> ########## ########## Simulation Dietary Fat Beispiel mit Random Effects mit dem rjags package

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

>

>

>

>

>

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

>

>

>

> ##### Clear data

> rm(list=ls())

>

>

>

> ##### load libraries

> library(rjags)

> library(coda)

> library(random)

> library(matrixStats) # zusätzl Paket, berechnet Median

> 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 rjags/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=ns, nt=nt, na=na, r=r, E=E, t=t) # names list of numbers

> #dat

>

>

>

> ##### 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.model( file = "DietaryFat\_Random.txt", data=dat, inits=all.inits, n.chains=3, n.adapt=5000)

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%

>

>

>

> ##### Initialization

> update(jags.m, 100000) # burn in

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

>

>

>

> ##### run JAGS and save posterior samples

> samps\_coda <- coda.samples( jags.m, variable.names=c("d[2]", "T[1]", "T[2]", "sd"), n.iter=100000, DIC=T )

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

>

>

>

>

>

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

>

>

>

>

> ##### summarize posterior samples # funktioniert nur mit coda.samples

> summary(window(samps\_coda, start=100001)) # burnin = 100000

Iterations = 105001:205000

Thinning interval = 1

Number of chains = 3

Sample size per chain = 1e+05

1. Empirical mean and standard deviation for each variable,

plus standard error of the mean:

Mean SD Naive SE Time-series SE

T[1] 0.06581 0.05716 0.0001044 0.0001044

T[2] 0.06525 0.05727 0.0001046 0.0001061

d[2] -0.01260 0.08867 0.0001619 0.0006531

sd 0.13148 0.12090 0.0002207 0.0018254

2. Quantiles for each variable:

2.5% 25% 50% 75% 97.5%

T[1] 0.011478 0.03002 0.04964 0.08237 0.2146

T[2] 0.011237 0.02957 0.04905 0.08157 0.2143

d[2] -0.191383 -0.06261 -0.01168 0.03761 0.1608

sd 0.004005 0.04556 0.10030 0.18124 0.4421

Warning messages:

1: In FUN(X[[i]], ...) : start value not changed

2: In FUN(X[[i]], ...) : start value not changed

3: In FUN(X[[i]], ...) : start value not changed

> # Anmerkung: mean für totresdev ist D\_res

>

>

>

> #### Trace Monitore

> #plot(samps\_coda)

>

>

>

> #### Median-Berechnung

> # Median für T1

> median(as.matrix(samps\_coda[,1]))

[1] 0.04964129

> # Median für T2

> median(as.matrix(samps\_coda[,2]))

[1] 0.04905075

> # Median für d[2]

> median(as.matrix(samps\_coda[,3]))

[1] -0.01168345

> # Median für sd

> median(as.matrix(samps\_coda[,4]))

[1] 0.1003046

>

>

>

> ##### run JAGS and save posterior samples

> samps\_jags <- jags.samples( jags.m, variable.names=c("dev", "totresdev", "theta", "deviance"), n.iter=100000, DIC=T )

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

>

>

>

> ##### Berechnung DIC und pD

> D\_res <- summary(samps\_jags$totresdev[])

> D\_res

Min. 1st Qu. Median Mean 3rd Qu. Max.

4.861 17.641 20.808 21.364 24.501 63.238

> pD <- var(samps\_jags$deviance)/2

> pD

[1] 13.96149

> DIC = D\_res + pD

> DIC

Min. 1st Qu. Median Mean 3rd Qu. Max.

18.82 31.60 34.77 35.33 38.46 77.20

>

>

>

>

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