R Lab Notebook

Business Analytics Foundations

Author:

PRACHI MITTAL

Contents

[1. Introduction to Business Analytics 4](#_Toc481177848)

[Computing power is a driver of analytics 4](#_Toc481177849)

[The methods of business analytics 4](#_Toc481177850)

[2. Introduction to R programming 5](#_Toc481177851)

[History of R 6](#_Toc481177852)

[What is an R package? 8](#_Toc481177853)

[Advantages of R 9](#_Toc481177854)

[Challenges of R 9](#_Toc481177855)

[R studio – Integrated Development Environment (IDE) for R 10](#_Toc481177856)

[The emergence of R studio 11](#_Toc481177857)

[Windows of RStudio 11](#_Toc481177858)

[3. Getting Started with R 12](#_Toc481177859)

[Basic commands in R 12](#_Toc481177860)

[Creating and displaying variables in R 12](#_Toc481177861)

[Precision 13](#_Toc481177862)

[Concatenation in R using c() function 13](#_Toc481177863)

[Operations on vectors 14](#_Toc481177864)

[Mathematical and logical operations with R objects 15](#_Toc481177865)

[Basic Datatypes in R 16](#_Toc481177866)

[Special values in R 17](#_Toc481177867)

[Objects in R 17](#_Toc481177868)

[Vector 18](#_Toc481177869)

[Matrix 19](#_Toc481177870)

[Data Frame 19](#_Toc481177871)

[List 20](#_Toc481177872)

[Factor 21](#_Toc481177873)

[4. Data Visualization in R 22](#_Toc481177874)

[Selected Chart Types in R 22](#_Toc481177875)

[Strip Charts 23](#_Toc481177876)

[Dot Charts (aka “Dot Plots”) 23](#_Toc481177877)

[Box Plots (aka "box and whisker plots") 24](#_Toc481177878)

[Histograms 25](#_Toc481177879)

[Density Plots 26](#_Toc481177880)

[Scatter Plots 27](#_Toc481177881)

[Bar Charts 28](#_Toc481177882)

[Line Charts 29](#_Toc481177883)

[Heat Maps 30](#_Toc481177884)

[QQ (Quantile-Quantile) Plots 31](#_Toc481177885)

[Scatterplot Matrices 31](#_Toc481177886)

[Corrgrams 32](#_Toc481177887)

[Mosaic Plots 33](#_Toc481177888)

[Bubble Plots 35](#_Toc481177889)

[Chloropleth Maps 36](#_Toc481177890)

[Violin Plots 37](#_Toc481177891)

[Frequency Ploygons 38](#_Toc481177892)

[Tree Maps 39](#_Toc481177893)

[Pie Charts 40](#_Toc481177894)

[5. Preparing data with R 41](#_Toc481177895)

[A general approach to prepare data 42](#_Toc481177896)

[6. Supervised Analytics Techniques 43](#_Toc481177897)

[Introduction to Regression 43](#_Toc481177898)

[Application of Regression 43](#_Toc481177899)

[Linear Regression 44](#_Toc481177900)

[Simple Regression 44](#_Toc481177901)

[Multiple Regression 46](#_Toc481177902)

[Stepwise Regression 47](#_Toc481177903)

[Linearity Assumption 49](#_Toc481177904)

[Assessing Regression Models 50](#_Toc481177905)

[Interpreting R2 50](#_Toc481177906)

[Measuring Prediction Accuracy 51](#_Toc481177907)

[Average Error 51](#_Toc481177908)

[Mean Absolute Error 51](#_Toc481177909)

[Mean Absolute Percent Error (MAPE) 52](#_Toc481177910)

[Root Mean Squared Error 52](#_Toc481177911)

[Regression Diagnostics 52](#_Toc481177912)

[Examining Residuals 53](#_Toc481177913)

[Influencial Observations 53](#_Toc481177914)

[Collinearity 56](#_Toc481177915)

[Logistic Regression 57](#_Toc481177916)

[Evaluating Model Performance- Basic Measures 63](#_Toc481177917)

[Evaluating Model Performance- ROC, Lift and Gain Curves 65](#_Toc481177918)

[K-Nearest neighbors 68](#_Toc481177919)

[Naïve Bayes 72](#_Toc481177920)

[Decision Trees 75](#_Toc481177921)

[Ensemble Models 85](#_Toc481177922)

[Bagging 85](#_Toc481177923)

[Boosting 86](#_Toc481177924)

[Random Forests 87](#_Toc481177925)

[Ensemble Models in R 88](#_Toc481177926)

[7. Unsupervised Techniques 89](#_Toc481177927)

[Handling Unbalanced Data 89](#_Toc481177928)

[Solutions for Imbalanced Data 89](#_Toc481177929)

[Cluster Analysis 90](#_Toc481177930)

[Hierarchical Clustering 92](#_Toc481177931)

[Non- hierarchical Clustering 105](#_Toc481177932)

[Principal Component Analysis 108](#_Toc481177933)

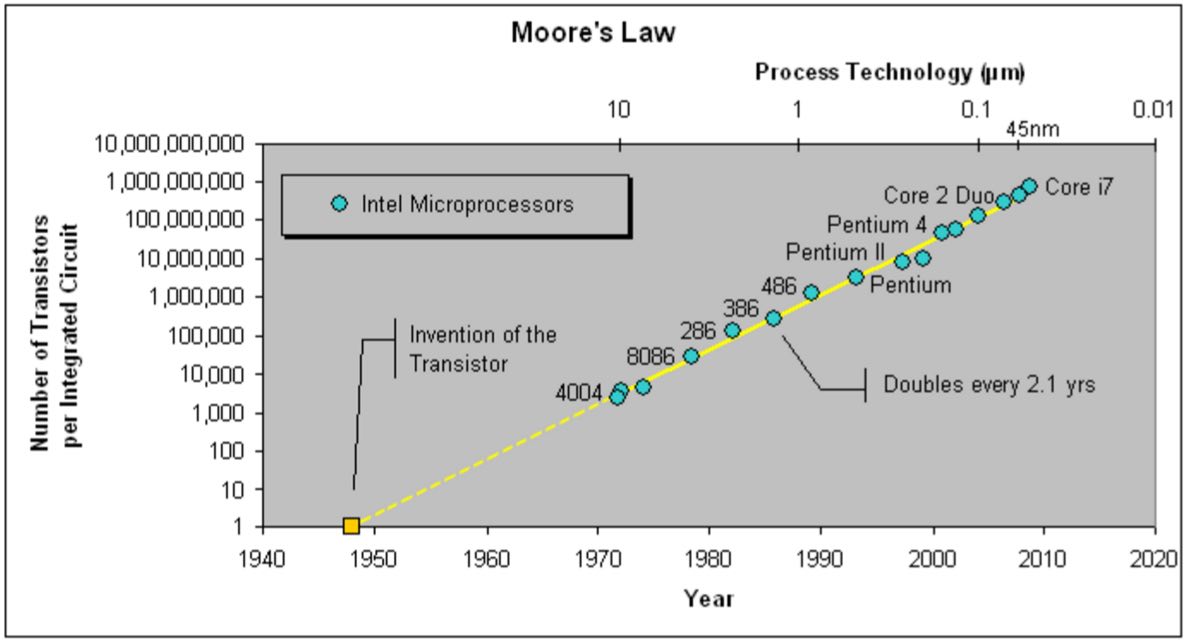
[Using Principal Component Analysis 111](#_Toc481177934)

[Some mathematics of Principal Components Analysis 114](#_Toc481177935)

# Introduction to Business Analytics

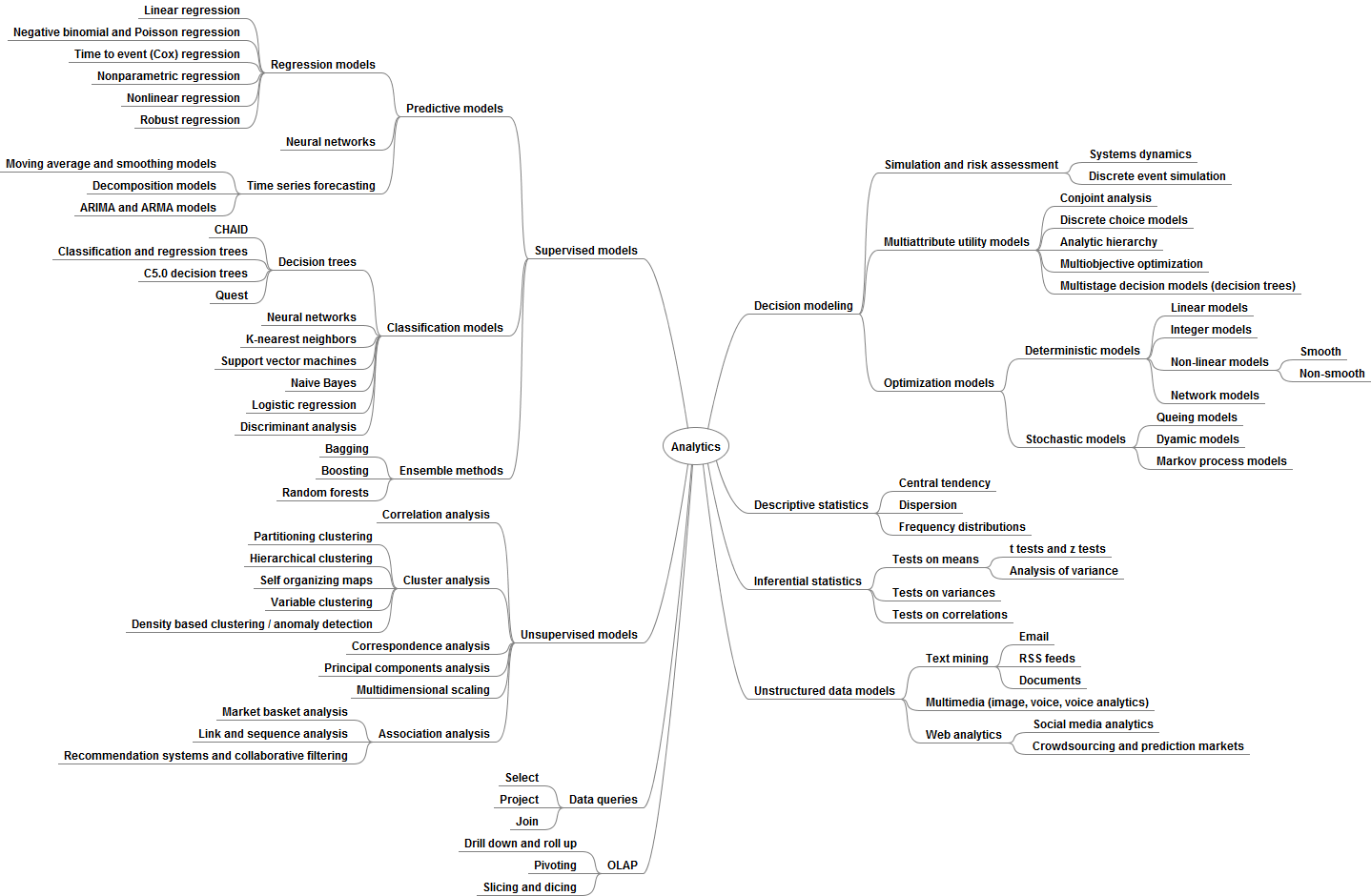
Business analytics (BA) is the practice of iterative, methodical exploration of an organization’s data with emphasis on statistical analysis. Business analytics is used by companies committed to data-driven decision making. Analytics is ...”the extensive use of data, statistical and quantitative analysis, explanatory and fact-based management to drive decisions and actions.” (Davenport and Harris, Competing on Analytics)

## Computing power is a driver of analytics

****

## The methods of business analytics

There are various methods of Business Analytics, including R Programming, which we will talk about in the next section.



In summary, the goal of business analytics is:

1. To add material value to organizations
   * Through the use of advanced data mining, pattern matching, data visualization and predictive modeling tools as well as dashboards, scorecards, and interactive reports.
2. To drive fact-based decision making and informed action.

# Introduction to R programming

R is an implementation of the S programming language combined with lexical scoping semantics inspired by Scheme. S was created by John Chambers while at Bell Labs. There are some important differences, but much of the code written for S runs unaltered.

* R is an *extensible* programming language which provides a wide variety of graphical and statistical techniques.
* It is the most widely used software environment for data science according to many polls.
* It is extensible because the functionalities can be added by using additional packages.
* A "package" in R is a collection of functions and data sets.

## History of R

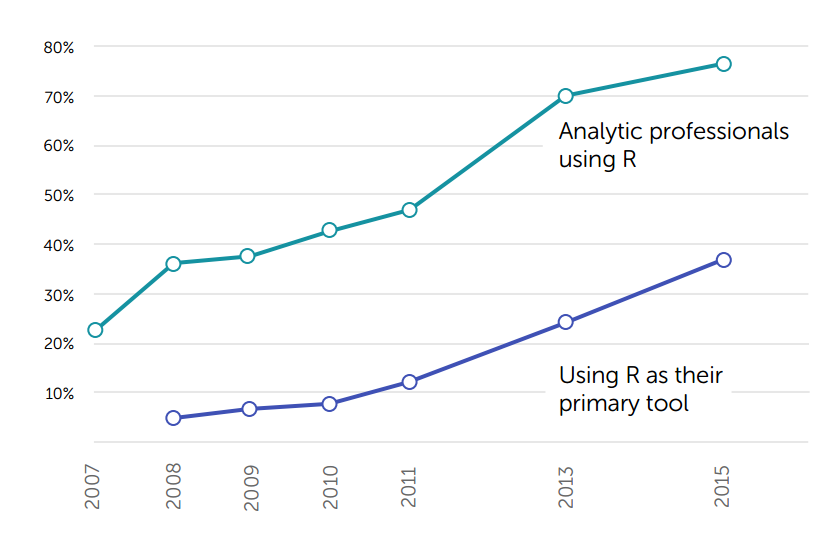
R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team, of which Chambers is a member. R is named partly after the first names of the first two R authors and partly as a play on the name of S. The project was conceived in 1992, with an initial version released in 1995 and a stable beta version in 2000.

R is a dialect of S. S was a language developed by John Chambers and at the now-defunct Bell Labs. It was initiated in 1976 as an internal statistical analysis environment, an environment that people at Bell Labs could use to analyze data. Initially it was implemented as a series of FORTRAN libraries to kind of implement routines that were tedious to have to do repeatedly, so there were FORTRAN libraries to repeat these statistical routines. Early versions of the language did not contain functions for statistical modelling. That did not come until roughly version three of the language. In 1988, the system was rewritten in the C language and to make it more portable across systems and it began to resemble the system that we have today. This was version three. And there was a seminal book the, called the Statistical Models in S written by John Chambers and Trevor Hastie. Sometimes referred to as the white book. And that documents, all the statistical analysis functionality that came into the version, that version of the language. Version four of the S language was released in 1998. And it's the version we use today. The book Programming with Data written by John Chambers sometimes called the green book documents version four of the S language. So, R is an implementation of the S language, that was originally developed in Bell Labs.

In 1993 Bell Labs gave a corporation called StatSci which became Insightful Corporation, an exclusive license to develop and sell the S language. In 2004, Insightful purchased the S language completely from Lucent. Bell Labs became Lucent Technology for $2 million, and became the current owner. In 2006, Alcatel purchased Lucent Technologies and it's now called Alcatel-Lucent. Insightful developed a product which was an implementation of the S language under the product name S-PLUS. And they built several fancy features into it for example graphical user interfaces and all kinds of a nice tools. that, so that's where the plus comes from in S-PLUS. In 2008 the Insightful Corporation was acquired a company called TIBCO for $25 million dollars and that's where it stands. TIBCO still develops as PLUS, although in a variety of different types of business analytic type products. And it continues to this day. The history of the language is a little bit tortured because of the various corporate acquisitions but it still survives to this day. The fundamentals of the S language have not changed since 1998 and the language that existed in 1998 looks like we, like what we use today at least superficially. And it's worth nothing that in 1998 the S language won the association for repeating machinery software system award.

In 1993, **R**oss Ihaka and **R**obert Gentleman at the University of Auckland experimented with a language they called R, similar to S. The first stable version of R was released in 2000, called version 1.0.0. R is now available for Mac, Windows and Unix systems. The latest version of R, called “Sincere pumpkin patch” was released in October, 2016.

**The rise of R usage can be shown by the figure below:**



Source: Rexer, 2015

## What is an R package?

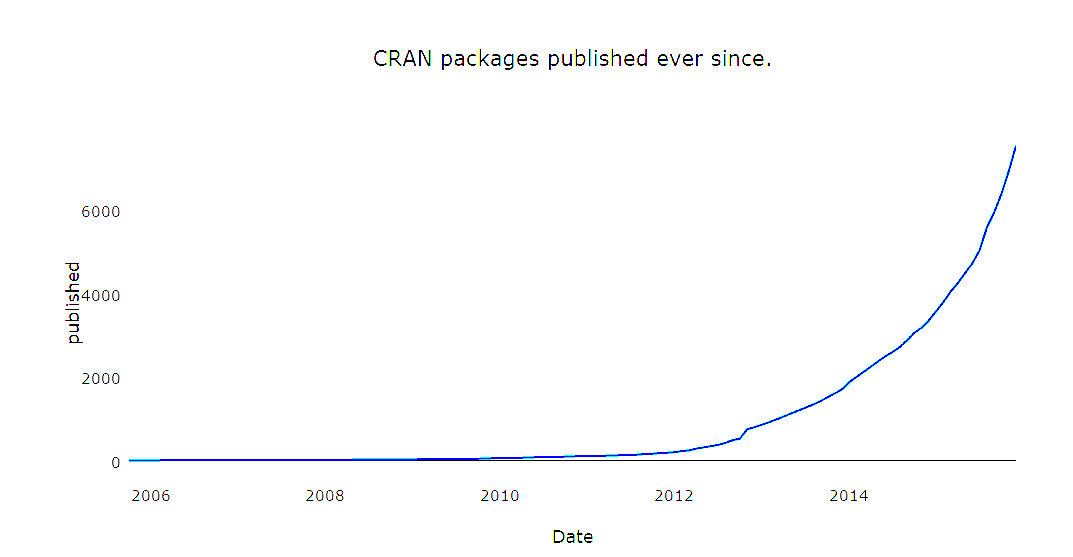
An R Package is a collection of R functions with comprehensive documents.

