2018

FIFA Player Value Estimator

MATH2269 - APPLIED BAYESIAN STATISTICS

ASSIGNMENT 3

SIMARPREET LUTHRA - S3706588 | AAKASH MAYUR KUMAR SHAH - S3636808

FIFA player Value estimator

Introduction

The game of soccer is a very popular game across the world. FIFA which stands for “International Federation of Association Football” is the major governing body for the sport across the world. Over 211 countries have an affiliation with FIFA which has earned it the nickname “the United Nations of Football”. Soccer has become such a popular sport that the video gaming franchise connected to it, “FIFA” is also a major success across the world. The game has been developed by Electronic Arts studios, through their subsidiary EA Sports. The firs version of the game was released in 1997. Since then the game has gone through over 20 iterations and has remained a crowd favorite.

The soccer video game has as much if not more fan following as the actual sport. The current version the game being “FIFA 2018” has been entailed to have the most realistic graphics and game play. The game allows players to create their own team, by buying players from auctions , and managing every aspect of the team. It also allows the players to compete against other players across the world online.

Aim

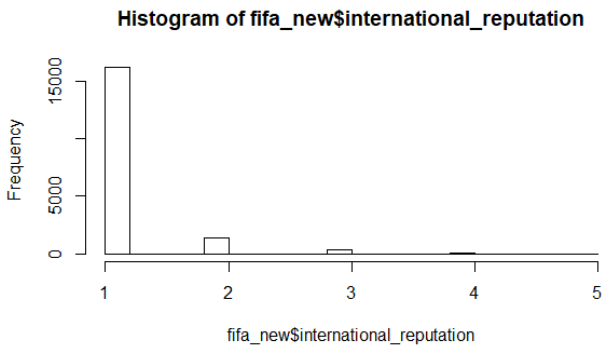
The aim of the project is to evaluate the net worth of the FIFA players on the basis of various factors such as their age, monthly wages, overall rating of their skills and international reputation.

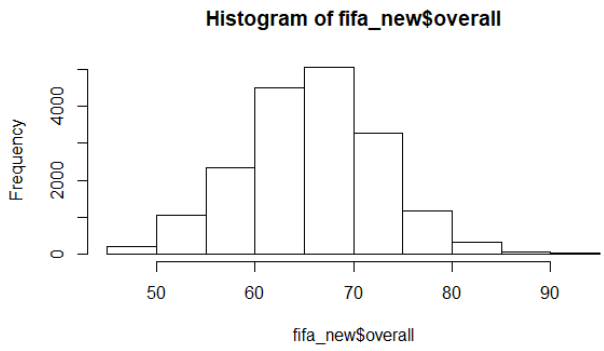
Data description

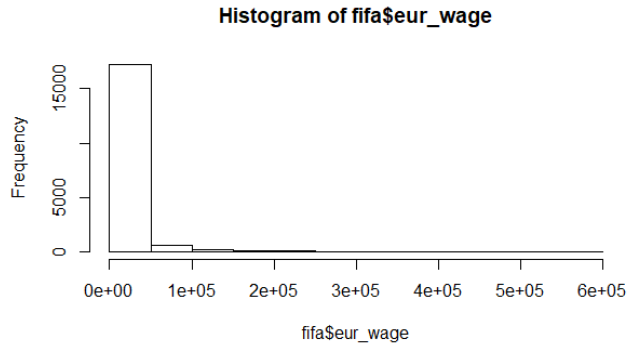
Each row in the dataset corresponds to a different FIFA player. The dataset explains the various statistics and attributes of the players such as the name of the player, the international reputation of the player, the player’s age, the overall skill rating of the player, the monthly earnings of the player in Euros, the net worth of the player, preferred foot, followed by the individual statistics such as strength, attack and so on. The data is open source and available on Kaggle, the world's largest community of data scientists and machine learners. Link to the data: https://www.kaggle.com/kevinmh/fifa-18-more-complete-player-dataset

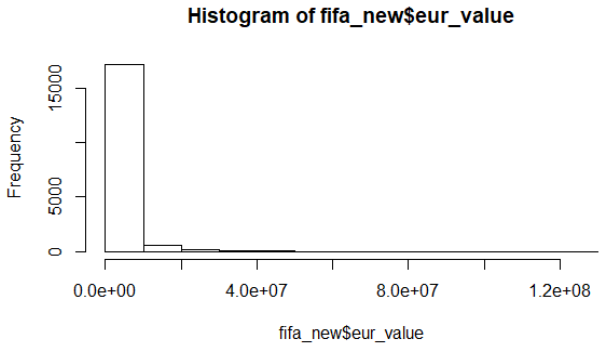
Preliminary analysis

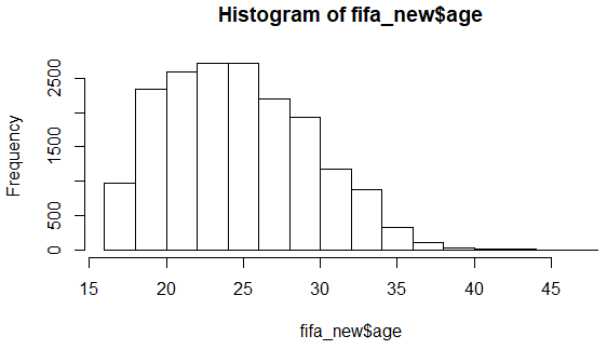
Scatter plots and histograms were plotted to investigate the relationship between the outcome variable and each of the predictors individually and the distribution of variables in the data. This revealed that the outcome variable, player wages and international reputation are highly left skewed. On the other hand, age is right skewed and overall rating of players is normally distributed.



