Assignment 4

Eveline Surbakti - 19200629

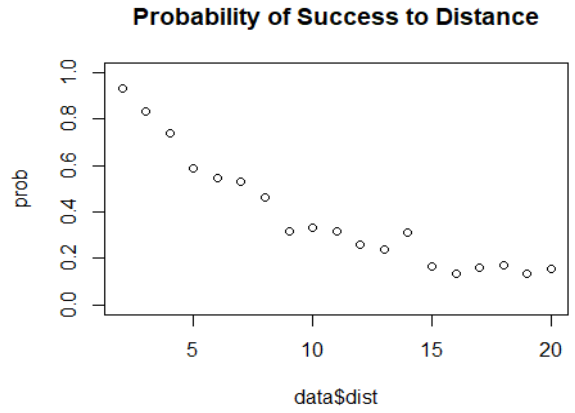
Introduction

Some data are available on the number of successful putts from various distances for professional golfers. In this short report, We will model the number of successful (Nsucc) putts out of a number of attempts (Ntrys) as being Binomially distributed with a probability of a success parameter (p say) that depends on some function of the distance. Since the Binomial probability parameter p must lie between 0 and 1 we should first transform our distance

(a) Plot the ratio of number of successes to number of attempts (y-axis) against each distance (x-axis).

Answer: From the data, we can see that the ratio of number of successes to number of attempts is higher when we have the closer distance and it decreases as the distance increases.

```
data<-read.table("putting.dat", header=TRUE)
prob=(data$Nsucc/data$Ntrys)

#PART A
plot(data$dist,prob,type = ,main = "Probability of Success to Distance", ylim =
c(0,1))
```



(b) Write JAGS code to produce 10,000 samples from the posterior for the linear model above for all unknown quantities. You may set uniform priors for alpha and beta with limits between -10 and 10. What 95% HDR intervals do you get for alpha and beta?

Answer:

```
require(rjags)
## Loading required package: rjags
## Loading required package: coda
## Linked to JAGS 4.3.0
## Loaded modules: basemod,bugs
library(tinytex)
#PART B
#the model
K=nrow(data)
n=data$Ntrys
X=data$dist
Y=data$Nsucc
golf<-list(K=K, n=n,X=X,Y=Y)
```

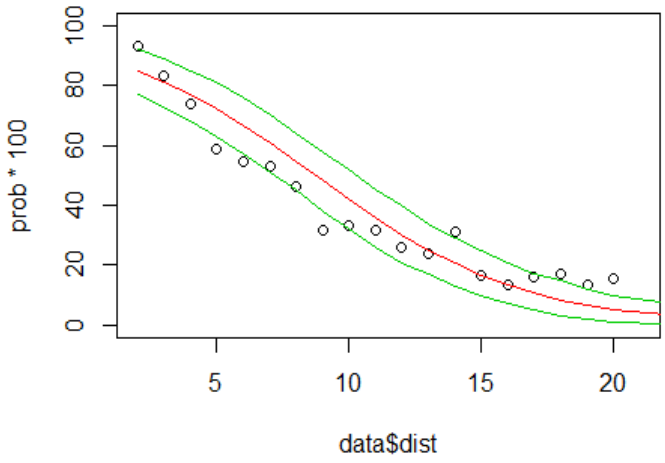
```
#the model
model_string <- "model{
    for (i in 1:K){
            Y[i] ~ dbin(p[i],n[i])
            logit(p[i]) <- alpha + beta*X[i]
            #unknown alpha and beta with - mean to make it neater
    }
    alpha ~ dunif(-10,10)
    beta ~ dunif(-10,10)}"

#the model
putting.model <- jags.model(textConnection(model_string),golf,n.chains=1)
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 2
##    Total graph size: 119
##
## Initializing model
#the sample
putting.samps=coda.samples(putting.model,variable.names=c("alpha","beta"),1e4)

#95% HDR intervals for alpha is
quantile(putting.samps[[1]][,"alpha"],c(0.025,0.975))
##     2.5%    97.5%
## 2.117166 2.345242
#95% HDR intervals for beta is
quantile(putting.samps[[1]][,"beta"],c(0.025,0.975))
##       2.5%      97.5%
## -0.2689787 -0.2430992
```

(c) Add code to produce samples from the posterior predicted number of successes at the observed distances and at 25 feet for 100 attempts at each distance.

