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

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

>

>

>

>

>

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

>

>

>

> ##### Clear data

> rm(list=ls())

>

>

>

> ##### load libraries

> library(rjags)

> library(runjags)

> library(random)

> library(coda)

> 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 runjags/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

>

>

>

> ##### 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 <- run.jags(model="DietaryFat\_Random.txt", monitor=c("d[2]", "T[1]", "T[2]", "sd" , "totresdev", "deviance", "pd", "dic" , "full.pd" ),

+ data=list("ns"=ns, "nt"=nt, "na"=na, "r"=r, "E"=E, "t"=t) , n.chains=3, inits=all.inits, burnin = 100000, sample = 100000, adapt = 5000)

Compiling rjags model...

Calling the simulation using the rjags method...

Adapting the model for 5000 iterations...

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

Burning in the model for 100000 iterations...

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

Running the model for 100000 iterations...

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

Extending 100000 iterations for pD/DIC estimates...

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

Simulation complete

Calculating summary statistics...

Calculating the Gelman-Rubin statistic for 6 variables....

Finished running the simulation

>

>

>

> #### optional, falls nicht konvergiert:

> #jags.m <- autorun.jags(model="DietaryFat\_Random.txt", monitor=c("d[2]", "T[1]", "T[2]", "sd" , "totresdev", "deviance", "pd", "dic" , "full.pd" ),

> # data=list("ns"=ns, "nt"=nt, "na"=na, "r"=r, "E"=E, "t"=t) , n.chains=3,

> # inits=all.inits, burnin = 100000, sample = 100000, adapt = 2000, , max.time="1hr")

>

>

>

>

>

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

>

>

>

> print(jags.m)

JAGS model summary statistics from 300000 samples (chains = 3; adapt+burnin = 105000):

Lower95 Median Upper95 Mean SD Mode MCerr MC%ofSD SSeff AC.10 psrf

d[2] -0.19099 -0.012215 0.15897 -0.013245 0.08878 -- 0.00077361 0.9 13170 0.2169 1.0001

T[1] 0.0054038 0.049991 0.16879 0.065692 0.056256 -- 0.00032479 0.6 30000 -0.0015864 0.99998

T[2] 0.0055297 0.049334 0.16828 0.065053 0.056384 -- 0.00032553 0.6 30000 0.0010535 1

sd 0.0000371 0.10095 0.36247 0.13187 0.12102 -- 0.0018568 1.5 4248 0.71376 1.0001

totresdev 11.755 20.746 31.93 21.312 5.2253 -- 0.031969 0.6 26716 0.025884 0.99995

deviance 114.47 123.47 134.65 124.03 5.2253 -- 0.031969 0.6 26716 0.025884 0.99995

Model fit assessment:

DIC = 133.0744 (range between chains: 132.9887 - 133.0233)

[PED not available from the stored object]

Estimated effective number of parameters: pD = 8.92366

Total time taken: 1.7 minutes

> summary(jags.m$mcmc) # für 2.5 - 97.5 CrI

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

d[2] -0.01260 0.08867 0.0001619 0.0006531

T[1] 0.06581 0.05716 0.0001044 0.0001044

T[2] 0.06525 0.05727 0.0001046 0.0001061

sd 0.13148 0.12090 0.0002207 0.0018254

totresdev 21.35766 5.26080 0.0096049 0.0196599

deviance 124.07657 5.26080 0.0096049 0.0196599

2. Quantiles for each variable:

2.5% 25% 50% 75% 97.5%

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

T[1] 0.011478 0.03002 0.04964 0.08237 0.2146

T[2] 0.011237 0.02957 0.04905 0.08157 0.2143

sd 0.004005 0.04556 0.10030 0.18124 0.4421

totresdev 12.583964 17.64984 20.80463 24.47654 33.2651

deviance 115.302874 120.36875 123.52354 127.19545 135.9840

>

>

>

>

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