A package includes:

* R functions
* Data examples
* Help files

The default installation of R includes only a small set of packages. To extend the capabilities of R, packages must be downloaded and installed.

Exponential growth in R packages can be shown by the figure below-



Source: <https://rpubs.com/filipstachura/rpackages>

## Advantages of R

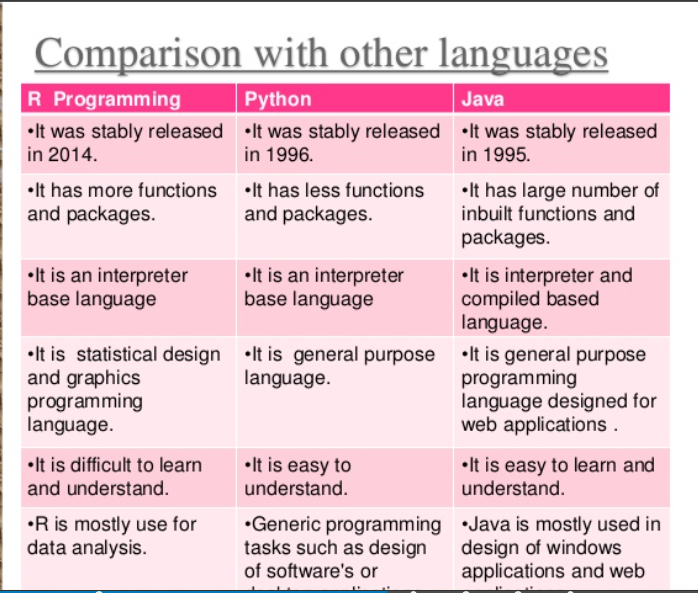
* R is free and an open source software.
* R is cross platform, it runs on many operating systems, including Macintosh.
* R has approximately 5000 cutting edge packages, providing the most comprehensive set of modeling tools available.
* R has many active groups which make the information on R easily accessible.

(See <https://www.r-bloggers.com/blogs-list/> for hundreds of R blogs.)

* R has outstanding graphical capabilities providing a totally programmable graphical package better than most other statistical packages.

## Challenges of R

* R is continually changing.
* R is slow; all objects are in memory.
* Complex data structures.
* Analyses that are run do not typically produce visible results automatically.
* Packages need to be loaded each time R is run.
* R syntax can present problems, making the learning curve fairly steep. (Syntax does not seem logical or consistent at times.)
* Thousands of individual and independent R developers with nobody responsible for the code.

**Comparison of R with other languages**

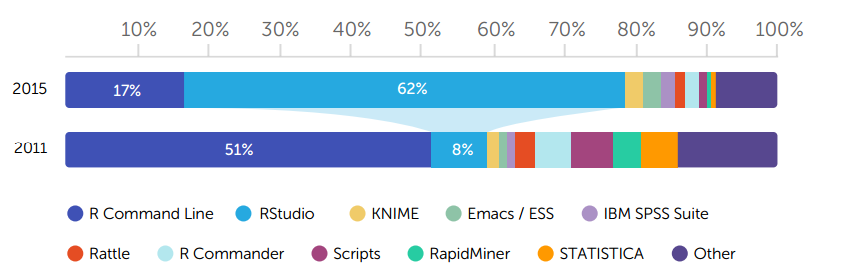
## R studio – Integrated Development Environment (IDE) for R

* RStudio is a free and open source integrated development environment (IDE) for R
* It makes programming very user friendly by providing an interface
* Installing R and RStudio is very easy and simple.
* An installation guide is available for both these software with this presentation



## The emergence of R studio

Data scientists interface with R in a variety of ways. However, in recent years RStudio has become the dominant R interface. In the 2015 survey 62% of R users report using RStudio, up from just 8% in 2011. In 2015 only 17% of R users report using the R command line interface, down from 51% in 2011. Small groups of R users report using a variety of other interfaces, such as RKWard, Revolution R GUI, and Rattle. Others report using R from within other tools such as IBM SPSS, STATISTICA and KNIME.



Source: Frank Acito, Business Analytics Foundations, Getting Started with R

### Windows of RStudio



# Getting Started with R

## Basic commands in R

* help() or ? or ??

help(boxplot)

* q()

To close R

* ls()

Lists all user defined objects.

* rm()

Removes objects from an environment.

E.g. rm(x) removes the user defined object “x” from the memory

E.g. rm(x,y) removes the user defined objects “x” and “y” from the memory

* demo()

Lists the demonstrations in the packages that are loaded.

E.g. demo(package = .packages(all.available = TRUE))

Lists the demonstrations in all installed packages.

* # to add comments

# These are single line comments.  
# Each new line requires another hash sign.

## Creating and displaying variables in R

* Use ‘<-’ symbol to create variables
  + ‘=‘ will usually work, but is considered bad form.
  + E.g. a<-3, b<-“x”, c=4
* The assignment symbol tells R to store the value at the right into the variable to the left.
* To display the value, type the variable and hit enter.
  + E.g. a will give an output [1] 3
  + The [1] in the output above indicates the index of the that value in the vector.

## Precision

* The options() function can set the precision of the output.
* pi is a pre-defined variable with the value of pi.
* E.g. options(digits = 22) will set the precision to 22 digits for the output

## Concatenation in R using c() function

* c() is a function that combines its arguments.
* The default method combines its arguments to form a vector.
* All arguments are coerced to a common type which is the type of the returned value, and all attributes except names are removed.
* The output type is determined from the highest type of the components in the hierarchy, e.g.,

NULL< logical < integer < real < complex < character

* E.g. w <- c(TRUE,3,"22") creates a variable called “w” and stores 3 arguments, “TRUE”, “3” and “22”. Note that all the values are implicitly coerced to a character since it has the highest priority or in other word, forms a superset. “w” in this case in a character vector.
* E.g. x <- c(TRUE, TRUE,FALSE) creates a variable called “x” and stores three arguments- TRUE, TRUE and FALSE. Note that in this case all the arguments were of the same form i.e. logical hence there was no need for coercion. “x” in this case is a logical vector.
* c() can also be used to combine variables
* E.g. x1 <- c(4,8,1:20), x2 <- c(34), x3 <- c(5,6,7,8:25), y <- c(x1,x2,x3) Here y just combines all the values pf x1, x2 and x3

**rbind(), cbind(), dim(), nrow(), ncol() functions in R**

* E.g. x1<- c(1,2,3,4)

x1

x2 <- c("a","b","c","d")

x2

x3 <- rbind(x1,x2)

x3

x4 <- cbind(x1,x2)

Here rbind() just combines the vectors x1 and x2 one below the other, i.e. row wise. The resultant vector will have 2 rows and 4 columns.

* Continuing the same example, dim(x3) will give output as [1] 2 4 i.e. 2 rows, 4 columns
* To find the number of rows, use nrow(x3) which will give the output as [1] 2
* To find the number of columns, use ncol(x3) which will give the output as [1] 4

And cbind() just combines the vectors x1 and x2 adjacent to the other, i.e. column wise. The resultant vector will have 4 rows and 2 columns.

* Continuing the same example, dim(x4) will give output as [1] 4 2 i.e. 4 rows, 2 columns
* To find the number of rows, use nrow(x4) which will give the output as [1] 4
* To find the number of columns, use ncol(x4) which will give the output as [1] 2

## Operations on vectors

* Use brackets to refer to specific values in a vector

Eg:

x <- c(1:10)

y <- x[3]

yy <- x[4:5]

z <- x[-2]

x\_sum <- sum(x)

xx <- c(x,NA)

sum(xx)

sum(xx, na.rm=TRUE)

## Mathematical and logical operations with R objects

# An addition

5 + 5 gives [1] 10

# A subtraction

5 - 5 gives [1] 0

# A multiplication

3 \* 5 gives [1] 15

# A division

(5 + 5) / 2 gives [1] 5

# Exponentiation

2^5 gives [1] 32

# Modulo

28%%6 gives [1] 4

abs(x) takes the absolute value  
sqrt(x) takes the square root  
cos(x), sin(x), tan(x) computes the trigonometric functions  
log(x) computes the natural logarithm  
log10(x) computes the logarithm to base 10  
exp(x) computes e^x

E.g. Assign a value to the variables my\_apples and my\_oranges and add them and store the value to my\_fruit variable

my\_apples <- 5

my\_oranges <- 6

my\_apples + my\_oranges gives [1] 11

# Create the variable my\_fruit

my\_fruit <- my\_apples + my\_oranges

my\_fruit gives [1] 11

Apart from the above arithmetic operators, R also has logical operators as shown below:

|  |  |
| --- | --- |
| Less than: | < |
| Less than or equal to: | <= |
| Greater than: | > |
| Greater than or equal to: | >= |
| Exactly equal to: | == |
| Not equal to: | != |
| Not x: | !x |
| x or y: | x|y |
| x and y: | x & y |
| Is x TRUE? | isTRUE(x) |

## Basic Datatypes in R

Datatypes or classes in R include logical, integer, double, complex, character, symbol. All of these datatypes will be used in later examples as programs.

class() function returns the class of a variable.

E.g. my\_numeric <- 42

my\_character <- "universe"

my\_logical <- FALSE

# Check class of my\_numeric

class(my\_numeric) returns numeric

# Check class of my\_character

class(my\_character) returns character

# Check class of my\_logical

class(my\_logical) returns logical

Further, datatypes can be set by using is() and as() functions as follows:

x <- TRUE

is.logical(x) returns TRUE

as.character(x) converts x to “TRUE”

### Special values in R

* NA stands for "not available".
* NA is reduced as an answer to an expression whenever one of the inputs is NA. NA means "missing value." To avoid this, use na.rm=TRUE.
* NULL is "the empty set." NULL has 0 length and no class (it's class is NULL). NULL means "does not exist."
* Inf stands for "infinity". (This only applies to vectors of class numeric.)
* NaN stands for "not a number"
* E.g.

x <- c(1,2,3,4,5,NA)

sum(x) will return [1] NA

whereas sum(x, na.rm = TRUE) will return [1] 15

## Objects in R

* Objects in R are specialized data structures.
* Each object possesses a **type**, a **mode**, and a **storage mode**.
  + Examples of **types**: NULL, symbol, builtin, logical, integer, double, complex, character
  + Examples of **modes**: numeric, complex, logical, character, list, function
  + Examples of **storage modes**: logical, integer, double, complex, character, symbol
* Examples of objects:
  + vectors, arrays
  + matrices
  + data frames
  + lists
  + factors

### Vector

* A vector is a sequence of elements of the same basic type.
  + Each member of the vector is called a component or member.
* Vectors can be created using the vector() function or by using c().
* Vectors have three attributes: type, length, and attributes.
  + These can be obtained using the functions typeof(), length() and attributes().

E.g. x1 <- c(2,3,5)

length(x1) returns [1] 3

typeof(x1) returns [1] “double”

attributes(x1) returns NULL

names(x1) <- c(“a”, ”b”, ”c”)

attributes(x1) returns $names [1] “a” “b” “c”

Retrieving values from a vector can be done in the following ways:

* Using an index

s = c("aa", "bb", "cc", "dd", "ee")   
 s[3]   
 [1] "cc"

* Using a negative index

s[-3]   
 [1] "aa" "bb" "dd" "ee"

* Using an out of range index

s[10]   
 [1] NA

* Using a range index

s[2:4]   
 [1] "bb" "cc" "dd"

### Matrix

* A matrix is a collection of data elements arranged in a two-dimensional rectangular layout.
* All columns in a matrix must have the same basic type (numeric, character, logical, etc.) and the same length.
* Data is filled column wise, starting from column 1, and going down, then to the next column by default.
* mymatrix <- matrix(*vector*, nrow=*r*, ncol=*c*, byrow=*FALSE*,   
   dimnames=list(*char\_vector\_rownames*, *char\_vector\_colnames*))

byrow=FALSE indicates that the matrix should be filled by columns (the default).

byrow=TRUE indicates that the matrix should be filled by rows.

dimnames provides optional labels for the columns and rows.

E.g. y<-matrix(1:20, nrow=5,ncol=4) generates a matrix of 5 rows and 4 columns.

y[2,3] returns the element in the 2nd row and 3rd column of the matrix

### Data Frame

* A data frame is more general than a matrix, in that different columns can have different type of data (numeric, character, factor, etc.).
* The data frame is going to be the most common data structure we will use in R, since it is an n X m table that contains numbers, characters, and factors.

Note: each column must have only one type of data.

* To create a data frame, combine a series of column vectors.

E.g., mydataframe <- data.frame(col1, col2, col3, ..)

#Example of creating a data frame.

patientID <- c(1,2,3,4)

age <- c(25,34,28,52)

diabetes <- c("Type1", "Type2", "Type1","Type1")

status <- c("Poor", "Improved", "Excellent", "Poor")

patientdata <- data.frame(patientID, age, diabetes, status)

patientdata

str(patientdata)

Note: str() function returns the structure of the underlying object. Here it returns “data.frame” along with the column names and data types of the columns in the data frame.

In the same example, if we want to subset the data frame,

patientdata[1,1] # Entry of 1st row, 1st column

patientdata[1:3] # Columns 1 through 3

patientdata[,3] # All rows of column 3

patientdata[3,] # Row 3, all columns

### List

* Lists are the most complex of the R data structures.
* A list is an ordered collection of objects, which may be of different types.

mylist <- list(object1, object2, ...) forms a list.

* E.g.

#Example of creating a list

g <- "My First List"

h <- c(25, 26, 18, 39)

j <- matrix(1:10, nrow=5)

k <- c("one", "two", "three")

mylist <- list(title=g, ages=h, j, k)

mylist$title returns [1] “My First List”

mylist[[3]] returns

[,1] [,2]

[1,] 1 6

[2,] 2 7

[3,] 3 8

[4,] 4 9

[5,] 5 10

### Factor

* Definition: A factor is a variable in R that takes on a small number of values, aka, "categorical variables."
* Factor variables can be numeric or strings (characters).
* Factors are used to classify cases.
* Factors can be ordered or unordered.
* Factors are treated differently from continuous variables in analyses.

Note: Factor variables used in statistical models will be implemented correctly in terms of assigning the number of degrees of freedom

* E.g.

ses <- c("low", "middle", "low", "low", "low", "low", "middle", "low",

"middle", "middle", "middle", "middle", "middle", "high", "high",

"low", "middle", "middle", "low", "high")

is.factor(ses) will return [1] FALSE

is.character(ses) will return [1] TRUE

ses.factor <- factor(ses)

is.factor(ses.factor) will return [1] TRUE

levels(ses.factor) will return [1] “high” “low” “medium”

# Data Visualization in R

Graphs for exploratory data analysis are used to suggest hypotheses about the causes of observed phenomena, assess assumptions on which statistical inference will be based, support the selection of appropriate statistical tools and techniques, provide a basis for further data collection through surveys or experiments, maximize insight into a data set, uncover underlying structure, extract important variables and detect outliers and anomalies.

## Selected Chart Types in R

Some principles of visualization for *presentation*

* Good visualizations start with good data and detailed analysis
* Good visualizations directly answer specific, focused questions
* Good visualizations get out of the way of the data; remove excessive clutter.

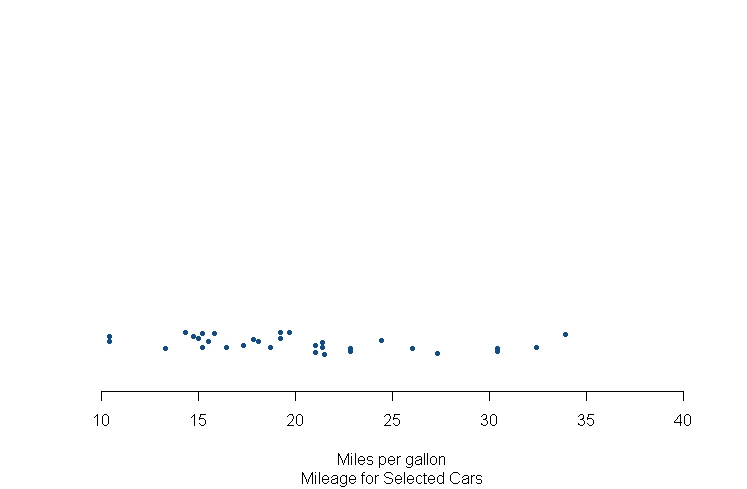
Guidelines:

* Maximize the data-ink ratio.
* Remove non-data-ink.
* Remove redundant data-ink.

Some common types of charts in R, are shown below:

### Strip Charts

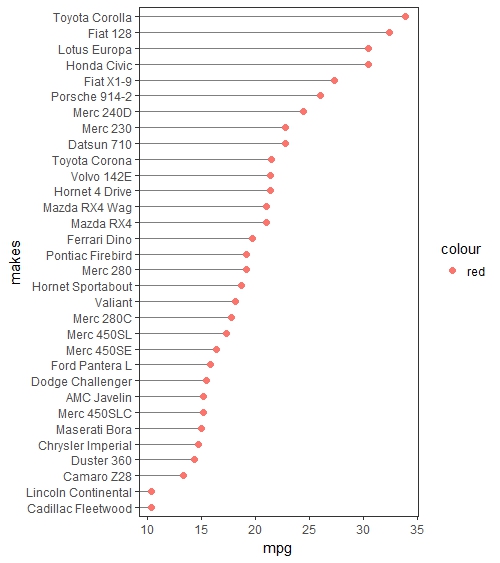
* Strip charts (aka "one dimenstional scatter plots") are used to display the distribution of a single variable.
  + The maximum, minimum and concentration of points is shown.
* The strip plot is most useful with fairly small data sets.
* With a strip chart, data points frequently overlap. To avoid this, "jitter" the data points to show where multiple points have the same value.
* For details, refer to the documentation on strip charts by entering ?stripchart in R.
* Eg: Strip Chart with additional parameters



### Dot Charts (aka “Dot Plots”)

* The dot plot is used to display the distribution of a single variable plus the labels on each observation.
* The dot plot is most useful with fairly small data sets.
* The data is typically sorted so that the dot chart shows the rank order of the variable being plotted.
* For details, refer to the documentation on dot charts by entering ?dotchart in R.