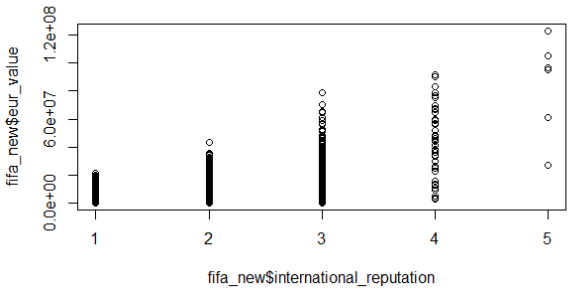


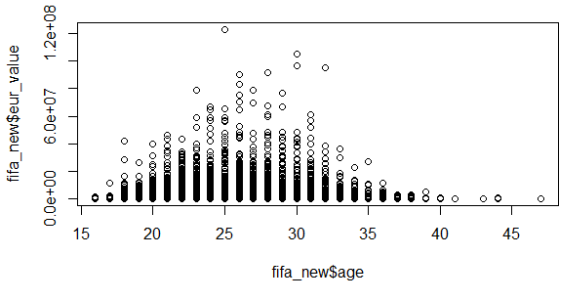


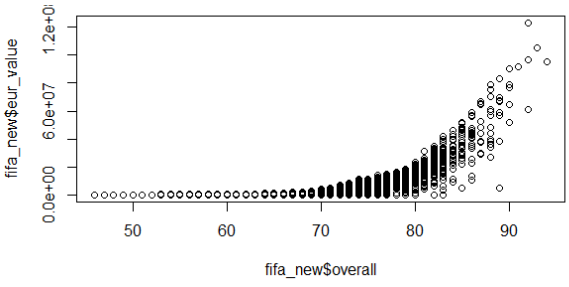




The worth of a player is linearly related to their wages and international reputation, that is, as wages/ international reputation increases, the player's worth increases proportionately. Furthermore, the age has a quadratic relationship with player's worth. Even though overall rating has an exponential relationship with player's worth, it can be approximated to have a linear relationship with a negative intercept.

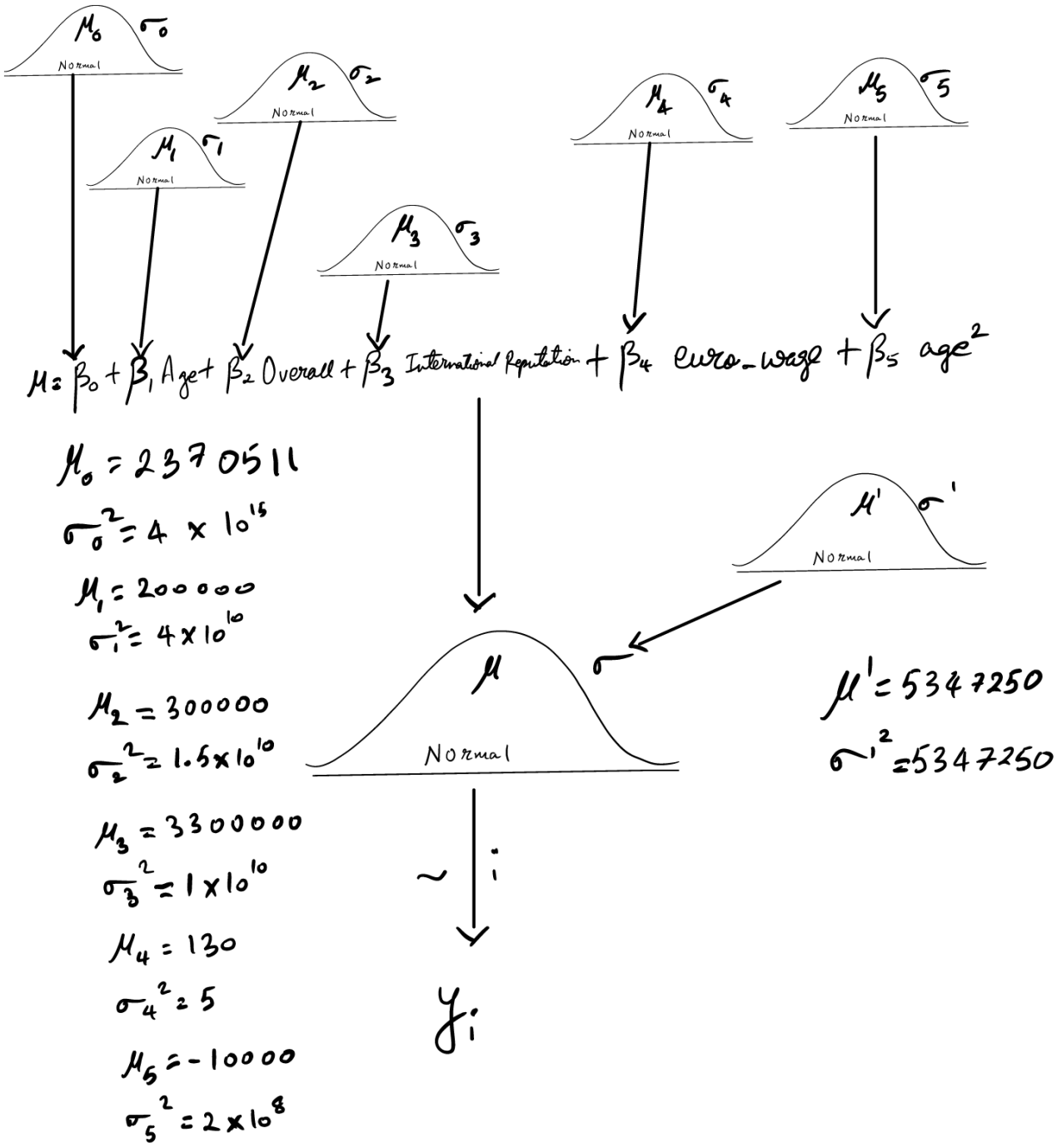






Model diagram

The model fitted on the player value is a regression model with age as a quadratic regressor, while the other predictors, which are overall rating, international reputation and monthly wages as linear terms. The dependent variable is assumed to be a normal prior distribution with high variance, which makes it non-informative, while the model is fit on mean values of the prior distribution of the dependant variable. Further, the intercept and each regression coefficient is assumed to be normally distributed.



