> ########## ########## Simulation Diabetes Beispiel mit Fixed 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("DietaryFat\_Data.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("DietaryFat\_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="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 )

> # .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 )

> #, .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

+ 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, n.chains = 3, n.iter = 20000, n.burnin = 10000,

+ model.file="Diabetes\_Fixed.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: 28

Total graph size: 1062

Initializing model

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

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

>

> #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

>

>

>

>

>

>

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

>

>

>

> print(jags.m)

Inference for Bugs model at "Diabetes\_Fixed.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] 0.065 0.067 0.008 0.023 0.044 0.081 0.251 1.001 3000

T[2] 0.052 0.054 0.006 0.018 0.035 0.064 0.204 1.001 3000

T[3] 0.062 0.064 0.007 0.022 0.042 0.076 0.242 1.001 3000

T[4] 0.051 0.054 0.006 0.018 0.035 0.063 0.203 1.001 3000

T[5] 0.047 0.050 0.005 0.016 0.031 0.057 0.186 1.001 3000

T[6] 0.043 0.046 0.005 0.015 0.028 0.052 0.166 1.001 3000

d[2] -0.248 0.056 -0.358 -0.288 -0.247 -0.210 -0.139 1.001 3000

d[3] -0.058 0.056 -0.168 -0.097 -0.058 -0.020 0.049 1.001 3000

d[4] -0.255 0.053 -0.356 -0.291 -0.254 -0.219 -0.154 1.001 3000

d[5] -0.360 0.053 -0.462 -0.397 -0.359 -0.324 -0.255 1.001 3000

d[6] -0.455 0.063 -0.580 -0.498 -0.454 -0.411 -0.335 1.001 3000

dev[1,1] 3.966 2.301 0.496 2.264 3.660 5.289 9.409 1.005 3000

dev[2,1] 0.711 0.986 0.001 0.084 0.337 0.949 3.536 1.003 1400

dev[3,1] 0.950 1.252 0.001 0.107 0.491 1.295 4.318 1.004 3000

dev[4,1] 1.277 1.488 0.004 0.205 0.765 1.832 5.316 1.001 3000

dev[5,1] 0.699 0.995 0.001 0.069 0.309 0.943 3.587 1.001 3000

dev[6,1] 0.672 0.927 0.001 0.069 0.330 0.893 3.285 1.001 3000

dev[7,1] 0.718 1.010 0.001 0.072 0.316 0.928 3.607 1.001 3000

dev[8,1] 1.916 2.115 0.004 0.350 1.237 2.789 7.712 1.003 710

dev[9,1] 1.510 1.708 0.002 0.251 0.939 2.181 6.209 1.001 3000

dev[10,1] 0.759 1.050 0.001 0.081 0.352 1.016 3.756 1.001 3000

dev[11,1] 3.162 2.677 0.039 1.146 2.528 4.459 10.099 1.001 3000

dev[12,1] 1.009 1.367 0.001 0.107 0.480 1.352 4.997 1.001 3000

dev[13,1] 0.674 0.947 0.001 0.069 0.306 0.906 3.392 1.001 3000

dev[14,1] 0.617 0.910 0.001 0.062 0.284 0.793 3.104 1.003 3000

dev[15,1] 0.624 0.871 0.001 0.071 0.290 0.848 3.003 1.003 1800

dev[16,1] 2.184 1.850 0.026 0.790 1.716 3.151 6.853 1.003 1200

dev[17,1] 4.990 3.518 0.193 2.260 4.379 6.887 13.617 1.002 3000

