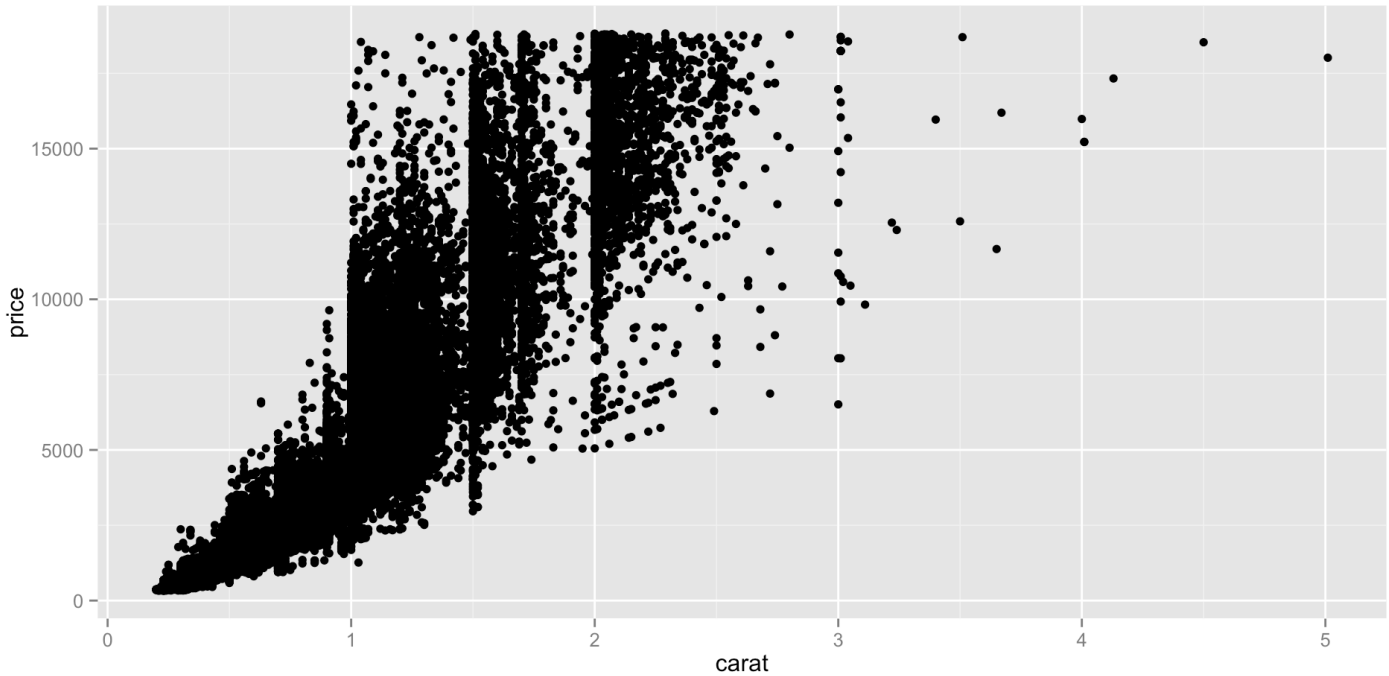
- Created by Hadley Wickham
- Used commonly to create publication quality graphics.

```
install.packages("ggplot2")
```

- Steeper learning curve.
- Can be used to plot complicated figures by chaining functions.

