PERFORMANCE EVALUATION OF BOWLERS USING R

ITCS 5102 Survey of Programming Languages

Team Members

KISHORE CHITTAMURU

REVANTH MALAY

SAITH KUMAR GUNDU

RAHUL REDDY NANNURU

SRI SAI KUMAR RAVIPATI

Abstract:

In this project we are going to analyze the bowling performance of cricketer based on certain attributes of the player such as frequency of the wickets taken in a particular range, wickets taken vs runs given etc.., Usually number of wickets taken is generally considered as the parameters for the evaluating performance of a bowler. But in our project we delve deeper into the analysis so that we can evaluate performance of the player in different perspectives. In this project we have used R language as it can perfectly capture the analysis of the data.

Software Requirements:

- Windows(OS).
- R Studio.

Hardware Requirements:

- RAM: >=4 GB

Introduction:

Basically in cricket we do have 3 kinds of formats. They are T20, Test and One day. Selection of bowler for a particular format is difficult task as some players are well suited for T20 but not for the test matches. Generally cricket teams will have 4 bowlers with one all-rounder(who can bat and bowl) so that the quota of the overs as well as the limit is taken care of and also choosing appropriate players for the team plays major role in winning the matches.

In order to choose players we have to see many parameters so that we can evaluate the performance of players in many angles. In this project we are presenting solution to the above problem using R language. The parameters what we had taken into consideration are as below.

- Number of wickets taken for e.g 1 wicket x%, 2 wickets y% etc.
- Number of wickets taken vs runs given.
- Average number of wickets at different venue.
- Average wickets under different opposition.
- Relative economy rate against wickets taken.
- Wickets taken in each year of their tenure.
- Future wickets forecast.
- Contribution to matches lost and won.
- Performance in home vs overseas.

Input file:

For this project we have taken statistics of top 3 spinners from Espncricinfo. For sample analysis we have taken the statics of the following players.

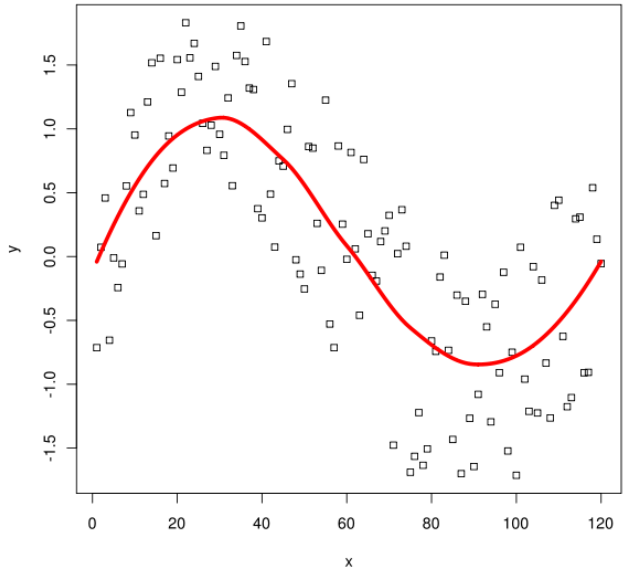
- Anil Kumble
- Shane Warne
- Muttiah Muralitharan

➢ **Note:** We have taken the statics of test matches for this implementation. So this performance evaluation can be considered while selecting players for the test matches.

Methods used:

LOWESS REGRESSION ANALYSIS:

LOWESS (Locally Weighted Scatterplot Smoothing), sometimes called LOESS (locally weighted smoothing), is a popular tool used in regression analysis that creates a smooth line through a time-plot or scatter plot to help you to see relationship between variables and foresee trends.



The smoothing process is viewed as local, where each smoothed value is determined by neighboring data points defined within the span. The regression weight function is defined for the data points contained within the span as the process is weighted. In addition to the regression weight function, a robust weight function can also be used, which makes the process resistant to outliers. Finally, the methods are differentiated by the model used in the regression: lowess uses a linear polynomial, while loess uses a quadratic polynomial**.**

Local Regression Smoothing Procedure The local regression smoothing process follows these steps for each data point:

The regression weights for each data point in the span is computed. The tricube function gives the weights as the shown below.

x is the predictor value associated with the response value to be smoothed, xi are the nearest neighbors of x as defined by the span, and d(x) is the distance along the abscissa from x to the most distant predictor value within the span.

