> ########## ########## Simulation Dietary Fat Beispiel mit Random 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/Dietary fat")

>

>

>

> ##### 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

[1] 10

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

> nt # check

[1] 3

> na <- data[,10]

> na # check

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

> r <- data[,7:9]

> r # Check

r..1. r..2. r..3.

[1,] 113 111 NA

[2,] 1 5 3

[3,] 24 20 NA

[4,] 248 269 NA

[5,] 31 28 NA

[6,] 65 48 NA

[7,] 3 1 NA

[8,] 28 39 NA

[9,] 177 174 NA

[10,] 2 1 NA

> E <- data[,4:6]

> E # Check

E..1. E..2. E..3.

[1,] 1917.0 1925.0 NA

[2,] 43.6 41.3 38

[3,] 393.5 373.9 NA

[4,] 4715.0 4823.0 NA

[5,] 715.0 751.0 NA

[6,] 885.0 895.0 NA

[7,] 87.8 91.0 NA

[8,] 1011.0 939.0 NA

[9,] 1544.0 1588.0 NA

[10,] 125.0 123.0 NA

> t <- data[,1:3]

> t # Check

t..1. t..2. t..3.

[1,] 1 2 NA

[2,] 1 2 2

[3,] 1 2 NA

[4,] 1 2 NA

[5,] 1 2 NA

[6,] 1 2 NA

[7,] 1 2 NA

[8,] 1 2 NA

[9,] 1 2 NA

[10,] 1 2 NA

>

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

> dat

[[1]]

[1] "ns"

[[2]]

[1] "nt"

[[3]]

[1] "na"

[[4]]

[1] "r"

[[5]]

[1] "E"

[[6]]

[1] "t"

>

>

>

> ##### 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="base::Wichmann-Hill", .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="base::Wichmann-Hill", .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="base::Wichmann-Hill", .RNG.seed=3 )

>

> # Achtung: "lecuyer::RngStream" funktioniert nicht mit jagsUI

> 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,

+ model.file="DietaryFat\_Random.txt",

+ n.chains=3,

+ n.iter=110000, n.burnin=90000,

+ store.data=TRUE)

Processing function input.......

Done.

Compiling model graph

Resolving undeclared variables

Allocating nodes

Graph information:

Observed stochastic nodes: 21

Unobserved stochastic nodes: 25

Total graph size: 363

Initializing model

Adaptive phase.....

Adaptive phase complete

Burn-in phase, 90000 iterations x 3 chains

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

Sampling from joint posterior, 20000 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

> #bei n.iter=100k & burn-in=100k:

> #Error in process.input(data, parameters.to.save, inits, n.chains, n.iter, :

> #Number of iterations must be larger than burn-in.

> #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 'DietaryFat\_Random.txt', generated by jagsUI.

Estimates based on 3 chains of 110000 iterations,

adaptation = 100 iterations (sufficient),

burn-in = 90000 iterations and thin rate = 1,

yielding 60000 total samples from the joint posterior.

MCMC ran for 0.686 minutes at time 2019-08-21 14:11:18.

