> ########## ########## Simulation Diabetes Beispiel mit Fixed 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]", "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]" "totresdev" " dev"

[14] "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 )

> # .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 )

>

> 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 )

> #, .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

+ 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] + d[t[i,k]] - d[t[i,1]] # 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

+ }

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

+ d[1]<-0 # treatment effect is zero for reference treatment

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

+ A ~ dnorm(-4.2,1.11)

+ for (k in 1:nt) {

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

+ }

+ } ",

+ file="Diabetes\_Fixed.txt")

>

>

>

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

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

+ inits=NULL,

+ parameters.to.save=params,

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

+ n.chains=3,

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

+ store.data=TRUE)

Processing function input.......

Done.

Compiling model graph

Resolving undeclared variables

Allocating nodes

Graph information:

Observed stochastic nodes: 48

Unobserved stochastic nodes: 28

Total graph size: 1062

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

>

> #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

> # => muss demnächst angepasst werden

>

>

>

>

>

>

> # 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\_Fixed.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 0.349 minutes at time 2019-08-27 11:21:38.

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

d[2] -0.249 0.056 -0.360 -0.248 -0.140 FALSE 1.000 1.001 3182

d[3] -0.059 0.056 -0.169 -0.059 0.052 TRUE 0.852 1.000 20038

d[4] -0.255 0.054 -0.360 -0.255 -0.149 FALSE 1.000 1.000 16444

d[5] -0.361 0.053 -0.465 -0.360 -0.257 FALSE 1.000 1.001 3554

d[6] -0.454 0.063 -0.577 -0.454 -0.330 FALSE 1.000 1.000 5989

T[1] 0.065 0.065 0.007 0.044 0.244 FALSE 1.000 1.000 30000

T[2] 0.051 0.053 0.005 0.035 0.195 FALSE 1.000 1.000 30000

T[3] 0.061 0.062 0.007 0.042 0.230 FALSE 1.000 1.000 30000

T[4] 0.051 0.053 0.005 0.035 0.195 FALSE 1.000 1.000 30000

T[5] 0.046 0.048 0.005 0.031 0.177 FALSE 1.000 1.000 30000

T[6] 0.042 0.045 0.004 0.028 0.162 FALSE 1.000 1.000 30000

totresdev 78.303 7.382 65.828 77.618 94.498 FALSE 1.000 1.000 30000

rhat[1,1] 32.002 3.148 26.233 31.870 38.437 FALSE 1.000 1.000 6245

rhat[2,1] 27.149 3.831 20.182 26.991 35.149 FALSE 1.000 1.000 30000

rhat[3,1] 146.330 9.386 128.473 146.157 165.302 FALSE 1.000 1.000 7676

rhat[4,1] 82.451 6.794 69.709 82.293 96.196 FALSE 1.000 1.000 30000

rhat[5,1] 305.653 14.153 278.876 305.511 334.355 FALSE 1.000 1.000 17006

rhat[6,1] 175.261 10.334 155.719 175.127 196.148 FALSE 1.000 1.000 19250

rhat[7,1] 199.023 11.241 177.719 198.755 221.697 FALSE 1.000 1.000 9411

rhat[8,1] 5.480 1.809 2.488 5.294 9.560 FALSE 1.000 1.000 20769

rhat[9,1] 166.451 9.773 147.978 166.202 186.251 FALSE 1.000 1.001 3656

rhat[10,1] 494.770 16.839 462.095 494.572 527.997 FALSE 1.000 1.000 15320

rhat[11,1] 136.548 8.729 120.065 136.317 154.369 FALSE 1.000 1.000 29438

rhat[12,1] 388.369 15.082 359.024 388.251 418.185 FALSE 1.000 1.000 6754

rhat[13,1] 200.779 11.133 179.594 200.560 223.030 FALSE 1.000 1.000 12917

rhat[14,1] 114.599 8.177 99.186 114.439 131.387 FALSE 1.000 1.000 30000