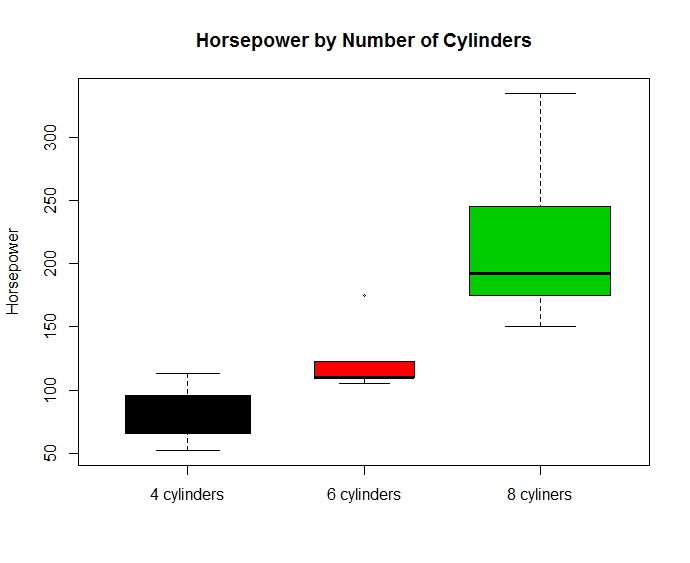
Eg: dot plot



### Box Plots (aka "box and whisker plots")

* Box plots split a variable into quartiles. The central "box" contains the interquartile range of the data.
* The vertical lines (whiskers) extend to the highest and lowest non-outlying values.
* Outliers are plotted as individual points, usually 1.5 times the upper and lower quartiles, respectively.

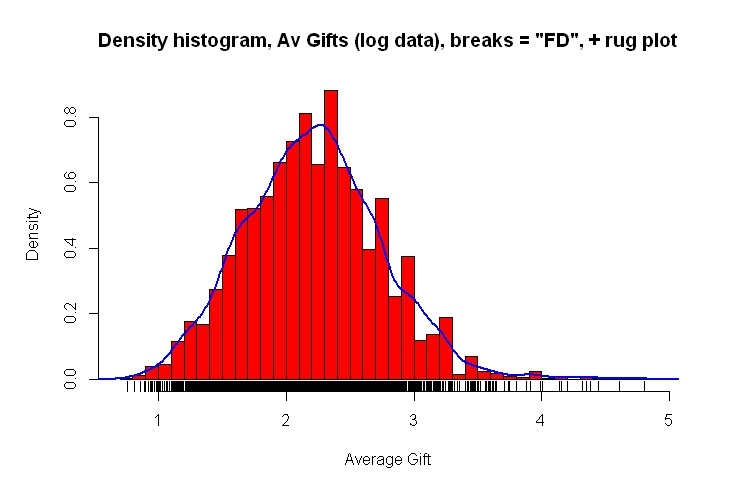
Eg: Elaborate box plot



### Histograms

* A histogram shows the distribution of a numerical variable. The histogram can show the approximate probability distribution of the variable.
* The values need to be "binned" before the histogram is created to divide the variable into a series of non-overlapping intervals.
* In R the number of bins can be "requested" by specifying breaks. For example:
  + breaks = c(110, 135, 160, 185)
  + breaks = 10
  + breaks = seq(0, 100, 10)
* However, R treats the specified number as a suggestion and may change the requested number to produce a more appropriate histogram.
* Documentation on histograms is found in R by entering ?hist
* Note histograms are not the same as bar charts; histograms are essentially bar charts of frequencies.

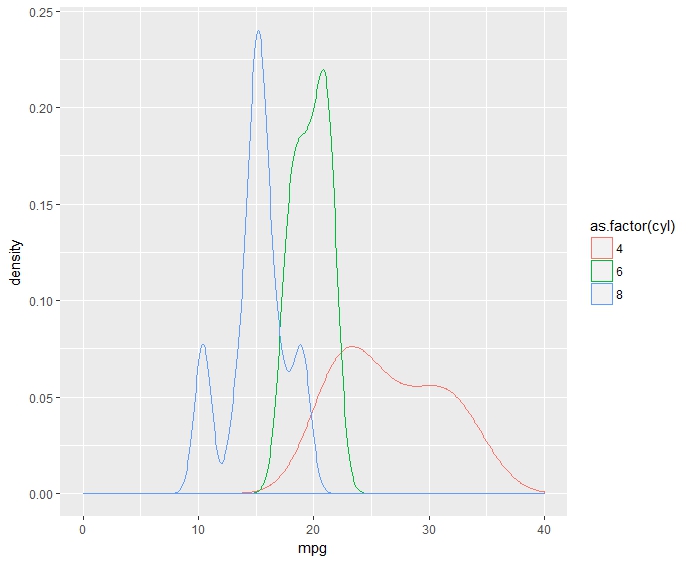
Eg: Histogram with density plot



### Density Plots

* Density plots (aka "kernel density plots") visualize the distribution of a variable over a continuous value.
* Density plots, in some sense, are like histograms, but smoothed, to average over noise and random variations.
* The advantage of density plots is that they are not affected by the number of bins or breaks.
* It should be understood that, the height of a density plot is not a probability. The total area under a density plot is 1.0. To obtain a probability, the area between two values on the x-axis can be computed.

Eg: comparison of density plot with qplot

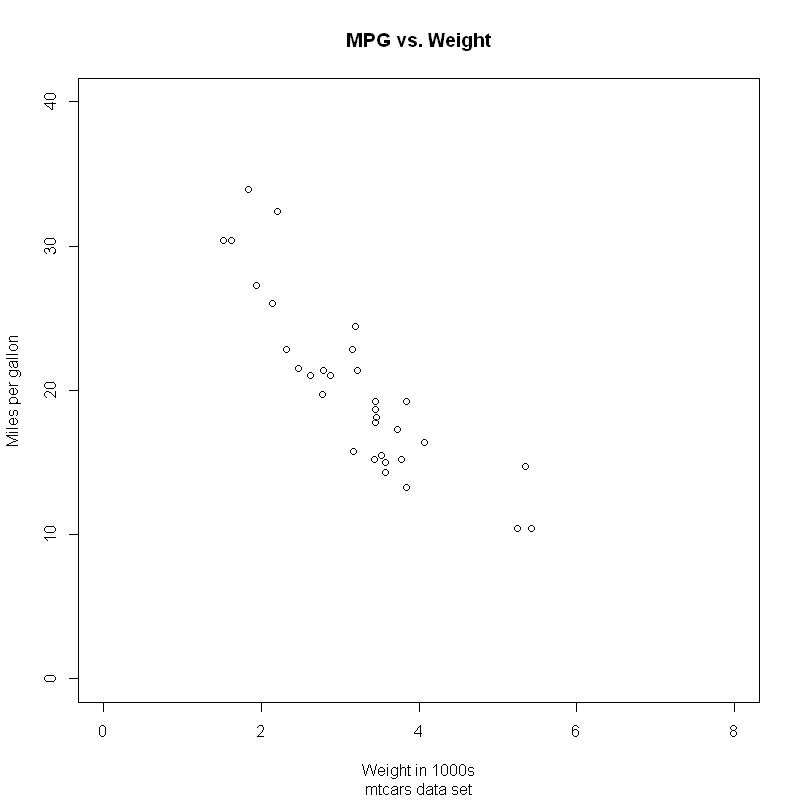


### Scatter Plots

Scatter plots are used to plot data points using a horizontal axis for one variable and vertical axis for another variable.

* The purpose of the scatter plot is to show the relationship between two variables.
* Scatter plots can be constructed in R in several ways; two basic ways are with the plot() function and the qplot() function.

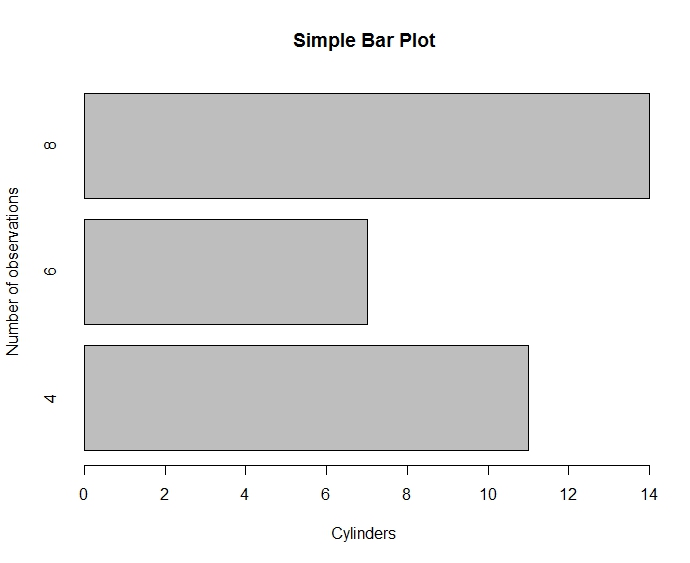
Eg: Scatter plot with lables, titles and axis limits



### Bar Charts

* A bar chart is made up of columns (or rows) plotted on a graph.
* Each column (row) is typically positioned over a label. The height (length) of a bar is proportional to the value defined by the label.
* There several types of bar charts, including:
  + Stacked bar charts
  + Grouped bar charts
  + Segmented bar chart (where each bar shows 100% of the total value)

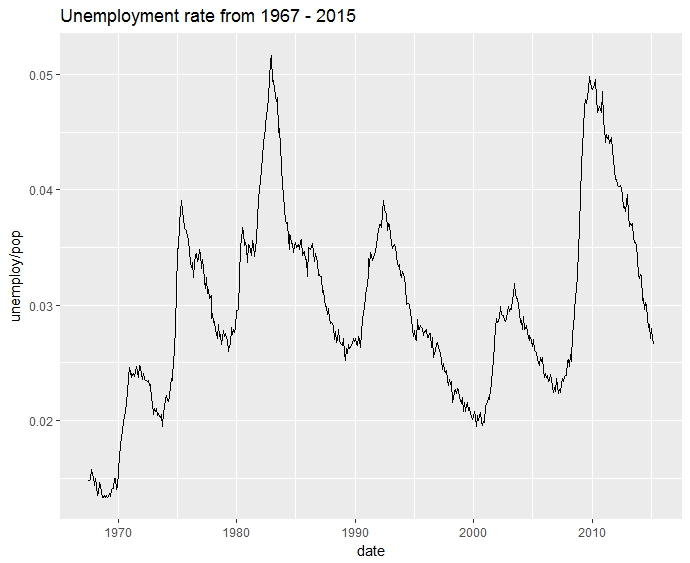
Eg: Simple Bar Chart



### Line Charts

* A line chart is essentially a scatter plot with lines connecting points from left to right.
* In many cases, the x-axis is a time variable and must be treated as such by R.

Eg: Time-Series Line Chart



### Heat Maps

A heat map (or heatmap) is a graphical representation of data where the individual values contained in a matrix are represented as colors. (Wikipeidia)

* One way to thing about a heat map is to consider it a two-way table where the entries are colors instead of numbers.
* According to LabEscape.com: "Heat maps leverage the human visual system to help users gain deeper and faster insights than other visualizations. Users can visually aggregate, determine relevance and detect micro-patterns in their data in ways other visualizations can’t match."

Eg: Colored Heat Map

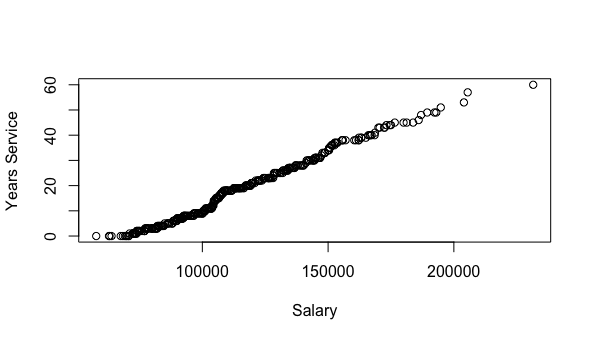


### QQ (Quantile-Quantile) Plots

QQ plots serve to visually check the normality of a sample distribution

* Data are sorted and mapped against a standard normal distribution
* A line is drawn to represent the expected normal distribution of the data
* If the plot appears linear along the expected distribution then it can be assumed the population of the sample is normally distribution
* Default quantiles are 25th and 75th percentile but can be specified through function keywords for additional detail
* Chart on next slide shows two variables. One which is noramlly distributed and one which is not.
* Help and documentation for the three functions used in creating QQ plots can be found by entering ?qqnorm, ?qqline, or ?qqplot

Eg: QQ Plot with 2 variables

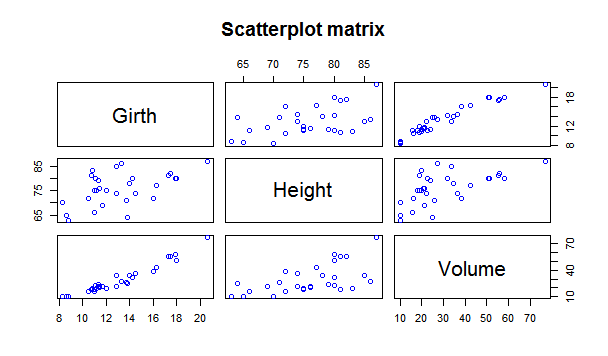


### Scatterplot Matrices

Scatterplot matrices are a great way to roughly determine if you have a linear correlation between multiple variables.

* This is particularly helpful in pinpointing specific variables that might have similar correlations to your genomic or proteomic data.
* Scatterplots related to Time are evenly distributed into columns or rows, suggesting that data was actually collected in a regimented fashion.
* Scatterplot matrices are **good** for determining rough linear correlations of metadata that contain continuous variables.
* Scatterplot matrices are **not so good** for looking at discrete variables.
* Documentation on scatterplot matrix is found in R by entering ?pairs

Eg: Scatterplot Matrices with headings

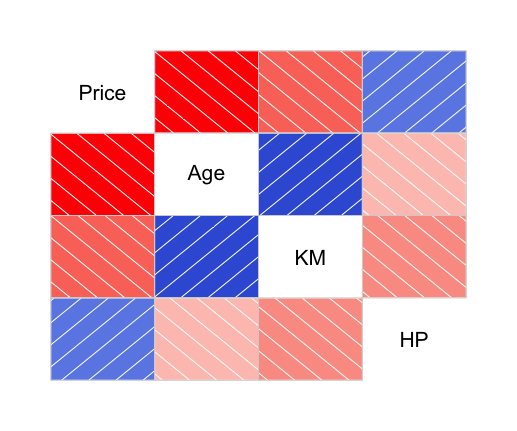


### Corrgrams

Corrgrams visualize correlation tables with color indicating direction

* Two colors are used to show either positive or negative correlation [-1,1]
* Variable names are written along the corrgram with intersecting cells representing the correlation between two variables
* Plot specified in R with the corrgram package and corrgram function
* The opacity of the color in each cell denotes the strength of the relationship
* Colors, correlation algorithm, and pie correlations can be specified
* Default colors are blue for positive correlations and red for negative correlations
* Pie charts which show proportion of perfect correlations can also be specified
* Help and documentation on Corrgrams can be found using ?corrgram

Eg: Corrgram- Default

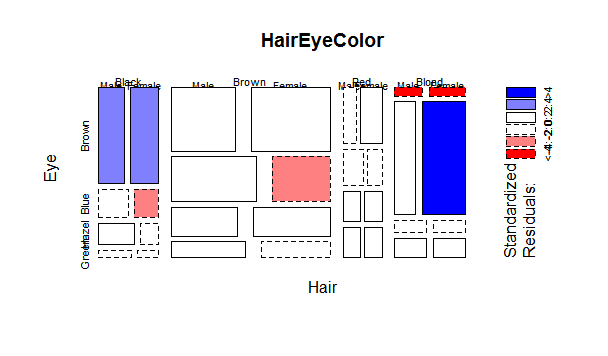


### Mosaic Plots

Mosaic plots (aka treemaps) are a great way to visualize hierarchical data.

* A collection of rectangles which represents elements to be visualized (customers, news items, blog posts), with size/color of the rectangle attribute.
* Where there is hierarchy (customer segments, news topics, post categories) those elements are collected and labeled together, with optional subcategories.
* Extended mosaic displays visualize standardized residuals of a loglinear model for the table by color and outline of the mosaic's tiles. (Standardized residuals are often referred to a standard normal distribution.) Cells representing negative residuals are drawn in shaded of red and with broken borders; positive ones are drawn in blue with solid borders.
* Documentation on scatterplot matrix is found in R by entering ?mosaicplot

Eg: Mosaic Plot with color



### Bubble Plots

Bubble plots are scatter plots that use color and size to express information

* Two variables are shown as bubbles at x and y axis intersects
* Color can be used to denote additional variables both categorical (factors) and continuous
* Bubble plots in R can be created using the plotly library which includes the plot\_ly function. Dataset and axis are specified in the function as well as variables to be visualized with color or size.
* Size is best used to express comparison of continuous variables
* Useful for communicating relationships between different variables and performing exploratory analysis of variable relationships
* Help and documentation creating bubble plots is found using ?ploy\_ly

Eg: Bubble Plot- Default

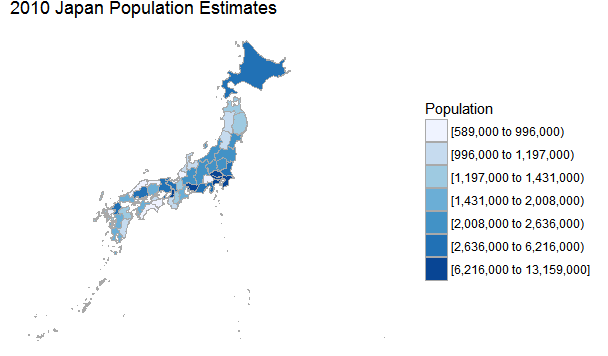


### Chloropleth Maps

A choropleth map is a thematic map with regions colored or shaded according to the value assumed by the variable of interest in that particular region.

