This post includes a template which you can use for analyzing the performances of cricketers, both batsmen and bowlers in Test, ODI and Twenty 20 cricket using my R package **cricketr**. To see actual usage of functions in the R package cricketr see [Introducing cricketr! : An R package to analyze performances of cricketers](https://gigadom.in/2015/07/04/introducing-cricketr-a-r-package-to-analyze-performances-of-cricketers/).

This template can be downloaded from Github at [cricketer-template](https://github.com/tvganesh/cricketr-template)

The ‘cricketr’ package uses the statistics info available in [ESPN Cricinfo Statsguru](http://stats.espncricinfo.com/ci/engine/stats/index.html). The current version of this package supports all formats of the game including Test, ODI and Twenty20 versions.

You should be able to install the package from GitHub and use the many functions available in the package. Please mindful of the [ESPN Cricinfo Terms of Use](http://www.espncricinfo.com/ci/content/site/company/terms_use.html)

Take a look at my short video tutorial on my R package cricketr on Youtube – [R package cricketr – A short tutorial](https://www.youtube.com/edit?o=U&video_id=q9uMPFVsXsI)

Do check out my interactive Shiny app implementation using the cricketr package – [Sixer – R package cricketr’s new Shiny avatar](https://tvganesh.shinyapps.io/Sixer)

**The cricketr package**

The cricketr package has several functions that perform several different analyses on both batsman and bowlers. The package has function that plot percentage frequency runs or wickets, runs likelihood for a batsman, relative run/strike rates of batsman and relative performance/economy rate for bowlers are available.

Other interesting functions include batting performance moving average, forecast and a function to check whether the batsmans in in-form or out-of-form.

The data for a particular player can be obtained with the getPlayerData() function. To do you will need to go to [ESPN CricInfo Player](http://www.espncricinfo.com/ci/content/player/index.html) and type in the name of the player for e.g Ricky Ponting, Sachin Tendulkar etc. This will bring up a page which have the profile number for the player e.g. for Sachin Tendulkar this would be <http://www.espncricinfo.com/india/content/player/35320.html>. Hence, Sachin’s profile is 35320. This can be used to get the data for Tendulkar as shown below

The cricketr package is now available from **CRAN!!!** You should be able to install directly with

**1. Install the cricketr package**

if (!require("cricketr")){

install.packages("cricketr",lib = "c:/test")

}

library(cricketr)

The cricketr package includes some pre-packaged sample (.csv) files. You can use these sample to test functions as shown below

# Retrieve the file path of a data file installed with cricketr

#pathToFile <- system.file("data", "tendulkar.csv", package = "cricketr")

#batsman4s(pathToFile, "Sachin Tendulkar")

# The general format is pkg-function(pathToFile,par1,...)

#batsman4s(,"Sachin Tendulkar")

“` The pre-packaged files can be accessed as shown above. To get the data of any player use the function in Test, ODI and Twenty20 use the following

**2. For Test cricket**

#tendulkar <- getPlayerData(35320,dir="..",file="tendulkar.csv",type="batting",homeOrAway=c(1,2), result=c(1,2,4))

**2a. For ODI cricket**

#tendulkarOD <- getPlayerDataOD(35320,dir="..",file="tendulkarOD.csv",type="batting")

**2b For Twenty 20 cricket**

#tendulkarT20 <- getPlayerDataTT(35320,dir="..",file="tendulkarT20.csv",type="batting")

**Analysis of batsmen**

**Important Note** This needs to be done only once for a player. This function stores the player’s data in a CSV file (for e.g. tendulkar.csv as above) which can then be reused for all other functions. Once we have the data for the players many analyses can be done. This post will use the stored CSV file obtained with a prior getPlayerData for all subsequent analyses

**Sachin Tendulkar’s performance – Basic Analyses**

The 3 plots below provide the following for Tendulkar

1. Frequency percentage of runs in each run range over the whole career
2. Mean Strike Rate for runs scored in the given range
3. A histogram of runs frequency percentages in runs ranges For example

**3. Basic analyses**

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#batsmanRunsFreqPerf("./tendulkar.csv","Tendulkar")

#batsmanMeanStrikeRate("./tendulkar.csv","Tendulkar")