GGPLOT Examples

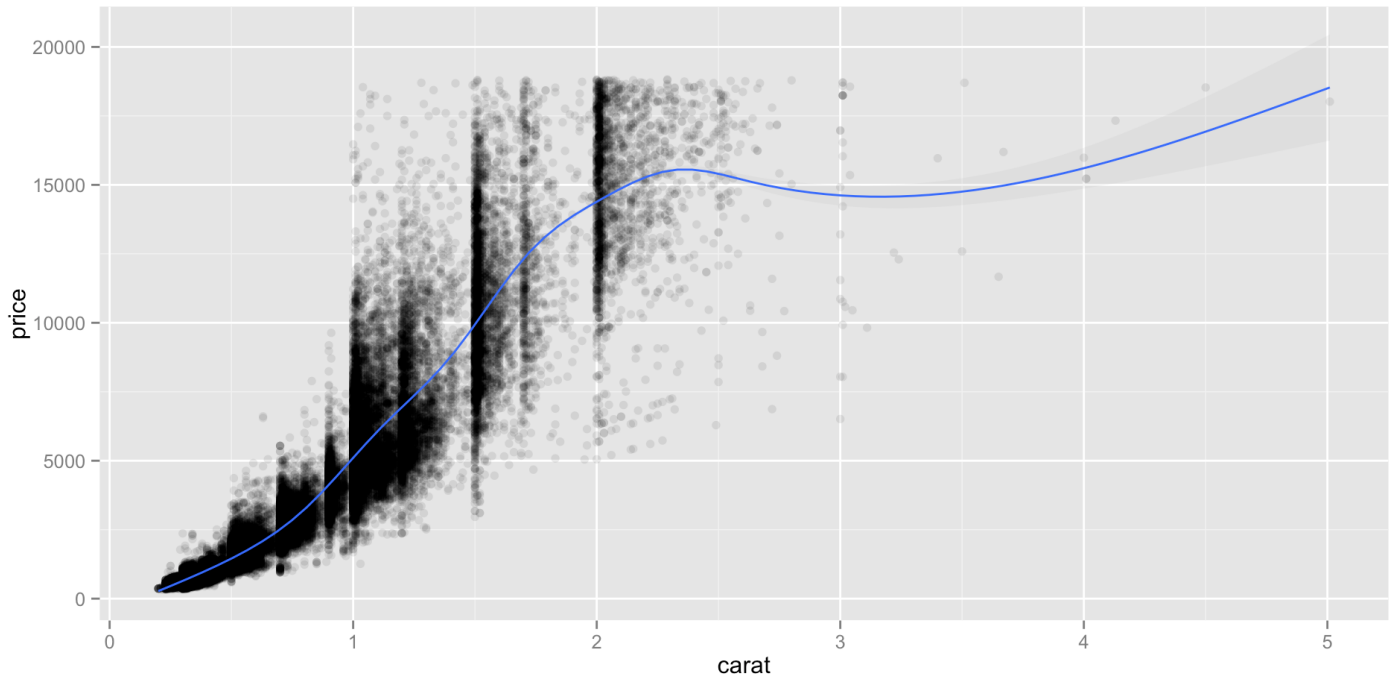
```
library("ggplot2")
qplot(carat, price, data = diamonds)
```



GGPLOT Examples ========== Fit a line, show error bar and change opacity.