The weights have these characteristics:

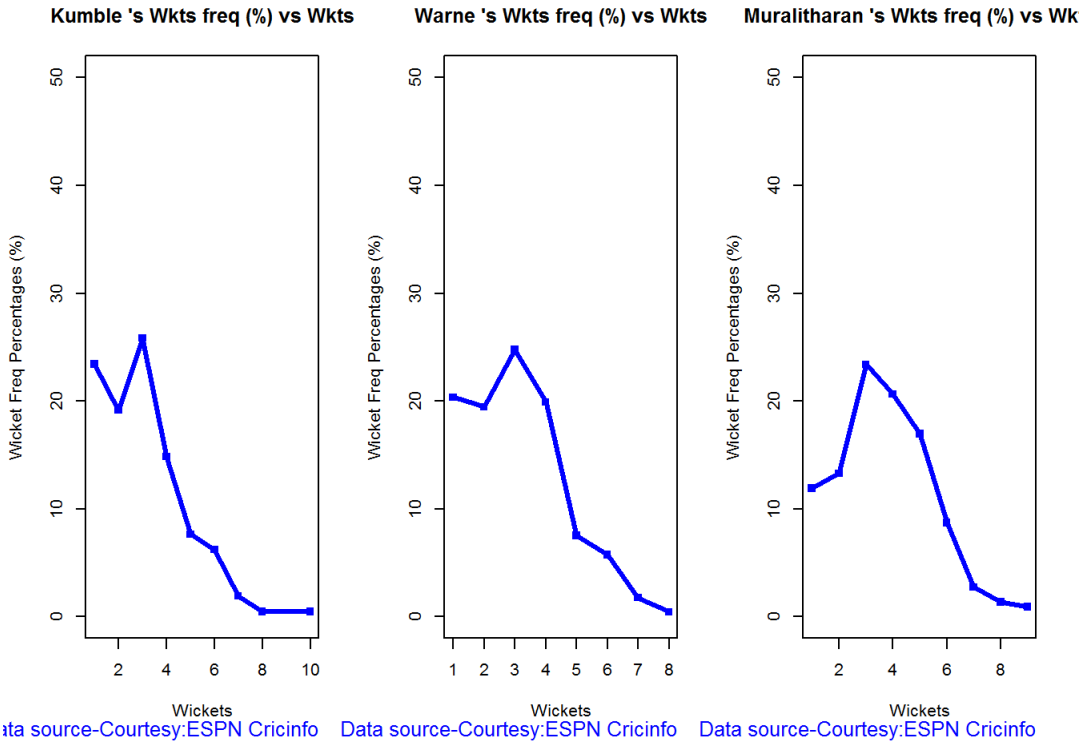
- The data point to be smoothed has the largest weight and the most influence on the fit.

- Data points outside the span have zero weight and no influence on the fit.

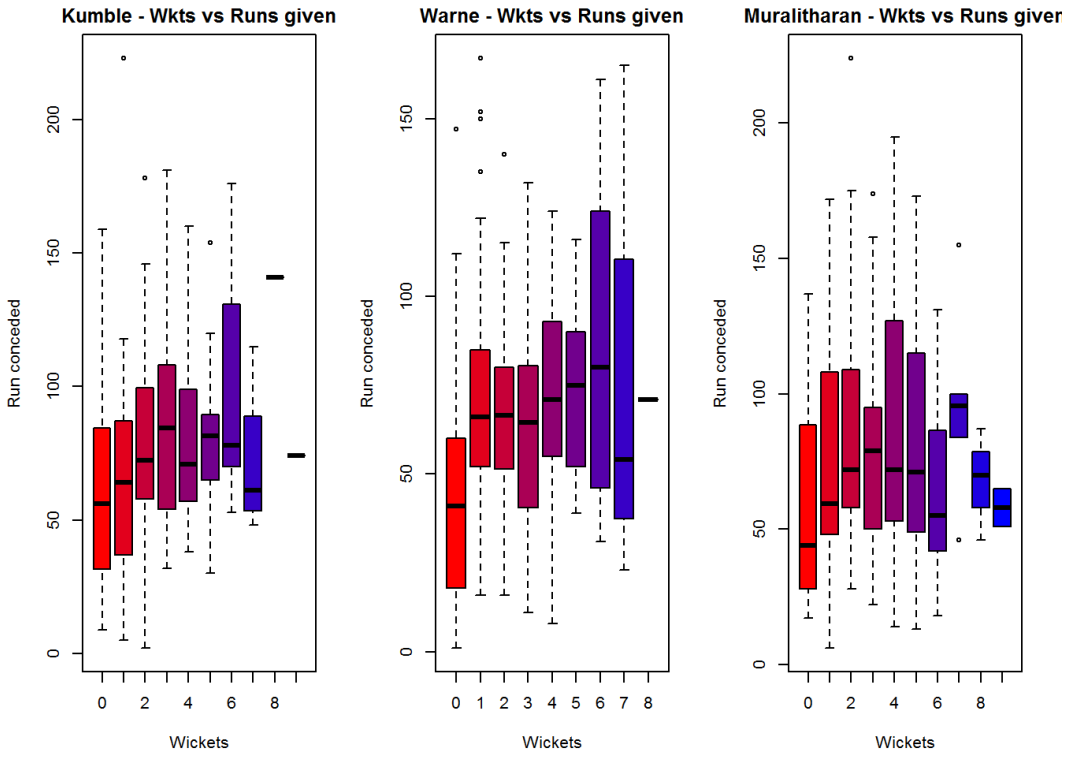
A weighted linear least squares regression is performed. For lowess, the regression uses a first degree polynomial. For loess, the regression uses a second degree polynomial.
The smoothed value is given by the weighted regression at the predictor value of interest. If the smooth calculation involves the same number of neighboring data points on either side of the smoothed data point, the weight function is symmetric.

