> ########## ########## Simulation Diabetes Beispiel mit Random Effects mit dem R2jags package

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

>

>

>

>

>

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

>

>

>

>

> ##### Clear data

> rm(list=ls())

>

>

>

> ##### load libraries

> library(rjags) # R2jags benötigt rjags

> library(R2jags)

> library(random)

> #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 R2jags/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, n.chains = 3, n.iter = 20000, n.burnin = 10000,

+ model.file="Diabetes\_Random.txt", DIC=TRUE, jags.module = c("glm","dic") )

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%

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

> # zusätzlich noch mehrere Argumente standardmäßig dabei, v.a. interessant: DIC, jags.module

>

>

>

> #### optional, falls nicht konvergiert:

> #jags.m.upd <- autojags(jags.m)

>

>

>

>

>

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

>

>

>

>

> print(jags.m)

Inference for Bugs model at "Diabetes\_Random.txt", fit using jags,

3 chains, each with 20000 iterations (first 10000 discarded), n.thin = 10

n.sims = 3000 iterations saved

mu.vect sd.vect 2.5% 25% 50% 75% 97.5% Rhat n.eff

T[1] 0.063 0.066 0.007 0.023 0.043 0.078 0.250 1.001 2100

T[2] 0.049 0.052 0.005 0.017 0.032 0.059 0.192 1.001 2200

T[3] 0.060 0.063 0.007 0.021 0.040 0.074 0.236 1.002 2000

T[4] 0.051 0.055 0.006 0.018 0.034 0.063 0.203 1.002 1900

T[5] 0.044 0.048 0.005 0.015 0.029 0.054 0.175 1.001 2100

T[6] 0.041 0.045 0.004 0.014 0.027 0.050 0.166 1.002 1700

d[2] -0.288 0.089 -0.473 -0.343 -0.286 -0.228 -0.123 1.001 3000

d[3] -0.074 0.087 -0.246 -0.131 -0.073 -0.014 0.091 1.002 1400

d[4] -0.239 0.085 -0.410 -0.293 -0.238 -0.186 -0.076 1.002 1500

d[5] -0.401 0.084 -0.574 -0.451 -0.400 -0.345 -0.248 1.001 3000

d[6] -0.473 0.108 -0.696 -0.542 -0.467 -0.402 -0.269 1.002 1300

dev[1,1] 2.309 2.136 0.009 0.620 1.779 3.390 7.725 1.001 3000

dev[2,1] 0.746 1.046 0.001 0.074 0.350 1.021 3.694 1.001 3000

dev[3,1] 0.958 1.333 0.001 0.098 0.453 1.273 4.832 1.002 1300

dev[4,1] 1.034 1.458 0.001 0.104 0.501 1.366 5.007 1.001 3000

dev[5,1] 0.814 1.139 0.001 0.081 0.376 1.080 4.106 1.001 3000

dev[6,1] 0.878 1.243 0.001 0.089 0.397 1.184 4.124 1.001 3000

dev[7,1] 0.827 1.141 0.001 0.082 0.392 1.111 3.795 1.001 3000

dev[8,1] 1.808 2.049 0.002 0.322 1.134 2.621 7.314 1.004 790

dev[9,1] 0.890 1.255 0.001 0.098 0.411 1.153 4.512 1.001 3000

dev[10,1] 0.924 1.310 0.001 0.093 0.393 1.212 4.669 1.001 3000

dev[11,1] 1.513 1.864 0.001 0.179 0.839 2.195 6.603 1.001 2800

dev[12,1] 0.948 1.375 0.001 0.088 0.435 1.236 4.860 1.001 3000

dev[13,1] 0.862 1.193 0.001 0.085 0.393 1.158 4.208 1.002 1800

dev[14,1] 0.781 1.127 0.001 0.077 0.334 1.046 3.947 1.001 3000

dev[15,1] 0.725 1.038 0.001 0.073 0.330 0.964 3.781 1.002 1400

dev[16,1] 1.268 1.566 0.002 0.158 0.684 1.798 5.663 1.002 1600

dev[17,1] 1.179 1.618 0.001 0.122 0.539 1.587 5.958 1.003 990