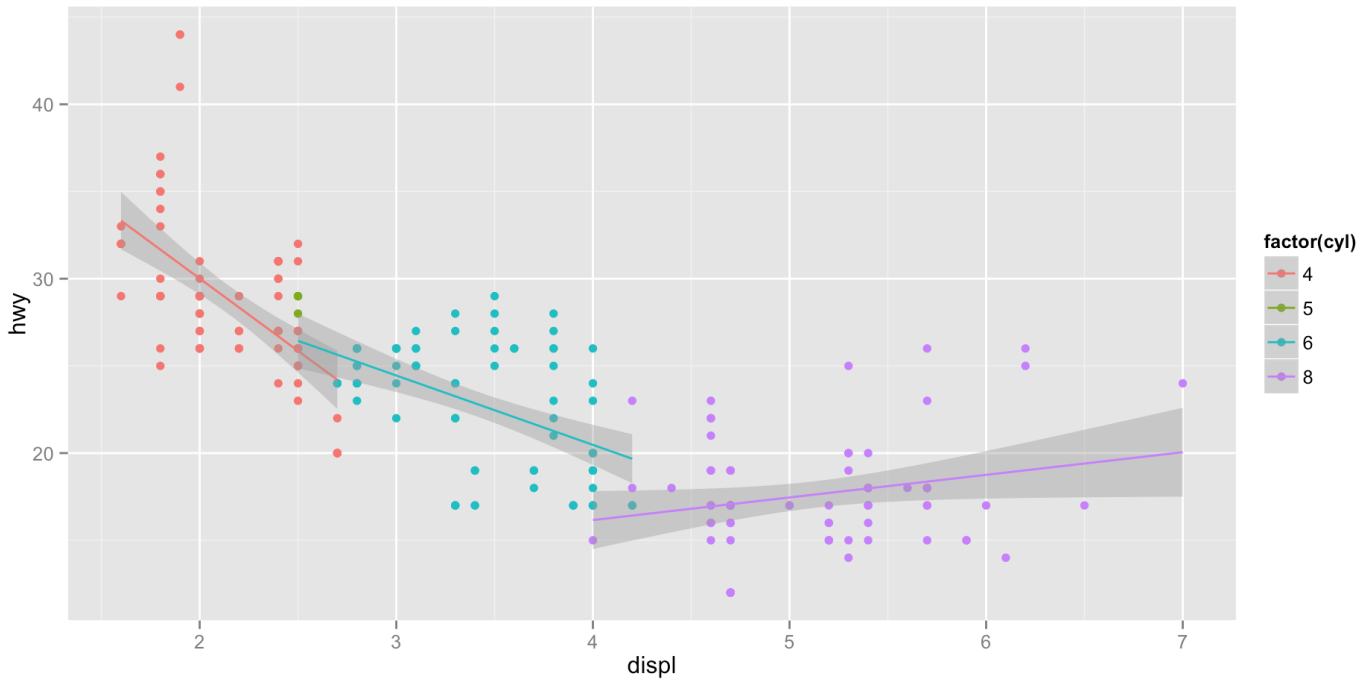
```
library("ggplot2")
qplot(carat, price, data = diamonds, geom = c("point", "smooth"), s, alpha = I(1/10))
```

```
## geom_smooth: method="auto" and size of largest group is >=1000, so using gam with formu
la: y ~ s(x, bs = "cs"). Use 'method = x' to change the smoothing method.
```



GGPLOT Examples

```
library("ggplot2")
ggplot(mpg, aes(displ, hwy, color = factor(cyl))) + geom_point() + stat_smooth(method = "l
m")
```



Showing Data on Maps

Downloading Area Maps using Google Maps API

