> ########## ########## 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(runjags)

> library(random)

> library(coda)

> 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 runjags/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

>

>

>

>

>

>

> ##### 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="lecuyer::RngStream", .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)))

> # .RNG.name="lecuyer::RngStream", .RNG.seed=2)

>

>

> 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="lecuyer::RngStream", .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 <- run.jags(model="Diabetes\_Random.txt", monitor=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", "deviance", "pd", "dic" , "full.pd" ),

+ data=list("ns"=ns, "nt"=nt, "na"=na, "r"=r, "time"=time, "t"=t, "n"=n) , n.chains=3, burnin = 10000, sample = 20000, adapt = 5000)

Compiling rjags model...

Calling the simulation using the rjags method...

Adapting the model for 5000 iterations...

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

Burning in the model for 10000 iterations...

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

Running the model for 20000 iterations...

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

Extending 20000 iterations for pD/DIC estimates...

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

Simulation complete

Calculating summary statistics...

Calculating the Gelman-Rubin statistic for 14 variables....

Finished running the simulation

Warning message:

No initial values were provided - JAGS will use the same initial values for all chains

>

>

>

> #### optional, falls nicht konvergiert:

> #jags.m <- autorun.jags(model="Diabetes\_Random.txt", monitor=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", "deviance", "pd", "dic" , "full.pd" ),

> # data=list("ns"=ns, "nt"=nt, "na"=na, "r"=r, "time"=time, "t"=t, "n"=n) , n.chains=3, inits=all.inits, burnin = 100000, sample = 100000, adapt = 5000, max.time="1hr")

>

>

>

>

>

>

>

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

>

>

>

> print(jags.m)

JAGS model summary statistics from 60000 samples (chains = 3; adapt+burnin = 15000):

Lower95 Median Upper95 Mean SD Mode MCerr MC%ofSD SSeff AC.10 psrf

d[2] -0.45863 -0.28476 -0.11302 -0.2883 0.087974 -- 0.001026 1.2 7353 0.11877 1.0002

d[3] -0.24904 -0.073758 0.098256 -0.07539 0.087744 -- 0.00095678 1.1 8410 0.088073 1.0003

d[4] -0.40968 -0.24224 -0.076076 -0.24248 0.083548 -- 0.00086867 1 9250 0.076508 1.0006

d[5] -0.57296 -0.39919 -0.23685 -0.40242 0.085318 -- 0.00093708 1.1 8289 0.090377 1.0001

d[6] -0.69374 -0.47028 -0.25808 -0.47352 0.10953 -- 0.0012166 1.1 8105 0.091421 1.0001

T[1] 0.0021124 0.044051 0.19883 0.065835 0.068422 -- 0.00039503 0.6 30000 -0.010856 0.99999

T[2] 0.0014445 0.033202 0.1536 0.050414 0.054283 -- 0.0003134 0.6 30000 -0.011145 0.99998

T[3] 0.0020919 0.040907 0.18755 0.061569 0.064825 -- 0.00037427 0.6 30000 -0.011028 0.99998

T[4] 0.0016212 0.034775 0.16146 0.052655 0.056468 -- 0.00032602 0.6 30000 -0.011677 0.99998

T[5] 0.0012881 0.029796 0.13839 0.04528 0.049323 -- 0.00028477 0.6 30000 -0.010912 0.99999

T[6] 0.001201 0.027635 0.12993 0.042417 0.046594 -- 0.00026901 0.6 30000 -0.010952 0.99999

sd 0.050227 0.12357 0.21578 0.12798 0.043924 -- 0.00083109 1.9 2793 0.37977 1.0003

totresdev 34.649 53.097 72.84 53.656 9.8628 -- 0.09917 1 9891 0.073331 1.0001

deviance 355.13 373.58 393.33 374.14 9.8628 -- 0.09917 1 9891 0.073331 1

Model fit assessment:

DIC = 412.2339 (range between chains: 412.2015 - 412.3995)

[PED not available from the stored object]

Estimated effective number of parameters: pD = 38.15376

Total time taken: 1.1 minutes

> summary(jags.m$mcmc) # für 2.5 - 97.5 CrI

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

d[2] -0.28835 0.08785 0.0003586 0.0010409

d[3] -0.07546 0.08758 0.0003575 0.0009395

d[4] -0.24268 0.08346 0.0003407 0.0008416

d[5] -0.40240 0.08513 0.0003475 0.0009039

d[6] -0.47357 0.10994 0.0004488 0.0011948

T[1] 0.06579 0.06823 0.0002786 0.0002766

T[2] 0.05040 0.05424 0.0002214 0.0002233

T[3] 0.06151 0.06465 0.0002639 0.0002660

T[4] 0.05258 0.05625 0.0002296 0.0002310

T[5] 0.04525 0.04915 0.0002007 0.0002017

T[6] 0.04238 0.04648 0.0001898 0.0001917

sd 0.12799 0.04382 0.0001789 0.0008249

totresdev 53.64736 9.85446 0.0402307 0.0958917

deviance 374.13302 9.85446 0.0402307 0.0958917

2. Quantiles for each variable:

2.5% 25% 50% 75% 97.5%

d[2] -0.471165 -0.34365 -0.28534 -0.22957 -0.12373

d[3] -0.252173 -0.13143 -0.07423 -0.01811 0.09448

d[4] -0.409565 -0.29599 -0.24264 -0.18817 -0.07794

d[5] -0.579057 -0.45551 -0.39967 -0.34592 -0.24155

d[6] -0.702838 -0.54188 -0.47040 -0.40098 -0.26505

T[1] 0.007086 0.02354 0.04433 0.08236 0.25474

T[2] 0.005289 0.01769 0.03339 0.06237 0.19872

T[3] 0.006542 0.02184 0.04108 0.07700 0.23983

T[4] 0.005531 0.01849 0.03493 0.06530 0.20588

T[5] 0.004679 0.01576 0.02996 0.05610 0.17951

T[6] 0.004360 0.01469 0.02783 0.05229 0.16866

sd 0.056037 0.09699 0.12354 0.15365 0.22683

totresdev 35.896699 46.78792 53.11635 59.90700 74.59757

deviance 356.382359 367.27357 373.60201 380.39266 395.08323

>

>

>

>

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