Answer:

```
#PART C
OBS20<-list(25,100,NA) #Hint: the easiest way to do this is to add a row to the data
with distance=25, number of tries=100, number of successes=NA.
data_mod<-rbind(data,OBS20)

K=nrow(data_mod)
n=data_mod$Ntrys
X=data_mod$dist
Y=data_mod$Nsucc
golf<-list(K=K, n=n,X=X,Y=Y)

model_string2 <- "model{
    for (i in 1:K){
            Y[i] ~ dbin(p[i],n[i])
            logit(p[i]) <- alpha + beta*X[i]
            PredY[i] ~ dbin(p[i],100)
    }
    alpha ~ dunif(-10,10)
    beta ~ dunif(-10,10)}"
```

```
#the model
putting.model.mod <- jags.model(textConnection(model_string2),
                      golf,n.chains=1)
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 23
##    Total graph size: 146
##
## Initializing model
#the sample
putting.samps.mod=coda.samples(putting.model.mod,variable.names=c("PredY"),1e4)

j<-(colMeans(putting.samps.mod[[1]][,grep("PredY",
colnames(putting.samps.mod[[1]]))]))

interval <- apply(putting.samps.mod[[1]], 2, quantile, c(0.025,0.975))

#Then create a plot that shows the data as per part (a) multiplied by 100, as well as
the mean and HDR number of successes out of 100 at each distance.
plot(data$dist,prob*100, type = , ylim=c(0,100), xlim=c(2, 21)) #extend the limit to
see the whole graph clearly
lines(data_mod$dist,j,type = 'l',col=2)
lines(data_mod$dist, interval[1,], type = 'l',col=3)
lines(data_mod$dist, interval[2,], type = 'l',col=3)
```



(d) What is the 95% HDR for the successes out of 100 at 25 feet? Does this seem reasonable based on your plot of the data?

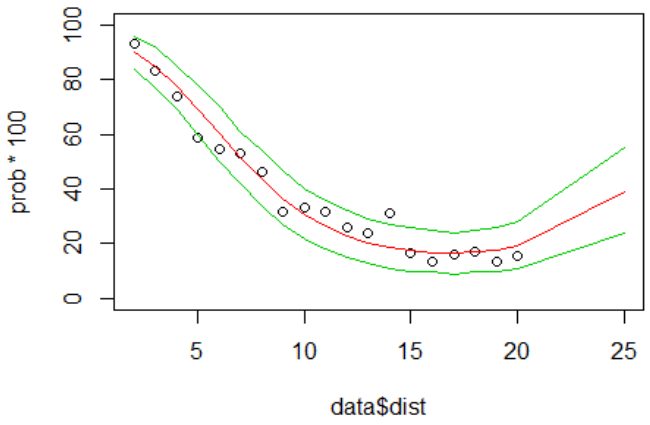
Answer: It does not seem to be reasonable; the plot of the data is not within the interval.

```
#PART D
quantile(putting.samps.mod[[1]][,"PredY[20]"],c(0.025,0.975))
##  2.5% 97.5%
##     0     4
```

(e) Next you will try a model where the probability of a miss depends both linearly and quadratically on distance.

Answer:

```
#PART E
K=nrow(data_mod)
n=data_mod$Ntrys
X=data_mod$dist
Y=data_mod$Nsucc
golf<-list(K=K, n=n,X=X,Y=Y)

#Code a model file for this model and again produce a 95% HDR for successes out of
100 at all observed distances and 25 feet. Create a plot similar to that obtained in
part (c)

model_string3 <- "model{
    for (i in 1:K){
            Y[i] ~ dbin(p[i],n[i])
            logit(p[i]) <- alpha + beta*X[i] + lambda*X[i]^2
            PredY[i] ~ dbin(p[i],100)
    }
    alpha ~ dunif(-10,10)
    beta ~ dunif(-10,10)
    lambda ~ dunif(-10,10)
}"

#the model
putting.model.mod2 <- jags.model(textConnection(model_string3),
                      golf,n.chains=1)
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 24
##    Total graph size: 188
##
## Initializing model
#the sample
putting.samps.mod2=coda.samples(putting.model.mod2,variable.names=c("PredY"),1e4)

j<-(colMeans(putting.samps.mod2[[1]][,grep("PredY",
colnames(putting.samps.mod2[[1]]))]))

interval <- apply(putting.samps.mod2[[1]], 2, quantile, c(0.025,0.975))

plot(data$dist,prob*100, type = , ylim=c(0,100), xlim=c(2, 25))
lines(data_mod$dist, j,type = 'l', col=2)
lines(data_mod$dist, interval[1,], type = 'l',col=3)
lines(data_mod$dist, interval[2,], type = 'l',col=3)
```



(f) Does this model lead to more believable predictions for successes at 25 feet? What is the HDR at 25 feet for successes out of 100 attempts?