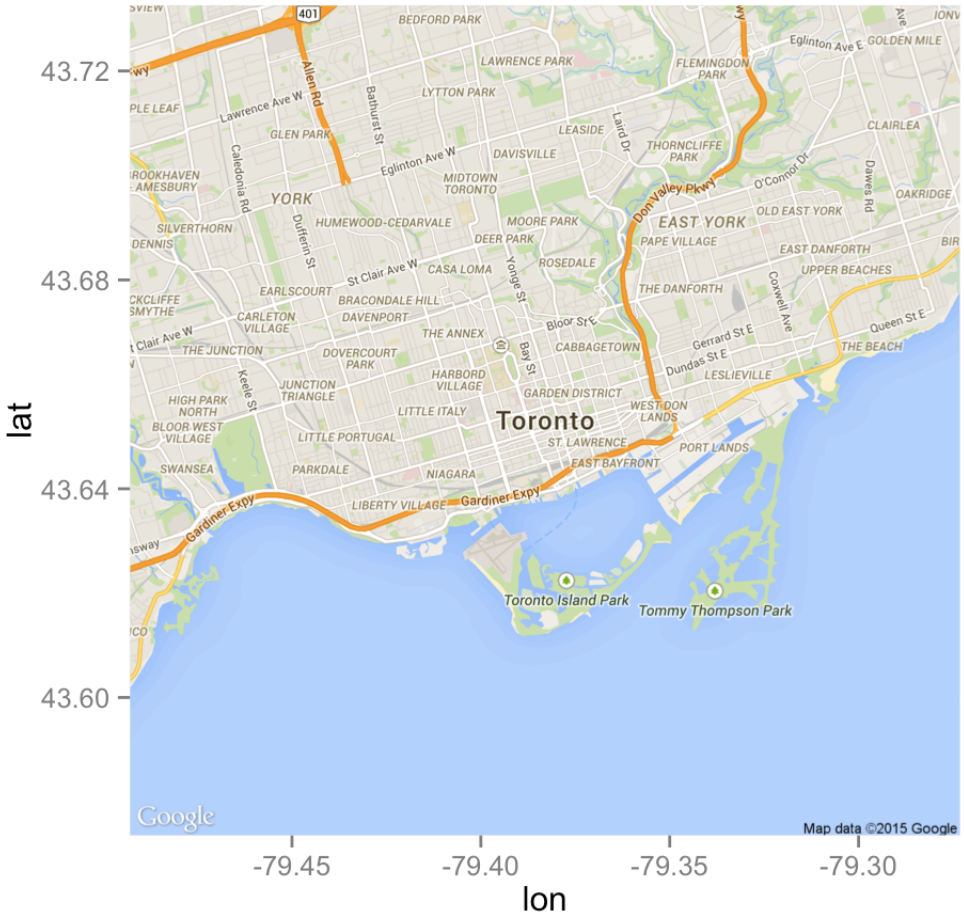
```
require("ggmap")
```

```
## Loading required package: ggmap
```

```
map <- get_map(location='Toronto', source="google", maptype="roadmap", zoom=12)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Toronto&zoom=12&size=%20640x640&scale=%202&maptype=roadmap&sensor=false
## Google Maps API Terms of Service : http://developers.google.com/maps/terms
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Toronto&sensor=false
## Google Maps API Terms of Service : http://developers.google.com/maps/terms
```

```
ggmap(map)
```



