Please name your submission using own name, e.g. Jane Doe e4.pdf

Produce a short report in pdf format (you can export from Word, use R Markdown, any other method to produce it) which answers the following questions and includes all used JAGS and R code and appropriate plots. As always, you may scan handwritten solutions to the non-code based parts as part of your answers and add these to your document. Please also upload your code ﬁles separately to that the tutor can run your code if necessary.

Some data are available on the number of successful putts from various distances for professional golfers. The data are as follows:

distance(feet) # of attempts # of successes x n y 2 1443 1346 3 694 577 4 455 337 5 353 208 6 272 149 7 256 136 8 240 111 9 217 69 10 200 67 11 237 75 12 202 52 13 192 46 14 174 54 15 167 28 16 201 27 17 195 31 18 191 33 19 147 20 20 152 24

For convenience these are also supplied in a ﬁle putting.dat. You 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 function to this interval. Looking at this in reverse, we want a function of p that transforms to the [−∞,∞]. interval. The logit (log-odds) function∗ is a good choice for this, because logit(p) could be any real number when p is between 0 and 1.

∗logit(p) = log( p 1−p

)

Now what can we use for the function of distance that logit(p) is equal to? In the usual linear model approach we could specify that for each observation:

logit(pi) = α + βdi

1. **Plot the ratio of number of successes to number of attempts (y-axis) against each distance (x-axis). [5]**

> a<- c (2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)

> b<- c(1443, 1346, 694 , 455 , 353 , 272 , 256 , 240 , 217 , 200 , 237 , 202 , 192 , 174 , 167 , 201 , 195 ,191 , 147 ,152)

> c<- c(1346,577, 337, 208, 149, 136, 111, 69, 67, 75, 52, 46, 54, 28, 27, 31, 33, 20, 24)

> puttin <- data.frame(a,b,c)

> puttin$d

NULL

> for(i in 1:19){

+ puttin$d[i] <- puttin$c[i]/puttin$b[i] }

> puttin

a b c d

1 2 1443 1346 0.9327789

2 3 694 577 0.8314121

3 4 455 337 0.7406593

4 5 353 208 0.5892351

5 6 272 149 0.5477941

6 7 256 136 0.5312500

7 8 240 111 0.4625000

8 9 217 69 0.3179724

9 10 200 67 0.3350000

10 11 237 75 0.3164557

11 12 202 52 0.2574257

12 13 192 46 0.2395833

13 14 174 54 0.3103448

14 15 167 28 0.1676647

15 16 201 27 0.1343284

16 17 195 31 0.1589744

17 18 191 33 0.1727749

18 19 147 20 0.1360544

19 20 152 24 0.1578947

> plot(puttin$b, puttin$d)

plot(putting$dist,putting$prop[i])

> plot(putting$dist,putting$prop)

1. **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 α and β with limits between -10 and 10. What 95% HDR intervals do you get for α and β? [10]**

putinList <- list(N=19, tries = putting$Ntrys, d= putting$dist, Y= putting$Nsucc)

model1b=jags.model(file="1b new.model", data=putinList)

samps1b=jags.samples(model1b,"alpha0", n.iter=1e5)

putting.samps=coda.samples(putting.model1b,c("alpha0","beta0"),1e5)

plot(putting.samps)

Beta is clearly negative, significantly further away from 0 than within standard dev reach, hence we get a decreasing function (starting from positive alpha), negatively correlated with distance. We observe the same pattern in the observed data graph, and by common sense. Hence model is a good fit.

For later parts, I've asked for plot(dist, 100\*Nsucc/Ntrys) and then the HDR intervals of the posterior predictive at each distance to be added.

Yes, do include this. Of course we expect the overall shape of the curves to match the shape of the mean success rates, so I've asked for this to (i) encourage you to think about how similar the curves should be (ii) use this to spot-check whether you've made a mistake. If the data curve and the posterior predictive curves look very dis-similar then you've gone wrong.