dev[18,1] 0.660 0.910 0.001 0.070 0.318 0.879 3.191 1.002 2200

dev[19,1] 0.777 1.098 0.000 0.072 0.342 1.026 3.938 1.001 3000

dev[20,1] 3.363 2.929 0.032 1.171 2.645 4.792 11.051 1.004 3000

dev[21,1] 1.213 1.546 0.002 0.144 0.621 1.714 5.735 1.003 3000

dev[22,1] 0.927 1.310 0.001 0.092 0.412 1.241 4.611 1.002 1700

dev[1,2] 7.041 3.411 1.651 4.563 6.619 9.099 15.066 1.002 2500

dev[2,2] 0.619 0.893 0.000 0.059 0.272 0.807 3.050 1.003 800

dev[3,2] 0.953 1.220 0.001 0.112 0.500 1.331 4.351 1.001 3000

dev[4,2] 1.255 1.463 0.002 0.173 0.714 1.871 5.162 1.001 2500

dev[5,2] 2.006 1.706 0.023 0.704 1.626 2.850 6.345 1.006 1300

dev[6,2] 0.571 0.821 0.000 0.059 0.257 0.745 2.911 1.001 2500

dev[7,2] 0.573 0.795 0.001 0.058 0.260 0.782 2.926 1.001 3000

dev[8,2] 2.636 1.642 0.256 1.405 2.351 3.610 6.435 1.003 990

dev[9,2] 1.668 1.891 0.003 0.280 1.035 2.395 6.563 1.002 3000

dev[10,2] 0.762 1.088 0.001 0.083 0.342 0.993 3.762 1.000 3000

dev[11,2] 3.566 2.553 0.125 1.637 3.057 5.007 9.721 1.001 3000

dev[12,2] 0.968 1.266 0.001 0.114 0.482 1.318 4.527 1.001 3000

dev[13,2] 0.577 0.826 0.001 0.058 0.258 0.764 2.885 1.001 3000

dev[14,2] 0.535 0.752 0.000 0.057 0.243 0.703 2.649 1.001 3000

dev[15,2] 0.841 0.812 0.003 0.227 0.622 1.249 2.910 1.002 3000

dev[16,2] 0.523 0.687 0.001 0.061 0.252 0.733 2.539 1.001 3000

dev[17,2] 5.899 3.723 0.644 3.149 5.242 7.927 14.467 1.002 3000

dev[18,2] 0.599 0.826 0.001 0.062 0.285 0.791 2.967 1.001 3000

dev[19,2] 0.817 1.139 0.001 0.091 0.380 1.059 4.115 1.002 1100

dev[20,2] 4.168 3.195 0.148 1.748 3.467 5.900 12.115 1.001 3000

dev[21,2] 1.336 1.551 0.002 0.195 0.793 1.923 5.405 1.002 1700

dev[22,2] 0.883 1.241 0.001 0.089 0.420 1.161 4.422 1.002 1500

dev[1,3] 1.579 1.334 0.015 0.544 1.272 2.282 4.940 1.001 3000

dev[5,3] 1.350 1.312 0.005 0.341 0.983 1.976 4.747 1.001 3000

dev[15,3] 1.391 1.348 0.005 0.347 1.006 2.029 4.829 1.001 3000

dev[16,3] 1.695 1.505 0.014 0.534 1.317 2.466 5.440 1.004 1200

rhat[1,1] 31.977 3.137 26.100 29.814 31.812 34.017 38.627 1.001 3000

rhat[2,1] 27.342 3.833 20.457 24.727 27.047 29.754 35.355 1.002 1400

rhat[3,1] 146.415 9.388 128.155 139.998 146.481 152.623 164.697 1.001 3000

rhat[4,1] 82.760 6.705 70.412 78.293 82.659 87.140 96.455 1.001 3000

rhat[5,1] 305.683 13.885 279.539 296.240 305.605 314.993 333.608 1.001 3000

rhat[6,1] 175.267 10.430 155.602 167.913 174.829 182.364 195.953 1.001 3000

rhat[7,1] 198.829 11.426 176.813 191.116 198.747 206.373 222.392 1.001 3000

rhat[8,1] 5.482 1.764 2.513 4.198 5.295 6.583 9.487 1.003 740

rhat[9,1] 166.211 9.826 147.745 159.358 166.005 172.710 186.389 1.001 3000

rhat[10,1] 494.331 16.698 462.036 482.878 494.297 505.435 527.544 1.001 3000

rhat[11,1] 136.592 8.730 119.488 130.746 136.492 142.388 153.980 1.001 3000

