> ########## ########## Simulation Diabetes Beispiel mit Random Effects mit dem rjags package

> ########## Die Working Directory muss auf Ihre Bedürfnisse angepasst werden

>

>

>

>

>

> # Teil Simulation mit JAGS ------------------------------------------------------------------

>

>

>

>

> ##### Clear data

> rm(list=ls())

>

>

>

> ##### load libraries

> library(rjags)

> library(coda)

> library(random)

> library(matrixStats) # zusätzl Paket, berechnet Median

> load.module("glm")

> load.module("lecuyer")

> load.module("dic")

>

>

>

> ##### Sichergehen richtiger Working directory

> setwd("C:/Users/IvanB/Desktop/Masterarbeit/Statistische Programme und Gibbs Sampler/Programm JAGS/Nachrechnen TSD2-Dokument/Nachrechnen mit rjags/Ex3 Diabetes")

>

>

>

> ##### Read the data into R.

> #data = read.table("Diabetes\_Data\_neu sortiert.txt", sep = "", header=F)

> data = as.matrix(read.table("Diabetes\_Data\_neu sortiert.txt", sep = "", header=T))

> head(data) # Shows the first six entries

time.. t..1. t..2. t..3. r..1. r..2. r..3. n..1. n..2. n..3. na..

[1,] 5.8 1 2 3 43 34 37 1081 2213 1102 3

[2,] 4.7 1 2 NA 29 20 NA 416 424 NA 2

[3,] 3.0 1 2 NA 140 118 NA 1631 1578 NA 2

[4,] 3.8 1 3 NA 75 86 NA 3272 3297 NA 2

[5,] 4.0 1 4 5 302 154 119 6766 3954 4096 3

[6,] 3.0 1 4 NA 176 136 NA 2511 2508 NA 2

> #data2 = as.data.frread.table("Diabetes\_Data\_Rest.txt")

> #head(data2) # Shows the first six entries

>

>

>

> ##### Values for simulation, prepare dat for JAGS (allocation values from data)

> ns <- nrow(data)

> #ns # check

> nt <- 6

> #nt # check

> na <- data[,11]

> #na # check

> r <- data[,5:7]

> #r # Check

> time <- data[,1]

> #time # Check

> t <- data[,2:4]

> #t # Check

> n <- data[,8:10]

> #n # Check

>

> dat <- list(ns=ns, nt=nt, na=na, r=r, time=time, t=t, n=n) # names list of numbers

> dat

$ns

[1] 22

$nt

[1] 6

$na

[1] 3 2 2 2 3 2 2 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2

$r

r..1. r..2. r..3.

[1,] 43 34 37

[2,] 29 20 NA

[3,] 140 118 NA

[4,] 75 86 NA

[5,] 302 154 119

[6,] 176 136 NA

[7,] 200 138 NA

[8,] 8 1 NA

[9,] 154 177 NA

[10,] 489 449 NA

[11,] 155 102 NA

[12,] 399 335 NA

[13,] 202 163 NA

[14,] 115 93 NA

[15,] 70 32 45

[16,] 97 95 93

[17,] 799 567 NA

[18,] 251 216 NA

[19,] 665 569 NA

[20,] 380 337 NA

[21,] 320 242 NA

[22,] 845 690 NA

$time

[1] 5.8 4.7 3.0 3.8 4.0 3.0 4.1 1.0 3.3 3.0 4.5 4.8 3.1 3.7 3.8 4.0 5.5 4.5 4.0 6.1 4.8 4.2

$t

t..1. t..2. t..3.

[1,] 1 2 3

[2,] 1 2 NA

[3,] 1 2 NA

[4,] 1 3 NA

[5,] 1 4 5

[6,] 1 4 NA

[7,] 1 5 NA

[8,] 1 6 NA

[9,] 2 4 NA

[10,] 2 5 NA

[11,] 2 5 NA

[12,] 2 5 NA

[13,] 2 6 NA

[14,] 2 6 NA

[15,] 3 4 5

[16,] 3 4 5

[17,] 3 4 NA

[18,] 3 4 NA

[19,] 3 4 NA

[20,] 3 5 NA

[21,] 3 6 NA

[22,] 4 6 NA

$n

n..1. n..2. n..3.

[1,] 1081 2213 1102

[2,] 416 424 NA

[3,] 1631 1578 NA

[4,] 3272 3297 NA

[5,] 6766 3954 4096

[6,] 2511 2508 NA

[7,] 2826 2800 NA

[8,] 196 196 NA

[9,] 4870 4841 NA

[10,] 2646 2623 NA

[11,] 2883 2837 NA

[12,] 3472 3432 NA

[13,] 2721 2715 NA

[14,] 2175 2167 NA

[15,] 405 202 410

[16,] 1960 1965 1970

[17,] 7040 7072 NA

[18,] 5059 5095 NA

[19,] 8078 8098 NA

[20,] 5230 5183 NA

[21,] 3979 4020 NA

[22,] 5074 5087 NA

>

>

>

> ##### read in inits with chains

> # Anmerkung: da cloglog als Link: wurden die Inits von JAGS generiert.