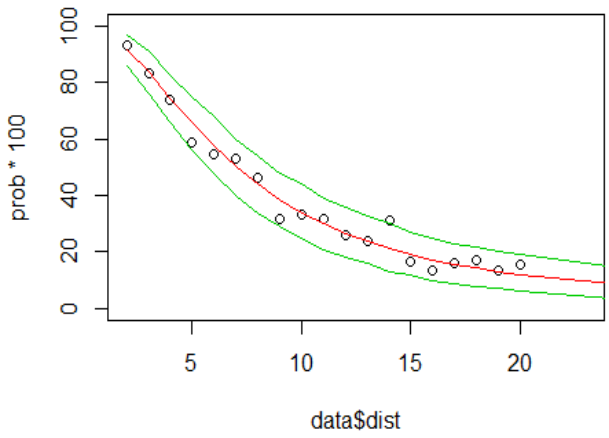
Answer: Yes, now the model gives a more believable predictions. The HDR at 25 feet for successes out of 100 attempts is:

```
#PART F
quantile(putting.samps.mod2[[1]][,"PredY[20]"],c(0.025,0.975))
##  2.5% 97.5%
##    24    55
```

(g) Now try a model based on the log of the distance. How does this compare (with reference to a plot similar to those created for the linear and quadratic models)?

```
#PART G
#model based on the log of the distance
model_string4 <- "model{
    for (i in 1:K){
            Y[i] ~ dbin(p[i],n[i])
            logit(p[i]) <- alpha + beta*log(X[i])
            PredY[i] ~ dbin(p[i],100)
    }
    alpha ~ dunif(-10,10)
    beta ~ dunif(-10,10)
}"

#the model
putting.model.mod3 <- jags.model(textConnection(model_string4),
                    golf,n.chains=1)
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 23
##    Total graph size: 166
##
## Initializing model
```

```
#the sample
putting.samps.mod3=coda.samples(putting.model.mod3,variable.names=c("PredY"),1e4)
j<-(colMeans(putting.samps.mod3[[1]][,grep("PredY",
colnames(putting.samps.mod3[[1]]))]))
interval <- apply(putting.samps.mod3[[1]], 2, quantile, c(0.025,0.975))
plot(data$dist,prob*100, type = , ylim=c(0,100), xlim=c(2, 23))
lines(data_mod$dist,j,type = 'l',col=2)
lines(data_mod$dist, interval[1,], type = 'l',col=3)
lines(data_mod$dist, interval[2,], type = 'l',col=3)
```



Based on the graphs, part G with log of the distance gives a better prediction. Because logically when the distance for professional golfer is increasing, the probability of success should be decreasing. Meanwhile, in part E there is a tendency of it to increase after 20 feet.

(h) Use DIC to choose the best model. Comment on the number of parameters in each of the 3 models.

Answer:

```
#PART H
#linear regression - 1st model
putting.model.mod=jags.model(textConnection(model_string2),golf,n.chains=4)
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 23
##    Total graph size: 146
##
## Initializing model
#linear and quadratic - 2nd model
putting.model.mod2=jags.model(textConnection(model_string3),golf,n.chains=4)
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 24
##    Total graph size: 188
```

```
##
## Initializing model
#log of the distance - 3rd model
putting.model.mod3=jags.model(textConnection(model_string4),golf,n.chains=4)
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 23
##    Total graph size: 166
##
## Initializing model
#the DIC
DIC1=dic.samples(putting.model.mod,1e4)
DIC2=dic.samples(putting.model.mod2,1e4)
DIC3=dic.samples(putting.model.mod3,1e4)

DIC1; DIC2; DIC3
## Mean deviance:  363.9
## penalty 1.969
## Penalized deviance: 365.9
## Mean deviance:  182.7
## penalty 3.006
## Penalized deviance: 185.7
## Mean deviance:  143
## penalty 2.061
## Penalized deviance: 145.1
```

Among the DIC values of all models, we can see that the log of distance (third model) has the lowest penalized deviance. The penalty between the first and the third model is about the same because they have the same model, except the third model use log which made the mean deviance way smaller than the first model. Based on the number of parameters in each of the 3 models, the second model has higher penalty because it has more parameters (lambda) than the rest of models.

