########## ########## Simulation Dietary Fat 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/DietaryFat")

>

>

>

> ##### Read the data into R.

> #data = read.table("DietaryFat\_Data.txt", sep = "", header=F)

> data = as.matrix(read.table("DietaryFat\_Data.txt", sep = "", header=T))

> head(data) # Shows the first six entries

t..1. t..2. t..3. E..1. E..2. E..3. r..1. r..2. r..3. na..

[1,] 1 2 NA 1917.0 1925.0 NA 113 111 NA 2

[2,] 1 2 2 43.6 41.3 38 1 5 3 3

[3,] 1 2 NA 393.5 373.9 NA 24 20 NA 2

[4,] 1 2 NA 4715.0 4823.0 NA 248 269 NA 2

[5,] 1 2 NA 715.0 751.0 NA 31 28 NA 2

[6,] 1 2 NA 885.0 895.0 NA 65 48 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

> nt <- ncol(data[,7:9])

> #nt # check

> na <- data[,10]

> #na # check

> r <- data[,7:9]

> #r # Check

> E <- data[,4:6]

> #E # Check

> t <- data[,1:3]

> #t # Check

>

>

>

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

> #dat

>

>

>

> ##### Parameter to monitor/save

> params <- c("d[2]", "T[1]", "T[2]", "sd" , "dev", "totresdev", "theta" )

> params

[1] "d[2]" "T[1]" "T[2]" "sd" "dev" "totresdev" "theta"

>

>

>

> ##### read in inits with chains

> inits1 <- list(d=c( NA, 0, 0),

+ sd=1,

+ mu=c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

+ .RNG.name="lecuyer::RngStream", .RNG.seed=1)

>

> inits2 <- list(d=c( NA, -1, -1),

+ sd=4,

+ mu=c(-3, -3, -3, -3, -3, -3, -3, -3, -3, -3),

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

>

> inits3 <- list(d=c( NA, 2, 2),

+ sd=2,

+ mu=c(-3, 5, -1, -3, 7, -3, -4, -3, -3, 0),

+ .RNG.name="lecuyer::RngStream", .RNG.seed=3 )

>

> all.inits <- list(inits1, inits2, 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] ~ dpois(theta[i,k]) # Poisson likelihood

+ theta[i,k] <- lambda[i,k]\*E[i,k] # failure rate \* exposure

+ log(lambda[i,k]) <- mu[i] + delta[i,k] # model for linear predictor

+ dev[i,k] <- 2\*((theta[i,k]-r[i,k]) + r[i,k]\*log(r[i,k]/theta[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 trial correction

+ taud[i,k] <- tau \*2\*(k-1)/k # precision of LOR distributions (with multi-arm trial 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 # treatment effect is zero for reference treatment

+ 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)

+

+ # zusätzlich eingefügt

+ A ~ dnorm(-3,1.77)

+ for (k in 1:nt) { log(T[k]) <- A + d[k] }

+ } ",

+ file="DietaryFat\_Random.txt")

>

>

>

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

> jags.m <- jags(data=dat, inits=all.inits, parameters.to.save=params, n.chains = 3, n.iter = 110000, n.burnin = 90000,

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

Compiling model graph

Resolving undeclared variables

Allocating nodes

Graph information:

Observed stochastic nodes: 21

Unobserved stochastic nodes: 25

Total graph size: 363

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 "DietaryFat\_Random.txt", fit using jags,