Showing Data on Maps

Finding Distances

```
from <- c("toronto", "toronto")
to <- c("montreal", "ottawa")
mapdist(from, to, mode = "driving")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/distancematrix/json?origins=
toronto&destinations=montreal%7cottawa&mode=driving&sensor=false
## Google Maps API Terms of Service : http://developers.google.com/maps/terms
```

```
##      from       to      m      km    miles seconds  minutes    hours
## 1 toronto montreal 540535 540.535 335.8884   19010 316.8333 5.280556
## 2 toronto   ottawa 449183 449.183 279.1223   15463 257.7167 4.295278
```

Showing Data on Maps

Getting Locations of Important Landmarks using Google Maps API

```
geocode("cn tower")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=cn+towe
r&sensor=false
## Google Maps API Terms of Service : http://developers.google.com/maps/terms
```

```
##         lon      lat
## 1 -79.38706 43.64257
```

```
geocode("ryerson university")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=ryerson
+university&sensor=false
## Google Maps API Terms of Service : http://developers.google.com/maps/terms
```

```
##        lon      lat
## 1 -79.3788 43.65766
```

Showing Data on Maps

Showing Locations on The Maps

```
lon<- c(-79.38706, -79.3788)
lat<- c(43.64257, 43.65766)
label <- c("CN Tower", "Ryerson University")
df<-data.frame(lon,lat,label)
map <- get_map(location='Toronto', source="google", maptype="roadmap", zoom=12 )
```