(i) Check your chosen model for mixing and convergence and comment on what you find.

Answer:

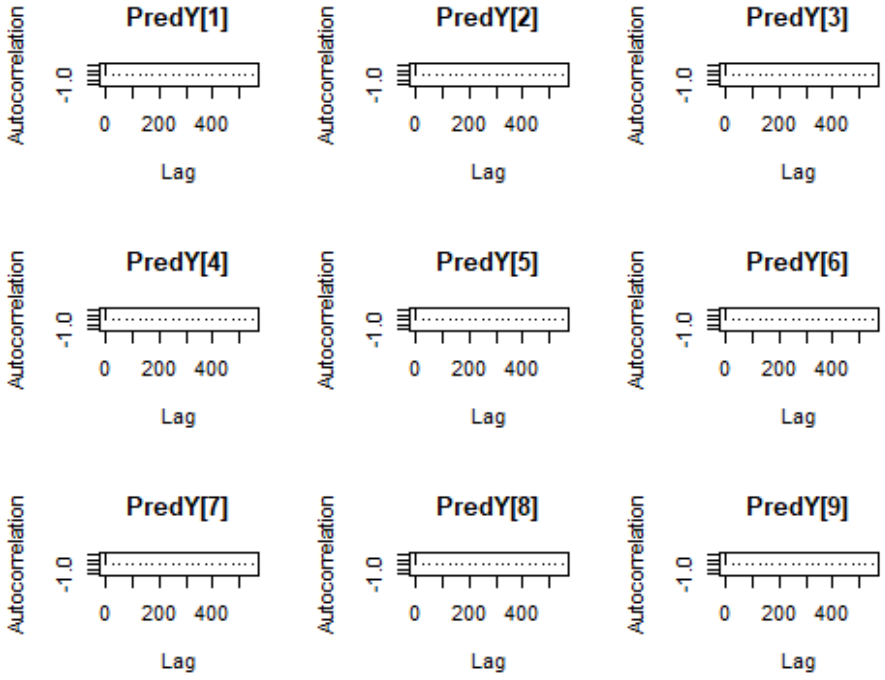
```
#modified the model with additional function = n.adapt and n.chains
putting.model.mod3A=jags.model(textConnection(model_string4),golf,n.chains=4,n.adapt
= 1e5)
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 19
##    Unobserved stochastic nodes: 23
##    Total graph size: 166
##
## Initializing model
#modified the sample with function thin
putting.samps.mod3A=coda.samples(putting.model.mod3,variable.names=c("alpha","beta","
PredY"),1e5,thin = 25)

effectiveSize(putting.samps.mod3A)
```