It has been seen that the performance of the players is correlated to their age and thus there is an impact on the net worth as such (Dendir, 2016). Thus based on this expert information the authors have chosen to put weak confidence on it, since there is proof that the age only impact performance not net worth directly. It has been found that the bigger the reputation of the player the more their worth, since they have bigger brand image and presence (Composite, 2018). In accordance to that the authors have strong confidence on it. Current research has shown that the greater the ranking of the player the higher has been the demand for the player and hence the price of the player goes up, thus increasing the net worth, keeping that in mind the authors have medium confidence in it (Dawson, 2018). It has been shown that the wage of the players depends on the skills and performance of the players, thus is already modelled and has found to be impacting on the net worth, since the higher the skills, the fitness, higher the rating and as such higher wage (Yaldo & Shamir, 2017)

The variable Wage has very Strong expert information, Reputation has Strong expert information, Overall Rating has good expert information, Age and Age squared have Weak expert information.

Code

```
source("DBDA2E-utilities.R")

#===============================================================================

genMCMC = function( data , xName="x" , yName="y" ,
                    numSavedSteps=10000 , thinSteps=1 , saveName=NULL  ,
                    runjagsMethod=runjagsMethodDefault ,
                    nChains=nChainsDefault , xPred = xPred) {
 require(runjags)
 #-------------------------------------------------------------------------
 # THE DATA.
 y = data[,yName]
 x = as.matrix(data[,xName],ncol=length(xName))
 # Do some checking that data make sense:
 if ( any( !is.finite(y) ) ) { stop("All y values must be finite.") }
 if ( any( !is.finite(x) ) ) { stop("All x values must be finite.") }
 cat("\nCORRELATION MATRIX OF PREDICTORS:\n ")
 show( round(cor(x),3) )
 cat("\n")
 flush.console()
 # Specify the data in a list, for later shipment to JAGS:
```

```
 dataList = list(
  x = x ,
  y = y ,
  Nx = dim(x)[2] ,
  Ntotal = dim(x)[1] ,
  xPred = xPred # Data for predictions
 )
 #------------------------------------------------------------------------
 # THE MODEL.
 modelString = "
 # Standardize the data:
 data {
  ym <- mean(y)
  ysd <- sd(y)
  for ( i in 1:Ntotal ) {
   zy[i] <- ( y[i] - ym ) / ysd
  }
  for ( j in 1:Nx ) {
   xm[j]  <- mean(x[,j])
   xsd[j] <-   sd(x[,j])
   for ( i in 1:Ntotal ) {
    zx[i,j] <- ( x[i,j] - xm[j] ) / xsd[j]
   }
  }
  # Specify the priors for original beta parameters
  # Prior locations to reflect the expert information
  mu0 <- -20000000 # Set to overall mean a priori based on the interpretation of constant term in
regression
  mu[1] <- 130 # Wage
```

```
  mu[2] <- 3300000 # Reputation
  mu[3] <- 200000 # Overall Rating
  mu[4] <- 150000 # Age
  mu5 <- -10000 # (Age) ^2

# Prior variances to reflect the expert information
  Var0 <- 40000000000000000000 # High variance to accomodate variability all variables
  Var[1] <- 5 # Wage - Very Strong expert information
  Var[2] <- 1000000000 # Reputation - Strong expert information
  Var[3] <- 35000000000 # Overall Rating - Good expert information
  Var[4] <- 80000000000 # Age - Weak expert information
  Var5 <- 500000000 # Age^2 - Weak expert information

  # Compute corresponding prior means and variances for the standardised parameters
  muZ[1:Nx] <-  mu[1:Nx] * xsd[1:Nx] / ysd
  muZ5 <- mu5*(xsd[1])^2/ysd
  muZ0 <- (mu0 + mu5*(xm[1])^2/(xsd[1])^2 + sum( mu[1:Nx] * xm[1:Nx] / xsd[1:Nx] )*ysd - ym) / ysd

  # Compute corresponding prior variances and variances for the standardised parameters
  VarZ[1:Nx] <- Var[1:Nx] * ( xsd[1:Nx]/ ysd )^2
  VarZ0 <- Var0 / (ysd^2)
  VarZ5 <- Var5*((xsd[1])^2/ysd)^2
}
# Specify the model for standardized data:
model {
  for ( i in 1:Ntotal ) {
    zy[i] ~ dnorm( zbeta0 + sum( zbeta[1:Nx] * zx[i,1:Nx] ) + zbeta5*(zx[i,1])^2 , 1/zsigma^2 )
  }
```

```
  # Priors vague on standardized scale:
  zbeta0 ~ dnorm( muZ0 , 1/VarZ0 )
  zbeta5 ~ dnorm( muZ5 , 1/VarZ5 )
  for ( j in 1:Nx ) {
    zbeta[j] ~ dnorm( muZ[j] , 1/VarZ[j] )
  }
  zsigma ~ dnorm(1,50)

  # Transform to original scale:
  beta[1:Nx] <- ( zbeta[1:Nx] / xsd[1:Nx] )*ysd
  beta0 <- zbeta0*ysd  + ym - sum( zbeta[1:Nx] * xm[1:Nx] / xsd[1:Nx] )*ysd
  beta5 <- (zbeta5/(xsd[1])^2)*ysd
  sigma <- zsigma*ysd

  # Compute predictions at every step of the MCMC
pred <- beta0 + beta[1] * xPred[1] + beta[2] * xPred[2] + beta[3] * xPred[3] + beta[4] * xPred[4] +
beta5*(xPred[1])^2
 }" # close quote for modelString
 # Write out modelString to a text file
 writeLines( modelString , con="TEMPmodel.txt" )
 #------------------------------------------------------------------------------
 # RUN THE CHAINS
 parameters = c( "beta0" ,  "beta" ,"beta5",  "sigma",
                 "zbeta0" , "zbeta" ,"zbeta5", "zsigma",  "pred" )
 adaptSteps = 1000  # Number of steps to "tune" the samplers
 burnInSteps = 2000
 runJagsOut <- run.jags( method=runjagsMethod , model="TEMPmodel.txt" ,
                         monitor=parameters , data=dataList , n.chains=nChains ,
                        adapt=adaptSteps , burnin=burnInSteps , sample=ceiling(numSavedSteps/nChains) ,
```

```
                    thin=thinSteps , summarise=FALSE , plots=FALSE )
  codaSamples = as.mcmc.list( runJagsOut )
  if ( !is.null(saveName) ) {
    save( codaSamples , file=paste(saveName,"Mcmc.Rdata",sep="") )
  }
  return( codaSamples )
} # end function
#===============================================================================
smryMCMC = function(  codaSamples ,
                      saveName=NULL) {
  summaryInfo = NULL
  mcmcMat = as.matrix(codaSamples,chains=TRUE)
  paramName = colnames(mcmcMat)
  for ( pName in paramName ) {
    summaryInfo = rbind( summaryInfo , summarizePost( mcmcMat[,pName] ) )
  }
  rownames(summaryInfo) = paramName

  if ( !is.null(saveName) ) {
    write.csv( summaryInfo , file=paste(saveName,"SummaryInfo.csv",sep="") )
  }
    return( summaryInfo)
}
#===============================================================================
plotMCMC = function( codaSamples , data , xName="x" , yName="y" ,
                     showCurve=FALSE ,  pairsPlot=FALSE ,
                     saveName=NULL , saveType="jpg" ) {
  # showCurve is TRUE or FALSE and indicates whether the posterior should
  #   be displayed as a histogram (by default) or by an approximate curve.
```

```
# pairsPlot is TRUE or FALSE and indicates whether scatterplots of pairs
#   of parameters should be displayed.
#-----------------------------------------------------------------------------
y = data[,yName]
x = as.matrix(data[,xName])
mcmcMat = as.matrix(codaSamples,chains=TRUE)