3 chains, each with 110000 iterations (first 90000 discarded), n.thin = 20

n.sims = 3000 iterations saved

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

T[1] 0.067 0.057 0.012 0.030 0.049 0.085 0.226 1.001 3000

T[2] 0.066 0.057 0.011 0.029 0.049 0.085 0.222 1.001 3000

d[2] -0.013 0.090 -0.203 -0.064 -0.013 0.038 0.163 1.006 460

dev[1,1] 0.732 1.047 0.001 0.071 0.322 0.971 3.609 1.001 3000

dev[2,1] 2.112 1.523 0.106 0.976 1.811 2.925 5.960 1.003 1700

dev[3,1] 0.711 1.021 0.001 0.072 0.313 0.918 3.657 1.002 3000

dev[4,1] 0.873 1.257 0.001 0.088 0.386 1.157 4.635 1.001 3000

dev[5,1] 0.743 1.019 0.001 0.081 0.351 0.979 3.520 1.003 3000

dev[6,1] 1.453 1.708 0.003 0.208 0.846 2.071 6.030 1.002 2100

dev[7,1] 1.256 1.549 0.001 0.162 0.671 1.797 5.335 1.001 3000

dev[8,1] 1.548 1.712 0.004 0.282 0.973 2.204 6.088 1.001 3000

dev[9,1] 0.799 1.134 0.001 0.075 0.357 1.045 3.989 1.001 3000

dev[10,1] 0.825 1.115 0.001 0.093 0.393 1.118 3.877 1.001 3000

dev[1,2] 0.790 1.151 0.001 0.073 0.352 1.034 4.136 1.004 1600

dev[2,2] 1.619 1.542 0.006 0.432 1.189 2.350 5.474 1.002 3000

dev[3,2] 0.622 0.903 0.001 0.060 0.285 0.831 3.063 1.002 1400

dev[4,2] 0.992 1.352 0.001 0.101 0.443 1.345 4.813 1.004 2100

dev[5,2] 0.652 0.893 0.000 0.066 0.312 0.875 3.243 1.001 3000

dev[6,2] 1.389 1.612 0.003 0.190 0.796 2.025 5.582 1.001 3000

dev[7,2] 0.845 1.056 0.001 0.113 0.466 1.174 3.716 1.001 2600

dev[8,2] 1.630 1.814 0.003 0.276 1.009 2.385 6.524 1.003 910

dev[9,2] 0.797 1.175 0.001 0.085 0.349 1.000 4.210 1.001 3000

dev[10,2] 0.521 0.727 0.001 0.055 0.252 0.677 2.573 1.003 2500

dev[2,3] 0.394 0.547 0.000 0.040 0.172 0.532 2.013 1.001 3000

sd 0.131 0.118 0.006 0.049 0.099 0.182 0.416 1.012 380

theta[1,1] 112.762 9.075 95.391 106.512 112.559 118.669 131.574 1.002 2000

theta[2,1] 3.136 1.108 1.357 2.337 3.006 3.797 5.725 1.002 1500

theta[3,1] 22.794 3.736 15.997 20.248 22.643 25.193 30.686 1.001 3000

theta[4,1] 252.447 14.277 224.851 242.930 252.353 261.642 281.481 1.001 3000

theta[5,1] 29.397 4.292 21.844 26.458 29.098 32.074 38.789 1.001 3000

theta[6,1] 58.809 6.673 47.244 54.255 58.393 62.891 73.074 1.001 3000

theta[7,1] 1.981 0.998 0.544 1.245 1.827 2.527 4.284 1.001 3000

theta[8,1] 33.460 4.854 23.940 30.227 33.375 36.551 43.160 1.002 1700

theta[9,1] 174.977 11.592 152.950 167.090 174.512 182.147 198.415 1.001 3000

theta[10,1] 1.523 0.872 0.334 0.876 1.362 1.999 3.612 1.001 2500

theta[1,2] 111.446 9.385 93.759 104.954 111.262 117.516 130.838 1.002 1000

theta[2,2] 3.083 1.090 1.411 2.308 2.945 3.688 5.653 1.001 3000

theta[3,2] 20.938 3.554 14.513 18.526 20.724 23.222 28.352 1.003 1200

theta[4,2] 264.767 15.540 235.971 254.030 264.218 274.979 296.595 1.003 720

theta[5,2] 29.542 4.194 21.689 26.637 29.354 32.288 38.183 1.001 3000

theta[6,2] 54.124 6.328 41.635 49.873 54.016 58.476 66.275 1.001 3000

theta[7,2] 1.964 0.999 0.531 1.236 1.816 2.498 4.322 1.001 3000

theta[8,2] 33.524 4.773 25.146 30.181 33.161 36.530 43.838 1.003 860

theta[9,2] 175.383 11.780 152.755 167.280 175.340 182.901 199.135 1.002 1400

theta[10,2] 1.469 0.846 0.315 0.833 1.319 1.930 3.525 1.001 3000

theta[2,3] 2.782 0.995 1.232 2.069 2.657 3.344 5.086 1.001 3000

totresdev 21.302 5.147 12.676 17.649 20.852 24.452 32.486 1.001 3000

deviance 124.021 5.147 115.395 120.368 123.571 127.171 135.205 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 = 13.2 and DIC = 137.3

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

>

>

>

> #### Median

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

$T

[1] 0.04948979 0.04868040

$d

[1] -0.0127305

$dev

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

[1,] 0.3220599 0.3519793 0.1719948

[2,] 1.8109191 1.1887885 0.3220599

[3,] 0.3129285 0.2847771 1.8109191

[4,] 0.3858026 0.4432322 0.3129285

[5,] 0.3512029 0.3124095 0.3858026

[6,] 0.8464654 0.7957427 0.3512029

[7,] 0.6708064 0.4658452 0.8464654

[8,] 0.9725977 1.0089395 0.6708064

[9,] 0.3573132 0.3494290 0.9725977

[10,] 0.3932408 0.2522601 0.3573132

$deviance

[1] 123.5707

$sd

[1] 0.09924095

$theta

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

[1,] 112.558836 111.261684 2.656692

[2,] 3.006105 2.945450 112.558836

[3,] 22.642593 20.723936 3.006105

[4,] 252.352776 264.218283 22.642593

[5,] 29.098230 29.353568 252.352776

[6,] 58.393012 54.015899 29.098230

[7,] 1.827356 1.815582 58.393012

[8,] 33.374906 33.160646 1.827356

[9,] 174.512322 175.340002 33.374906

[10,] 1.361995 1.319057 174.512322

$totresdev

[1] 20.85181

>

>

>

>

>

> # 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] 10.82707

>

>

>

>

>

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