rhat[12,1] 388.513 15.279 358.379 378.408 388.588 398.377 418.574 1.001 3000

rhat[13,1] 200.722 11.102 179.877 193.176 200.529 207.802 223.161 1.001 3000

rhat[14,1] 114.614 8.144 99.239 109.022 114.534 120.154 130.531 1.001 3000

rhat[15,1] 67.466 5.277 57.695 63.768 67.293 71.058 77.845 1.001 3000

rhat[16,1] 110.432 6.719 98.049 105.770 110.095 114.996 124.212 1.002 1600

rhat[17,1] 745.103 21.247 704.311 730.935 744.438 759.571 787.702 1.001 3000

rhat[18,1] 254.954 12.090 232.416 246.832 255.052 263.066 278.791 1.002 2000

rhat[19,1] 673.415 20.339 635.078 659.786 672.679 687.091 713.996 1.002 1900

rhat[20,1] 411.516 16.641 379.677 400.626 411.283 422.458 445.556 1.001 2300

rhat[21,1] 332.855 14.509 305.554 322.834 332.648 342.598 362.686 1.003 940

rhat[22,1] 836.989 24.066 790.539 820.891 837.006 853.070 884.126 1.001 3000

rhat[1,2] 51.229 4.903 41.977 47.869 51.083 54.487 61.520 1.001 3000

rhat[2,2] 21.915 3.152 16.230 19.776 21.739 23.823 28.434 1.003 960

rhat[3,2] 111.674 7.522 97.435 106.549 111.393 116.760 127.047 1.001 3000

rhat[4,2] 78.715 6.416 66.811 74.134 78.703 82.924 91.562 1.001 2400

rhat[5,2] 139.233 7.288 125.286 134.333 138.983 144.019 153.589 1.002 2500

rhat[6,2] 136.817 8.581 120.455 130.864 136.616 142.608 154.035 1.002 3000

rhat[7,2] 138.987 8.663 122.452 133.110 138.756 144.940 156.374 1.001 3000

rhat[8,2] 3.497 1.127 1.592 2.682 3.378 4.217 5.935 1.003 820

rhat[9,2] 164.179 9.828 145.554 157.527 164.197 170.790 183.607 1.001 3000

rhat[10,2] 442.877 15.505 413.138 432.848 442.639 453.355 473.328 1.001 3000

rhat[11,2] 120.502 7.729 105.389 115.198 120.292 125.754 135.965 1.001 3000

rhat[12,2] 345.547 13.911 318.379 336.308 345.359 354.531 373.038 1.001 3000

rhat[13,2] 164.027 9.404 145.852 157.636 163.592 170.423 183.050 1.001 3000

rhat[14,2] 93.339 6.920 80.455 88.632 93.013 97.950 107.335 1.001 3000

rhat[15,2] 28.099 2.248 23.826 26.489 28.053 29.591 32.421 1.001 3000

rhat[16,2] 91.461 5.667 80.671 87.432 91.292 95.297 102.799 1.002 1600

rhat[17,2] 621.216 18.725 585.512 608.407 620.754 633.518 657.912 1.001 3000

rhat[18,2] 211.993 10.201 193.025 204.825 211.865 218.692 232.850 1.002 1900

rhat[19,2] 559.098 17.966 524.246 547.023 558.706 571.060 593.658 1.001 2400

rhat[20,2] 304.896 13.398 278.710 295.588 304.952 314.038 330.616 1.001 2100

rhat[21,2] 229.462 11.314 208.516 221.767 229.176 237.003 252.724 1.003 810

rhat[22,2] 697.831 21.785 656.150 682.830 697.397 712.327 741.067 1.001 3000

rhat[1,3] 30.770 2.995 25.212 28.666 30.648 32.761 37.066 1.001 3000

rhat[5,3] 130.055 6.944 116.551 125.243 129.963 134.734 143.925 1.002 2600

rhat[15,3] 51.715 4.234 43.841 48.765 51.600 54.534 60.147 1.001 3000

rhat[16,3] 82.743 5.442 72.669 78.958 82.593 86.310 93.967 1.001 2000