chainLength = NROW( mcmcMat )
zbeta0 = mcmcMat[,"zbeta0"]
zbeta  = mcmcMat[,grep("^zbeta$|^zbeta\\[",colnames(mcmcMat))]
zbeta5 = mcmcMat[,"zbeta5"]
if ( ncol(x)==1 ) { zbeta = matrix( zbeta , ncol=1 ) }
zsigma = mcmcMat[,"zsigma"]
beta0 = mcmcMat[,"beta0"]
beta  = mcmcMat[,grep("^beta$|^beta\\[",colnames(mcmcMat))]
beta5 = mcmcMat[,"beta5"]
if ( ncol(x)==1 ) { beta = matrix( beta , ncol=1 ) }
sigma = mcmcMat[,"sigma"]
pred = mcmcMat[,"pred"]
#-----------------------------------------------------------------------------
# Compute R^2 for credible parameters:
YcorX = cor( y , x ) # correlation of y with each x predictor
Rsq = zbeta %*% matrix( YcorX , ncol=1 )
#-----------------------------------------------------------------------------
if ( pairsPlot ) {
  # Plot the parameters pairwise, to see correlations:
  openGraph()
  nPtToPlot = 1000
  plotIdx = floor(seq(1,chainLength,by=chainLength/nPtToPlot))
  panel.cor = function(x, y, digits=2, prefix="", cex.cor, ...) {
```

```
  usr = par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r = (cor(x, y))
  txt = format(c(r, 0.123456789), digits=digits)[1]
  txt = paste(prefix, txt, sep="")
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex=1.25 ) # was cex=cex.cor*r
 }
 pairs( cbind( beta0 , beta, beta5 , sigma )[plotIdx,] ,
       labels=c( "beta[0]" ,
             paste0("beta[",1:ncol(beta),"]\n",xName),"beta[5]" ,
             expression(sigma) ) ,
       lower.panel=panel.cor , col="skyblue" )
 if ( !is.null(saveName) ) {
  saveGraph( file=paste(saveName,"PostPairs",sep=""), type=saveType)
 }
}
#-------------------------------------------------------------------------
# Marginal histograms:
 decideOpenGraph = function( panelCount , saveName , finished=FALSE ,
                     nRow=2 , nCol=3 ) {
 # If finishing a set:
 if ( finished==TRUE ) {
  if ( !is.null(saveName) ) {
   saveGraph( file=paste0(saveName,ceiling((panelCount-1)/(nRow*nCol))),
           type=saveType)
  }
  panelCount = 1 # re-set panelCount
  return(panelCount)
```

```
  } else {
  # If this is first panel of a graph:
  if ( ( panelCount %% (nRow*nCol) ) == 1 ) {
    # If previous graph was open, save previous one:
    if ( panelCount>1 & !is.null(saveName) ) {
      saveGraph( file=paste0(saveName,(panelCount%/%(nRow*nCol))),
                 type=saveType)
    }
    # Open new graph
    openGraph(width=nCol*7.0/3,height=nRow*2.0)
    layout( matrix( 1:(nRow*nCol) , nrow=nRow, byrow=TRUE ) )
    par( mar=c(4,4,2.5,0.5) , mgp=c(2.5,0.7,0) )
  }
  # Increment and return panel count:
  panelCount = panelCount+2
  return(panelCount)
  }
}
  # Original scale:
panelCount = 1
panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
histInfo = plotPost( beta0 , cex.lab = 1.75 , showCurve=showCurve ,
                     xlab=bquote(beta[0]) , main="Intercept" )
for ( bIdx in 1:ncol(beta) ) {
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
  histInfo = plotPost( beta[,bIdx] , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(beta[.(bIdx)]) , main=xName[bIdx] )
}
panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
```

```
  histInfo = plotPost( beta5 , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(beta[5]) , main="age^2" )
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
  histInfo = plotPost( sigma , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(sigma) , main=paste("Scale") )
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
  histInfo = plotPost( Rsq , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(R^2) , main=paste("Prop Var Accntd") )
  panelCount = decideOpenGraph( panelCount , finished=TRUE ,
saveName=paste0(saveName,"PostMarg") )
  histInfo = plotPost( pred , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(pred) , main="Prediction" )
  # Standardized scale:
  panelCount = 1
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMargZ") )
  histInfo = plotPost( zbeta0 , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(z*beta[0]) , main="Intercept" )
  for ( bIdx in 1:ncol(beta) ) {
    panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMargZ") )
    histInfo = plotPost( zbeta[,bIdx] , cex.lab = 1.75 , showCurve=showCurve ,
                         xlab=bquote(z*beta[.(bIdx)]) , main=xName[bIdx] )
  }
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMargZ") )
  histInfo = plotPost( zbeta5 , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(z*beta[5]) , main="age^2" )
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMargZ") )
  histInfo = plotPost( zsigma , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(z*sigma) , main=paste("Scale") )
 panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMargZ") )
```

```
  histInfo = plotPost( Rsq , cex.lab = 1.75 , showCurve=showCurve ,
                   xlab=bquote(R^2) , main=paste("Prop Var Accntd") )
  panelCount = decideOpenGraph( panelCount , finished=TRUE ,
saveName=paste0(saveName,"PostMargZ") )
    #-------------------------------------------------------------------------
}
#===============================================================================
graphics.off()
setwd("D:/study/RMIT/1-2/Applied Bayesian Statistics/Final project")
myData = read.csv("fifa.csv")
yName = "eur_value" ; xName = c("age","eur_wage","overall","international_reputation")
fileNameRoot = "fifa"
numSavedSteps = 10000 ; thinSteps=3
graphFileType = "eps"
#-------------------------------------------------------------------------
# Load the relevant model into R's working memory:
source("Final Project.R")
#-------------------------------------------------------------------------
# Generate the MCMC chain:
#startTime = proc.time()
xPred = c(32 ,     565000 ,            94 ,        5)
mcmcCoda = genMCMC( data=myData , xName=xName , yName=yName ,
                  numSavedSteps=numSavedSteps , thinSteps=thinSteps ,
                  saveName=fileNameRoot , xPred = xPred )
#stopTime = proc.time()
#duration = stopTime - startTime
#show(duration)
#-------------------------------------------------------------------------
# Display diagnostics of chain, for specified parameters:
```

```
parameterNames = varnames(mcmcCoda) # get all parameter names
for ( parName in parameterNames ) {
  diagMCMC( codaObject=mcmcCoda , parName=parName ,
          saveName=fileNameRoot , saveType="png" )
}
#-------------------------------------------------------------------------------------------------------------------------
# Get summary statistics of chain:
summaryInfo = smryMCMC( mcmcCoda ,
                    saveName=fileNameRoot  )
show(summaryInfo)
# Display posterior information:
plotMCMC( mcmcCoda , data=myData , xName=xName , yName=yName ,
        pairsPlot=TRUE , showCurve=FALSE ,
        saveName=fileNameRoot , saveType="png" )
#-------------------------------------------------------------------------------------------------------------------------
```