However, the weight function is not symmetric if the number of neighboring points is not symmetric about the smoothed data point, then. Note that unlike the moving average smoothing process, the span never changes. For example, when you smooth the data point with the smallest predictor value, the shape of the weight function is truncated by one half, the leftmost data point in the span has the largest weight, and all the neighboring points are to the right of the smoothed value. The weight function for an end point and for an interior point is shown below for a span of 31 data points. Using the lowess method with a span of five, the smoothed values and associated regressions for the first four data points of a generated data set are shown below. Notice that the span does not change as the smoothing process progresses from data point to data point. However, depending on the number of nearest neighbors, the regression weight function might not be symmetric about the data point to be smoothed. In particular, plots (a) and (b) use an asymmetric weight function, while plots (c) and (d) use a symmetric weight function. For the loess method, the graphs would look the same except the smoothed value would be generated by a second-degree polynomial.

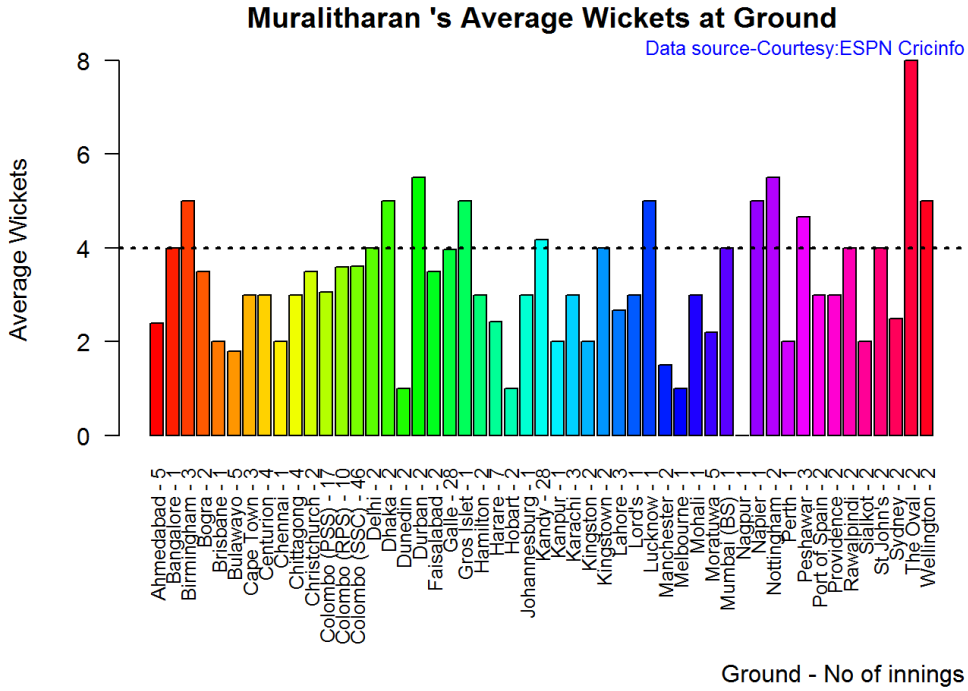
Output:



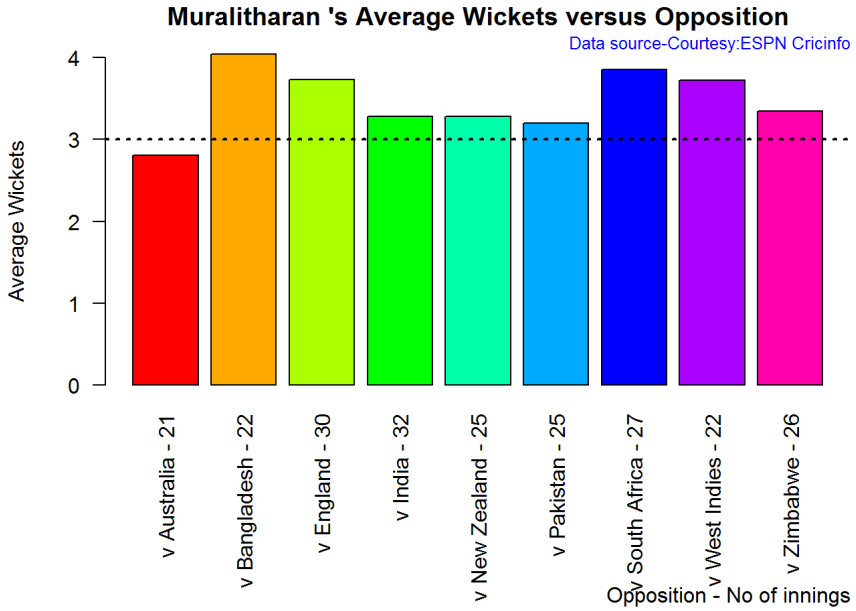

```
par(mfrow=c(1,3))
par(mar=c(4,4,2,2))
bowlerWktsFreqPercent("./kumble.csv","Kumble")
bowlerWktsFreqPercent("./warne.csv","Warne")
bowlerWktsFreqPercent("./murali.csv","Muralitharan")
```



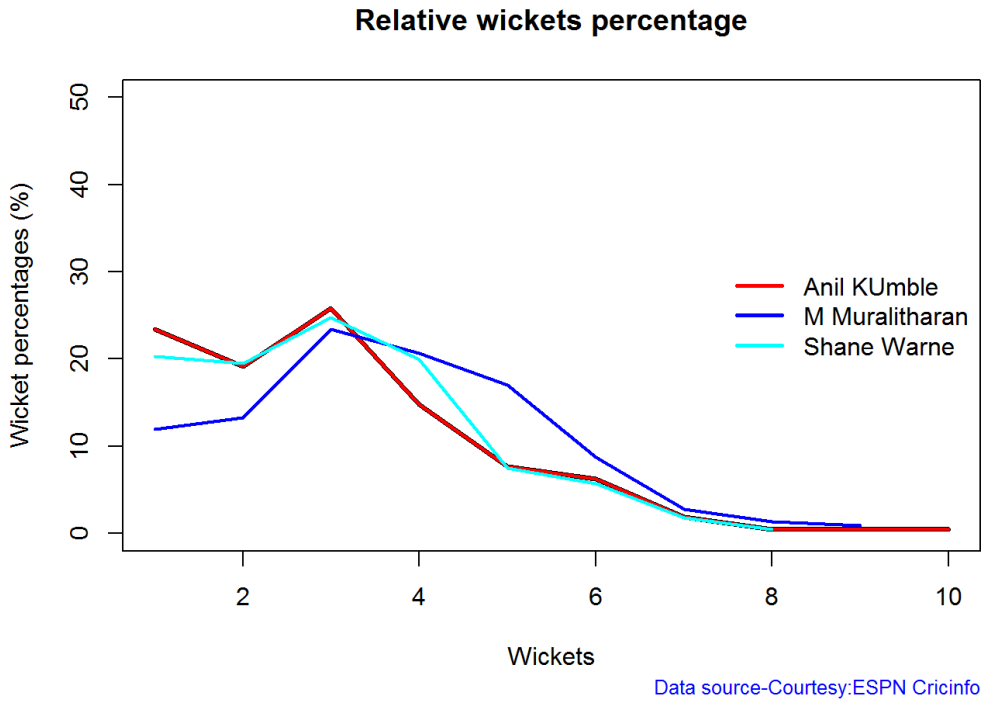

```
par(mfrow=c(1,3))
par(mar=c(4,4,2,2))
bowlerWktsRunsPlot("./kumble.csv","Kumble")
bowlerWktsRunsPlot("./warne.csv","Warne")
bowlerWktsRunsPlot("./murali.csv","Muralitharan")
```