* We make use of three packages: choropleth, choroplethAdmin1, choroplethMaps (plus their dependencies) to produce the maps in this post.
* maptools: a set of tools for reading and handling spatial objects. In particular, it will be used to read .shp files, a common format used by geographic information systems software.
* ggplot2: one of the powerful graphics engines available to R users
* ggmap: a package for spatial visualization using popular on-line mapping systems, such as GoogleMaps or OpenStreetMap.
* Documentation on choropleth maps is found in R by entering ?choroplethr

Eg: Chloropleth Map for Population of Japan

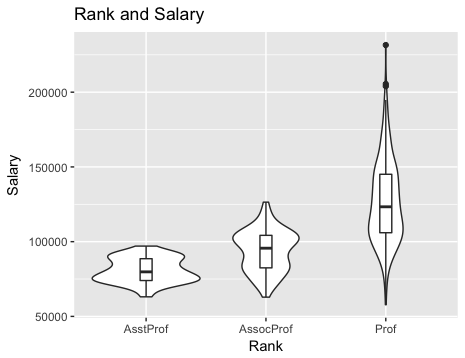


### Violin Plots

Violin plots combine a boxplot structure and probability density visualization

* Violin plots are used to provide information regarding both the distribution of a numerical variable and the probability density
* As with traditional boxplots the central box of the chart represents the interquartile range, but whiskers represent a 95% confidence interval
* The width of the probability density denotes the frequency of observations at different values of the variable
* Violin plots are often charted with multiple plots split by a categorical variable
* Violin plots in R are created using ggplot2 library and the geom\_violin function
* Help and documentation can be found using ?geom\_violin

Eg: Violin Plots with boxes

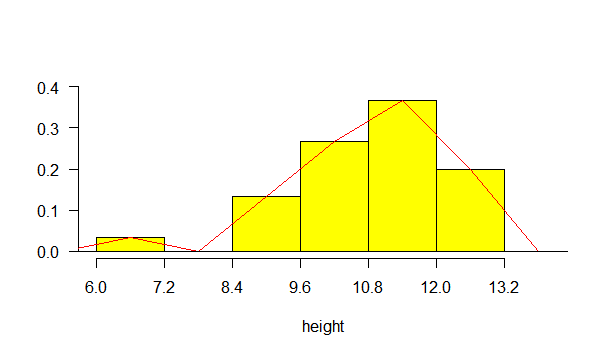


### Frequency Ploygons

Frequency Ploygons visualize the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin.

* Histograms (geom\_histogram) display the count with bars; frequency polygons (geom\_freqpoly), display the counts with lines.
* Frequency polygons are more suitable when you want to compare the distribution across a the levels of a categorical variable.
* By default, the underlying computation (stat\_bin) uses 30 bins
* geom\_histogram uses the same aesthetics as geom\_bar; geom\_freqpoly uses the same aesthetics as geom\_line.
* Documentation on scatterplot matrix is found in R by entering ?geom\_freqpoly

Eg: Frequency Polygon with Histogram



### Tree Maps

Tree Maps use a spatial structure to display categorical hierarchy using boxes of varying size and color. Categories are ordered and displayed as breakdowns within the prior category.

* Quantitative variables are visualized within hierarchies using color or size of the individual boxes
* In R Tree Maps can be created using the treemap package
* treemap() – default requires a dataframe, a vector of variable names, and a quantitative variable for vSize
* An alternative package to use is portfolio
* Data names should be changed to be clear for use on treemap
* Help and documentation can be found using ?treemap in the R interpreter

Eg: Tree Map using Color Brewer



### Pie Charts

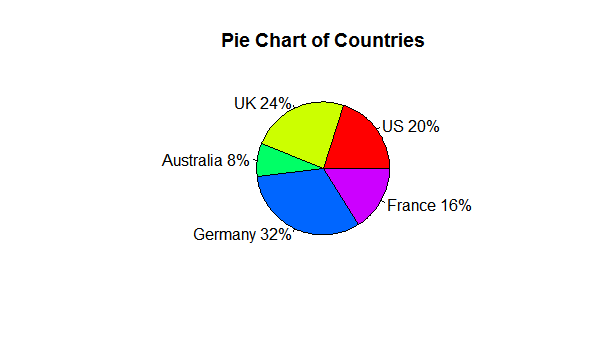
A pie-chart is a representation of values as slices of a circle with different colors. The slices are labeled and the numbers corresponding to each slice are shown

* In R the pie chart is created using the pie() function which takes positive numbers as a vector input. The additional parameters are used to control labels, color, title etc.
* The basic syntax for creating a pie-chart using the R is −

pie(x, labels, radius, main, col, clockwise)

* A pie chart with 3 dimensions can be drawn using additional packages. The package plotrix has a function called pie3D() that is used for this.
* Documentation on scatterplot matrix is found in R by entering ?pie

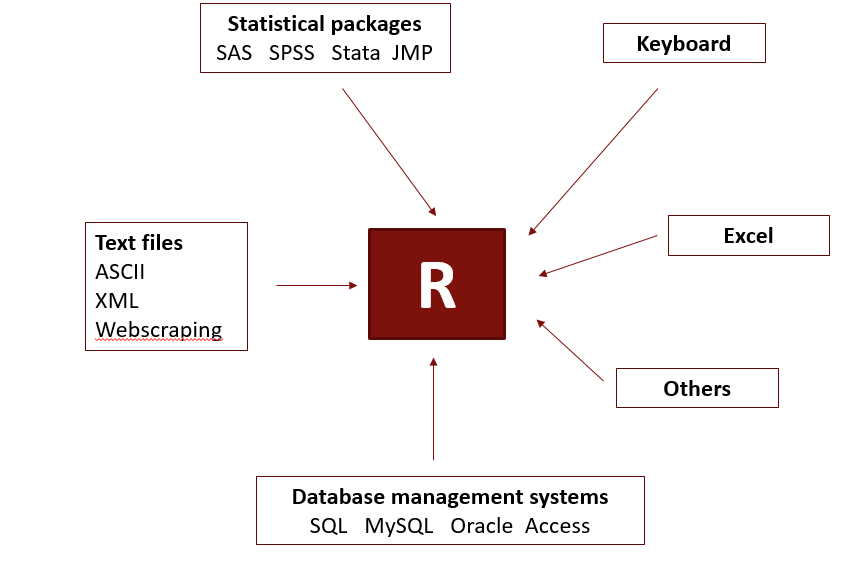
Eg: Pie Charts with Annotated Percentages



**Visualizations in R (with code)**

# Preparing data with R

R provides means to get data from many types of resources as shown below:



Source: Frank Acito, Data Management with R, Business Analytics Foundations

Note that there can be many issues while getting the data like

* Data formats
* Headings
* Default values
* Missing value codes

## A general approach to prepare data

1. Import your data with read.csv() or another appropriate function and check that the importation process gives no errors.
2. Convert it to a data frame if needed (my.data <- data.frame(DATA) and set column names (names(my.data) <- c(...)) if needed.
3. Examine dim() to check that the data frame has the expected number of rows and columns.
4. Use head() and tail(my.data) to check the first few and last few rows; make sure that header rows at the beginning and blank rows at the end were not included accidentally. Also check that no good rows were skipped at the beginning.
5. Use some() from the car package to examine a few sets of random rows.
6. Check the data frame structure with str() to ensure that variable types and values are appropriate. Change the type of variables—especially to factor types—as necessary.
7. Run summary() and look for unexpected values, especially min and max that are unexpected.
8. Load the psych library and examine basic descriptives with describe().
9. Reconfirm the observation counts by checking that n is the same for each variable, and check trimmed mean and skew (if relevant).

# Supervised Analytics Techniques

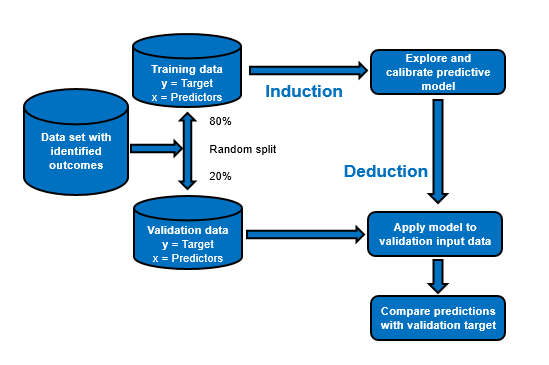
## Introduction to Regression

* Regression can be used to predict the value of the dependent variable (Y) given any set of values for the independent (X) variables.
* Test hypotheses about whether each particular independent (or predictor) variable is statistically significant.
* Estimate the size of the effect of changing each independent variable on the dependent variable, holding all of the other independent variables constant. (The coefficients are called “partial regression coefficients.”)

### Application of Regression

* Description and explanation
  + E.g., describe the unit sales of automobiles in sales regions as a function of population, per capita income, and degree of urbanization.
* Control
  + E.g., look at 125 branch offices of an insurance company and model the annual operating cost as a function of the number of policies written, the number of claims made, and the number of people working in the office. Management can then set standards for controlling expenses.
* Prediction
  + E.g., a telco firm wants to predict the number of minutes of cell phone use per month for customers as a function of age, education, gender, occupation, marital status, and size of household.

**Strategy for building Regression Models:**



## Linear Regression

* Linear regression is the classic method for analysis of dependence between variables.
* Used to predict the relationship between an outcome (target) variable that is continuous (interval or ratio) and single or multiple independent variables
* Independent variables can be categorical, interval, ratio
* Relationship between predictor and target variable must be linear or at least able to be transformed to linear
* For statistical analysis, regression has several strict assumptions:
  + Normal distribution of errors
  + Uncorrelated errors
  + Homogeneous variance of errors

The basic idea of linear regression is that it tries to “fit” a line through the scattered distribution of data. The straight line is chosen such that the sum of errors between the predicted value and actual value of the dependent variable is the minimum.

### Simple Regression

* Simple linear regression identifies the best fitting straight line for a scatterplot between two variables- one dependent and one independent
* The relationship between the independent variable and the target variable is seldom, if ever perfect.
* The straight line is chosen to minimize the squared error between the predicted and actual value of the target variable
* The measure used to select the best fitting line is call R-squared value which is explained as follows:
  + In the absence of a regression model, the best “guess” for any predicted outcome is the mean value. This is our base model. Any regression model developed should have an error that is lower than this base model. The error in the base model can be written as ST which is the total variance that is to be explained by our regression model. If “y” is the dependent variable then,

**ST = ∑ (y – mean(y))2**

* + Now, suppose we develop a regression model. This model will also not be perfect and will contain some errors. Let SE be the error in our model.

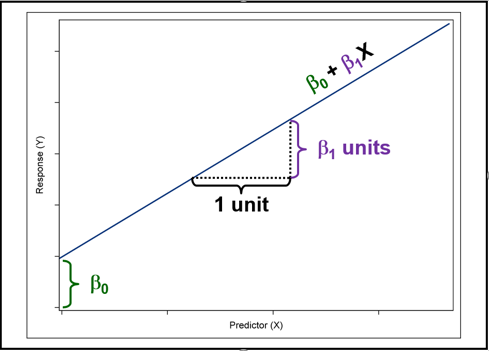
**SE = ∑ (yactual – ypredicted)2**

* + For any reasonable regression model, **SE < ST.** Thus, our regression model only explains a portion of the total variance **ST** in the base model. This ratio of the variance explained by our model is called R-squared value.

**R-squared = 1 – (SE/ ST)**

* + The greater the error in the model, SE, lower will be the R-squared value. This implies that a greater value of R-squared means the model can explain more variance between the dependent and independent variables and has a lower error.
  + R-squared value is very important to compare two regression models to each other. A “better” model will have a greater R-squared and will explain more variance in the data.
  + One disadvantage of R-squared is that it always increases as the number of independent variables increase, irrespective of the significance of the variables. Thus, a very unimportant variable (which makes our model more complex and adds no meaning to the model) also increases the value of R-squared.
  + Hence, a more accurate measure is **adjusted R-squared** which penalizes the model for the inclusion of unimportant independent variables.

**A simple regression:**



**Simple Regression with R: **

### Multiple Regression

* Multiple regression is a more complex version of simple regression. The only difference between a simple linear regression and a multiple linear regression is that multiple linear regression involves multiple independent variables.
* This means visualizing a multiple regression model is more difficult as compared to a simple linear regression model containing 2 variables that can be plotted on an X-Y plane
* The same concepts of least squared error and adjusted R-squared values for determining a better model hold true for multiple linear regression as well
* Adjusted R-squared is even more critical in multiple linear regression as there are multiple independent variables to choose from.

### Stepwise Regression

**The regression modeling problem:**

* After a set of explanatory variables has been selected, after the functional form (linear, quadratic, logarithmic, etc.) and whether any interactions are to be included, the question becomes which subset of should be retained?
* If too many predictors are included:
  + Maintenance of the model becomes difficult.
  + Obtaining measures on all the predictors can be expensive and time consuming.
  + The model can be overfitted – resulting in large variances in the estimates and a lack of generalizability of the model.
* On the other hand, omitting a key variable can lead to biased estimates on the regression parameters, the statistical tests, and predictions.

**Selecting a subset of predictors**

* The object is to find the simplest model that performs sufficiently well, because:
* It is more likely to be robust (instability due to multicollinearity)
* May be less expensive
* Less likely to have missing variables
* May actually be more accurate (fewer variables included which are subject to measurement error)
* Parsimony is itself a desirable goal
* In general, the number of models that can be formed from n possible predictors taken r at a time is:

**Automated Variable Selection Methods**

* Forward: Start with *no* predictors; Add them one by one (add the one with largest contribution); Stop when the addition is not statistically significant.
* Backward: Start with *all* predictors; Successively eliminate least useful predictors one by one; Stop when all remaining predictors have statistically significant contribution.
* Both: Like Forward Selection; except at each step, also consider dropping non-significant predictors.
* Exhaustive Search: All possible subsets of predictors assessed (single, pairs, triplets, etc.); Computationally intensive. (For example, with 11 predictors, there are 462 possible models with 5 of the 11 in the equation.)

***None of these methods can guarantee that the "best" subset of predictors.***

**Issues with Stepwise Models:**

* **Cook and Weisberg**
  + "We do not recommend such stopping rules for routine use since they can reject perfectly reasonable sub-models from further consideration. Stepwise procedures are easy to explain, inexpensive to compute, and widely used. The comparative simplicity of the results from stepwise regression with model selection rules appeals to many analysts. But, such algorithmic model selection methods must be used with caution."
* **What can happen with stepwise?**
  + r-squared is biased too high.
  + Standard errors on the parameters are to low.
  + When multicollinearity is present, variable selection becomes arbitrary.
* **What to do?**
  + Only use stepwise when you have large (e.g., > 1,000 cases) and a ratio of cases to potential preditors at least 10 to 1.
  + If stepwise is used, validation using a holdout sample is necessary.

### Linearity Assumption

* ***Assumption****:* The assumption is that the dependent variable is linearly related to the predictor variable or variables – that is the dependent variable increases the same amount for each unit increase in a particular independent variable.
* ***Consequences****:* If the relationship is not linear, the regression coefficients will be biased and the model may not fit the data well.
* ***Detection****:* Examine a plot of the residuals. Look for U-shaped patterns. Also, plot the dependent variable versus each of the predictors to visually assess linearity.
* ***Remedies****:* Add polynomial term or transform one or more of the predictors.
* ***Some useful transformations:***

## Assessing Regression Models

* R2 is probably the most common statistic used to assess regression results.
  + Desirable property: R2 varies between 0 and 1 and can be interpreted as a percentage.
  + R2 is the square of the correlation between the actual and fitted values in a regression.
  + But, R2 will always increase with additional predictors. Therefore, adjusted R2 is often used.
* Adjusted R2 formula:
  + where p=# of predictors (not including the intercept) and n=sample size.
  + There are also other formulas in use for adjusted R2.
  + Adjusted R2 tries to estimate the R2 from the population, whereas the unadjusted R2 is for the sample. As the sample becomes large compared with the number of predictors, the adjusted and unadjusted R2 converge.

### Interpreting R2

* In general, higher R2 is better than lower R2.
* But, R2 cannot be compared between models that have different transformations of the dependent variable.
* Low R2 models can still be useful (and high R2 models are not always useful). It depends.

## Measuring Prediction Accuracy

* This is not the same as “goodness-of-fit.”
* We want to know how well the model predicts new data, not how well it fits the data it was trained with.
* A key component of most measures is difference between *actual* y and *predicted* y which is prediction error.
* The prediction error is frequently called the residual in regression programs.

**Measures of Prediction Error**

* Average error: Simple arithmetic mean of errors.
* MAE or MAD: Mean absolute error (deviation).
* MAPE: Mean absolute percentage error.
* RMSE (root-mean-squared-error): Square the errors, find their average, take the square root.  
  All of these should be calculated using a validation sample that was not used to estimate the regression.

As might be expected, no single measure of accuracy is best in all circumstances.

### Average Error

This is the average of differences between the actual and predicted values from the regression equation. In the training sample this will always be exactly zero. In the validation or test sample, this will indicate whether there is bias in the predictions.

### Mean Absolute Error

* This measure avoids the problem that positive and negative errors tend to cancel each other out.

### Mean Absolute Percent Error (MAPE)

This measure has a simple and intuitive meaning and is frequently used. However, it does have some limitations. It cannot be used if there are zero values in the y variable. Also, by focusing on percentages, the relative magnitude of errors is masked. Consider a 10% error on a product selling 100 units. Is that as serious as a 10% error on a product selling 10,000 units?

### Root Mean Squared Error

This measure gives greater weight to very unfavorable forecasts because the error is squared. However, this also makes RMSE very sensitive to outlying values.

## Regression Diagnostics

**Objectives**

* Review the assumptions of linear regression.
* Examine the assumptions with scatter plots and residual plots.
* Using statistics to identify potentially influential observations.
* Determine whether collinearity exists in a model.
* Generate output to evaluate the strength of the collinearity and what variables are involved in the collinearity.
* Determine methods that can minimize collinearity in a model.

### Examining Residuals

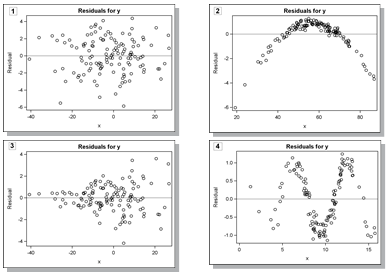
**Residuals:** The difference between the observed value of the dependent variable (*y*) and the predicted value (*ŷ*) is called the **residual** (*e*). Each data point has one residual.