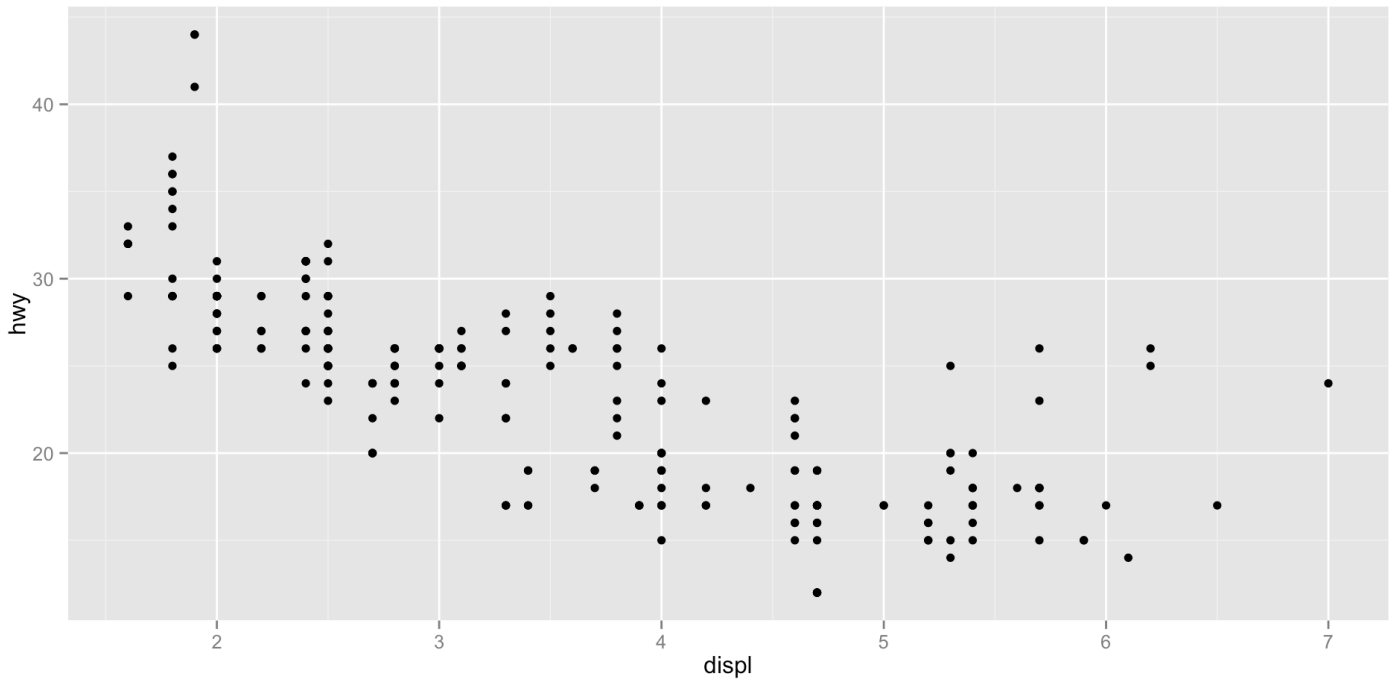
```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Toronto&zoom=12&size=%20640x640&scale=%202&maptype=roadmap&sensor=false
## Google Maps API Terms of Service : http://developers.google.com/maps/terms
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Toronto&sensor=false
## Google Maps API Terms of Service : http://developers.google.com/maps/terms
```

```
ggmap(map) + geom_point(data = df,
                  aes(x = lon, y = lat),
                  fill = "red",
                  colour = "black",
                  size = 3,
                  shape = 21) +
               geom_text(data = df,
                  aes(x = lon,
                      y = lat,
                      label = label),
                  color = 'blue')
```

Dealing with Many Data Points

- Jittering

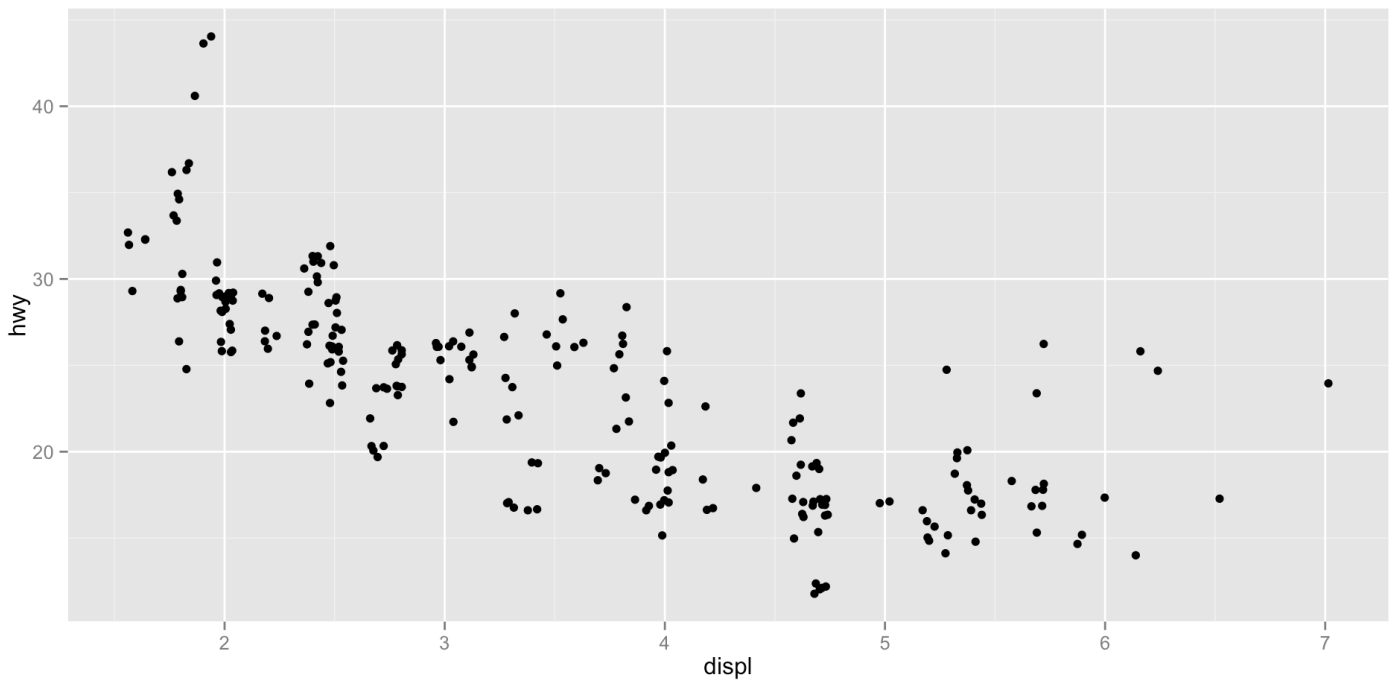
```
p <- ggplot(mpg, aes(displ, hwy))
p + geom_point()
```