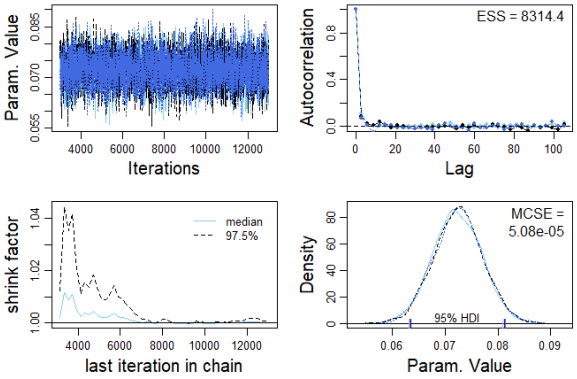
Diagnostics

Diagnostics refer to the plots of MCMC chains for each parameter to access whether the MCMC chains provide a good enough representation of the target distribution. The diagnostics accessed are:

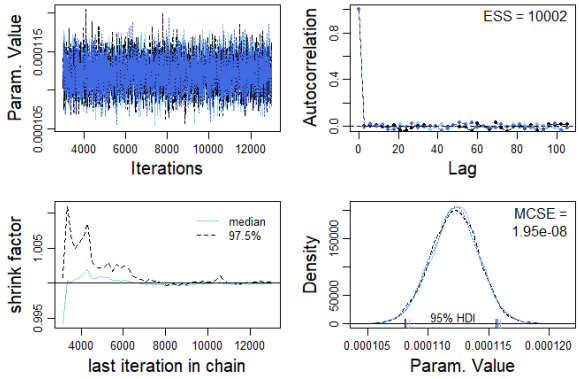
- Well mixing chains
- Autocorrelation between the chains
- Effective sample size generated by the chains
- Shrink factor
- Overlapping posteriors
- Monte Carlo Standard Error

Since, the values of all the parameters were standardised before fitting the model, we observe that the diagnostics for all the standardised values are perfect. The chains are well mixing. There is little or no autocorrelation between the chains. The effective sample size is fairly huge. Shrink factor is less than 1.1 threshold. The posteriors are overlapping and MCSE is very close to zero. The actual parameter values are also very well diagnosed, the chains are well mixing. There is little or no autocorrelation between the chains. The effective sample size is fairly huge. Shrink factor is less than 1.1 threshold. The posteriors are overlapping but MCSE is a little bit high, but well within the expected range. The diagnostics of the predicted value are also all perfect except the huge MCSE as it accounts for the total MCSE of all the predictor parameters in the model.