> # Die manuellen Inits sind zwar drin, können aber rausgelassen werden

>

> inits1 <- list(d=c(NA,0,0,0,0,0),

+ sd=1,

+ mu=c(0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 0,0,0,0,0, 0,0),

+ A=0 ,

+ delta= structure(.Data= c(NA, 0, 0, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, 0, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA,

+ 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0,0, NA, 0, 0, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA,

+ NA, 0, NA), .Dim=c(22, 3)))

> # .RNG.name="base::Wichmann-Hill", .RNG.seed=1

>

> inits2 <- list(d=c(NA,-1,4,-1,2,3),

+ sd=3,

+ mu=c(1,1,0,1,0, 0,1,0,0,0, 1,1,0,0,0, 0,1,0,0,0, 1,1),

+ A=1 ,

+ delta= structure(.Data= c(NA, 0, 0, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, 0, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA,

+ 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0,0, NA, 0, 0, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA,

+ NA, 0, NA), .Dim=c(22, 3)))

>

> inits3 <- list(d=c(NA,1,4,-3,-2,3),

+

+ sd=4.5,

+ mu=c(1,1,0,1,0, 0,1,0,0,0, 1,1,0,-2,0, 0,1,0,-2,0, 1,1),

+ A=2 ,

+ delta= structure(.Data= c(NA, 0, 0, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, 0, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA,

+ 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0,0, NA, 0, 0, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA, NA, 0, NA,

+ NA, 0, NA), .Dim=c(22, 3)))

> #, .RNG.name="base::Wichmann-Hill", .RNG.seed=3

>

> all.inits <- list(inits1, inits2, inits3)

> # all.inits <- list(inits2)

>

>

>

> ##### define JAGS model within R

> cat("model{ # \*\*\* PROGRAM STARTS

+ for(i in 1:ns){ # LOOP THROUGH STUDIES

+ w[i,1] <- 0 # adjustment for multi-arm trials is zero for control arm

+ delta[i,1] <- 0 # treatment effect is zero for control arm

+ mu[i] ~ dnorm(0,.0001) # vague priors for all trial baselines

+ for (k in 1:na[i]) { # LOOP THROUGH ARMS

+ r[i,k] ~ dbin(p[i,k],n[i,k]) # Binomial likelihood

+

+ cloglog(p[i,k]) <- log(time[i]) + mu[i] + delta[i,k] # model for linear predictor

+ rhat[i,k] <- p[i,k] \* n[i,k] # expected value of the numerators

+ dev[i,k] <- 2 \* (r[i,k] \* (log(r[i,k])-log(rhat[i,k])) + (n[i,k]-r[i,k]) \* (log(n[i,k]-r[i,k]) - log(n[i,k]-rhat[i,k]))) #Deviance contribution

+ }

+ resdev[i] <- sum(dev[i,1:na[i]]) # summed residual deviance contribution for this trial

+ for (k in 2:na[i]) { # LOOP THROUGH ARMS

+ delta[i,k] ~ dnorm(md[i,k],taud[i,k]) # trial-specific LOR distributions

+ md[i,k] <- d[t[i,k]] - d[t[i,1]] + sw[i,k] # mean of LOR distributions (with multi-arm correction)

+ taud[i,k] <- tau \*2\*(k-1)/k # precision of LOR distributions (with multi-arm correction)

+ w[i,k] <- (delta[i,k] - d[t[i,k]] + d[t[i,1]]) # adjustment for multi-arm RCTs

+ sw[i,k] <- sum(w[i,1:(k-1)])/(k-1) # cumulative adjustment for multi-arm trials

+ }

+ }

+ totresdev <- sum(resdev[]) #Total Residual Deviance

+

+ d[1]<- 0

+ for (k in 2:nt){ d[k] ~ dnorm(0,.0001) } # vague priors for treatment effects

+ sd ~ dunif(0,5) # vague prior for between-trial SD

+ tau <- pow(sd,-2) # between-trial precision = (1/between-trial variance)

+ A ~ dnorm(-4.2,1.11)

+ for (k in 1:nt) {

+ cloglog(T[k]) <- log(3) + A + d[k]

+ }

+ } ",

+ file="Diabetes\_Random.txt")

>

>

>

> ##### Set up the JAGS model and settings

> jags.m <- jags.model( file = "Diabetes\_Random.txt", data=dat, n.chains=3, n.adapt=5000)

Compiling model graph

Resolving undeclared variables

Allocating nodes

Graph information:

Observed stochastic nodes: 48

Unobserved stochastic nodes: 55

Total graph size: 1145

Initializing model

|++++++++++++++++++++++++++++++++++++++++++++++++++| 100%

>

>

>

> ##### Initialization

> update(jags.m, 10000) # burn in

|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| 100%

>

>

>

> ##### run JAGS and save posterior samples

> samps\_coda <- coda.samples( jags.m, variable.names=c("d[2]", "d[3]", "d[4]", "d[5]", "d[6]", "T[1]", "T[2]", "T[3]", "T[4]", "T[5]", "T[6]", "sd" , "totresdev"), n.iter=20000, DIC=T )

|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| 100%

>

>

>

>

>

> # Ausgabe posteriore Werte, Berechnung Median und Berechnung DIC --------------------------