#batsmanRunsRanges("./tendulkar.csv","Tendulkar")

dev.off()

## null device

## 1

1. Player 1
2. Player 2
3. Player 3
4. Player 4

**4. More analyses**

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#batsman4s("./player1.csv","Player1")

#batsman6s("./player1.csv","Player1")

#batsmanMeanStrikeRate("./player1.csv","Player1")

# For ODI and T20

#batsmanScoringRateODTT("./player1.csv","Player1")

dev.off()

## null device

## 1

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#batsman4s("./player2.csv","Player2")

#batsman6s("./player2.csv","Player2")

#batsmanMeanStrikeRate("./player2.csv","Player2")

# For ODI and T20

#batsmanScoringRateODTT("./player1.csv","Player1")

dev.off()

## null device

## 1

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#batsman4s("./player3.csv","Player3")

#batsman6s("./player3.csv","Player3")

#batsmanMeanStrikeRate("./player3.csv","Player3")

# For ODI and T20

#batsmanScoringRateODTT("./player1.csv","Player1")

dev.off()

## null device

## 1

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#batsman4s("./player4.csv","Player4")

#batsman6s("./player4.csv","Player4")

#batsmanMeanStrikeRate("./player4.csv","Player4")

# For ODI and T20

#batsmanScoringRateODTT("./player1.csv","Player1")

dev.off()

## null device

## 1

*Note*: For mean strike rate in ODI and Twenty20 use the function batsmanScoringRateODTT()

**5.Boxplot histogram plot**

This plot shows a combined boxplot of the Runs ranges and a histogram of the Runs Frequency

#batsmanPerfBoxHist("./player1.csv","Player1")

#batsmanPerfBoxHist("./player2.csv","Player2")

#batsmanPerfBoxHist("./player3.csv","Player3")

#batsmanPerfBoxHist("./player4.csv","Player4")

**6. Contribution to won and lost matches**

For the 2 functions below you will have to use the getPlayerDataSp() function. I have commented this as I already have these files. This function can only be used for Test matches

#player1sp <- getPlayerDataSp(xxxx,tdir=".",tfile="player1sp.csv",ttype="batting")

#player2sp <- getPlayerDataSp(xxxx,tdir=".",tfile="player2sp.csv",ttype="batting")

#player3sp <- getPlayerDataSp(xxxx,tdir=".",tfile="player3sp.csv",ttype="batting")

#player4sp <- getPlayerDataSp(xxxx,tdir=".",tfile="player4sp.csv",ttype="batting")

par(mfrow=c(2,2))

par(mar=c(4,4,2,2))

#batsmanContributionWonLost("player1sp.csv","Player1")

#batsmanContributionWonLost("player2sp.csv","Player2")

#batsmanContributionWonLost("player3sp.csv","Player3")

#batsmanContributionWonLost("player4sp.csv","Player4")

dev.off()

## null device

## 1

**7, Performance at home and overseas**

This function also requires the use of getPlayerDataSp() as shown above. This can only be used for Test matches

par(mfrow=c(2,2))

par(mar=c(4,4,2,2))

#batsmanPerfHomeAway("player1sp.csv","Player1")

#batsmanPerfHomeAway("player2sp.csv","Player2")

#batsmanPerfHomeAway("player3sp.csv","Player3")

#batsmanPerfHomeAway("player4sp.csv","Player4")

dev.off()

## null device

## 1

**8. Batsman average at different venues**

par(mfrow=c(2,2))

par(mar=c(4,4,2,2))

#batsmanAvgRunsGround("./player1.csv","Player1")

#batsmanAvgRunsGround("./player2.csv","Player2")

#batsmanAvgRunsGround("./player3.csv","Ponting")

#batsmanAvgRunsGround("./player4.csv","Player4")

dev.off()

## null device

## 1

**9. Batsman average against different opposition**

par(mfrow=c(2,2))

par(mar=c(4,4,2,2))

#batsmanAvgRunsOpposition("./player1.csv","Player1")

#batsmanAvgRunsOpposition("./player2.csv","Player2")

#batsmanAvgRunsOpposition("./player3.csv","Ponting")

#batsmanAvgRunsOpposition("./player4.csv","Player4")

dev.off()

