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

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

>

>

>

>

>

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

>

>

>

>

> ##### Clear data

> rm(list=ls())

>

>

>

> ##### load libraries

> #library(rjags) # jagsUI benötigt dieses Paket

> library(lattice)

> #library(coda)

> library(jagsUI)

> #library(random)

>

>

>

> ##### Sichergehen richtiger Working directory

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

[1] 22

> nt <- 6

> nt # check

[1] 6

> na <- data[,11]

> na # check

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

> r <- data[,5:7]

> r # Check

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 <- data[,1]

> time # Check

[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 <- data[,2:4]

> t # Check

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 <- data[,8:10]

> 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

>

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

> dat

[[1]]

[1] "ns"

[[2]]

[1] "nt"

[[3]]

[1] "na"

[[4]]

[1] "r"

[[5]]

[1] "time"

[[6]]

[1] "t"

[[7]]

[1] "n"

>

>

>

> ##### Parameter to monitor/save

> params <- 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"," dev", "rhat" )

> params

[1] "d[2]" "d[3]" "d[4]" "d[5]" "d[6]" "T[1]" "T[2]" "T[3]" "T[4]" "T[5]" "T[6]" "sd" "totresdev"

[14] " dev" "rhat"

>

>

>

> ##### 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(data=dat,

+ inits=NULL,

+ parameters.to.save=params,

+ model.file="Diabetes\_Random.txt",

+ n.chains=3,

+ n.iter=20000, n.burnin=10000, n.thin=1,

+ store.data=TRUE)

Processing function input.......

Done.

Compiling model graph

Resolving undeclared variables

Allocating nodes

Graph information:

Observed stochastic nodes: 48

Unobserved stochastic nodes: 55

Total graph size: 1145

Initializing model

Adaptive phase.....

Adaptive phase complete

Burn-in phase, 10000 iterations x 3 chains

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

Sampling from joint posterior, 10000 iterations x 3 chains

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

Calculating statistics.......

Done.

Warning messages:

1: In process.input(data, parameters.to.save, inits, n.chains, n.iter, :

Suppling a list of character strings to the data argument will be deprecated in the next version

2: In process.input(data, parameters.to.save, inits, n.chains, n.iter, :

Suppling a character vector to the data argument will be deprecated in the next version

>

>

>

>

>

> # Anzeigen der posterioren Werte und des Medians --------------------------

>

>

>

>

> #traceplot(jags.m) # zeigt Abbildungen einzeln nach Eingabe der Entertaste an

> jags.View(jags.m)

> jags.m

JAGS output for model 'Diabetes\_Random.txt', generated by jagsUI.

Estimates based on 3 chains of 20000 iterations,

adaptation = 100 iterations (sufficient),

burn-in = 10000 iterations and thin rate = 1,

yielding 30000 total samples from the joint posterior.

MCMC ran for 0.341 minutes at time 2019-08-27 10:59:16.