>

>

>

>

> ##### summarize posterior samples # funktioniert nur mit coda.samples

> summary(window(samps\_coda, start=10001)) # burnin = 100000

Iterations = 15001:35000

Thinning interval = 1

Number of chains = 3

Sample size per chain = 20000

1. Empirical mean and standard deviation for each variable,

plus standard error of the mean:

Mean SD Naive SE Time-series SE

T[1] 0.06528 0.06715 0.0002741 0.0002719

T[2] 0.05004 0.05344 0.0002182 0.0002196

T[3] 0.06106 0.06364 0.0002598 0.0002602

T[4] 0.05219 0.05533 0.0002259 0.0002263

T[5] 0.04492 0.04851 0.0001980 0.0001984

T[6] 0.04200 0.04577 0.0001869 0.0001874

d[2] -0.28771 0.08880 0.0003625 0.0010367

d[3] -0.07475 0.08775 0.0003583 0.0009500

d[4] -0.24235 0.08408 0.0003432 0.0008502

d[5] -0.40200 0.08542 0.0003487 0.0009112

d[6] -0.47408 0.11034 0.0004504 0.0012158

sd 0.12783 0.04408 0.0001799 0.0008472

totresdev 53.72503 9.97318 0.0407153 0.1066715

2. Quantiles for each variable:

2.5% 25% 50% 75% 97.5%

T[1] 0.007044 0.02353 0.04420 0.08130 0.25044

T[2] 0.005285 0.01766 0.03341 0.06182 0.19507

T[3] 0.006541 0.02184 0.04115 0.07588 0.23600

T[4] 0.005496 0.01851 0.03492 0.06476 0.20277

T[5] 0.004691 0.01579 0.02978 0.05527 0.17657

T[6] 0.004363 0.01470 0.02778 0.05183 0.16489

d[2] -0.470828 -0.34379 -0.28490 -0.22796 -0.12104

d[3] -0.252041 -0.13118 -0.07359 -0.01683 0.09639

d[4] -0.410605 -0.29617 -0.24141 -0.18781 -0.07943

d[5] -0.579741 -0.45583 -0.39908 -0.34519 -0.24029

d[6] -0.703406 -0.54321 -0.47044 -0.40142 -0.26594

sd 0.053393 0.09719 0.12339 0.15336 0.22814

totresdev 35.855314 46.74355 53.11193 60.05750 74.93459

Warning messages:

1: In FUN(X[[i]], ...) : start value not changed

2: In FUN(X[[i]], ...) : start value not changed

3: In FUN(X[[i]], ...) : start value not changed

> # Anmerkung: mean für totresdev ist D\_res

>

>

>

> #### Trace Monitore

> #plot(samps\_coda)

>

>

>

>

>

> #### Median-Berechnung

>

> #as.matrix(samps\_coda[])

>

> # Median für T1

> median(as.matrix(samps\_coda[,1]))

[1] 0.04419726

> # Median für T2

> median(as.matrix(samps\_coda[,2]))

[1] 0.03340914

> # Median für T3

> median(as.matrix(samps\_coda[,3]))

[1] 0.04114925

> # Median für T4

> median(as.matrix(samps\_coda[,4]))

[1] 0.03491905

> # Median für T5

> median(as.matrix(samps\_coda[,5]))

[1] 0.02977856

> # Median für T6

> median(as.matrix(samps\_coda[,6]))

[1] 0.02777897

>

> # Median für d[2]

> median(as.matrix(samps\_coda[,7]))

[1] -0.2849025

> # Median für d[3]

> median(as.matrix(samps\_coda[,8]))

[1] -0.07358553

> # Median für d[4]

> median(as.matrix(samps\_coda[,9]))

[1] -0.2414103

> # Median für d[5]

> median(as.matrix(samps\_coda[,10]))

[1] -0.3990785

> # Median für d[6

> median(as.matrix(samps\_coda[,11]))

[1] -0.470438

>

> # Median für sd

> median(as.matrix(samps\_coda[,12]))

[1] 0.1233881

>

>

>

> ##### run JAGS and save posterior samples

> samps\_jags <- jags.samples( jags.m, variable.names=c("dev", "totresdev", "rhat", "deviance"), n.iter=20000, DIC=T )

|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| 100%

>

>

>

> ##### Berechnung DIC und pD

> D\_res <- summary(samps\_jags$totresdev[])

> D\_res

Min. 1st Qu. Median Mean 3rd Qu. Max.

19.57 46.52 52.89 53.55 59.92 112.34

> pD <- var(samps\_jags$deviance)/2

> pD

[1] 50.11928

> DIC = D\_res + pD

> DIC

Min. 1st Qu. Median Mean 3rd Qu. Max.

69.69 96.64 103.01 103.67 110.04 162.46

>

>

>

>

>

> ########## ########## ########## Simulation beendet ########## ########## ##########