totresdev 78.190 7.473 65.567 72.656 77.403 82.767 94.765 1.001 3000

deviance 398.675 7.473 386.053 393.142 397.889 403.253 415.250 1.001 3000

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 = 27.9 and DIC = 426.6

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

>

>

>

> #### Median

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

$T

[1] 0.04425101 0.03487106 0.04199911 0.03461548 0.03108007 0.02846137

$d

[1] -0.24698242 -0.05820739 -0.25421214 -0.35923001 -0.45422140

$dev

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

[1,] 3.6602991 6.6185623 1.2718227

[2,] 0.3365512 0.2721563 0.9832107

[3,] 0.4905089 0.5000151 1.0063830

[4,] 0.7646208 0.7143010 1.3168828

[5,] 0.3087828 1.6263061 3.6602991

[6,] 0.3300914 0.2567848 0.3365512

[7,] 0.3163748 0.2600370 0.4905089

[8,] 1.2372635 2.3507409 0.7646208

[9,] 0.9393786 1.0347005 0.3087828

[10,] 0.3521914 0.3422449 0.3300914

[11,] 2.5282231 3.0573240 0.3163748

[12,] 0.4797581 0.4821878 1.2372635

[13,] 0.3056906 0.2582139 0.9393786

[14,] 0.2835036 0.2425589 0.3521914

[15,] 0.2904234 0.6217423 2.5282231

[16,] 1.7157416 0.2524165 0.4797581

[17,] 4.3789209 5.2419154 0.3056906

[18,] 0.3178299 0.2854792 0.2835036

[19,] 0.3415282 0.3797201 0.2904234

[20,] 2.6448323 3.4668897 1.7157416

[21,] 0.6208470 0.7926414 4.3789209

[22,] 0.4124969 0.4204343 0.3178299

$deviance

[1] 397.8889

$rhat

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

[1,] 31.81213 51.08320 30.64765

[2,] 27.04707 21.73919 129.96340

[3,] 146.48112 111.39287 51.60042

[4,] 82.65935 78.70265 82.59328

[5,] 305.60475 138.98293 31.81213

[6,] 174.82947 136.61647 27.04707

[7,] 198.74686 138.75648 146.48112

[8,] 5.29494 3.37805 82.65935

[9,] 166.00461 164.19674 305.60475

[10,] 494.29660 442.63943 174.82947

[11,] 136.49236 120.29230 198.74686

[12,] 388.58764 345.35932 5.29494

[13,] 200.52916 163.59158 166.00461

[14,] 114.53388 93.01315 494.29660

[15,] 67.29311 28.05341 136.49236

[16,] 110.09499 91.29152 388.58764

[17,] 744.43838 620.75372 200.52916

[18,] 255.05184 211.86493 114.53388

[19,] 672.67863 558.70611 67.29311

[20,] 411.28301 304.95183 110.09499

[21,] 332.64779 229.17593 744.43838

[22,] 837.00555 697.39695 255.05184

$totresdev

[1] 77.40323

>

>

>

>

>

> # 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=5, nrows=21)

> #Hilf\_data

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

> #Hilf\_dev

>

>

>

> # manuelle Berechnung von pD

> #dev ist Std-Abweichung jedes einzelnen Werts

> # insg 21 Werte

> Var\_manuell <- sum(Hilf\_dev)^2/21

> pD\_manuell <- Var\_manuell/2

> pD\_manuell

[1] 4.281322

>

>

>

>

>

> # 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 ########## ########## ##########