dev[18,1] 0.869 1.246 0.001 0.087 0.391 1.153 4.215 1.001 3000

dev[19,1] 0.910 1.289 0.001 0.089 0.395 1.191 4.683 1.001 2400

dev[20,1] 1.299 1.774 0.001 0.142 0.640 1.743 6.196 1.001 3000

dev[21,1] 0.970 1.388 0.001 0.093 0.453 1.254 5.015 1.001 3000

dev[22,1] 0.972 1.350 0.001 0.103 0.445 1.296 4.965 1.001 3000

dev[1,2] 4.155 3.091 0.103 1.779 3.549 5.809 11.476 1.001 2400

dev[2,2] 0.587 0.842 0.001 0.057 0.260 0.766 2.973 1.001 3000

dev[3,2] 0.939 1.340 0.001 0.100 0.425 1.233 4.972 1.003 1200

dev[4,2] 1.106 1.450 0.001 0.125 0.545 1.483 5.194 1.001 3000

dev[5,2] 1.036 1.402 0.001 0.110 0.463 1.416 5.131 1.002 2000

dev[6,2] 0.801 1.127 0.001 0.087 0.364 1.079 4.072 1.002 1200

dev[7,2] 0.786 1.137 0.001 0.089 0.360 1.001 4.041 1.002 1900

dev[8,2] 2.474 1.657 0.244 1.216 2.186 3.385 6.593 1.001 3000

dev[9,2] 0.934 1.377 0.001 0.083 0.440 1.227 4.644 1.001 3000

dev[10,2] 0.914 1.301 0.001 0.091 0.417 1.200 4.528 1.001 3000

dev[11,2] 1.588 1.938 0.003 0.223 0.906 2.270 6.776 1.002 3000

dev[12,2] 0.858 1.191 0.001 0.093 0.399 1.148 4.053 1.001 3000

dev[13,2] 0.819 1.184 0.001 0.082 0.363 1.028 4.472 1.001 3000

dev[14,2] 0.748 1.077 0.001 0.078 0.346 0.955 3.844 1.003 1100

dev[15,2] 0.674 0.832 0.001 0.088 0.363 0.961 3.006 1.001 3000

dev[16,2] 0.657 0.934 0.001 0.062 0.286 0.873 3.364 1.001 3000

dev[17,2] 1.266 1.716 0.002 0.145 0.615 1.725 6.099 1.001 3000

dev[18,2] 0.841 1.142 0.001 0.086 0.404 1.159 4.061 1.001 3000

dev[19,2] 0.863 1.246 0.001 0.085 0.397 1.140 4.444 1.001 2800

dev[20,2] 1.312 1.790 0.001 0.138 0.610 1.810 6.552 1.001 3000

dev[21,2] 0.970 1.352 0.001 0.091 0.426 1.311 4.737 1.001 3000

dev[22,2] 0.920 1.287 0.001 0.101 0.432 1.239 4.575 1.001 3000

dev[1,3] 1.164 1.318 0.003 0.174 0.749 1.677 4.841 1.004 3000

dev[5,3] 0.835 1.148 0.001 0.086 0.385 1.134 4.131 1.001 3000

dev[15,3] 0.902 1.180 0.001 0.103 0.455 1.257 4.286 1.002 1400

dev[16,3] 1.345 1.595 0.003 0.181 0.775 1.916 5.741 1.001 3000

rhat[1,1] 35.305 4.410 27.455 32.194 34.972 38.197 44.581 1.001 3000

rhat[2,1] 27.947 4.211 20.449 25.043 27.745 30.578 37.227 1.001 3000

rhat[3,1] 144.482 10.429 124.813 137.311 144.151 151.486 165.720 1.003 900

rhat[4,1] 79.724 7.827 65.395 74.341 79.483 84.885 95.384 1.001 2500

rhat[5,1] 303.322 15.328 274.419 292.834 303.201 313.602 334.315 1.001 2200

rhat[6,1] 174.912 11.875 152.833 166.870 174.562 182.742 199.048 1.001 2400

rhat[7,1] 200.903 12.420 177.759 192.260 200.654 209.197 225.275 1.001 3000

rhat[8,1] 5.611 1.827 2.605 4.297 5.410 6.692 9.804 1.002 1500

rhat[9,1] 157.830 11.133 136.477 150.210 157.853 165.014 180.515 1.002 1800

rhat[10,1] 490.611 19.170 453.061 477.866 490.338 503.095 528.410 1.001 3000

rhat[11,1] 145.742 10.883 125.829 137.878 145.282 152.809 168.026 1.001 3000

rhat[12,1] 395.876 17.908 362.152 383.550 395.497 407.215 432.106 1.001 3000

rhat[13,1] 200.718 12.553 176.634 192.012 200.547 208.857 225.882 1.001 3000

rhat[14,1] 114.159 9.105 96.913 108.049 114.041 120.122 132.808 1.001 3000