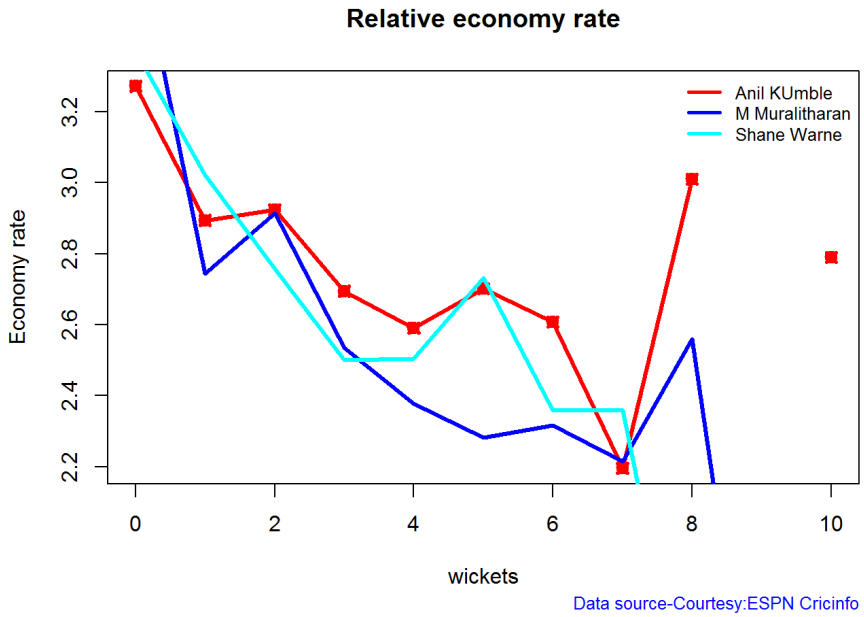

```
bowlerAvgWktsGround("./murali.csv","Muralitharan")
```




```
bowlerAvgWktsOpposition("./murali.csv","Muralitharan")
```




```
frames <- list("./kumble.csv","./murali.csv","warne.csv")
names <- list("Anil KUmble","M Muralitharan","Shane Warne")
relativeBowlingPerf(frames,names)
```




```
frames <- list("./kumble.csv","./murali.csv","warne.csv")
names <- list("Anil KUmble","M Muralitharan","Shane Warne")
relativeBowlingER(frames,names)
```

Language History:

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. 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.

Elements of the language:

Keywords: Keyword	**Description**
if, else, while, for, next, repeat, break	These are the basic control-flow constructs of the R language.
function	These functions provide the base mechanisms for defining new functions in the R language.
TRUE & FALSE	These symbols represent the logical values “true” and “false”. The result of a condition is a logical value.
NULL	NULL represents the null object in R. NULL is often returned by expressions and functions whose value is undefined.
NA	This symbol represents a missing or unknown value.
Inf	This symbol is used to represent infinity (as the result of an arithmetic expression).
NaN	This symbol is used to represent an arithmetic result that is undefined (Not A Number).
	Description

Data Types: Data type	
Vector	Vectors are the simplest R objects, an ordered list of primitive R objects of a given type (e.g. real numbers, strings, and logicals).
Factor	factor() transforms a vector into a factor. It is a sequence assigning a category to each index.
Matrix & Array	Both are vectors with ‘2’ dimensions and ‘n’ dimensions simultaneously.
List	A collection of R objects.

Data Frame	A dataframe has been referred to as "a list of variables/vectors of the same length".
Environment	Hash table. A collection of key-value pairs.

Description of Language Syntax:

R is a case-sensitive, interpreted language. The syntax of R language has a superficial similarity with C, but the semantics are of the FPL (functional programming language) variety with stronger affinities with Lisp and APL.
In particular, it allows "computing on the language", which in turn makes it possible to write functions that take expressions as input, something that is often useful for statistical modeling and graphics.
It is possible to get quite far using R interactively, executing simple expressions from the command line. Some users may never need to go beyond that level, others will want to write their own functions either in an ad hoc fashion to systematize repetitive work or with the perspective of writing add-on packages for new functionality.