1. **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. 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. 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. [10]and d**

puttingc <- read.delim("putting 1c 25.dat", header = TRUE, sep = "")

putingList <- list(N=20, tries = puttingc$Ntrys, d= puttingc$dist, Y= puttingc$Nsucc) #could have used nrow()

model1c=jags.model(file="1c.model", data=putingList)

samps1b=jags.samples(model1b,"alpha0", n.iter=1e5)

putting.samps=coda.samples(putting.model1b,c("alpha0","beta0"),1e5)

plot(puttingc$dist,100\* rowMeans(putting.samps[[1]][,3]))

miss\_ids=which(is.na(puttingList$Nsucc),arr.ind=T)

**par(ask=TRUE)**

**for (i in 1:nrow(miss\_ids)) { # there is just 1**

**plot(puttingc$dist)-mean(log(hep.data$dist)), puttingc $Y[miss\_ids[i,2],], ylim=range(puttingc$Nsucc,na.rm=TRUE))**

**# add in a boxplot**

**boxplot(as.numeric(puttingc[[1]][,paste0("Nsucc[",paste0(miss\_ids[i,],collapse=","),"]")]),add=T,at=log(puttingc$dist[miss\_ids[i,2]])-mean(log(puttingc$dist)),border=1)**

**}**

[HPD](https://rdrr.io/cran/BayesTwin/man/HPD.html)(putting.samps$)

[apply](https://rdrr.io/r/base/apply.html)(…$putting.samps, 1, function (x) [HPD](https://rdrr.io/cran/BayesTwin/man/HPD.html)(x, 0.95))

As per Lab 4 Task 2, you then need to add one new line to the model file so sample from the posterior predictive distribution.

i.e. the same likelihood as the data but 100 tries at each distance and p based on the sampled alpha and beta values.

summary(putting.samps, quantiles = c(0.025, 0.5, 0.975))

Iterations = 101001:201000

Thinning interval = 1

Number of chains = 1

Sample size per chain = 1e+05

1. Empirical mean and standard deviation for each variable,

plus standard error of the mean:

Mean SD Naive SE Time-series SE

alpha0 2.2323 0.058482 1.849e-04 5.648e-04

beta0 -0.2558 0.006695 2.117e-05 6.482e-05

2. Quantiles for each variable:

2.5% 50% 97.5%

alpha0 2.118 2.2323 2.3467

beta0 -0.269 -0.2558 -0.2427

for 25, 1/1+exp(-(2.2323-0.2558\*25)) = dist(50%quantile) in sumary stats, same algorithm for other intervals, model is a good fit

summary(putting.samps, quantiles = c(0.025, 0.5, 0.975),HPDinterval(putting.samps,95))

1. Empirical mean and standard deviation for each variable,

plus standard error of the mean:

Mean SD Naive SE Time-series SE

alpha0 2.2323 0.058482 1.849e-04 5.648e-04

beta0 -0.2558 0.006695 2.117e-05 6.482e-05

**2. Quantiles for each variable:**

**2.5% 50% 97.5%**

**alpha0 2.118 2.2323 2.3467**

**beta0 -0.269 -0.2558 -0.2427**

SD very small in comparison to distance of parameters from 0, parameters are significant.

You need to sample values from the posterior predictive distribution, not just replot the data (with one extra line). Look over Task 2 in Lab 4 for example.

1. **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? [5]**

Run same code of plotting with hdr included and hardcoding [25] if I got other result in 25 than keeping it as NA.

CI\_95=HPDinterval(glm1.samps[[1]],prob=0.95)[grep("ystar",colnames(glm1.samps[[1]])),]

mle <- glm(eh$statistics~putingList$d,family="binomial", link= logit)

Where eh is saved summary stats for previous example (b)

> itrows <- putting$dist

> plot(itrows ~ eh[itrows])

Błąd w poleceniu '(function (formula, data = NULL, subset = NULL, na.action = na.fail, ':

niepoprawny typ (list) dla zmiennej 'eh[itrows]'

> cbind(itrows, eh)

HPDinterval – select mle[[1]] with ystar as name.

1. **Next you will try a model where the probability of a miss depends both linearly and quadratically on distance: logit(pi) = α + βdi + λd2 i 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) [5]**

You'll then want to think (especially in part (e)) about whether you think the model is behaving correctly based on the plot you get.

1. **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?**

After multiplying by 100, curve that looked like it fits test small sample data well, was actually overfitted and predicted new values badly. Curvature recorded random error and less steepness in 2nd der. predicts first part of slope well, but then we lose the property of negative correlation and that’s extremely inaccurate if we believe the further away we are, the chances fall steeply not start growing again…

1. **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)?**

Not better than previous two.

puttinge <- read.delim("putting.dat", header = TRUE, sep = "")

putingListe <- list(N=19, tries = puttingc$Ntrys, d= puttingc$dist, Y= puttingc$Nsucc) #could have used nrow()

model1e=jags.model(file="1e.model", data=putingListe)

samps1e=jags.samples(model1e,"alpha0", n.iter=1e5) #random name, maybe gamma better

puttingc.samps=coda.samples(putting.model1b,c("alpha0","beta0", 1e5)