rhat[15,1] 67.377 5.222 57.432 67.273 77.781 FALSE 1.000 1.000 11013

rhat[16,1] 110.523 6.623 97.834 110.410 124.078 FALSE 1.000 1.000 30000

rhat[17,1] 745.147 21.707 703.163 744.865 788.206 FALSE 1.000 1.000 30000

rhat[18,1] 254.917 12.126 231.789 254.646 279.014 FALSE 1.000 1.000 30000

rhat[19,1] 674.282 20.355 635.027 673.930 715.178 FALSE 1.000 1.000 30000

rhat[20,1] 411.824 16.655 379.531 411.708 444.935 FALSE 1.000 1.000 28468

rhat[21,1] 332.342 14.669 304.085 332.065 361.791 FALSE 1.000 1.000 9258

rhat[22,1] 836.547 24.125 789.949 836.305 884.373 FALSE 1.000 1.000 6682

rhat[1,2] 51.249 4.875 42.190 51.089 61.198 FALSE 1.000 1.000 4530

rhat[2,2] 21.748 3.123 16.058 21.604 28.225 FALSE 1.000 1.000 30000

rhat[3,2] 111.574 7.536 97.293 111.426 126.927 FALSE 1.000 1.000 20253

rhat[4,2] 78.409 6.482 66.265 78.227 91.565 FALSE 1.000 1.000 30000

rhat[5,2] 139.228 7.353 125.215 139.076 154.188 FALSE 1.000 1.000 4879

rhat[6,2] 136.823 8.524 120.600 136.646 153.851 FALSE 1.000 1.000 14759

rhat[7,2] 139.050 8.494 122.794 138.835 156.075 FALSE 1.000 1.000 21475

rhat[8,2] 3.502 1.170 1.588 3.375 6.130 FALSE 1.000 1.000 10347

rhat[9,2] 164.476 9.687 145.865 164.305 183.991 FALSE 1.000 1.000 17400

rhat[10,2] 443.200 15.637 412.827 443.128 474.103 FALSE 1.000 1.000 6308

rhat[11,2] 120.446 7.786 105.526 120.279 136.240 FALSE 1.000 1.000 30000

rhat[12,2] 345.360 13.818 318.863 345.189 372.877 FALSE 1.000 1.000 13250

rhat[13,2] 164.286 9.522 146.111 164.086 183.475 FALSE 1.000 1.000 30000

rhat[14,2] 93.443 6.882 80.295 93.283 107.235 FALSE 1.000 1.000 30000

rhat[15,2] 28.068 2.251 23.801 28.020 32.584 FALSE 1.000 1.000 13032

rhat[16,2] 91.554 5.582 81.008 91.413 102.956 FALSE 1.000 1.000 30000

rhat[17,2] 621.371 19.014 584.745 621.093 659.491 FALSE 1.000 1.000 30000

rhat[18,2] 212.006 10.258 192.490 211.737 232.464 FALSE 1.000 1.000 30000

rhat[19,2] 559.931 17.934 525.602 559.670 595.747 FALSE 1.000 1.000 30000

rhat[20,2] 305.014 13.529 279.183 304.823 332.315 FALSE 1.000 1.000 10361

rhat[21,2] 229.338 11.186 207.778 229.151 251.922 FALSE 1.000 1.000 30000

rhat[22,2] 698.046 22.114 655.444 697.842 741.973 FALSE 1.000 1.000 30000

rhat[1,3] 30.790 3.000 25.281 30.679 36.989 FALSE 1.000 1.000 4674

rhat[5,3] 129.966 6.856 116.912 129.816 143.758 FALSE 1.000 1.001 2005

rhat[15,3] 51.629 4.232 43.628 51.541 60.167 FALSE 1.000 1.000 30000

rhat[16,3] 82.781 5.390 72.652 82.602 93.737 FALSE 1.000 1.000 8745

dev[1,1] 3.949 2.292 0.541 3.618 9.233 FALSE 1.000 1.001 5749

dev[2,1] 0.751 1.035 0.001 0.348 3.669 FALSE 1.000 1.000 30000

dev[3,1] 0.941 1.283 0.001 0.446 4.569 FALSE 1.000 1.000 13405

dev[4,1] 1.237 1.446 0.002 0.721 5.198 FALSE 1.000 1.000 28018

dev[5,1] 0.725 1.016 0.001 0.334 3.641 FALSE 1.000 1.000 30000

dev[6,1] 0.662 0.957 0.001 0.301 3.362 FALSE 1.000 1.000 20464

dev[7,1] 0.692 0.978 0.001 0.318 3.501 FALSE 1.000 1.000 30000

dev[8,1] 1.958 2.155 0.004 1.248 7.822 FALSE 1.000 1.001 5852

dev[9,1] 1.540 1.712 0.003 0.966 6.160 FALSE 1.000 1.001 2773

dev[10,1] 0.782 1.091 0.001 0.353 3.890 FALSE 1.000 1.000 30000

dev[11,1] 3.174 2.637 0.034 2.578 9.745 FALSE 1.000 1.000 30000

dev[12,1] 1.001 1.354 0.001 0.485 4.835 FALSE 1.000 1.001 3788

dev[13,1] 0.679 0.960 0.001 0.310 3.381 FALSE 1.000 1.001 11576

dev[14,1] 0.620 0.878 0.001 0.281 3.129 FALSE 1.000 1.001 6390

dev[15,1] 0.624 0.872 0.001 0.289 3.075 FALSE 1.000 1.001 10086

dev[16,1] 2.198 1.798 0.030 1.796 6.791 FALSE 1.000 1.000 30000

dev[17,1] 5.016 3.649 0.184 4.309 13.963 FALSE 1.000 1.000 22029

