> ########## ########## Simulation Dietary Fat 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/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]", "dev", "totresdev", "theta" )

> params

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

>

>

>

> ##### read in inits with chains

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

+ 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),

+ 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),

+ 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

+ 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] # event rate \* exposure

+ log(lambda[i,k]) <- mu[i] + d[t[i,k]] - d[t[i,1]] # 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

+ }

+ 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

+

+ # 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 = 21000, n.burnin = 9000,

+ 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: 13

Total graph size: 329

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 21000 iterations (first 9000 discarded), n.thin = 12

n.sims = 3000 iterations saved

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

T[1] 0.066 0.055 0.012 0.030 0.050 0.082 0.212 1.001 3000

T[2] 0.065 0.056 0.012 0.030 0.050 0.082 0.219 1.001 3000

d[2] -0.010 0.053 -0.115 -0.045 -0.009 0.026 0.093 1.001 3000

dev[1,1] 0.579 0.825 0.001 0.059 0.248 0.757 2.996 1.002 2100

dev[2,1] 2.184 1.486 0.142 1.066 1.950 2.977 5.764 1.001 3000

dev[3,1] 0.663 0.933 0.001 0.067 0.288 0.893 3.409 1.001 2300

dev[4,1] 0.919 1.229 0.001 0.094 0.431 1.261 4.428 1.003 3000

dev[5,1] 0.693 0.957 0.001 0.066 0.334 0.918 3.433 1.002 1800

dev[6,1] 1.925 1.975 0.006 0.451 1.352 2.773 7.012 1.001 3000

dev[7,1] 1.272 1.566 0.002 0.171 0.710 1.742 5.728 1.001 3000

dev[8,1] 1.973 1.860 0.009 0.537 1.504 2.853 6.828 1.002 2000

dev[9,1] 0.709 0.975 0.001 0.072 0.322 0.945 3.704 1.002 1900

dev[10,1] 0.809 1.098 0.001 0.087 0.390 1.104 3.943 1.001 2400

dev[1,2] 0.571 0.805 0.001 0.061 0.270 0.747 2.831 1.003 1800

dev[2,2] 1.698 1.605 0.008 0.488 1.229 2.481 5.760 1.001 3000

dev[3,2] 0.569 0.776 0.000 0.061 0.266 0.774 2.724 1.001 3000

dev[4,2] 1.043 1.425 0.001 0.108 0.510 1.425 5.145 1.001 3000

dev[5,2] 0.627 0.840 0.001 0.074 0.301 0.845 2.904 1.002 3000

dev[6,2] 1.792 1.794 0.006 0.436 1.277 2.550 6.679 1.008 1000

dev[7,2] 0.911 1.090 0.001 0.125 0.503 1.343 3.869 1.001 3000

dev[8,2] 1.928 1.917 0.010 0.478 1.355 2.750 7.294 1.010 550

dev[9,2] 0.692 0.955 0.001 0.077 0.319 0.943 3.422 1.004 2100

dev[10,2] 0.522 0.742 0.001 0.051 0.232 0.692 2.648 1.001 2400

dev[2,3] 0.371 0.525 0.000 0.033 0.166 0.494 1.798 1.003 860

theta[1,1] 112.312 8.002 96.697 106.971 112.145 117.535 128.144 1.001 3000

theta[2,1] 3.195 1.080 1.425 2.415 3.110 3.832 5.606 1.001 3000

theta[3,1] 22.620 3.498 16.166 20.103 22.497 24.918 29.705 1.001 3000

theta[4,1] 256.618 12.908 232.885 247.409 256.148 265.458 282.619 1.001 3000

theta[5,1] 28.880 3.801 21.891 26.179 28.774 31.425 36.582 1.001 3000

theta[6,1] 56.323 5.638 45.922 52.482 56.089 59.961 68.111 1.001 3000

theta[7,1] 1.974 1.000 0.502 1.258 1.807 2.536 4.331 1.001 3000

theta[8,1] 35.072 4.384 26.947 32.008 35.001 37.913 44.191 1.003 680

theta[9,1] 174.083 10.633 153.764 166.752 173.916 181.298 195.704 1.003 1200

theta[10,1] 1.530 0.871 0.323 0.891 1.361 1.999 3.701 1.001 3000

theta[1,2] 111.709 7.970 96.457 106.189 111.604 117.092 127.694 1.001 3000

theta[2,2] 2.995 1.007 1.356 2.252 2.912 3.599 5.170 1.001 3000

theta[3,2] 21.292 3.320 15.363 18.922 21.127 23.490 28.153 1.001 3000

theta[4,2] 260.029 13.600 233.493 250.925 259.693 269.000 287.852 1.001 3000

theta[5,2] 30.043 3.933 22.653 27.272 29.929 32.691 37.819 1.001 3000

theta[6,2] 56.411 5.580 46.048 52.561 56.259 59.925 68.198 1.002 1500

theta[7,2] 2.026 1.024 0.523 1.280 1.842 2.628 4.421 1.001 3000

theta[8,2] 32.264 4.023 24.473 29.540 32.175 34.928 40.468 1.004 520

theta[9,2] 177.332 10.659 156.763 169.829 177.183 184.439 198.929 1.002 1800

theta[10,2] 1.492 0.849 0.314 0.861 1.323 1.951 3.594 1.001 3000

theta[2,3] 2.756 0.927 1.248 2.072 2.680 3.311 4.757 1.001 3000

totresdev 22.452 4.791 15.140 18.915 21.739 25.327 33.029 1.002 1300

deviance 125.171 4.791 117.859 121.634 124.458 128.046 135.748 1.002 1200

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 = 11.5 and DIC = 136.6

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

>

>

>

> #### Median

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

$T

[1] 0.05026178 0.04980645

$d

[1] -0.009227022

$dev

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

[1,] 0.2480065 0.2697797 0.1664712

[2,] 1.9501069 1.2293048 0.2480065

[3,] 0.2875132 0.2663906 1.9501069

[4,] 0.4310788 0.5102746 0.2875132

[5,] 0.3340354 0.3010117 0.4310788

[6,] 1.3522570 1.2772183 0.3340354

[7,] 0.7103174 0.5026009 1.3522570

[8,] 1.5044301 1.3554954 0.7103174

[9,] 0.3221207 0.3186361 1.5044301

[10,] 0.3898220 0.2318283 0.3221207

$deviance

[1] 124.4576

$theta

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

[1,] 112.145426 111.604169 2.679754

[2,] 3.109523 2.912469 112.145426

[3,] 22.497387 21.127260 3.109523

[4,] 256.148359 259.692857 22.497387

[5,] 28.774046 29.928506 256.148359

[6,] 56.089132 56.259161 28.774046

[7,] 1.806963 1.842075 56.089132

[8,] 35.001172 32.174792 1.806963

[9,] 173.915838 177.182748 35.001172

[10,] 1.361369 1.323063 173.915838

$totresdev

[1] 21.73869

>

>

>

>

>

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