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

d[2] -0.013 0.089 -0.194 -0.012 0.164 TRUE 0.561 1.001 2659

T[1] 0.066 0.058 0.012 0.050 0.216 FALSE 1.000 1.001 60000

T[2] 0.065 0.058 0.011 0.049 0.217 FALSE 1.000 1.001 31218

sd 0.129 0.116 0.004 0.100 0.439 FALSE 1.000 1.002 1071

dev[1,1] 0.743 1.079 0.001 0.331 3.754 FALSE 1.000 1.000 45616

dev[2,1] 2.115 1.508 0.109 1.827 5.802 FALSE 1.000 1.000 60000

dev[3,1] 0.716 1.006 0.001 0.333 3.548 FALSE 1.000 1.000 60000

dev[4,1] 0.925 1.295 0.001 0.419 4.612 FALSE 1.000 1.000 23871

dev[5,1] 0.757 1.049 0.001 0.356 3.735 FALSE 1.000 1.000 19863

dev[6,1] 1.450 1.728 0.002 0.811 6.139 FALSE 1.000 1.002 2475

dev[7,1] 1.243 1.524 0.002 0.690 5.554 FALSE 1.000 1.000 60000

dev[8,1] 1.541 1.722 0.003 0.963 6.130 FALSE 1.000 1.000 18938

dev[9,1] 0.814 1.153 0.001 0.371 4.116 FALSE 1.000 1.000 60000

dev[10,1] 0.842 1.153 0.001 0.405 4.116 FALSE 1.000 1.000 20751

dev[1,2] 0.734 1.045 0.001 0.328 3.741 FALSE 1.000 1.001 7337

dev[2,2] 1.608 1.561 0.007 1.164 5.716 FALSE 1.000 1.000 60000

dev[3,2] 0.626 0.909 0.001 0.278 3.180 FALSE 1.000 1.000 60000

dev[4,2] 0.957 1.333 0.001 0.441 4.789 FALSE 1.000 1.000 60000

dev[5,2] 0.689 0.964 0.001 0.321 3.458 FALSE 1.000 1.000 60000

dev[6,2] 1.374 1.641 0.002 0.775 5.893 FALSE 1.000 1.001 4708

dev[7,2] 0.879 1.091 0.001 0.481 3.920 FALSE 1.000 1.000 52900

dev[8,2] 1.637 1.841 0.003 1.021 6.628 FALSE 1.000 1.000 26401

dev[9,2] 0.820 1.177 0.001 0.366 4.125 FALSE 1.000 1.000 21766

dev[10,2] 0.533 0.770 0.000 0.239 2.702 FALSE 1.000 1.000 60000

dev[2,3] 0.381 0.536 0.000 0.173 1.945 FALSE 1.000 1.001 5517

totresdev 21.384 5.238 12.532 20.834 33.022 FALSE 1.000 1.000 6031

theta[1,1] 112.632 9.138 95.771 112.277 131.476 FALSE 1.000 1.000 7855

theta[2,1] 3.139 1.099 1.361 3.018 5.629 FALSE 1.000 1.000 60000

theta[3,1] 22.926 3.815 16.099 22.681 31.006 FALSE 1.000 1.000 25288

theta[4,1] 252.356 14.748 223.967 252.196 281.844 FALSE 1.000 1.000 26149

theta[5,1] 29.350 4.308 21.654 29.085 38.513 FALSE 1.000 1.001 3334

theta[6,1] 58.993 6.826 47.056 58.536 73.583 FALSE 1.000 1.001 1547

theta[7,1] 2.001 1.024 0.520 1.829 4.493 FALSE 1.000 1.000 21357

theta[8,1] 33.443 4.852 24.148 33.339 43.193 FALSE 1.000 1.000 7420

theta[9,1] 175.331 11.775 153.266 175.019 199.486 FALSE 1.000 1.000 8557

theta[10,1] 1.537 0.899 0.308 1.364 3.738 FALSE 1.000 1.000 53263

theta[1,2] 111.418 9.047 94.445 111.183 129.919 FALSE 1.000 1.000 42393

theta[2,2] 3.083 1.070 1.364 2.960 5.542 FALSE 1.000 1.000 60000

theta[3,2] 21.071 3.553 14.638 20.863 28.656 FALSE 1.000 1.000 7867

theta[4,2] 264.674 15.202 235.916 264.202 295.415 FALSE 1.000 1.000 60000

theta[5,2] 29.594 4.322 21.692 29.421 38.619 FALSE 1.000 1.000 50389

theta[6,2] 53.998 6.383 41.650 53.890 66.799 FALSE 1.000 1.000 4001

theta[7,2] 1.994 1.020 0.519 1.822 4.454 FALSE 1.000 1.000 60000

theta[8,2] 33.571 4.850 25.080 33.193 44.095 FALSE 1.000 1.000 6941

theta[9,2] 175.794 11.925 153.144 175.565 199.806 FALSE 1.000 1.000 10462

theta[10,2] 1.474 0.866 0.293 1.309 3.596 FALSE 1.000 1.000 13173

theta[2,3] 2.780 0.961 1.224 2.673 4.968 FALSE 1.000 1.000 60000

deviance 124.103 5.238 115.251 123.553 135.741 FALSE 1.000 1.000 6031

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 = 13.7 and DIC = 137.818

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=all.inits,

+ parameters.to.save=params,

+ model.file="DietaryFat\_Random.txt",

+ n.chains=3,

+ n.iter=110000, n.burnin=90000)

Processing function input.......

Done.

Compiling model graph

Resolving undeclared variables

Allocating nodes

Graph information:

Observed stochastic nodes: 21

Unobserved stochastic nodes: 25

Total graph size: 363

Initializing model

Adaptive phase.....

Adaptive phase complete

Burn-in phase, 90000 iterations x 3 chains

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

Sampling from joint posterior, 20000 iterations x 3 chains

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

MCMC took 0.579 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

> #ja, funktioniert

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

>

>

>

> #### Berechnung Median

> # Median von T1

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

[1] 0.04990442

>

> # Median von T2

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

[1] 0.04932665

>

> # Median von d

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

[1] -0.01237286

>

> # Median von sd

> median(jagsbasic.m\_II$`sd`)

[1] 0.09951937

>

>

>

>

>

>

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