## null device

## 1

**10. Runs Likelihood of batsman**

par(mfrow=c(2,2))

par(mar=c(4,4,2,2))

#batsmanRunsLikelihood("./player1.csv","Player1")

#batsmanRunsLikelihood("./player2.csv","Player2")

#batsmanRunsLikelihood("./player3.csv","Ponting")

#batsmanRunsLikelihood("./player4.csv","Player4")

dev.off()

## null device

## 1

**11. Moving Average of runs in career**

par(mfrow=c(2,2))

par(mar=c(4,4,2,2))

#batsmanMovingAverage("./player1.csv","Player1")

#batsmanMovingAverage("./player2.csv","Player2")

#batsmanMovingAverage("./player3.csv","Ponting")

#batsmanMovingAverage("./player4.csv","Player4")

dev.off()

## null device

## 1

**12. Cumulative Average runs of batsman in career**

par(mfrow=c(2,2))

par(mar=c(4,4,2,2))

#batsmanCumulativeAverageRuns("./player1.csv","Player1")

#batsmanCumulativeAverageRuns("./player2.csv","Player2")

#batsmanCumulativeAverageRuns("./player3.csv","Ponting")

#batsmanCumulativeAverageRuns("./player4.csv","Player4")

dev.off()

## null device

## 1

**13. Cumulative Average strike rate of batsman in career**

par(mfrow=c(2,2))

par(mar=c(4,4,2,2))

#batsmanCumulativeStrikeRate("./player1.csv","Player1")

#batsmanCumulativeStrikeRate("./player2.csv","Player2")

#batsmanCumulativeStrikeRate("./player3.csv","Ponting")

#batsmanCumulativeStrikeRate("./player4.csv","Player4")

dev.off()

## null device

## 1

**14. Future Runs forecast**

Here are plots that forecast how the batsman will perform in future. In this case 90% of the career runs trend is uses as the training set. the remaining 10% is the test set.

A Holt-Winters forecating model is used to forecast future performance based on the 90% training set. The forecated runs trend is plotted. The test set is also plotted to see how close the forecast and the actual matches

Take a look at the runs forecasted for the batsman below.

par(mfrow=c(2,2))

par(mar=c(4,4,2,2))

#batsmanPerfForecast("./player1.csv","Player1")

#batsmanPerfForecast("./player2.csv","Player2")

#batsmanPerfForecast("./player3.csv","Player3")

#batsmanPerfForecast("./player4.csv","Player4")

dev.off()

## null device

## 1

**15. Relative Mean Strike Rate plot**

The plot below compares the Mean Strike Rate of the batsman for each of the runs ranges of 10 and plots them. The plot indicate the following

frames <- list("./player1.csv","./player2.csv","player3.csv","player4.csv")

names <- list("Player1","Player2","Player3","Player4")

#relativeBatsmanSR(frames,names)

**16. Relative Runs Frequency plot**

The plot below gives the relative Runs Frequency Percetages for each 10 run bucket. The plot below show

frames <- list("./player1.csv","./player2.csv","player3.csv","player4.csv")

names <- list("Player1","Player2","Player3","Player4")

#relativeRunsFreqPerf(frames,names)

**17. Relative cumulative average runs in career**

frames <- list("./player1.csv","./player2.csv","player3.csv","player4.csv")

names <- list("Player1","Player2","Player3","Player4")

#relativeBatsmanCumulativeAvgRuns(frames,names)

**18. Relative cumulative average strike rate in career**

frames <- list("./player1.csv","./player2.csv","player3.csv","player4.csv")

names <- list("Player1","Player2","Player3","player4")

#relativeBatsmanCumulativeStrikeRate(frames,names)

**19. Check Batsman In-Form or Out-of-Form**

The below computation uses Null Hypothesis testing and p-value to determine if the batsman is in-form or out-of-form. For this 90% of the career runs is chosen as the population and the mean computed. The last 10% is chosen to be the sample set and the sample Mean and the sample Standard Deviation are caculated.

The Null Hypothesis (H0) assumes that the batsman continues to stay in-form where the sample mean is within 95% confidence interval of population mean The Alternative (Ha) assumes that the batsman is out of form the sample mean is beyond the 95% confidence interval of the population mean.