Residual = Observed value - Predicted value   
*e* = *y* - *ŷ*

Source: <http://stattrek.com/regression/residual-analysis.aspx?Tutorial=AP>

**Residual Plots:**

A **residual plot** is a graph that shows the residuals on the vertical axis and the independent variable on the horizontal axis. If the points in a residual plot are randomly dispersed around the horizontal axis, a linear regression model is appropriate for the data; otherwise, a non-linear model is more appropriate.



### Influencial Observations

**Outliers versus Influencial Observations**

* An outlier is an observation that is “extreme”, being distant from the rest of the data (definition of “distant” is deliberately vague).
* Outliers can have disproportionate influence on models (a problem if it is spurious). Such extreme values can distort an analysis. The effects of outliers may be to increase or suppress a correlation, for example, which can lead to misleading results. These are influential observations.
* They also may be indicative of an incorrect model. Influential observations may not be detected by simply looking at residuals. Furthermore, not all outlying cases have a strong influence on the fitted regression function.
* An important step in data pre-processing is detecting influential observations.
* Once detected, domain knowledge is required to determine if there are errors, or truly extreme values.

**Diagnostic Statistics**

Statistics that help identify influential observations are the following are computed in R using the following functions:

#### rstandard

Standardized residuals are obtained by dividing the residuals by their standard errors.

* Suggested cutoffs are as follows:
  + |SR| > 2 for data sets with a relatively small number of observations
  + |SR| > 3 for data sets with a relatively large number of observations
* In R:
  + rstandard(model, ...)

#### rstudent

* Studentized residuals are obtained by dividing the residuals by a standard error computed by omitting the ith observation.
* Suggested cutoffs are as follows:
  + |SR| > 2 for data sets with a relatively small number of observations
  + |SR| > 3 for data sets with a relatively large number of observations
* In R:
  + rstudent(model, ...)

#### cooks.distance

* Cook’s D statistic is a measure of the simultaneous change in the parameter estimates when the *i*th observation is deleted from the analysis.
* A value of Cook's D is computed for every observation



* Some textbooks suggest a cutoff of 1.0. Others suggest a cutpoint for influence using this expression:
* In R:
  + cooks.distance(model, ...)

#### dffits

* DFFITS*i* measures the impact that the *i*th observation   
  has on the predicted value.



* A suggested cutoff for influence using DFFITS is 1 for small to medium data sets and the following for large data sets:
* In R:
  + dffits(model, ...)

#### dfbetas

* Measure of change in the *j*th parameter estimate with deletion of the *i*th observation
* One DFBETA per parameter per observation
* Helpful in explaining on which parameter coefficient the influence most lies
* Suggested cutoff for influence:



* In R:
  + dfbeta(model, ...)

**How to handle Influencial Observations**

* Recheck the data to ensure that no transcription or data entry errors occurred.
* Do not drop "outliers" without careful checking.
* If the data is valid, one possible explanation is that the model is not adequate.
  + A model with higher-order terms, such as polynomials and interactions between the variables, might be necessary to fit the data well.

**R Code for Assessing Regression:** 

### Collinearity

#### Multicollinearity

* Multicollinearity is due to the relationships among predictor variables.
* The effects of severe multicollinearity:
  + The coefficients on the predictors can have very wide variance and instability.
  + The relative effect of predictor variables (i.e., importance) can be difficult to detect.
* What to do about multicollinearity
  + Ignore it: ***Note: Multicollinearity does not affect the overall fit of a model nor does it produce bad predictions.***
  + Remove highly correlated predictors.
  + Make linear combinations of predictors – e.g., adding them or averaging them.
* Note: If interaction terms or polynomial terms are present in a regression, multicollinearity can appear to be high. This can be prevented by standardizing the variables involved in the interaction or polynomical term.

**Informal diagnostics for serious multicollinearity:**

1. Large changes in the estimated regression coefficients when a predictor variable is added or deleted or when an observation is added or deleted.
2. Nonsignificant results in individual tests on the regression coefficients while the overall regression is significant.
3. Estimated regression coefficients where the algebraic sign is opposite of what is expected based on experience or theory.
4. Large coefficients of the simple correlations between pairs of predictor variables in the correlation matrix.

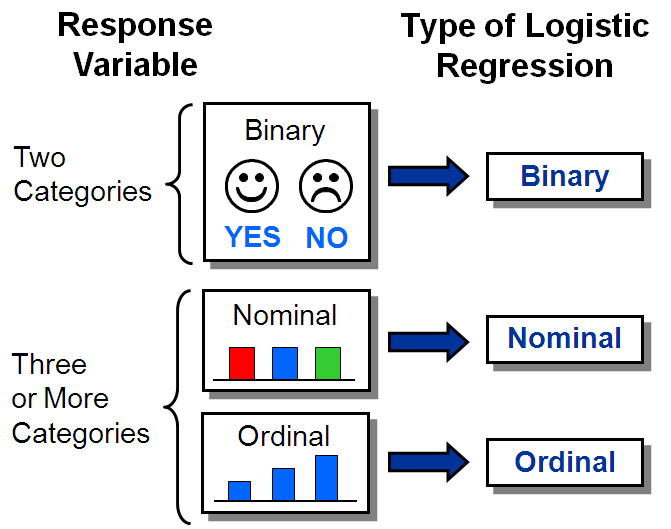
#### Variance Inflation Factor

* The *VIF* is a relative measure of the increase in the variance because of collinearity. R-squared in the formula from regressing the ith variable on all of the other predictors (not the dependent variable).
* A VIFi > 10 indicates that collinearity is a problem.

## Logistic Regression

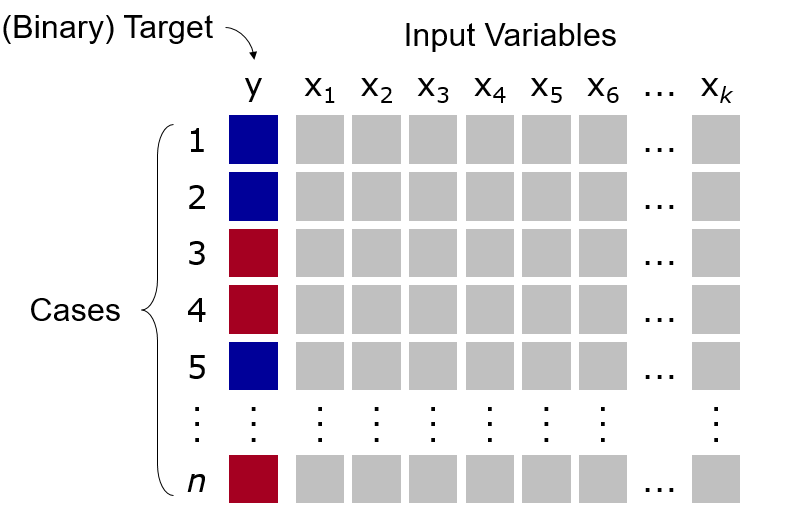
Logistic regression is used when the target variable is binary i.e. 0/1. Many cases can have this scenario like:

* + Buy or not buy
  + Default on loan or not
  + Vote Yes or No
  + Diagnosis is cancer or not
* Similar in some respects to multiple regression and discriminant analysis. (Probit analysis is very much like logistic regression, but with a different transformation. The results of probit and logit are usually very similar).
* More flexible – no assumptions about distributions of independent variables; the error distribution is assumed to be logistic.
* No assumption of equality of variance among groups.
* Cannot produce negative probability predictions.
* Types of Logistic Regressions:



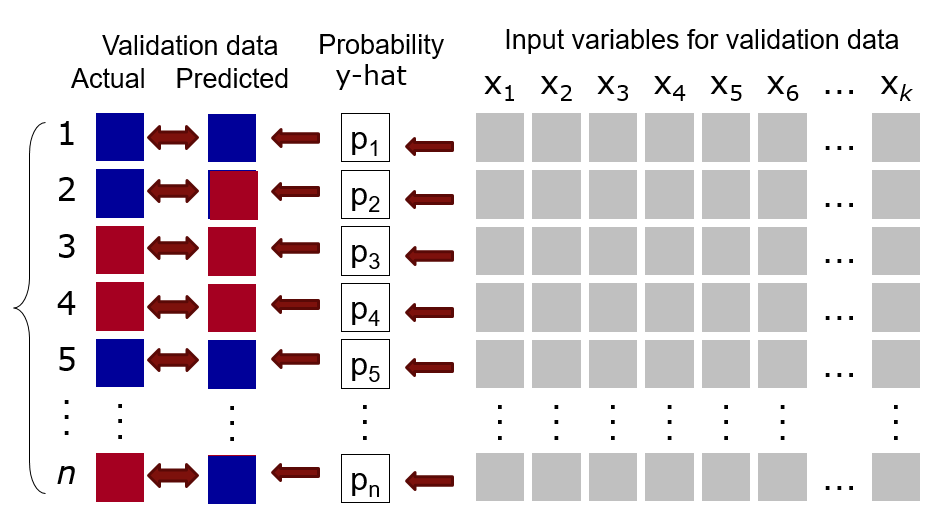
**Supervised classification data structure: training data**

* The data is split up into training, validation and testing as required. The model is first developed by the training data. The training data set should be random and resemble both the populations proportionally. If any one of the populations is under represented, it could lead to unbalanced data and would be needed to mitigate via techniques like under sampling the majority population or over sampling the minority population.

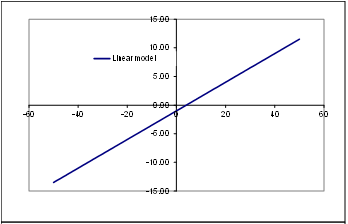


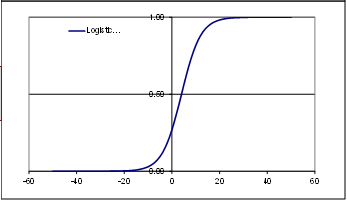
**Supervised classification: validation data**

* Next, the model is validated on the validation data set. A confusion matrix is created which has the count of true positives, true negatives, false positives and false negatives. This result should be like the result obtained from the training data set to ensure there is no over-fitting. Once this is verified, the user can add weights to all the four categories (true positives, true negatives, false positives and false negatives) and decide for himself if the model is useful. If accuracy is high enough and there is no overfitting, the model can then be used against testing data set.



**The logistic transformation of a linear model**





**Why not use ordinary regression?**

* Some of the results may actually be similar; but:
  + The error term is not normal with binary dependent variables:

when Yi = 1: εi = 1 - β0 - β1Xi when Yi = 0: εi = - β0 - β1Xi

* + So, the error term can only take on two values, which violates normality
  + Also, the error variance is not constant, which violates another assumption or ordinary regression:

σ2 = πi (1 - πi) = (β0 + β1Xi)(1- β0 - β1)  
 Note that σ2 depends on Xi

* With regression, the predicted response (target) is not constrained to be between 0 and 1

**The Binary Logit Model**

* The logistic distribution was selected from many possible distributions that could be used with dichotomous data because:
  + The logistic model is flexible and workable from a mathematical perspective
  + The interpretation of results using the logistic model makes intuitive sense
* For a single independent variable, the logistic regression model is:

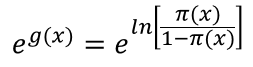


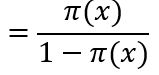
* The model is simplified by transformations to the log of odds.
* Logit is the “log of odds”. The logit transformation is given by:



* The logit is linear in its parameters, it can be continuous, and it can range from -∞ to +∞.
* This gives the logit. To get the probability of accepting the loan,

we, need to solve by exponentiating both sides of the equation:





= e 

**Summary**

* Logistic regression is similar to linear regression, except that it is used with a categorical response
* It can be used for explanatory tasks (=profiling) or predictive tasks (=classification)
* The predictors are related to the response Y via a nonlinear function called the logit
* As in linear regression, reducing predictors can be done via variable selection
* Logistic regression can be generalized to more than two classes
* Issues: overfitting, missing values, need for training and validation samples.

**R code for Logistic Regression:** **** ****

### Evaluating Model Performance- Basic Measures

Basic questions

* + How accurate is the model developed?
  + Which of several models performs best?
  + Is the model useful?

Classification vs. prediction

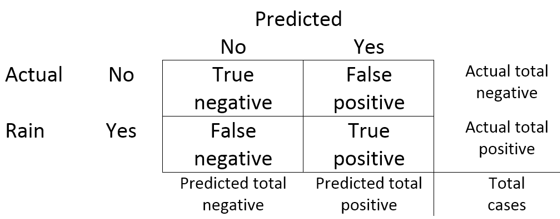
* + Classification involves assignment to a group – could be one of two groups, one of three, etc.
  + Prediction refers to the estimation of a numerical value.
  + Different metrics are used for each.

The focus is on the evaluation of binary predictive models.

**Use of Evaluation Measures**

* Multiple methods are available to classify or predict outcomes. Analysts may therefore try several different models.
* For each method, multiple choices are available for settings, making the total number of possible models very large.
* To choose best model, there need to assess each model’s performance.
* It is important to have a separate data set, one not used to build the model, to make the performance assessments.

The output of a logistic regression model is a probability of the dependent variable being 1. The threshold can be changed from the default threshold of 0.5 to predict either a 1 or 0. As the threshold is increased, the number of predicted positives will decrease and the number of predicted negatives will increase.



The various measures of accuracy can be described from the figure above as follows:

Accuracy = (True positive + True negative) / Total cases

* + Also known as the “fraction correct.”

Sensitivity = (True positive) / (True positive + False negative)

* + Also known as “recall,” “hit rate,” or the “true positive rate.”

Specificity = (True negative) / ( True negative + False positive)

* + Also known as “true negative rate.”

Precision = (True positive) / (True positive + False positive)

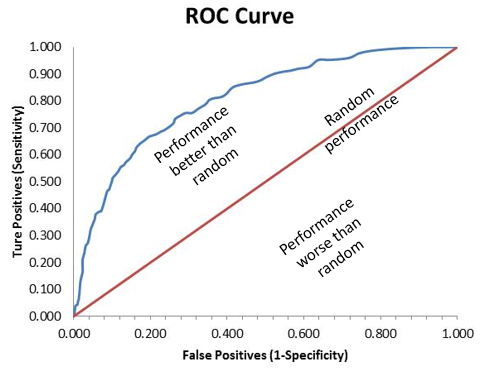
* + Also known as “positive predictive value.”

Negative predictive value = (True negative) / (True negative + False negative).

### Evaluating Model Performance- ROC, Lift and Gain Curves

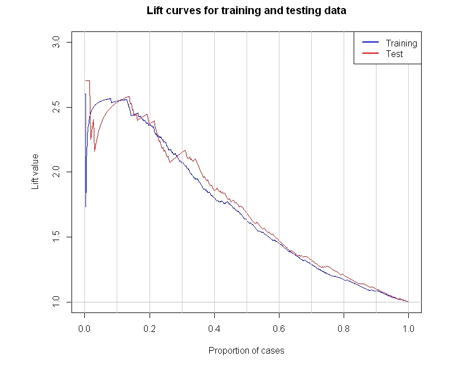
#### ROC Curves

* These curves show the tradeoff between sensitivity and (1-specificity) or the tradeoff between the true positive rate and the false positive rate for a predictive model.
* Interesting history – developed in World War II.
* The area under the curve (or AUC) is a summary measure of the performance; the range of AUC is .5 to 1.0)
* ROC curves are especially useful in comparing the performance of two or more models.



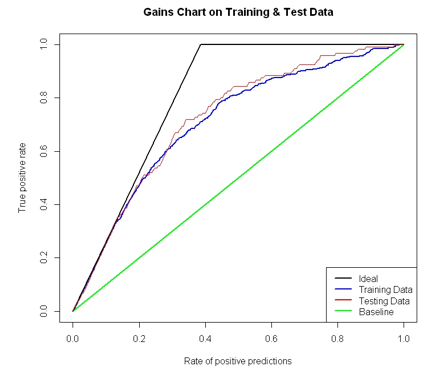
#### Lift Curves

* Lift is a performance measure used with predicting a binary outcome.
* For example, in a data set of 5,000 cases, 30% of the individuals purchased a given product and the rest did not.
  + So, the total number of purchasers in the sample is 1,500.
* If any random sample is taken of 20% of the data (1,000 cases), the expected number of purchasers is 300.
* Now assume a predictive model has been developed which assigns a probability of purchase to each of the 5,000 cases.
* Sort the 5,000 cases from highest to lowest probability of purchase.
* Now, take the top 20% of the sorted cases.
* If the model works, the top 20% of the sorted cases should have many more than 300 purchases.
* The ratio of the number of purchases in the top 20% of the sorted data to the 20% of the random selection is called lift.



#### Gains Curves

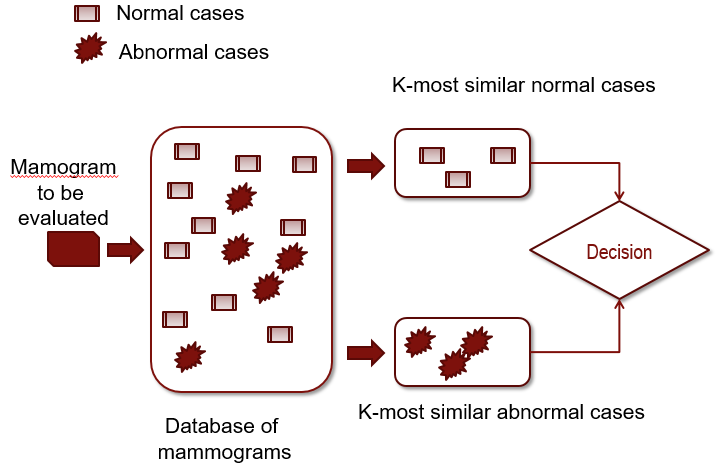
* Gains curves use data similar to lift curves and measure performance of a model in predicting a binary outcome.
* For example, in a data set of 5,000 cases, 30% of the individuals purchased a given product and the rest did not.
  + So, the total number of purchasers in the sample is 1,500.
* As with lift curves, assume a predictive model has been developed which assigns a probability of purchase to each of the 5,000 cases. Sort the 5,000 cases from highest to lowest probability of purchase.
* Plot on the x-axis the percentage of the 5,000 cases
* On the y-axis show the percentage of purchasers.
* If the model works, the percentage on the y-axis should always be larger than the percentage on the x-axis.
* One line is drawn to show no predictive ability of the model. Another line is drawn to indicate the best possible gain with a model that predicts perfectly.