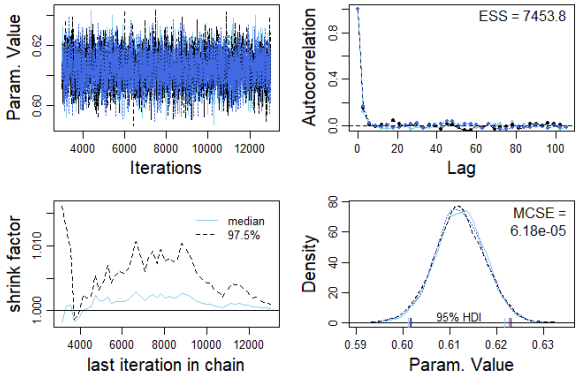
zbeta0



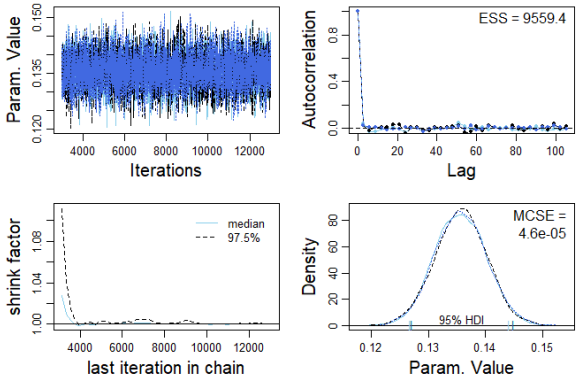
zbeta[1]



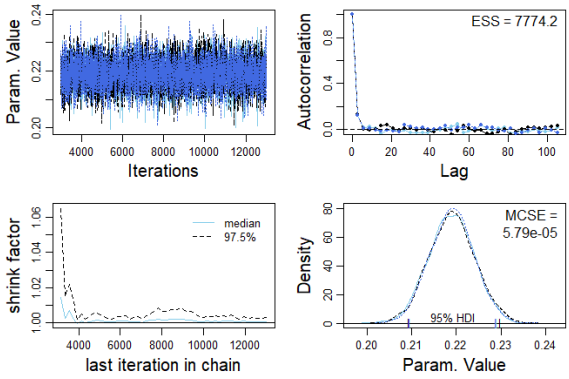
zbeta[2]



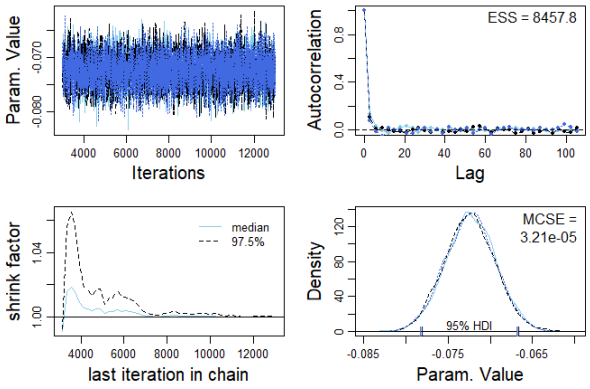
zbeta[3]



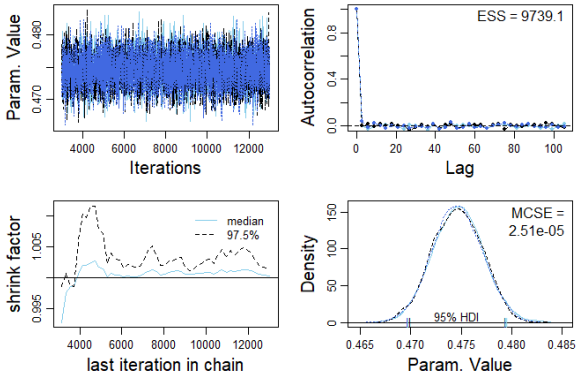
zbeta[4]