A significance value of 0.05 is chosen and p-value us computed If p-value >= .05 – Batsman In-Form If p-value < 0.05 – Batsman Out-of-Form

**Note** Ideally the p-value should be done for a population that follows the Normal Distribution. But the runs population is usually left skewed. So some correction may be needed. I will revisit this later

This is done for the Top 4 batsman

#checkBatsmanInForm("./player1.csv","Player1")

#checkBatsmanInForm("./player2.csv","Player2")

#checkBatsmanInForm("./player3.csv","Player3")

#checkBatsmanInForm("./player4.csv","Player4")

**20. 3D plot of Runs vs Balls Faced and Minutes at Crease**

The plot is a scatter plot of Runs vs Balls faced and Minutes at Crease. A prediction plane is fitted

par(mfrow=c(1,2))

par(mar=c(4,4,2,2))

#battingPerf3d("./player1.csv","Player1")

#battingPerf3d("./player2.csv","Player2")

par(mfrow=c(1,2))

par(mar=c(4,4,2,2))

#battingPerf3d("./player3.csv","Player3")

#battingPerf3d("./player4.csv","player4")

dev.off()

## null device

## 1

**21. Predicting Runs given Balls Faced and Minutes at Crease**

A multi-variate regression plane is fitted between Runs and Balls faced +Minutes at crease.

BF <- seq( 10, 400,length=15)

Mins <- seq(30,600,length=15)

newDF <- data.frame(BF,Mins)

#Player1 <- batsmanRunsPredict("./player1.csv","Player1",newdataframe=newDF)

#Player2 <- batsmanRunsPredict("./player2.csv","Player2",newdataframe=newDF)

#ponting <- batsmanRunsPredict("./player3.csv","Player3",newdataframe=newDF)

#sangakkara <- batsmanRunsPredict("./player4.csv","Player4",newdataframe=newDF)

#batsmen <-cbind(round(Player1$Runs),round(Player2$Runs),round(Player3$Runs),round(Player4$Runs))

#colnames(batsmen) <- c("Player1","Player2","Player3","Player4")

#newDF <- data.frame(round(newDF$BF),round(newDF$Mins))

#colnames(newDF) <- c("BallsFaced","MinsAtCrease")

#predictedRuns <- cbind(newDF,batsmen)

#predictedRuns

**Analysis of bowlers**

1. Bowler1
2. Bowler2
3. Bowler3
4. Bowler4

player1 <- getPlayerData(xxxx,dir=“..”,file=“player1.csv”,type=“bowling”) *Note* For One day you will have to use getPlayerDataOD() and for Twenty20 it is getPlayerDataTT()

**21. Wicket Frequency Plot**

This plot below computes the percentage frequency of number of wickets taken for e.g 1 wicket x%, 2 wickets y% etc and plots them as a continuous line

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#bowlerWktsFreqPercent("./bowler1.csv","Bowler1")

#bowlerWktsFreqPercent("./bowler2.csv","Bowler2")

#bowlerWktsFreqPercent("./bowler3.csv","Bowler3")

dev.off()

## null device

## 1

**22. Wickets Runs plot**

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#bowlerWktsRunsPlot("./bowler1.csv","Bowler1")

#bowlerWktsRunsPlot("./bowler2.csv","Bowler2")

#bowlerWktsRunsPlot("./bowler3.csv","Bowler3")

dev.off()

## null device

## 1

**23. Average wickets at different venues**

#bowlerAvgWktsGround("./bowler3.csv","Bowler3")

**24. Average wickets against different opposition**

#bowlerAvgWktsOpposition("./bowler3.csv","Bowler3")

**25. Wickets taken moving average**

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#bowlerMovingAverage("./bowler1.csv","Bowler1")

#bowlerMovingAverage("./bowler2.csv","Bowler2")

#bowlerMovingAverage("./bowler3.csv","Bowler3")

dev.off()

## null device

## 1

**26. Cumulative Wickets taken**

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#bowlerCumulativeAvgWickets("./bowler1.csv","Bowler1")

#bowlerCumulativeAvgWickets("./bowler2.csv","Bowler2")

#bowlerCumulativeAvgWickets("./bowler3.csv","Bowler3")

dev.off()

## null device

## 1

**27. Cumulative Economy rate**

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#bowlerCumulativeAvgEconRate("./bowler1.csv","Bowler1")

#bowlerCumulativeAvgEconRate("./bowler2.csv","Bowler2")

#bowlerCumulativeAvgEconRate("./bowler3.csv","Bowler3")

dev.off()

## null device

## 1

**28. Future Wickets forecast**

Here are plots that forecast how the bowler will perform in future. In this case 90% of the career wickets trend is used as the training set. the remaining 10% is the test set.