**ROC, Lift Curves, Gains Curve Illustrations: **

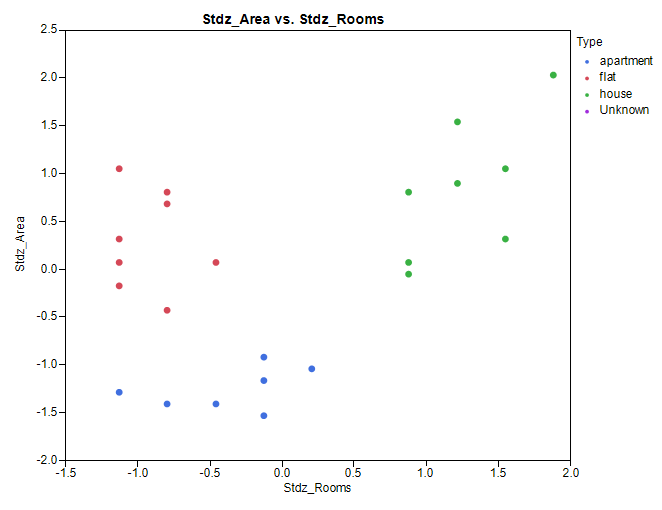
## K-Nearest neighbors

* It is a non-parametric model and does not require distribution assumptions regarding the variables and does not make statistical inferences to a population
* Does not require a “training” phase – just apply the model “right out of the box.”
  + However, the “testing” or analysis phase can be slow because all of the data must be used.
  + Training is used to select the proper number of “neighbors”
* The classification rule is: A new object is classified to the class Y based on a majority “vote” of its k neighbors where k is a positive integer (typically small) measure on X variables
* E.g. Anomalies in mammograms

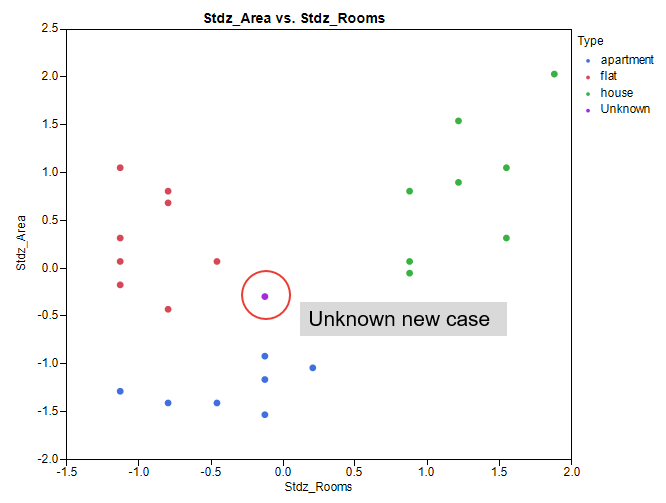


**How does it work?**

* Each case is considered a point in a multidimensional space (aka a feature space)
* This allows the computation of “distances” among the points (or cases) based on locations (values) in the feature space.
* Also needed is a value of “k” – the number of nearest points or neighbors to consider in making an assignment to a category.
* Usually an odd number is selected for k so that no ties are formed, but this is not always necessary.
* If an even number is selected for k, then a random choice is made for ties.
* The most common way is to use Euclidean distance as a measure of dissimilarity
* Euclidean distance is very fast to compute and works well in the algorithm
* An important point to note is that the variables should be standardized before running K-nearest neighbors or else large values will dominate the measure
* The process can be illustrated as below:



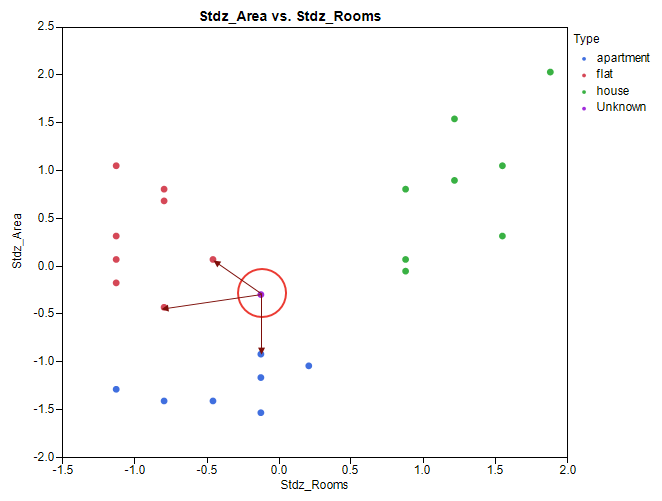
There are three distinct groups in the above figure, any new data point must be grouped into one of these.



Compute distance of unknown to all the points and assign to the same group as the closest point (smallest Euclidean distance)



The object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors



* There are multiple R packages that provide K nearest neighbors. The most common amongst them are:
* **Package 'class'**
  + k-nearest neighbor classification for test set from training set. For each row of the test set, the k-nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random.
* **Package 'caret'**
  + k-nearest neighbor classification is performed with the function 'train'. This function sets up a grid of tuning parameters for k-nearest neighbors, fits each model and calculates a resampling based performance measure.
* **Package 'knnGarde'**
  + Multi-distance based k-Nearest Neighbors Classification with K Threshold Value. Missing Observations Filling.
* k-NN is a type of [instance-based learning](https://en.wikipedia.org/wiki/Instance-based_learning), or [lazy learning](https://en.wikipedia.org/wiki/Lazy_learning), where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms.

**R code for k-NN & Digit Recognizer: **

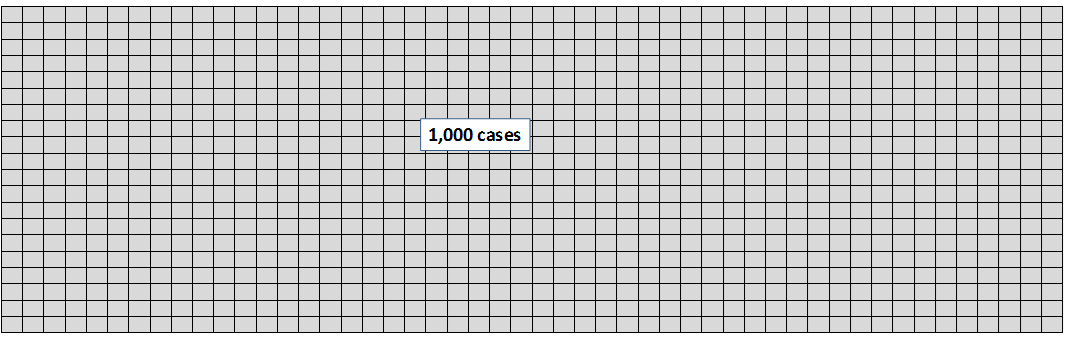
**R code for Digit Images: **

## Naïve Bayes

* Bayes theorem provides a way of calculating *posterior* probabilities.
* A *prior* probability is an initial probability value originally obtained before any additional information is obtained.
* A *posterior* probability is a probability value that has been revised by using additional information that is later obtained.
* Bayes theorem provides a way relating the likelihood of some outcome given some informative prior information.

**Bayes theorem can be illustrated in the following way:**

* 1. A sample of 1,000 customers was offered a new service from the Cable Company. The company would like to be able to predict who among other customers would be most likely to purchase the new service

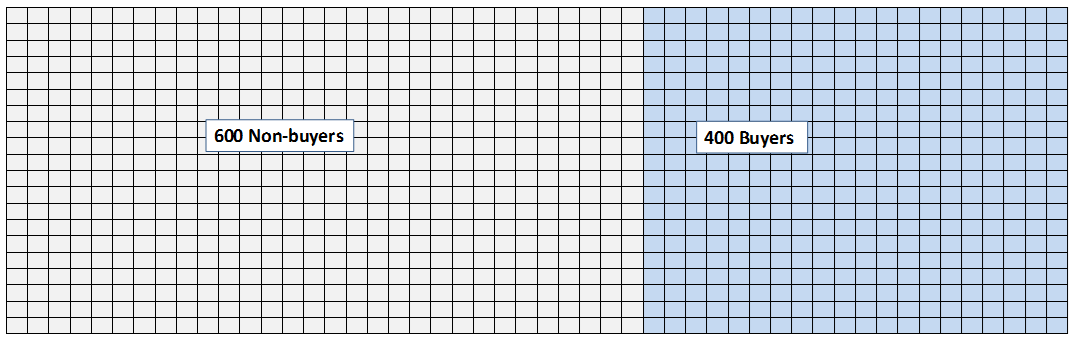


* 1. 400 of the customers purchased the new service.

Without additional information, the probability that another customer will purchase the new service is 400/1000 = .40.

This is known as the *prior probability*.

Prob(Purchase) = .40.



* 1. What if we have some additional information about the customers that purchased and did not purchase the new service?

Specifically, assume that we know which ones were male and which were female. It turns out that 600 were female and 400 were male and the relative proportions of males and females that purchased were different.

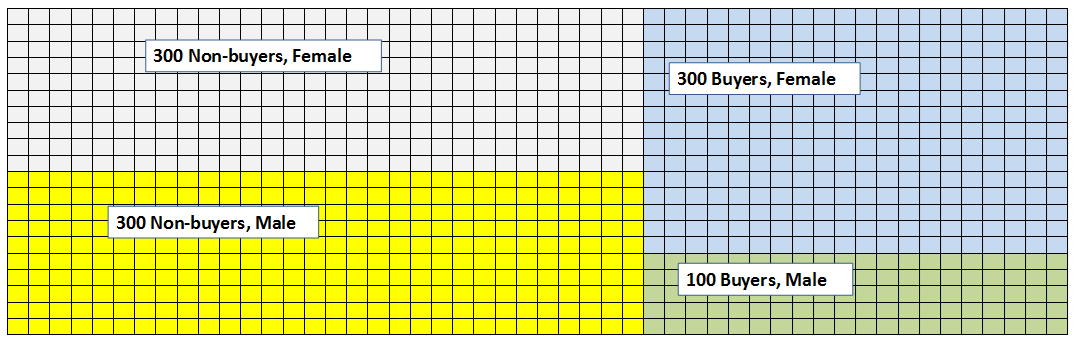
* 1. Of the 600 female customers, 300 purchased the product.

Of the 400 male customers, 100 purchased the product.

With this information we can compute the posterior probabilities:

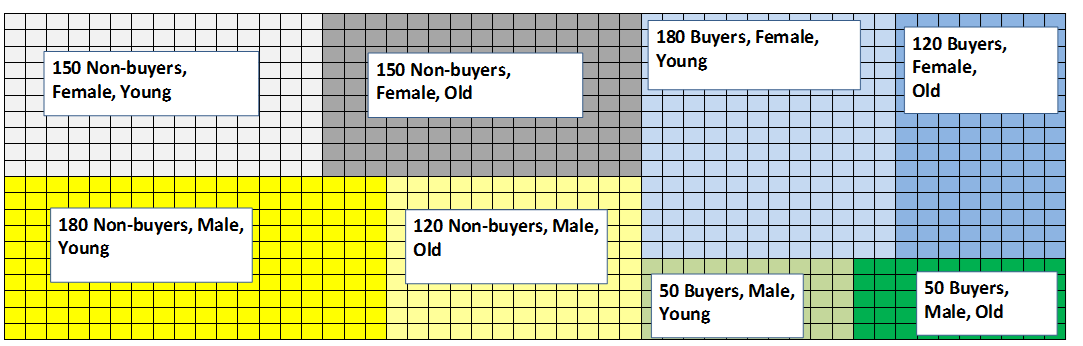
Prob(Purchase | Male) = 100/400 = .25.

Prob(Purchase | Female) = 300/600 = .50.



* 1. Now, assume we have some more information: the customer’s approximate age, either 40 or younger or over 40. We’ll call the two groups Young (40 or younger) and Old (over 40). It turns out the 560 of the customers were “young” and 440 were “old.”

How do the probabilities purchase change if we know both the customer’s gender and age?



* 1. There are 4 posterior probabilities that can be calculated:

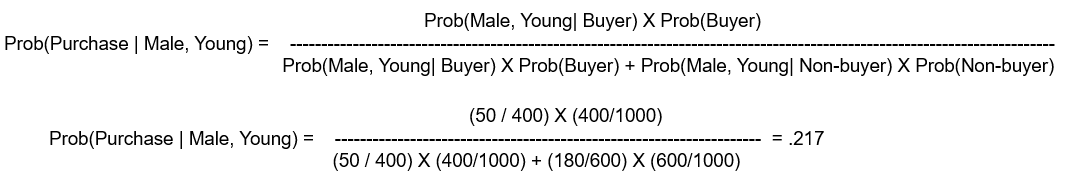
Prob(Purchase | Male, Young) = 50/(50+180) = .217

Prob(Purchase | Male, Old) = 50/(50+120) = .294

Prob(Purchase | Female, Young) = 180/(180+150) =.545

Prob(Purchase | Female, Old) = 120/(120+150) = .444

* 1. The same computation can be done using Bayes theorem as follows:



**R code for Naives Bayes **

## Decision Trees

Decision trees are a commonly used technique in data mining to perform multivariate analysis, typically with a single dependent variable (which can be continuous or categorical) and multiple predictor variables (which also can be continuous or categorical).

**Uses of Decision Trees**

* Market segmentation
* Stratification – dividing cases into high/medium/low risk, for example
* Prediction – create rules and use them to predict future outcomes
* Data reduction and variable screening – for screening large number of variables to identify best prospects
* Interaction detection – to find effects which differ by values of selected variables
* Category merging – to recode variables with large numbers of levels into few categories without losing predictive information

**An Automatic Technique**

* The technique is “automatic” in that the independent variables are selected by searching for optimal splits using a measure of “purity” or effect size.
* Much like stepwise regression, a decision tree is not an inferential technique that is well-suited to hypothesis testing.
  + The search itself violates the logical premises of classical statistical testing.
* However, when there are many observations and many variables and in cases where no well-defined theory exists, decision trees can help the researcher to “discover” relationships that can be tested later on a different sample.
* Decision Trees differ in terms of Criteria for branching, Type of target variables, Levels of predictor variables, Stopping rules, “Pruning” approaches, Graphic portrayal of results.

**Wide variety of decision tree models**

Differ in terms of:

* Criteria for branching
* Type of target variables
* Levels of predictor variables
* Stopping rules
* “Pruning” approaches
* Graphic portrayal of results

**Types of decision tree models**

Decision trees go by a variety of names which refer to the same or closely related techniques. All can handle predictor (input) variables that are continuous, nominal, ordinal, or binary.

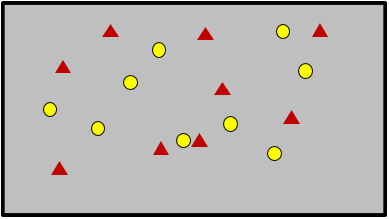
* CHAID (“Chi-Square Automatic Interaction Detection”, uses a chi-square test to identify splits).
  + - The target variable can be continuous, nominal, ordinal, or binary.
    - Binary or multiple category splits.
* CART (“Classification and Regression Trees” – aka “C&RT”; proprietary model)
  + - The target variable can be continuous, nominal, ordinal, or binary.
    - Binary splits only.
* QUEST (Quick, Unbiased, Efficient Statistical Trees)
  + - It is designed to reduce processing time when large data sets are used.
    - The target variable must be categorical or binary.
    - Drops cases with missing data.
* C 5.0 (proprietary model from Rulequest Corporation).
  + - The target variable must be categorical or binary.
    - Robust to the presence of missing values.

**Criteria for splitting and growing a tree**

* Several measures have been developed for evaluating splits in decision trees.
* For categorical target variables:
  + The Gini index
  + Entropy (also called information gain)
  + Chi-square (in the program CHAID)
* For continuous numeric target variables:
  + Reduction in variance
  + The F-Test

**Illustration of "purity" in a tree**

The "root" node: 9 red triangles; 9 yellow circles.



**The Gini impurity index**

* The Gini index is a measure of purity or diversity.
* For binary prediction, the Gini measure works like this. The measure of impurity of a sample A is a function of p0 the proportion of cases that are at level "0" and and p1 the proportion that are at level "1".

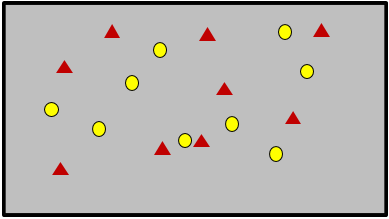
**The Gini index as a function of p0 and p1**



Gini

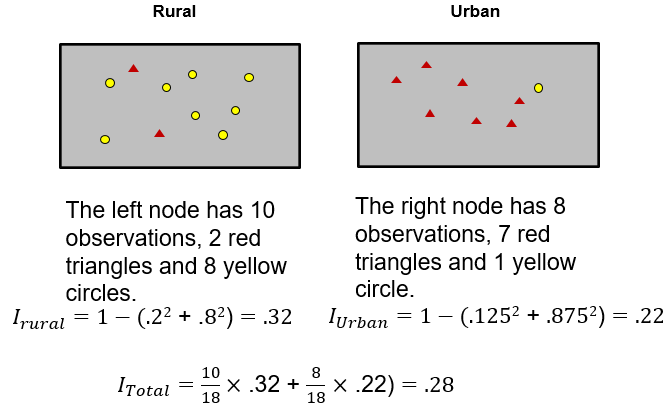
**The Gini index applied to the example**

The "root" node: 9 red triangles; 9 yellow circles.