mean sd 2.5% 50% 97.5% overlap0 f Rhat n.eff

d[2] -0.286 0.088 -0.471 -0.283 -0.120 FALSE 0.999 1.002 1087

d[3] -0.072 0.088 -0.252 -0.071 0.097 TRUE 0.797 1.001 2314

d[4] -0.239 0.084 -0.409 -0.238 -0.074 FALSE 0.996 1.001 3058

d[5] -0.400 0.085 -0.578 -0.398 -0.240 FALSE 1.000 1.001 2174

d[6] -0.471 0.111 -0.699 -0.468 -0.259 FALSE 1.000 1.001 1626

T[1] 0.065 0.066 0.007 0.045 0.250 FALSE 1.000 1.000 23990

T[2] 0.050 0.053 0.005 0.034 0.193 FALSE 1.000 1.000 24135

T[3] 0.061 0.063 0.007 0.041 0.235 FALSE 1.000 1.000 27259

T[4] 0.052 0.055 0.006 0.035 0.204 FALSE 1.000 1.000 30000

T[5] 0.045 0.048 0.005 0.030 0.175 FALSE 1.000 1.000 23516

T[6] 0.042 0.045 0.004 0.028 0.166 FALSE 1.000 1.000 28129

sd 0.129 0.044 0.057 0.124 0.228 FALSE 1.000 1.002 1651

totresdev 53.590 9.932 35.720 52.973 74.674 FALSE 1.000 1.001 4058

rhat[1,1] 35.180 4.486 27.267 34.854 44.923 FALSE 1.000 1.002 1642

rhat[2,1] 27.833 4.143 20.338 27.625 36.488 FALSE 1.000 1.000 11178

rhat[3,1] 144.045 10.393 124.188 143.897 165.036 FALSE 1.000 1.000 7737

rhat[4,1] 79.876 7.694 65.403 79.658 95.592 FALSE 1.000 1.000 30000

rhat[5,1] 303.619 16.003 272.975 303.431 335.672 FALSE 1.000 1.001 1743

rhat[6,1] 175.040 11.761 152.709 174.758 198.574 FALSE 1.000 1.000 30000

rhat[7,1] 200.691 12.441 176.897 200.521 225.942 FALSE 1.000 1.000 17465

rhat[8,1] 5.599 1.872 2.569 5.376 9.846 FALSE 1.000 1.000 13763

rhat[9,1] 157.866 11.284 136.511 157.661 180.972 FALSE 1.000 1.001 3241

rhat[10,1] 490.407 19.192 453.742 490.156 528.595 FALSE 1.000 1.000 12751

rhat[11,1] 145.715 10.839 125.388 145.386 167.854 FALSE 1.000 1.000 30000

rhat[12,1] 395.578 17.805 361.541 395.255 430.899 FALSE 1.000 1.000 30000

rhat[13,1] 200.740 12.610 177.127 200.445 226.263 FALSE 1.000 1.000 26177

rhat[14,1] 114.224 9.362 96.589 113.892 133.312 FALSE 1.000 1.000 30000

rhat[15,1] 68.400 6.100 56.993 68.224 80.870 FALSE 1.000 1.000 8471

rhat[16,1] 104.583 8.427 88.568 104.467 121.423 FALSE 1.000 1.000 11518

rhat[17,1] 786.592 26.632 734.939 786.290 839.654 FALSE 1.000 1.000 12285

rhat[18,1] 251.067 14.235 223.722 250.860 279.584 FALSE 1.000 1.000 11519

rhat[19,1] 665.579 23.635 620.151 665.238 712.480 FALSE 1.000 1.000 8818

rhat[20,1] 390.322 18.761 354.166 390.278 427.186 FALSE 1.000 1.000 16665

rhat[21,1] 323.952 16.445 292.538 323.623 357.100 FALSE 1.000 1.000 30000

rhat[22,1] 845.023 26.125 794.931 844.542 897.161 FALSE 1.000 1.000 17115

rhat[1,2] 46.245 5.675 35.334 46.176 57.608 FALSE 1.000 1.001 2854

rhat[2,2] 21.136 3.381 15.033 20.945 28.269 FALSE 1.000 1.000 10529

rhat[3,2] 113.648 9.026 96.791 113.413 132.176 FALSE 1.000 1.000 6024

rhat[4,2] 81.054 7.772 66.821 80.721 97.071 FALSE 1.000 1.000 30000

rhat[5,2] 148.461 10.534 128.798 148.129 170.327 FALSE 1.000 1.000 30000

rhat[6,2] 136.961 10.219 117.587 136.750 157.736 FALSE 1.000 1.000 10375

rhat[7,2] 137.275 10.167 118.088 137.015 157.792 FALSE 1.000 1.000 23298

rhat[8,2] 3.373 1.168 1.500 3.234 6.018 FALSE 1.000 1.000 16885

rhat[9,2] 173.064 11.949 150.435 172.701 197.564 FALSE 1.000 1.000 19875

rhat[10,2] 447.176 18.443 411.775 446.840 484.175 FALSE 1.000 1.000 17793

rhat[11,2] 111.368 9.377 93.406 111.246 130.215 FALSE 1.000 1.000 30000

rhat[12,2] 338.052 16.419 306.674 337.860 371.080 FALSE 1.000 1.000 16472

rhat[13,2] 164.339 11.421 142.695 164.062 187.368 FALSE 1.000 1.000 16613

rhat[14,2] 93.810 8.278 78.297 93.567 110.675 FALSE 1.000 1.000 30000