A Holt-Winters forecating model is used to forecast future performance based on the 90% training set. The forecated wickets trend is plotted. The test set is also plotted to see how close the forecast and the actual matches

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#bowlerPerfForecast("./bowler1.csv","Bowler1")

#bowlerPerfForecast("./bowler2.csv","Bowler2")

#bowlerPerfForecast("./bowler3.csv","Bowler3")

dev.off()

## null device

## 1

**29. Contribution to matches won and lost**

As discussed above the next 2 charts require the use of getPlayerDataSp(). This can only be done for Test matches

#bowler1sp <- getPlayerDataSp(xxxx,tdir=".",tfile="bowler1sp.csv",ttype="bowling")

#bowler2sp <- getPlayerDataSp(xxxx,tdir=".",tfile="bowler2sp.csv",ttype="bowling")

#bowler3sp <- getPlayerDataSp(xxxx,tdir=".",tfile="bowler3sp.csv",ttype="bowling")

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#bowlerContributionWonLost("bowler1sp","Bowler1")

#bowlerContributionWonLost("bowler2sp","Bowler2")

#bowlerContributionWonLost("bowler3sp","Bowler3")

dev.off()

## null device

## 1

**30. Performance home and overseas.**

This can only be done for Test matches

par(mfrow=c(1,3))

par(mar=c(4,4,2,2))

#bowlerPerfHomeAway("bowler1sp","Bowler1")

#bowlerPerfHomeAway("bowler2sp","Bowler2")

#bowlerPerfHomeAway("bowler3sp","Bowler3")

dev.off()

## null device

## 1

**31 Relative Wickets Frequency Percentage**

frames <- list("./bowler1.csv","./bowler3.csv","bowler2.csv")

names <- list("Bowler1","Bowler3","Bowler2")

#relativeBowlingPerf(frames,names)

**32 Relative Economy Rate against wickets taken**

frames <- list("./bowler1.csv","./bowler3.csv","bowler2.csv")

names <- list("Bowler1","Bowler3","Bowler2")

#relativeBowlingER(frames,names)

**33 Relative cumulative average wickets of bowlers in career**

frames <- list("./bowler1.csv","./bowler3.csv","bowler2.csv")

names <- list("Bowler1","Bowler3","Bowler2")

#relativeBowlerCumulativeAvgWickets(frames,names)

**34 Relative cumulative average economy rate of bowlers**

frames <- list("./bowler1.csv","./bowler3.csv","bowler2.csv")

names <- list("Bowler1","Bowler3","Bowler2")

#relativeBowlerCumulativeAvgEconRate(frames,names)

**35 Check for bowler in-form/out-of-form**

The below computation uses Null Hypothesis testing and p-value to determine if the bowler is in-form or out-of-form. For this 90% of the career wickets is chosen as the population and the mean computed. The last 10% is chosen to be the sample set and the sample Mean and the sample Standard Deviation are caculated.

The Null Hypothesis (H0) assumes that the bowler continues to stay in-form where the sample mean is within 95% confidence interval of population mean The Alternative (Ha) assumes that the bowler is out of form the sample mean is beyond the 95% confidence interval of the population mean.

A significance value of 0.05 is chosen and p-value us computed If p-value >= .05 – Batsman In-Form If p-value < 0.05 – Batsman Out-of-Form

**Note** Ideally the p-value should be done for a population that follows the Normal Distribution. But the runs population is usually left skewed. So some correction may be needed. I will revisit this later

**Note:** The check for the form status of the bowlers indicate

#checkBowlerInForm("./bowler1.csv","Bowler1")

#checkBowlerInForm("./bowler2.csv","Bowler2")

#checkBowlerInForm("./bowler3.csv","Bowler3")

dev.off()

## null device

## 1