Dealing with Many Data Points

- Jittering

```
p <- ggplot(mpg, aes(displ, hwy))
p + geom_point(position="jitter", width=0.05, height=1)
```



Dealing with Many Data Points

- Would plotting weight vs height of every person in the world be meaningful?
- Group large data before plotting.

- Use methods such as histograms to show the distributions.
- Do not forget: Computer screens have limited pixels. Humans have limited cognitive capability.

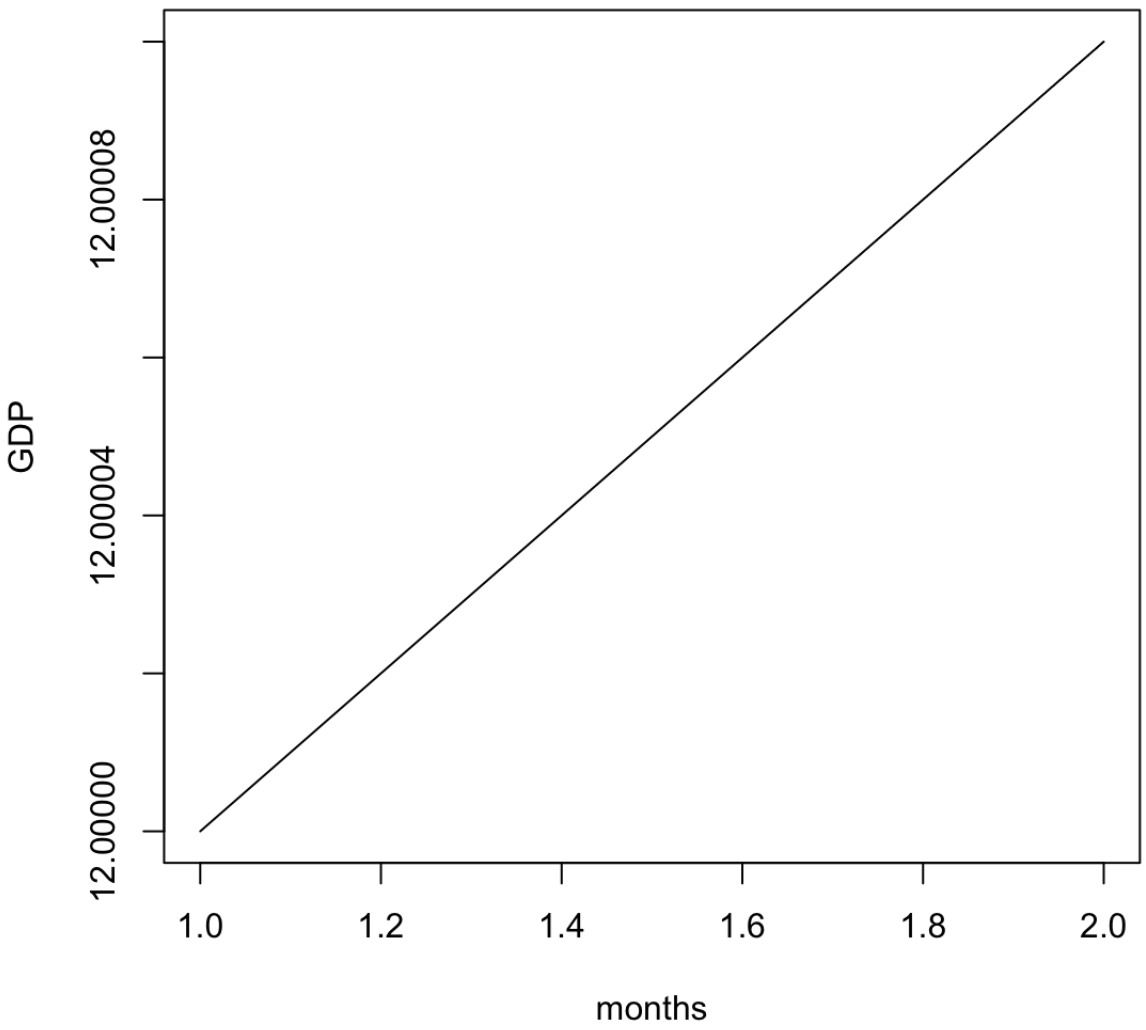
A Few Tips

- Use colours sparingly. - Use multiple subplots for complex data. - Use 3d plots sparingly. - Use jittering with large data. For very large datasets you may try summarizing the data before visualizing it. - Do not forget that computer screen can show just a couple of million pixels.

A Few Tips

- Relative axis limits might make small changes look dramatic. Avoid that if possible.
- A good plot should not be deceiving.

```
plot(c(1,2), c(12, 12.0001), type = "l", ylab="GDP", xlab="months")
```



Further References

- ggplot tutorial: https://www.youtube.com/watch?v=TaxJwC_MP9Q (https://www.youtube.com/watch?v=TaxJwC_MP9Q) - R Graphics Cookbook - Winston Chang - O'REILLY - http://www.cookbook-r.com/Graphs/ (http://www.cookbook-r.com/Graphs/) - GGmap - http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf (http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf) - Check web for some really cool examples. - The Visual Display of Quantitative Information, 2nd edition - Edward R. Tufte - Interactive visualization with Shiny: http://shiny.rstudio.com/ (http://shiny.rstudio.com/) - Some Public Datasets: http://vincentarelbundock.github.io/Rdatasets/datasets.html (http://vincentarelbundock.github.io/Rdatasets/datasets.html)

Lab Session

Preparation

- Load the computer price data (doc) (http://vincentarelbundock.github.io/Rdatasets/doc/Ecdat/Computers.html):

```
library(RCurl)
u <- getURL("http://vincentarelbundock.github.io/Rdatasets/csv/Ecdat/Computers.csv")
c_prices <- read.csv(text = u)
```

- Load Sales Data of Men's Fashion Stores (doc) (http://vincentarelbundock.github.io/Rdatasets/doc/Ecdat/Clothing.html):

```
library(RCurl)
u <- getURL("http://vincentarelbundock.github.io/Rdatasets/csv/Ecdat/Clothing.csv")
c_sales <- read.csv(text = u)
```

- Save your plots as images after you solve a problem.

Lab Session

Part 1

- Plot computer price vs harddisk space. How can you deal with data points with same values?
- Plot a pie chart of computer count with different amount of RAM.
- Show relation of every pair of variables in the clothing sales dataset on a single plot..
- Plot a histogram of computer harddisk storage spaces.

Lab Session

Part 2

- Compare the prices of computer with different amount of ram using box plot.
- Is the price distribution of computers similar to a normal distribution?
- Show Pisa Tower on a map.
- Find the walking distance between New York City and Los Angeles using ggmaps.