rhat[15,2] 29.570 3.269 23.643 29.371 36.535 FALSE 1.000 1.000 10221

rhat[16,2] 94.607 7.678 80.416 94.280 110.568 FALSE 1.000 1.000 30000

rhat[17,2] 579.081 22.926 535.015 578.899 625.014 FALSE 1.000 1.000 18441

rhat[18,2] 215.734 13.004 191.202 215.370 242.311 FALSE 1.000 1.000 30000

rhat[19,2] 568.550 21.871 526.118 568.403 611.983 FALSE 1.000 1.000 9212

rhat[20,2] 326.294 17.132 293.733 326.058 360.842 FALSE 1.000 1.000 25995

rhat[21,2] 237.615 14.104 211.031 237.456 266.141 FALSE 1.000 1.000 30000

rhat[22,2] 689.618 23.670 643.761 689.446 736.712 FALSE 1.000 1.000 30000

rhat[1,3] 32.589 4.104 25.272 32.314 41.318 FALSE 1.000 1.000 30000

rhat[5,3] 122.810 9.302 104.883 122.689 141.226 FALSE 1.000 1.000 9493

rhat[15,3] 49.015 5.049 39.492 48.895 59.246 FALSE 1.000 1.000 26442

rhat[16,3] 85.699 7.473 72.111 85.345 101.413 FALSE 1.000 1.000 17692

dev[1,1] 2.391 2.185 0.011 1.839 7.945 FALSE 1.000 1.001 2515

dev[2,1] 0.740 1.023 0.001 0.340 3.641 FALSE 1.000 1.000 30000

dev[3,1] 0.929 1.297 0.001 0.433 4.721 FALSE 1.000 1.000 30000

dev[4,1] 1.028 1.395 0.001 0.494 4.995 FALSE 1.000 1.001 9080

dev[5,1] 0.888 1.272 0.001 0.402 4.492 FALSE 1.000 1.001 9312

dev[6,1] 0.860 1.204 0.001 0.401 4.297 FALSE 1.000 1.000 12015

dev[7,1] 0.831 1.170 0.001 0.382 4.192 FALSE 1.000 1.000 8024

dev[8,1] 1.864 2.084 0.004 1.178 7.465 FALSE 1.000 1.000 12373

dev[9,1] 0.914 1.294 0.001 0.418 4.663 FALSE 1.000 1.001 9668

dev[10,1] 0.925 1.298 0.001 0.424 4.648 FALSE 1.000 1.001 5931

dev[11,1] 1.511 1.907 0.002 0.800 6.829 FALSE 1.000 1.000 30000

dev[12,1] 0.945 1.328 0.001 0.433 4.699 FALSE 1.000 1.001 15218

dev[13,1] 0.868 1.227 0.001 0.396 4.304 FALSE 1.000 1.002 16473

dev[14,1] 0.821 1.154 0.001 0.374 4.074 FALSE 1.000 1.000 7464

dev[15,1] 0.712 1.005 0.001 0.326 3.621 FALSE 1.000 1.000 30000

dev[16,1] 1.265 1.579 0.001 0.677 5.606 FALSE 1.000 1.000 30000

dev[17,1] 1.249 1.731 0.001 0.570 6.180 FALSE 1.000 1.001 6185

dev[18,1] 0.850 1.196 0.001 0.386 4.250 FALSE 1.000 1.000 30000

dev[19,1] 0.915 1.307 0.001 0.412 4.659 FALSE 1.000 1.000 6979

dev[20,1] 1.249 1.711 0.001 0.599 6.005 FALSE 1.000 1.001 5388

dev[21,1] 0.951 1.363 0.001 0.425 4.821 FALSE 1.000 1.001 19742

dev[22,1] 0.969 1.368 0.001 0.445 4.831 FALSE 1.000 1.000 30000

dev[1,2] 4.177 3.096 0.088 3.606 11.617 FALSE 1.000 1.001 3300

dev[2,2] 0.607 0.865 0.001 0.278 3.059 FALSE 1.000 1.000 27266

dev[3,2] 0.977 1.337 0.001 0.455 4.777 FALSE 1.000 1.001 8534

dev[4,2] 1.112 1.471 0.001 0.546 5.190 FALSE 1.000 1.000 30000

dev[5,2] 1.015 1.376 0.001 0.472 4.903 FALSE 1.000 1.000 30000

dev[6,2] 0.810 1.140 0.001 0.369 4.086 FALSE 1.000 1.000 29447

dev[7,2] 0.801 1.140 0.001 0.360 4.057 FALSE 1.000 1.002 7108

dev[8,2] 2.472 1.691 0.191 2.146 6.578 FALSE 1.000 1.000 20279

dev[9,2] 0.966 1.335 0.001 0.447 4.817 FALSE 1.000 1.000 30000

dev[10,2] 0.929 1.319 0.001 0.425 4.622 FALSE 1.000 1.002 9566

dev[11,2] 1.602 1.939 0.002 0.911 6.909 FALSE 1.000 1.000 21544

dev[12,2] 0.909 1.275 0.001 0.416 4.588 FALSE 1.000 1.000 30000

dev[13,2] 0.851 1.206 0.001 0.392 4.281 FALSE 1.000 1.000 30000

dev[14,2] 0.766 1.096 0.001 0.351 3.830 FALSE 1.000 1.001 30000

dev[15,2] 0.679 0.872 0.001 0.360 3.089 FALSE 1.000 1.000 8962

dev[16,2] 0.658 0.936 0.001 0.300 3.337 FALSE 1.000 1.000 30000

dev[17,2] 1.247 1.716 0.001 0.588 6.189 FALSE 1.000 1.000 30000

dev[18,2] 0.819 1.150 0.001 0.380 4.140 FALSE 1.000 1.000 30000