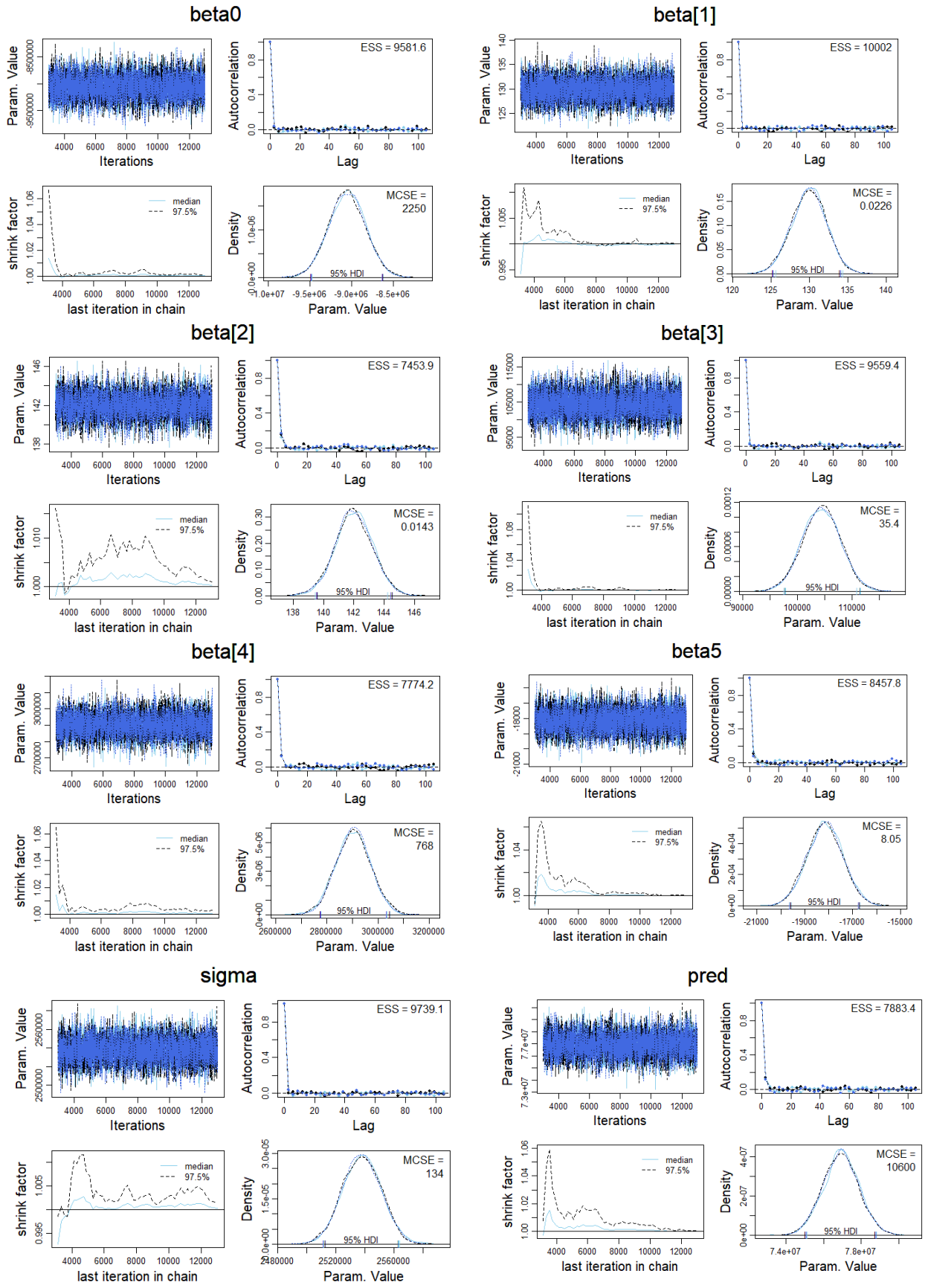


zbeta5



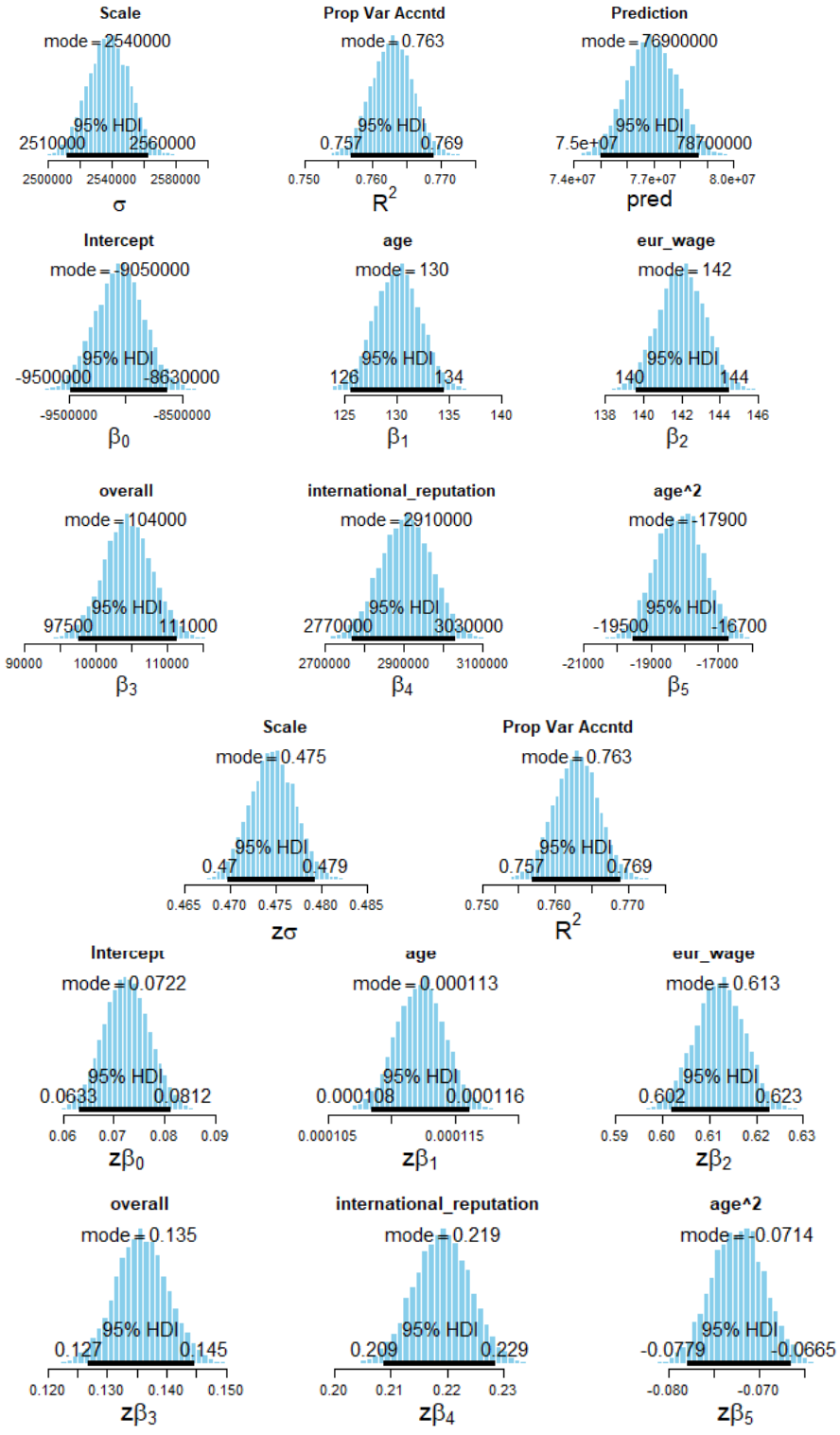
zsigma




beta0
Param. Value
Iterations
Autocorrelation
ESS = 9581.6
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 2250
95% HDI
Param. Value
beta[1]
ESS = 10002
MCSE = 0.0226
beta[2]
ESS = 7453.9
MCSE = 0.0143
beta[3]
ESS = 9559.4
MCSE = 35.4
beta[4]
ESS = 7774.2
MCSE = 768
beta5
ESS = 8457.8
MCSE = 8.05
sigma
ESS = 9739.1
MCSE = 134
pred
ESS = 7883.4
MCSE = 10600

Posterior Plots

The posterior plots are histogram plots for the posterior distribution estimated by the MCMC chains, which give us an idea about the shape of the posterior distribution and the HDI interval for each parameter.



All the posteriors are normally distributed, with mode values given as follows:

Scale = 2540000

Intercept = -9050000

Coefficient of Age = 130

Coefficient of Wages = 142

Coefficient of Overall Rating = 104000

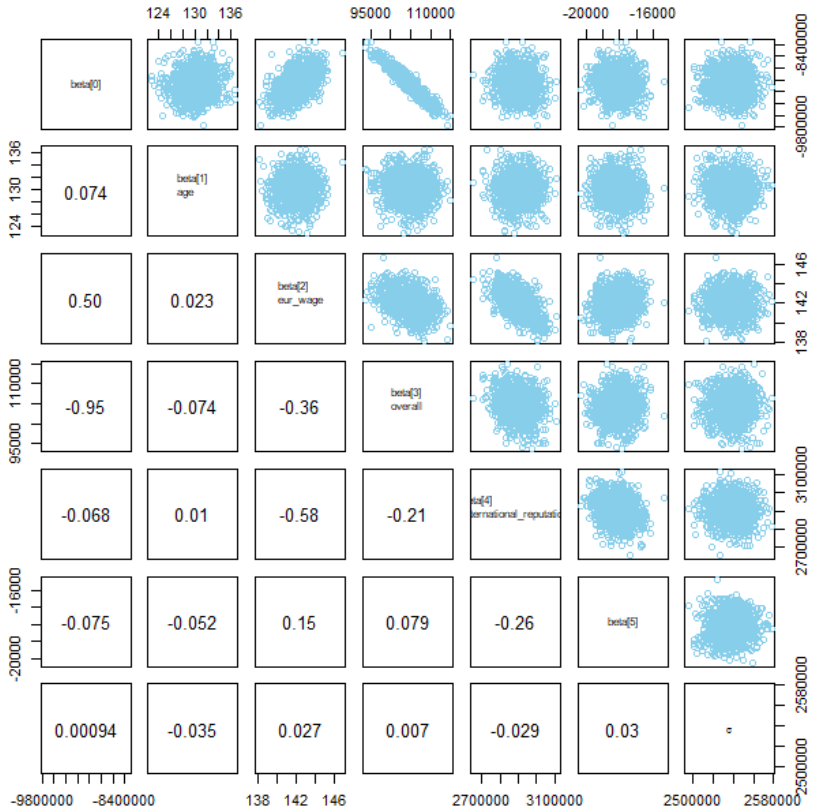
Coefficient of International Reputation = 2910000

Coefficient of Age squared = -17900

The model explains approximately 76% variation in player value and the prediction value for the tested observation is estimated as 7.69 * 10^7, which is close to actual value that is 7.90 * 10^7.

Correlation