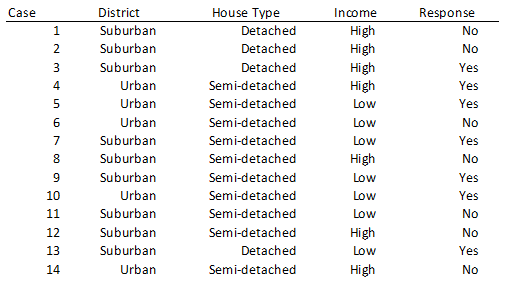


**The Gini index for split on location**

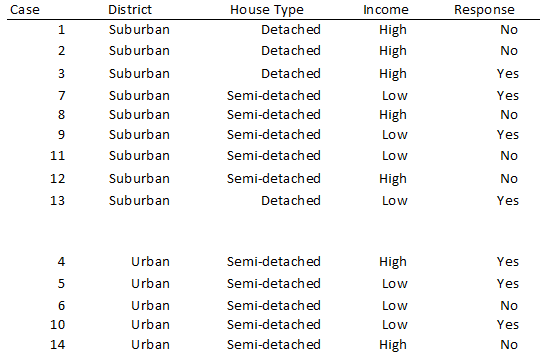


**Example of Decision Trees**

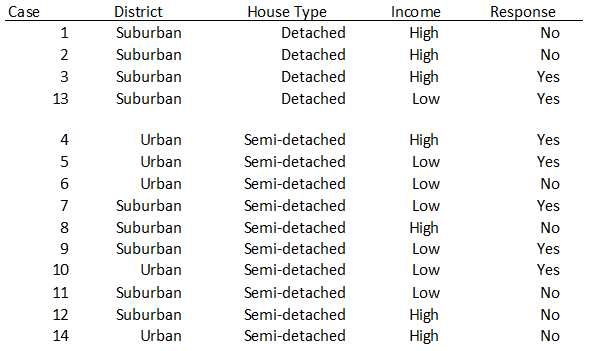
1. Raw Data:



1. Break by District



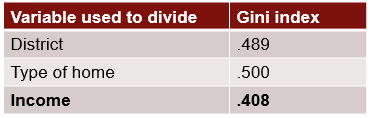
1. Break by type of house



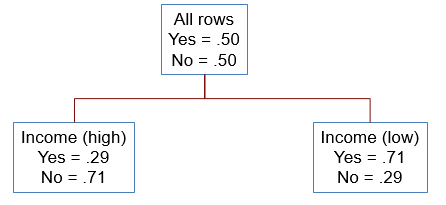
Break by Income



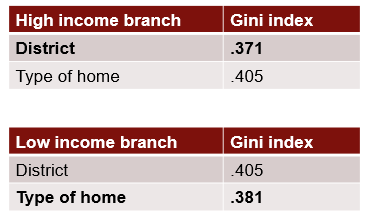
**Calculations**



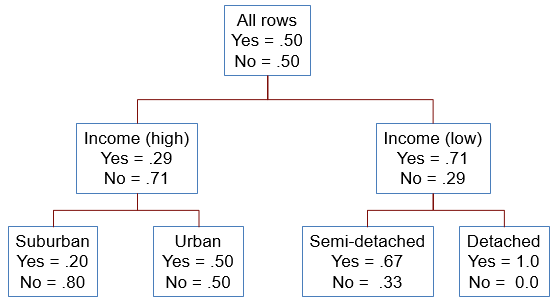
First split on Income



Calculation on the two branches



**The final decision tree**



**The over-fitting problem**

* Natural end of process is 100% purity in each leaf.
* This over-fits the data, which end up fitting noise in the data.
* Over-fitting leads to low predictive accuracy of new data.
* Past a certain point, the error rate for the validation data starts to increase.

**Over-fitting only helps on the training data**



**Pruning decision trees: Two approaches**

1. Avoid growing large trees – by providing a stopping rule such as the minimum number of observations in a node or the maximum number of splits.
2. Grow large tree, cut afterwards, which is known as pruning. The full tree is grown (early stopping might additionally be used), and each split is examined, if it brings a reliable improvement.

**R packages for decision trees**

* Package tree
* Package rpart
* Package party

**Strengths of decision trees and regression trees**

* Interpretation is usually straightforward and easy to demonstrate and explain.
* There are few underlying assumptions that must be met.
* The results are displayed in a tree-like structure, which is intuitively appealing. Rules generated are transparent.
* Interactions among predictor variables can be identified.
* Outliers and missing values can be handled without problems (with most algorithms).
* Non-linear relationships are handled without problems.
* Predictor variable selection is automatic.
* Binary, categorical, ordinal, and interval target and predictor variables can be used.

**Weaknesses of decision trees and regression trees**

* Slight changes in the data set can produce dramatically different results.
* Large datasets are needed.
* Careful validation is required to avoid over fitting.
* The data snooping process can be misleading.
* Considerable judgment and experimentation may be needed to develop a suitable result.

**R code for ChurnTree:** 

## Ensemble Models

**Key Criteria**

* *Diversity of opinion* – Each person should have private information even if its just an eccentric interpretation of the known facts.
* *Independence* – Peoples opinions arent determined by the opinions of those around them.
* *Decentralization* – People are able to specialize and draw on local knowledge.
* *Aggregation* – Some mechanism exists for turning private judgments into a collective decision.

**How do we obtain multiple data sets to build models?**

Answer: Bootstrapping

### Bagging

* Bagging – short for "bootstrap aggregation"
* Assume a data set of size n. Set M = number of repetitions.
* **The bagging process:**
  1. Create a random sample of size n with replacement from the data set. This is sometimes called bootstrapping.
  2. Train a model on the n observations
  3. Record the classification prediction for each observation.
  4. Repeat this process M times.
  5. Take a "vote" on the predicted classifications for a final prediction.

**Evaluation of Bagging**

* The averaging process across multiple bootstrapped samples of the training data can reduce error.
* The bagging process does not use the testing data. The averaging process, however, should lead to better performance on unseen data.
* Bagging also has some limitations:
  + Bagging may not work well if one of the output classes is rare. If this is the case, then oversampling of the rare outcome may be necessary.
  + Bagging works best with "unstable" algorithms (e.g., neural nets, decision trees, regression trees, but not KNN).
  + Bagging can lead to high variances in the accuracy measures if there are outliers in the data. The outliers will be included only in some of the bootstrapped samples and in these cases, the classifier function may be quite different.
  + The explanatory diagram that is provided by decision trees cannot be developed with bagging, since there are literally hundreds of different trees.

### Boosting

* Assume a data set of size n. Set M = the number of trees to create; max depth = number of levels in each tree.
* The boosting process:
  1. Train a decision model using all n obsevations. However, force the model to be a "weak learner" by restricting the number of levels in the decision tree to a small number , e.g., 3, 4, or 5.
  2. For obserations which are predicted incorrectly, increase the importance weight in the sample. Decrease the weights on the correct predictions. Repeat this process M times.
  3. After the M runs, construct a weighted average prediction using all M runs, with weights proportional to the accuracy.

**Evaluation of Boosting**

* Boosting has been shown to be a very powerful method, frequently outperforming other techniques.
* By using weak models, overfitting is usually avoided with boosting.
* The entire data set is used at each iteration; no test data is required to select a good model.
* Boosting also has some limitations:
  + If the training data has a lot of noise (e.g., misclassified obserations) then boosting may not work very well.
  + Boosting can sometimes overfit.
  + The model can fail to converge, instead osciallating predictions with subsequent models.

### Random Forests

* Assume a data set of size n and k predictors; set j << k to be a subset of the k predictors. Set M = number of repetitions.
* The random forest process:
  1. Create a random sample of size n with replacement from the data set.
  2. Also take a random sample of size j from the k preditors.
  3. Form a decision tree; assess the prediction accuracy.
  4. Repeat the process M times.
  5. Average the results for the final predictions.

**Evaluation of Random Forests**

* This method combines the ideas of bagging and boosting.
* Random forests are usually less likely to overfit the training data.
* This approach is useful when the predictors are correlated. By using only subsets of the predictors, every predictor has a "chance" to be in a model.
* Limitations of random forests
  + No tree diagram can be produced.
  + Overfitting is sometimes a problem.

### Ensemble Models in R

* ipred – Bagging
* adabag – Bagging and boosting
* randomForest – Random forest
* gbm – Stochastic gradient boosting
* party – Random forest
* mboost – Boosting glm
* caretEnsemble – Ensemble models using caret (rpart, ipred, gbm, etc.)

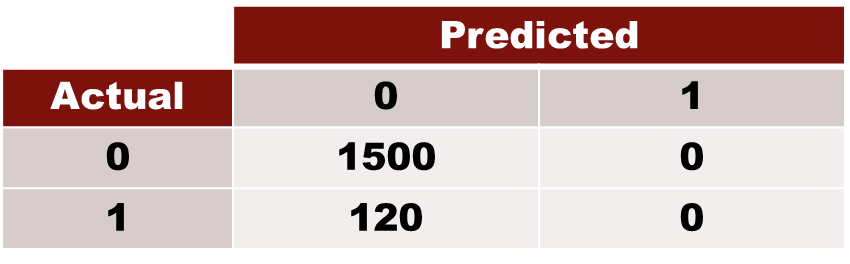
**R code for Ensemble Models: **

# Unsupervised Techniques

## Handling Unbalanced Data

**Problems with Unbalanced data:**

* The problem: A model to predict a binary outcome (0 vs. 1) ends up predicting 100% of the cases in the training (and test) sets to one or the other level.
* The model appears to have very high accuracy.
* Closer inspection shows the result is not at all useful.
* The confusion matrix looks something like this:



Accuracy = 92.6%  
Sensitivity = 0

Specificity = 100%

* If a particular class in a predictive model is sharply underrepresented, the data set is said to be imbalanced.
* Most classification algorithms will perform poorly with imbalanced data because algorithms are designed to minimize overall error.
* There are several approaches to dealing with this problem.

**Example of Imbalanced Data**

* Total number of observations: 10,000
* Fraudulent observations: 200 (code = "1")
* Non-fraudulent observations: 9,800 (code = "0")
* Event rate: 2%

### Solutions for Imbalanced Data

**Under-sampling in Majority class**

* Use all fraudulent observations: 200 (code = "1")
* Sample 10% of the non-fraudulent observations: 980 (code = "0")
* Event rate: 200/1180 = 17%

**Over-sampling in Minority class**

* Replicate the fraudulent observations 20 times yielding a total of 4,000 (code = "1")
* Use all of the non-fraudulent observations: 9,800 (code = "0")
* Event rate: 4000 / (9,800 + 4,000) = 29%

**Synthetic Over-sampling the Minority class**

* Create 20 synthetic instances of the fraudulent observations 20 times yielding a total of 4,000 (code = "1")
  + K nearest neighbors is used to create additional cases.
* Use all of the non-fraudulent observations: 9,800 (code = "0")
* Event rate: 4000 / (9,800 + 4,000) = 29%

**Other Approaches**

* Lower the cutoff to increase the number of predictions of the minority class.
* Adjust the prior probabilities (e.g., in naïve Bayes or discriminant analysis).
* Use unequal case weights (weight the minority class more heavily).
* Use costs to differentially weight specific types of errors.

**R code for Handling Imbalanced Data: **

## Cluster Analysis

* It’s a way of finding groups of observations or grouping observations according to measure of *proximity*
  + Proximity can be assessed as: *similarity or dissimilarity*
* It is essentially the process of finding groups of objects (records, people, items, documents) such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups - within group homogeneity and between group heterogeneity.
  + E.g., 10 objects in cluster A should be quite similar to one another; 8 objects in cluster B should be similar; but the objects in A should be very distinct from those in B
  + Seems like a clear and simple idea, but it’s difficult to operationalize

**Determining number of Clusters**

* No assumptions are made a priori regarding the number of groups or their structure; the goal is to discover “natural groups” within a data structure
* Discovering natural groups is not always easy because of the number of possibilities
  + One approach might be to investigate all possible groupings and then decide (using some criterion) which is the “best” approach.

Example:

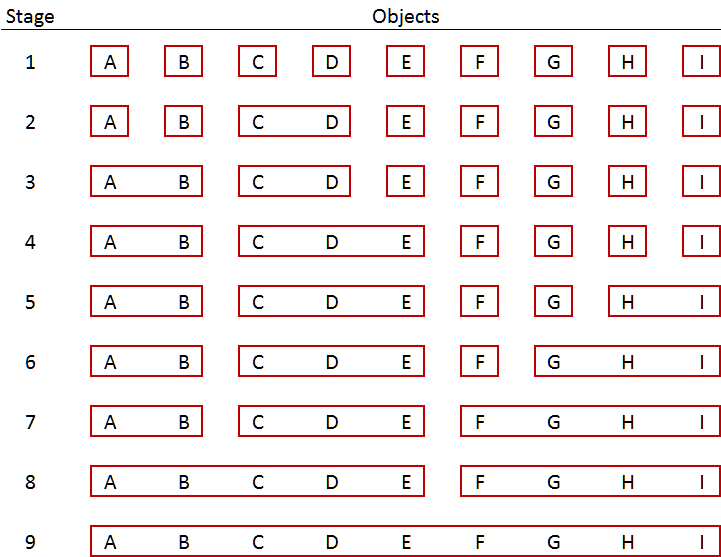
Possible number of partitions with 4 items



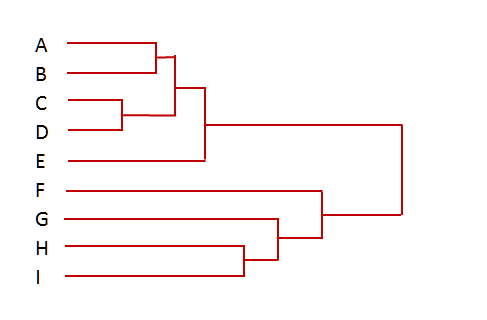
### Hierarchical Clustering

* Each observation begins in a cluster by itself. The two closest observations are merged to form a cluster that replaces the two observations. Merging of the clusters and observations is repeated until only one cluster is left. Once an object or group is joined to another group, it is not allowed to separate at a later stage.
* Clusters are formed based on a pre-specified similarity (or dissimilarity) criterion and a set of decision rules which guide the determination of which groups to form at each stage.
  + *The various clustering methods differ in how the* ***distance between two records is computed and how the distance between two clusters is computed.***
* Hierarchical clustering is not practical for very large data sets because, with most methods, the CPU time varies as the square or cube of the number of observations.

**Example of Hierarchical Clustering:**



**The Dendogram**

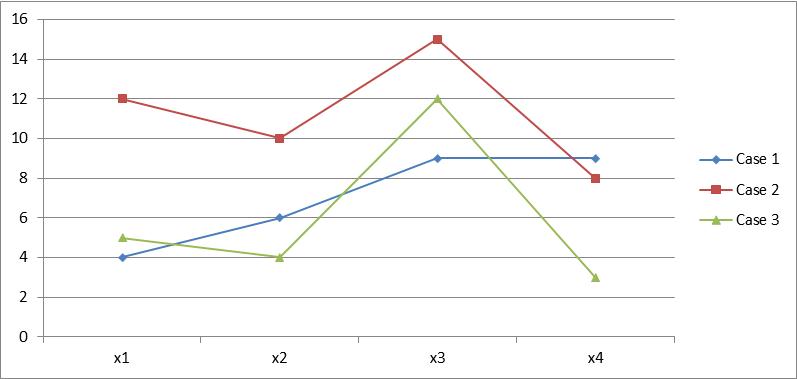
****

**Measuring similarity/dissimilarity**

* To cluster objects, it is necessary to define what criterion is to be used to group the objects. Similarity or distance must be computed between (Case 1 and Case 2), (Case 1 and Case 3) and (Case 2 and Case 3).



**Plot of the data: Which are most similar?**



**Euclidean distance can measure dissimilarity**

* Euclidean distance is typically used to measure dissimilarity. Euclidean distance can be generalized to any number of dimensions. Consider the three cases below. Euclidean distance can be computed between (Case 1 and Case 2), (Case 1 and Case 3) and (Case 2 and Case 3) using the following formula.





**The Euclidean distances:**



**Most similar**

**Correlation is a measure of similarity**



**Most similar**

**Distance (similarity) measures**

* The distance measure provides a quantitative index of the similarity between records or objects
* The most common measure of distance is (squared) Euclidean distance
* This is straightforward, but there can be complications
  + Variables used in computing distance may have vastly different scales
  + Variables may be correlated, which causes extra emphasis on certain variables in defining distance

**Binary data**

****

**Clustering binary data with the Jaccard Measure**

****

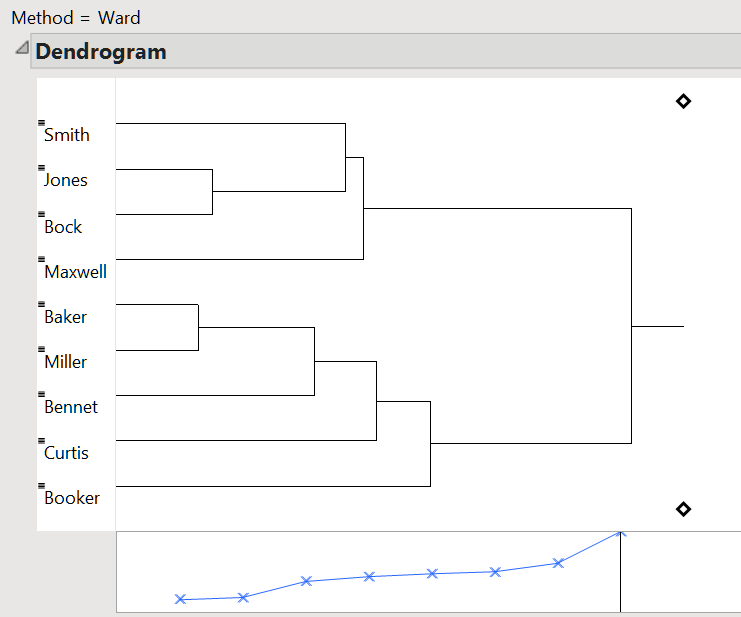
**Similarity based on Jaccard measures**



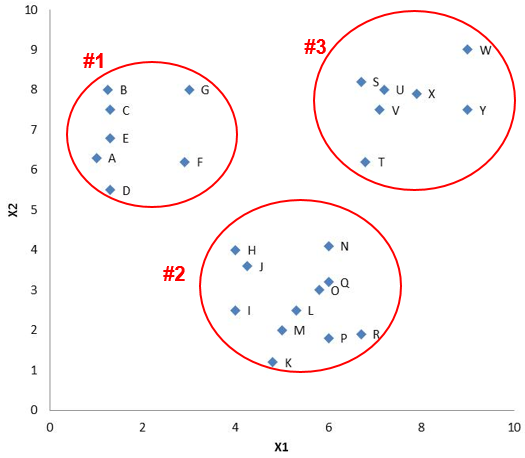
**Conversion to distance matrix**



**Cluster results**



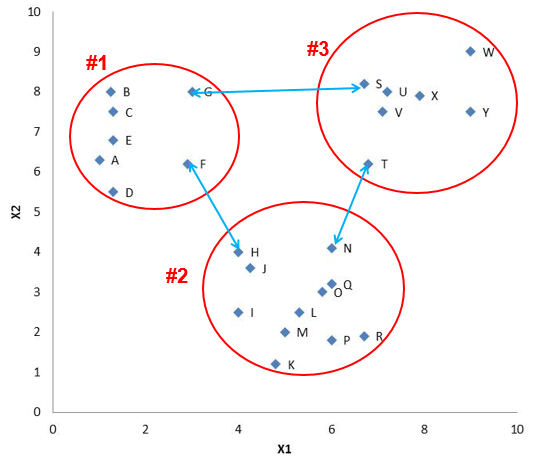
**Measuring distances between clusters**



**Which Cluster to join? It Depends!**

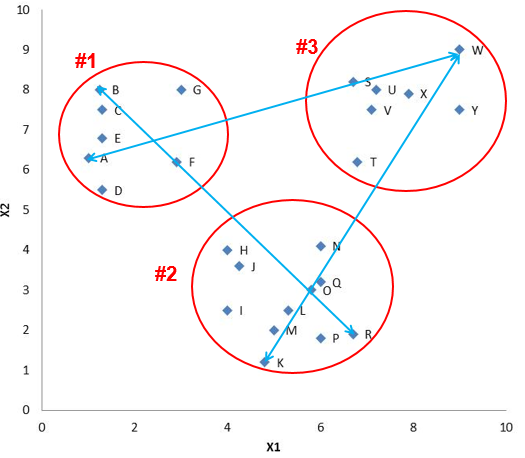
**Nearest neighbor (single linkage)**

Similarity can be defined as the distance is the minimum distance between a pair of points in two clusters (nearest neighbor). (Clusters #2 and #3 are joined.)