rhat[15,1] 68.207 6.083 56.622 64.034 68.064 72.209 80.480 1.001 2800

rhat[16,1] 104.610 8.422 88.295 98.788 104.374 110.229 121.563 1.001 2300

rhat[17,1] 787.355 26.089 736.789 769.325 787.335 804.686 839.198 1.001 2500

rhat[18,1] 251.493 14.406 223.796 241.686 251.560 260.952 280.281 1.002 1300

rhat[19,1] 665.135 23.572 618.751 649.367 664.606 680.428 713.809 1.001 3000

rhat[20,1] 390.915 18.933 353.517 378.137 391.031 403.357 428.266 1.001 3000

rhat[21,1] 323.738 16.656 292.403 312.289 323.418 334.652 357.401 1.001 2500

rhat[22,1] 844.481 26.135 793.155 826.771 844.777 862.230 895.686 1.001 3000

rhat[1,2] 46.210 5.660 35.549 42.301 46.070 49.872 57.440 1.002 1800

rhat[2,2] 21.187 3.298 15.162 18.963 21.047 23.340 27.985 1.001 3000

rhat[3,2] 113.877 8.925 96.575 107.954 113.669 119.523 132.543 1.001 3000

rhat[4,2] 81.035 7.737 66.876 75.802 80.794 85.963 97.198 1.001 3000

rhat[5,2] 148.377 10.618 128.354 141.184 148.131 155.420 170.227 1.001 3000

rhat[6,2] 136.848 10.173 117.832 129.980 136.574 143.683 157.728 1.001 3000

rhat[7,2] 137.304 10.044 117.746 130.367 137.433 144.209 156.964 1.001 3000

rhat[8,2] 3.377 1.146 1.571 2.530 3.262 4.072 6.027 1.001 3000

rhat[9,2] 173.375 11.852 151.436 165.082 173.315 180.879 196.795 1.001 3000

rhat[10,2] 447.135 18.272 411.200 434.611 447.303 459.561 484.005 1.001 3000

rhat[11,2] 111.460 9.209 93.725 105.178 111.279 117.534 129.933 1.001 2500

rhat[12,2] 337.854 15.962 307.183 326.923 337.582 348.666 369.813 1.001 3000

rhat[13,2] 164.212 11.204 142.548 156.690 164.009 171.646 187.477 1.001 3000

rhat[14,2] 93.774 8.197 78.544 88.145 93.487 99.227 110.570 1.002 1500

rhat[15,2] 29.556 3.244 23.728 27.286 29.334 31.621 36.401 1.001 3000

rhat[16,2] 94.541 7.672 80.513 89.359 94.253 99.345 110.660 1.001 3000

rhat[17,2] 579.531 22.907 533.936 564.389 579.937 595.136 624.173 1.001 2700

rhat[18,2] 215.526 13.156 191.140 206.291 215.215 224.250 242.053 1.002 1300

rhat[19,2] 568.497 21.369 528.472 553.730 568.484 582.536 610.840 1.001 3000

rhat[20,2] 326.671 16.933 293.465 314.783 326.618 337.868 360.900 1.001 3000

rhat[21,2] 237.532 13.921 211.548 228.063 237.401 246.414 266.709 1.001 2200

rhat[22,2] 689.699 23.421 644.236 673.258 689.539 705.377 736.110 1.001 3000

rhat[1,3] 32.582 4.012 25.316 29.797 32.209 35.103 40.948 1.001 3000

rhat[5,3] 122.737 9.355 104.704 116.388 122.617 129.048 141.391 1.001 3000

rhat[15,3] 48.889 4.998 39.446 45.444 48.744 52.252 59.204 1.001 3000

rhat[16,3] 85.810 7.475 72.198 80.673 85.438 90.631 101.440 1.001 3000

sd 0.129 0.042 0.057 0.099 0.124 0.155 0.225 1.001 3000

totresdev 52.975 9.968 35.300 45.966 52.323 59.272 74.231 1.001 2000

deviance 373.461 9.968 355.785 366.452 372.808 379.758 394.716 1.002 2000

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

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

DIC info (using the rule, pD = var(deviance)/2)

pD = 49.7 and DIC = 423.1

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

>

>

>

> #### Median

> jags.m[["BUGSoutput"]][["median"]]