dev[18,1] 0.662 0.925 0.001 0.309 3.263 FALSE 1.000 1.000 30000

dev[19,1] 0.802 1.120 0.001 0.365 4.049 FALSE 1.000 1.000 30000

dev[20,1] 3.415 2.948 0.027 2.717 10.852 FALSE 1.000 1.001 16214

dev[21,1] 1.189 1.527 0.001 0.612 5.522 FALSE 1.000 1.001 3547

dev[22,1] 0.942 1.344 0.001 0.430 4.731 FALSE 1.000 1.000 7780

dev[1,2] 7.051 3.397 1.735 6.623 14.771 FALSE 1.000 1.000 4948

dev[2,2] 0.591 0.812 0.001 0.279 2.877 FALSE 1.000 1.001 30000

dev[3,2] 0.968 1.235 0.001 0.501 4.425 FALSE 1.000 1.001 4296

dev[4,2] 1.332 1.536 0.002 0.799 5.488 FALSE 1.000 1.000 30000

dev[5,2] 2.016 1.732 0.018 1.606 6.379 FALSE 1.000 1.000 5232

dev[6,2] 0.564 0.793 0.001 0.258 2.825 FALSE 1.000 1.000 12684

dev[7,2] 0.551 0.771 0.001 0.255 2.761 FALSE 1.000 1.000 26048

dev[8,2] 2.652 1.714 0.252 2.347 6.772 FALSE 1.000 1.000 11645

dev[9,2] 1.598 1.781 0.003 1.010 6.425 FALSE 1.000 1.000 30000

dev[10,2] 0.762 1.081 0.001 0.346 3.825 FALSE 1.000 1.001 15366

dev[11,2] 3.555 2.574 0.134 3.053 9.865 FALSE 1.000 1.000 30000

dev[12,2] 0.947 1.251 0.001 0.472 4.479 FALSE 1.000 1.000 30000

dev[13,2] 0.594 0.849 0.001 0.267 3.037 FALSE 1.000 1.000 30000

dev[14,2] 0.531 0.756 0.000 0.239 2.649 FALSE 1.000 1.001 7516

dev[15,2] 0.851 0.806 0.004 0.633 2.930 FALSE 1.000 1.000 9811

dev[16,2] 0.504 0.687 0.001 0.242 2.424 FALSE 1.000 1.000 14507

dev[17,2] 5.947 3.756 0.593 5.306 14.952 FALSE 1.000 1.000 30000

dev[18,2] 0.605 0.833 0.001 0.284 2.967 FALSE 1.000 1.000 18999

dev[19,2] 0.783 1.070 0.001 0.370 3.838 FALSE 1.000 1.000 30000

dev[20,2] 4.153 3.164 0.098 3.496 11.908 FALSE 1.000 1.000 11617

dev[21,2] 1.338 1.570 0.002 0.788 5.658 FALSE 1.000 1.000 30000

dev[22,2] 0.912 1.294 0.001 0.417 4.580 FALSE 1.000 1.000 30000

dev[1,3] 1.572 1.333 0.014 1.259 4.874 FALSE 1.000 1.001 4322

dev[5,3] 1.326 1.293 0.005 0.958 4.687 FALSE 1.000 1.001 2247

dev[15,3] 1.366 1.341 0.005 0.989 4.841 FALSE 1.000 1.000 30000

dev[16,3] 1.677 1.491 0.011 1.315 5.450 FALSE 1.000 1.000 9525

deviance 398.788 7.382 386.314 398.104 414.983 FALSE 1.000 1.000 30000

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 = 27.2 and DIC = 426.032

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\_Fixed.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: 28

Total graph size: 1062

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 0.339 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

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

>

>

>

> #### Berechnung Median

>

> # Median von T1

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

[1] 0.04432673

>

> # Median von T2

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

[1] 0.0345513

>

> # Median von T3

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

[1] 0.04170786

>

> # Median von T4

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

[1] 0.03438489

>

> # Median von T5

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

[1] 0.03100329

>

> # Median von T6

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

[1] 0.0282365

>

> # Median von d2

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

[1] -0.2487837

>

> # Median von d3

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

[1] -0.05932262

>

> # Median von d4

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

[1] -0.2564713

>

> # Median von d5

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

[1] -0.360495

>

> # Median von d6

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

[1] -0.4560547

>

>

>

>

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