We observe from the correlation matrix that none of the parameters are correlated, except coefficient of overall ranking and intercept. This is because of the linear approximation with negative intercept of the non-linear (exponential) trend between overall rating and player value. Thus, the major part of the intercept value comes from the overall rating, which explains the high correlation between the random variables.



Conclusions

We conclude from the analysis that a FIFA player's value can be fairly determined by the linear combination of international reputation, overall rating, monthly wages and quadratic term of age, which explains seventy-six percent variation in player value.

References

Composite, G. (2018). Footballers' net worth: How much do Ronaldo, Messi & the top stars earn? | Goal.com. Retrieved from https://www.goal.com/en-au/news/footballers-net-worth-how-much-do-ronaldo-messi-neymar-earn/qz0262wyw1511sdj60pbq06l7

Dawson, A. (2018). RANKED: The 25 most valuable soccer players in Europe, all worth over €100 million. Retrieved from https://www.businessinsider.com.au/most-valuable-european-soccer-players-ranked-2018-6?r=US&IR=T%20

Dendir, S. (2016). When do soccer players peak? A note. Journal Of Sports Analytics, 2(2), 89-105. doi: 10.3233/jsa-160021

Yaldo, L., & Shamir, L. (2017). Computational Estimation of Football Player Wages. International Journal Of Computer Science In Sport, 16(1), 18-38. doi: 10.1515/ijcss-2017-0002