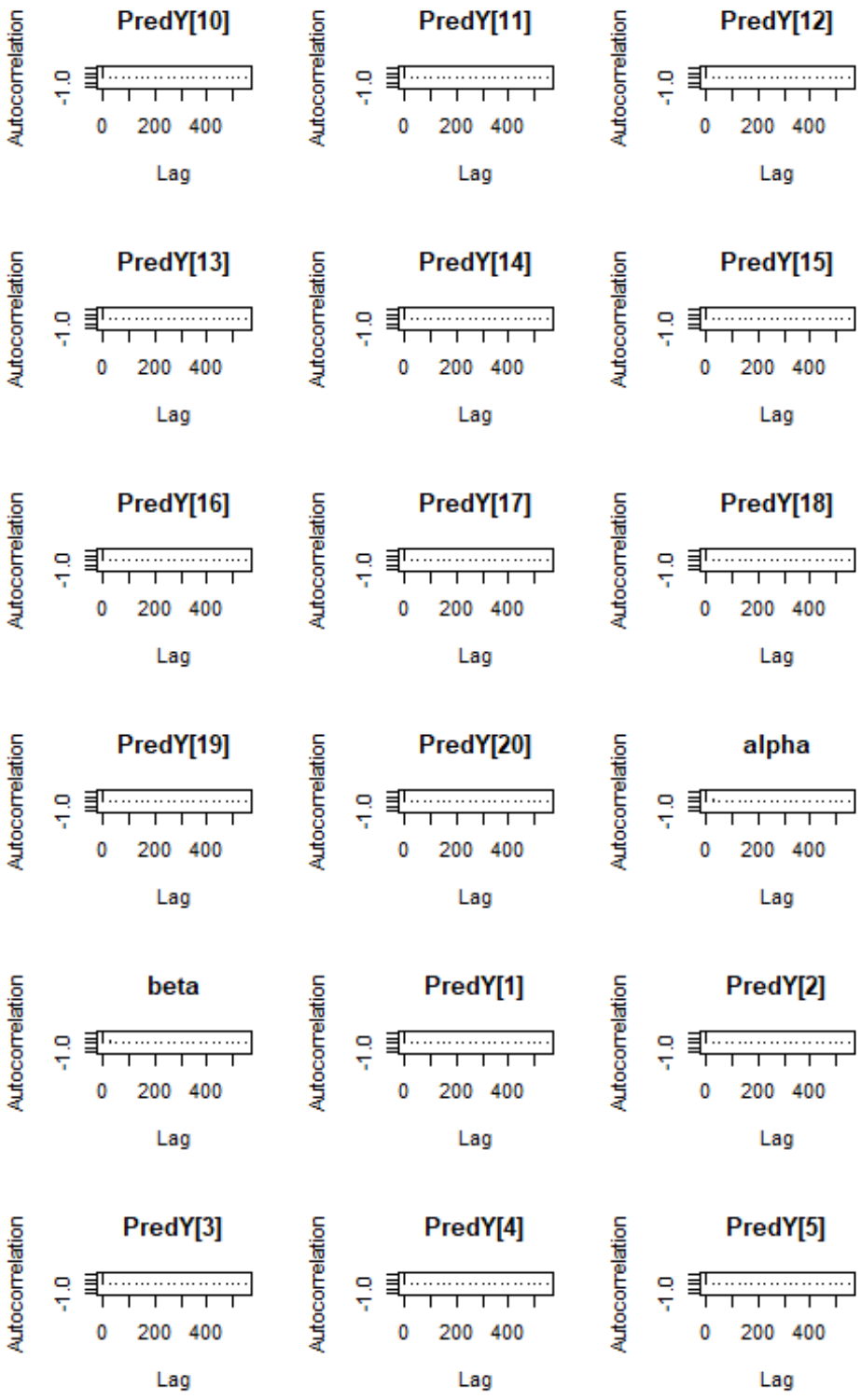
```
##  PredY[1]  PredY[2]  PredY[3]  PredY[4]  PredY[5]  PredY[6]  PredY[7]  PredY[8]
##  16000.00  15835.47  16000.00  16333.86  16470.82  16000.00  16873.07  16191.43
##  PredY[9] PredY[10] PredY[11] PredY[12] PredY[13] PredY[14] PredY[15] PredY[16]
##  16011.12  16000.00  15811.86  16000.00  16000.00  16417.51  15788.79  15370.93
## PredY[17] PredY[18] PredY[19] PredY[20]     alpha      beta
##  16000.00  16195.46  16181.26  16000.00  11856.33  12074.09
dim(putting.samps.mod3A[[1]])
## [1] 4000   22
dim(putting.samps.mod3A[[2]])
## [1] 4000   22
dim(putting.samps.mod3A[[3]])
## [1] 4000   22
dim(putting.samps.mod3A[[4]])
## [1] 4000   22
```

Approximate convergence is diagnosed when the upper limit is close to 1 as below.

```
gelman.diag(putting.samps.mod3A)
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## PredY[1]           1          1
##   …                1          1
## PredY[20]          1          1
## alpha              1          1
## beta               1          1
##
## Multivariate psrf
##
## 1
autocorr.plot(putting.samps.mod3A)
```

The autocorrelation is ideal. No more autocorrelation after lag 1.




PredY[10]
PredY[11]
PredY[12]
PredY[13]
PredY[14]
PredY[15]
PredY[16]
PredY[17]
PredY[18]
PredY[19]
PredY[20]
alpha
beta
PredY[1]
PredY[2]
PredY[3]
PredY[4]
PredY[5]
Autocorrelation
-1.0
0
200
400
Lag