Control Structure:

Conditional Execution: (If Statements)

- **If-else** expression runs different branches of code and also evaluates to a different value depending on the Boolean expression given.

Syntax: if (condition) expr_1 else expr_2

Where 'condition' must evaluate to a single logical value and the result of the entire expression is then evident.
There is a vectorized version of the if/else construct, the 'ifelse' function.
Syntax: ifelse(condition, a, b)
It returns a vector of the length of its longest argument, with elements a[i] if condition[i] is true, otherwise b[i].

Repetitive Execution: (while, for & repeat)

- **While Loop:** The while expression is used to perform iterative execution (looping) while a specified test condition is true.

Syntax: while (condition) expr
Where 'condition' is the any Boolean expression, expr is any statement evaluated when condition is true.

- **For loop**: This looping construct is used to iterate over the matches of a pattern in an enumerable collection such as a range expression, sequence, list, array, or other construct that supports enumeration.

Syntax: for (name in expr_1) expr_2
Where name is the loop variable, expr_1 is a vector expression, (often a sequence like 1:20), expr_2 is repeatedly evaluated as name ranges through the values in the vector result of expr_1.

- **Repeat:** This is used to execute a block of statements until it satisfies a certain condition which is explicitly defined by the user.

Syntax: repeat *expr*
Where 'expr' is any statement which is executed on a repetitive criteria.

- The **break** statement can be used to terminate any loop, possibly abnormally. This is the only way to terminate repeat loops.

- The **next** statement can be used to discontinue one particular cycle and skip to the "next".

Programming Paradigm:

- R supports a mixture of object-oriented and functional programming paradigms.
- R is a functional programming language, meaning that functions are first-order data types, they can be declared and used in exactly the same way that any other variable can be used.

Functional programming makes no distinction between values and functions, so we can consider functions to be equal to all other data types. That means that we can:

- create a token (the function variable name) and associate it with a type
- assign it a value (the actual calculation)
- interrogate its value (perform the calculation)
- pass a function as a parameter of another function or sub-routine
- return a function as the result of another function

On the functional side it:

- has first class functions
- has lazy evaluation of arguments
- encourages pure, side-effect free functions

But,

- it does not implement tail call recursion
- and it's easy to create non-pure functions

On the object Oriented side it:

- Has three built in OO paradigms: S3 and S4, which are immutable and support generic function style OO, and reference classes (aka R5) which are mutable, and support the more common message-passing style OO.
- S4 is heavily influenced by the OO-style of common lisp (CLOS) and dylan.
- There are also a number of contributed packages that provide other types of OO: proto, mutatr, R.oo, OOP.

But,

- The built-in OO tools provide little in the way of syntactic sugar.

Advantages and Disadvantages:

Advantages:

R is the most comprehensive statistical analysis package available. It incorporates all of the standard statistical tests, models, and analyses, as well as providing a comprehensive language for managing and manipulating data.

R is a programming language and environment developed for statistical analysis by practicing statisticians and researchers. It reflects well on a very competent community of computational statisticians. The graphical capabilities of R are outstanding, providing a fully programmable graphics language that surpasses most other statistical and graphical packages.

Apart From above mentioned strengths here are few

- **Open Language:**

R is free and open source software, allowing anyone to use and, importantly, to modify it. R is licensed under the GNU General Public License, with copyright held by The R Foundation for Statistical Computing. R has no license restrictions (other than ensuring our freedom to use it at our own discretion), and so we can run it anywhere and at any time, and even sell it under the conditions of the license.

- **Cross Platform:**

R is cross-platform. R runs on many operating systems and different hardware. It is popularly used on GNU/Linux, Macintosh, and Microsoft Windows, running on both 32 and 64 bit processors.

- **High Performance:**

R is well suited for performing heavy numeric computations across large data sets. It has been successfully used in financial, statistical, parallel, scientific, engineering, testing, and event-processing software components.

- **Extensibility**

R plays well with many other tools, importing data, for example, from CSV files, SAS, and SPSS, or directly from Microsoft Excel, Microsoft Access, Oracle, MySQL, and SQLite. It can also produce graphics output in PDF, JPG, PNG, and SVG formats, and table output for LATEX and HTML.

Disadvantages:

Many R commands give little thought to memory management, and so R can very quickly consume all available memory. This can be a restriction when doing data mining. There are various solutions, including using 64 bit operating systems that can access much more memory than 32 bit ones.