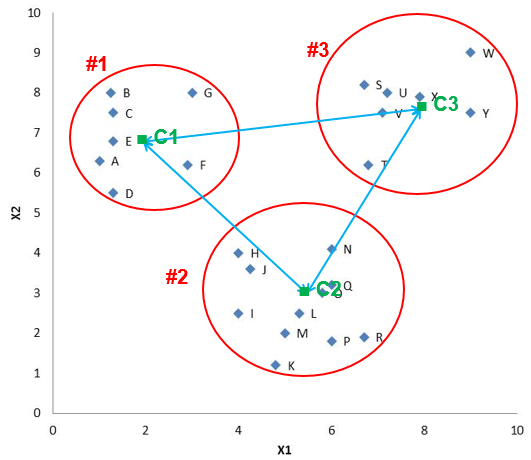
**Farthest neighbor (complete linkage)**

This is a conservative measure of distance based on the maximum distance between any two points (“farthest neighbor” or complete linkage). (Clusters #1 and #2 are joined.)



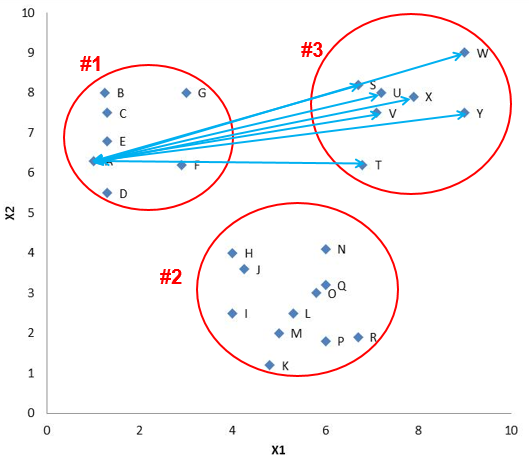
**Centroid method**

Distances between clusters are set equal to the distance between centroids. Note that none of the objects actually have the centroid values. (Clusters #1 and #2 are joined.)



**Average linkage**

This is based on the average of all distances among all pairs of points in two clusters. For examples, cluster #1 contains seven points and cluster #2 seven points. There are 49 interpoint distances and we could take the average of these.



**Ward’s method**

This is based on the loss of information from grouping clusters. The criterion is the minimum within cluster variation. The combination which produces the smallest within-group variance is selected.



**Choosing the linkage method...**

* The choice of linkage relationship can have a dramatic effect on the clustering result.
* Which one should be used?
  + You know what the answer is to the question of which one to use: it depends.
  + We know that the single linkage techniques tend to produce string-like clusters. It also is very sensitive to errors in the distances among observations. However, the single linkage technique is better at identifying clusters that have “curvy” shapes and it tends to be less sensitive to outliers.
  + Ward’s method also seems to be more robust to outliers compared with the average linkage method and the complete linkage method. The centroid method is also less sensitive to outliers.
  + A good approach is to try several methods and to assess whether or not there is general agreement in the results (which would give one clue as to whether well-defined clusters exist).

**Hierarchical cluster analysis in R**

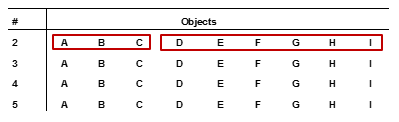
* Before performing a cluster analysis on coordinate data, it is necessary to consider scaling or transforming the variables since variables with large variances tend to have more effect on the resulting clusters than those with small variances.
  + One option is to standardize the data by case to mean 0 and standard deviation 1. (Standardization is not always appropriate, however.) Other options are to row center the data (mean = 0, but spread is not changed).

**R code for Cluster Analysis: **

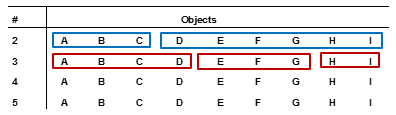
### Non- hierarchical Clustering

**K-means (non-hierarchical or partitioning) clustering**

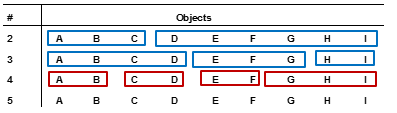
* The k-means procedure performs a disjoint cluster analysis on the basis of distances computed from one or more quantitative variables.
* The observations are divided into clusters such that every observation belongs to one and only one cluster; the clusters do not form a tree structure as they do in the hierarchical procedures.
* The number of clusters must be pre-specified. If you want separate analyses for different numbers of clusters, you must run the k-means procedure once for each analysis.
* The k-means procedure uses Euclidean distances, so the cluster centers (centroids) are based on least-squares estimation. The cluster centers are the means of the observations assigned to each cluster.
* Initial cluster centers (seeds) are chosen randomly in a first pass through the data. Cluster centers change on each pass. It is also possible to read initial centers from a file.
* The k-means procedure can handle large data sets.
* If variables are not standardized, should consider standardization to prevent those with large means and variances from dominating cluster results.
* k-means works like this:
  + Select initial centers (seeds) for specified number of clusters
  + Assign each case to nearest center based on Euclidean distance
  + Compute new centers
  + Reassign cases to nearest new centers
  + Repeat until maximum number of iterations is reached or until the largest change in any cluster center is less than 2% of the minimum distance between initial centers. (These defaults can be overridden.)
* Non-hierarchical (partitioning) methods
  + With this approach clusters are formed directly from proximity (similarity or dissimilarity) measures. A pre-specified number of clusters is formed in a single stage. If the analysis is carried out for different numbers of clusters, objects do not necessarily remain in the same clusters with other objects.
* **K-means: 2 clusters**



**K-means: 3 clusters**



**K-means: 4 clusters**



**Summary**

* Cluster analysis includes a broad set of techniques
* Results are sensitive to researcher decisions
* It can be helpful, but unlikely to yield definitive answers; the main criterion for evaluation is usefulness of the results.
* Hierarchical methods are useful for smaller data sets and allow more options for similarity measures
* Non-hierarchical methods can be used for large data sets, but are sensitive to starting conditions and outliers

**R code for K Means Example:**

## Principal Component Analysis

* It is an unsupervised model – no partitioning of the variables into dependent and independent subsets.
* The objective is to account for as much variance in the set of correlated variables by a sequence of components, where the first component accounts for the most variance, the second component accounts for the second most variance, etc.
* Furthermore, each successive component is uncorrelated (technically, orthogonal) with the other components.
* Each component is merely a weighted average of the observed variables.
* There are potentially as many components as there are variables in the data set.
* However, since the components account for the variance in the data in order of size of the variance, it is frequently possible to approximate the data set with fewer components than the number of variables.

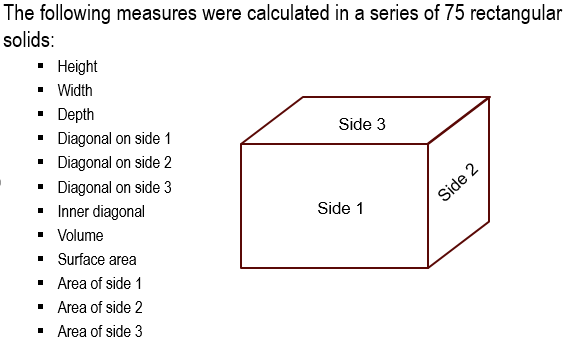
**Dimension Reduction**

* A common situation in data mining and predictive analytics is that the number of potential variables that could be included in an analysis is very large. While data mining is usually associated with automatic methods, where the computer is allowed to search for the “best” variables, there are several problems with doing so.
* Techniques are available to reduce the dimensionality of a data set without sacrificing *much* in the way of predictive or descriptive power.
* Principal components analysis is one such technique. The aim of principal components analysis is to reduce the dimensionality of a data matrix from p variates to a smaller number of k variates so that most of the information in the original data matrix is preserved.

**Why dimension reduction is needed**

* First, if an algorithm is used to select variables, the selection can be unstable in the sense that a different sample of data for the same situation will result in different variables being selected as “best.” There are typically correlations among variables and it is not clear which of the correlated variables should be selected. Sampling variation will often lead to different selections.
* Second, all measurements contain error. In many cases, combining several related measures into one summary measure can reduce error, resulting in a measure that is superior to the individual variables.
* Third, including a large number of variables in an analysis can increase computation time with little overall benefit to accuracy.
* Fourth, when a large number of variables is included, interpretation and communication of results can become difficult.

**Example of dimension reduction**

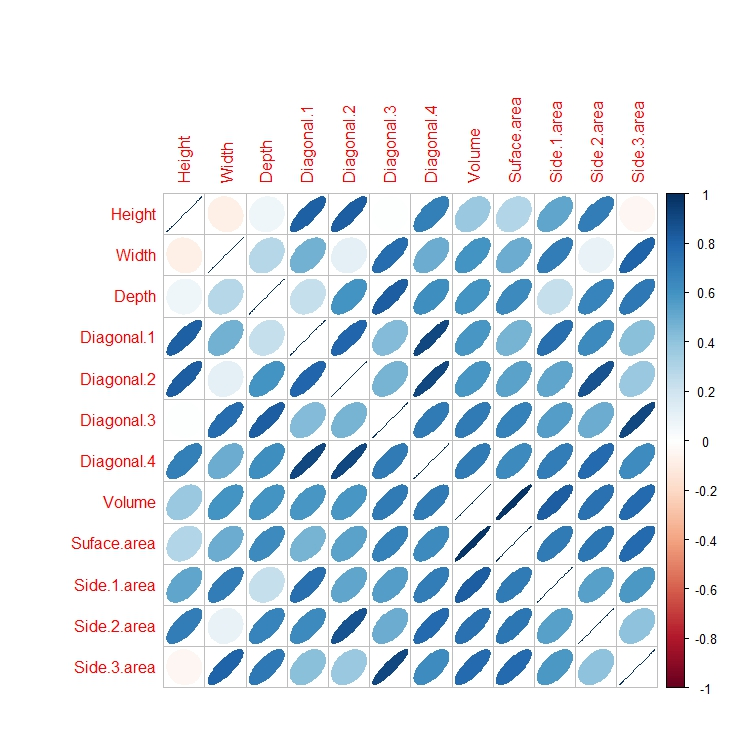


**Approaches to variable reduction**

* Apply domain knowledge
* Examine relationships of variables with the target variable one at a time
* Employ an automated search procedure such as stepwise logit or stepwise regression
* Use a variable reduction technique such as principal components analysis.

**Correlation Matrix**

* Principal components analysis can be applied to the correlation matrix or the covariance matrix.
* The most common approach is to use the correlation matrix.
* Since there are 12 variables, the total variance in the correlation matrix is 12.0.



**Total variance of this correlation matrix is 12.0**



**Data Reduction**

What does Data Reduction mean?

* The loading matrix for all components times the transpose of the loading matrix will reproduce the original correlation matrix.
* The loading matrix for the first three components times the transpose of that matrix will approximate the original correlation matrix.

### Using Principal Component Analysis

**Steps in using PCA include:**

* Selecting variables.
* Obtaining measurements on the variables (the data matrix).
* Computing the correlation (or covariance) matrix. Typically, PCA is performed on the covariance or correlation matrix; the results are not the same.
* Extracting a set of components.
* Determining how many components to retain.
* Interpreting the components, possibly after further rotation.
* Developing scores on the components for further analyses.

**Selecting Variables**

* Typically continuous variables are used in principal components analysis. Binary variables can be used if the overall proportion is not close to 1.0 or 0.0.
  + For continuous measures, should have at least three levels for each variable.
* Variables should not be highly skewed.
* Variables should be highly enough correlated so that principal components makes sense. (Tests are available, but are not suited for large samples.)

**Scaling the raw data**

* Principal components analysis can be used with raw data (cross-products matrix), the covariance matrix, or the correlation matrix.
* Since principal components analysis seeks to maximize the variance accounted for by the components, it is sensitive to the scale differences among variables.
  + For example, including a variable such as “corporate sales” expressed in dollars will have a much larger impact on the results than a smaller number such as number of employees if the variables are not standardized.
  + Standardizing insures that the data are expressed in comparable units.
* Most applications of principal components analysis analyze the correlation matrix.
  + However, know that information is “lost” when the correlation matrix is used, since it essentially standardizes the data.

**Selecting the number of components to retain**

* PCA is typically used to reduce the dimensionality of the data.
  + Statistical tests have been developed, but these are sensitive to sample size and will usually indicate too many components if the sample is large.
* An easy rule of thumb developed by Kaiser:
  + Retain only the principal components with eigenvalues exceeding one, assuming the correlation matrix is being analyzed.
  + This rule is based on the notion that any principal component should account for at least as much variance as the any one of the original variables (which were all scaled to have variance equal to 1.0).
* A better way: examine the scree plot (proposed by Cattell). Plot the eigenvalue (or variance) accounted for by each principal component in order from the largest to the smallest. Look for an “elbow” in the plot; retain only the components above the elbow.
* Other procedures are based on simulation using random data.

**Rotating the loading matrix**

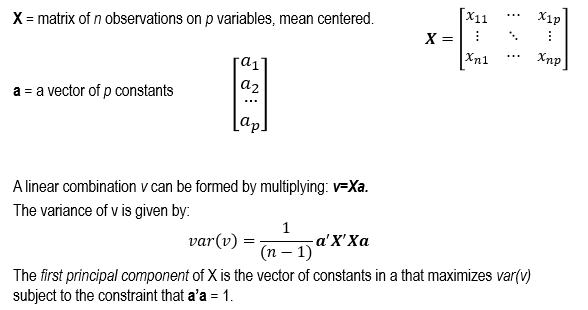
* The principal component solution can be rotated since the solution is not unique. This is known as rotational indeterminacy. The initial solution is done to maximize the variance explained by the first component, but this result may not be easily interpretable.
* Therefore, a rotation of the axes of the extracted components is obtained, such that the loading matrix is simplified for easier interpretation.
* The goal of the rotation is to find a set of axes such that each variable loads highly only on one (or at most a couple) of components and nearly zero on the rest. A Varimax solution can do this. An added advantage of Varimax is that it keeps the components uncorrelated.

**Some issues with PCA**

* No strong criteria exist to test the solution; typically, there are no hypothesis tests.
* Deciding on the number of components is somewhat arbitrary.
* Virtually any data set (even “bad” data) can be submitted to PCA and results will be obtained.
* Interpretation can be difficult and ambiguous.

### Some mathematics of Principal Components Analysis

**Mechanics of PCA**



**Eigenvalues and Eigenvectors**

* When a vector is multiplied by a matrix A, the vector can be rotated and/or stretched.
* It so happens that there are special vectors for which multiplication by a matrix A results in no rotation, but only stretching of the vector. This vector is known as an eigenvector and is defined as follows:



**Finding Eigenvalues and Eigenvectors**

* To find the eigenvectors and eigenvalues of a matrix A, the following equation (called the characteristic equation) is solved:
* The λ are found as solutions to the following:

det

**Implications of the properties just described**

* Obtain a set of *n* observations on *p* variables and place these in a matrix X.
* Find the correlation (or covariance) matrix of X; call it R (or C).
* Find the eigenvalues and eigenvectors of R; place the eigenvalues in a diagonal matrix D (with the largest eigenvalue in the upper left corner).
  + Place the corresponding eigenvectors in U.
* Rearrange the expression D = U ′RU by pre-multiplying by U and post-multiplying by U′, so we have UDU ′ = UU ′RUU ′ = R, sinceU ′=U-1.

**PCA on correlation matrices**

* Begin with an n x p matrix of observed values **X**.
* Standardize **X** to **X**stdz.
* Form the correlation matrix **R**xx = 1/(n-1) **X’**stdz **X**stdz.
* Find the eigenvector matrix **V** of **R**xx. **V** is p x p.
* Compute the component scores:
  + - **Z**scores = **X**stdz **V.**
    - **Z** is n x p.
* Standardize **Z**scores to **Z**stdz.
* Next, compute the “loadings” of the variables on the components (which are the correlatetions between the original values and the component scores).
  + - **R**xz = 1/(n-1) **X’**stdz **Z**stdz

**R code for Principal Component Analysis:**

--------------------------------------------------------xxxxxxxxxxxxxxxxxx---------------------------------------------------------

**Reference:** Dr. Frank Acito, Kelley School of Business, Indiana University