dev[19,2] 0.907 1.289 0.001 0.415 4.564 FALSE 1.000 1.000 30000

dev[20,2] 1.361 1.833 0.001 0.646 6.522 FALSE 1.000 1.001 3547

dev[21,2] 0.991 1.387 0.001 0.460 4.915 FALSE 1.000 1.000 30000

dev[22,2] 0.941 1.329 0.001 0.422 4.780 FALSE 1.000 1.000 10390

dev[1,3] 1.188 1.375 0.002 0.718 4.888 FALSE 1.000 1.000 30000

dev[5,3] 0.831 1.135 0.001 0.392 4.057 FALSE 1.000 1.000 30000

dev[15,3] 0.935 1.227 0.001 0.476 4.349 FALSE 1.000 1.000 14564

dev[16,3] 1.366 1.614 0.002 0.783 5.783 FALSE 1.000 1.000 26183

deviance 374.075 9.932 356.206 373.459 395.160 FALSE 1.000 1.001 4058

Successful convergence based on Rhat values (all < 1.1).

Rhat is the potential scale reduction factor (at convergence, Rhat=1).

For each parameter, n.eff is a crude measure of effective sample size.

overlap0 checks if 0 falls in the parameter's 95% credible interval.

f is the proportion of the posterior with the same sign as the mean;

i.e., our confidence that the parameter is positive or negative.

DIC info: (pD = var(deviance)/2)

pD = 49.3 and DIC = 423.38

DIC is an estimate of expected predictive error (lower is better).

> # mit jags-Funktion keine weiterführende Diagnostik möglich: das Objekt jags.m zeigt nur eine Liste von 24 Elementen an.

> # ebenso wenig mit jags.basic

>

>

>

> # Versuch mit jagsbasic

> jagsbasic.m <- jags.basic(data=dat,

+ inits=NULL,

+ parameters.to.save=params,

+ model.file="Diabetes\_Random.txt",

+ n.chains=3,

+ n.iter=20000, n.burnin=10000)

Processing function input.......

Done.

Compiling model graph

Resolving undeclared variables

Allocating nodes

Graph information:

Observed stochastic nodes: 48

Unobserved stochastic nodes: 55

Total graph size: 1145

Initializing model

Adaptive phase.....

Adaptive phase complete

Burn-in phase, 10000 iterations x 3 chains

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

Sampling from joint posterior, 10000 iterations x 3 chains

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

MCMC took 0.354 minutes.

Warning messages:

1: In process.input(data, parameters.to.save, inits, n.chains, n.iter, :

Suppling a list of character strings to the data argument will be deprecated in the next version

2: In process.input(data, parameters.to.save, inits, n.chains, n.iter, :

Suppling a character vector to the data argument will be deprecated in the next version

>

> jagsbasic.m\_II <- do.call(rbind.data.frame, jagsbasic.m)

>

>

>

> #### Berechnung Median

>

> # Median von T1

> median(jagsbasic.m\_II$`T[1]`)

[1] 0.04386464

>

> # Median von T2

> median(jagsbasic.m\_II$`T[2]`)

[1] 0.03313067

>

> # Median von T3

> median(jagsbasic.m\_II$`T[3]`)

[1] 0.04092542

>

> # Median von T4

> median(jagsbasic.m\_II$`T[4]`)

[1] 0.03462993

>

> # Median von T5

> median(jagsbasic.m\_II$`T[5]`)

[1] 0.02962521

>

> # Median von T6

> median(jagsbasic.m\_II$`T[6]`)

[1] 0.02768601

>

> # Median von d2

> median(jagsbasic.m\_II$`d[2]`)

[1] -0.2851851

>

> # Median von d3

> median(jagsbasic.m\_II$`d[3]`)

[1] -0.07274048

>

> # Median von d4

> median(jagsbasic.m\_II$`d[4]`)

[1] -0.239737

>

> # Median von d5

> median(jagsbasic.m\_II$`d[5]`)

[1] -0.3977464

>

> # Median von d6

> median(jagsbasic.m\_II$`d[6]`)

[1] -0.4688324

>

> # Median von sd

> median(jagsbasic.m\_II$`sd`)

[1] 0.1225618

>

>

>

>

>

>

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