$T

[1] 0.04282274 0.03208214 0.03953010 0.03365302 0.02881554 0.02662009

$d

[1] -0.2864384 -0.0730349 -0.2384141 -0.4001148 -0.4670184

$dev

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

[1,] 1.7789029 3.5486409 0.7492199

[2,] 0.3498753 0.2601412 0.3845998

[3,] 0.4527489 0.4250459 0.4547986

[4,] 0.5012999 0.5449422 0.7753714

[5,] 0.3757843 0.4625025 1.7789029

[6,] 0.3966181 0.3643267 0.3498753

[7,] 0.3923679 0.3601963 0.4527489

[8,] 1.1340167 2.1860338 0.5012999

[9,] 0.4108558 0.4396744 0.3757843

[10,] 0.3933144 0.4169558 0.3966181

[11,] 0.8391794 0.9059275 0.3923679

[12,] 0.4350774 0.3986960 1.1340167

[13,] 0.3930996 0.3632575 0.4108558

[14,] 0.3341636 0.3457615 0.3933144

[15,] 0.3298687 0.3632674 0.8391794

[16,] 0.6835854 0.2857257 0.4350774

[17,] 0.5392852 0.6154301 0.3930996

[18,] 0.3907100 0.4039077 0.3341636

[19,] 0.3946105 0.3971773 0.3298687

[20,] 0.6400298 0.6103647 0.6835854

[21,] 0.4531382 0.4259633 0.5392852

[22,] 0.4452840 0.4318182 0.3907100

$deviance

[1] 372.8085

$rhat

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

[1,] 34.972081 46.070062 32.208731

[2,] 27.744828 21.047403 122.616512

[3,] 144.150960 113.669121 48.744153

[4,] 79.483365 80.794114 85.438403

[5,] 303.201209 148.131086 34.972081

[6,] 174.561602 136.573877 27.744828

[7,] 200.653837 137.433162 144.150960

[8,] 5.410201 3.262192 79.483365

[9,] 157.853380 173.314658 303.201209

[10,] 490.338486 447.303464 174.561602

[11,] 145.282439 111.278940 200.653837

[12,] 395.496990 337.582041 5.410201

[13,] 200.547352 164.009208 157.853380

[14,] 114.040577 93.487494 490.338486

[15,] 68.063728 29.334014 145.282439

[16,] 104.373989 94.252603 395.496990

[17,] 787.334920 579.936922 200.547352

[18,] 251.559741 215.214859 114.040577

[19,] 664.605732 568.483769 68.063728

[20,] 391.031053 326.617983 104.373989

[21,] 323.417895 237.401453 787.334920

[22,] 844.777195 689.538735 251.559741

$sd

[1] 0.1244244

$totresdev

[1] 52.32283

>

>

>

>

>

> # nachträgliche Berechnung von pD --------------------------------------

>

>

>

>

> #jags.m\_levPlot[["BUGSoutput"]][["summary"]]

> out\_lePlo <- capture.output( jags.m[["BUGSoutput"]][["summary"]])

> cat("Hilf\_pD", out\_lePlo, file="Hilf.txt", sep="\n", append=TRUE)

>

>

>

> Hilf\_data = read.table("Hilf.txt", sep = "", header=F, skip=13, nrows=48)

> #Hilf\_data

> Hilf\_dev <- Hilf\_data[1:21,2]

> #Hilf\_dev

>

>

>

> # manuelle Berechnung von pD

> #dev ist Std-Abweichung jedes einzelnen Werts

> # insg 48 Werte

> Var\_manuell <- sum(Hilf\_dev)^2/48

> pD\_manuell <- Var\_manuell/2

> pD\_manuell

[1] 5.416967

>

>

>

>

>

> # zusätzliche Diagnostik, bei Bedarf aktivieren --------------------------------------------------------------------

>

>

>

>

> #pdf("DietaryFat\_Random\_trace.pdf")

> #plot(jags.m)

> #traceplot(jags.m)

> #dev.off()

>

>

>

> # Generate MCMC object for analysis

> #Anm: es scheint, dass die Zeile "jags.m.mcmc <- as.mcmc(jags.m) " manuell ausgeführt werden muss

> #jags.m.mcmc <- as.mcmc(jags.m)

> #jags.m.mcmc

>

>

>

> #pdf("jags.m.mcmc.autocorr.pdf") # Autocorrelation plot

> #autocorr.plot(jags.m.mcmc)

> #dev.off()

>

>

>

> # Other diagnostics using CODA:

> #gelman.plot(jags.m.mcmc)

> #geweke.diag(jags.m.mcmc)

> #geweke.plot(jags.m.mcmc)

> #raftery.diag(jags.m.mcmc)

> #heidel.diag(jags.m.mcmc)

>

>

>

>

>

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