> ########## ########## Simulation Blocker Beispiel mit Fixed Effects mit NIMBLE

> ########## Verwendung readBUGS

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

>

>

>

>

>

> # Teil 1 Creating a model ------------------------------------------------------------------

>

>

>

>

> ##### Clear data

> rm(list=ls())

>

>

>

> ##### load libraries

> library(nimble)

> library(car)

> #library(igraph)

> library(coda)

>

>

>

> ##### Sichergehen richtiger Working directory

> setwd("C:/Users/IvanB/Desktop/Masterarbeit/Statistische Programme und Gibbs Sampler/NIMBLE/Nachrechnen TSD2/DietaryFat")

>

>

>

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

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

> # dieser Schritt ist für die Erzeugung des leverage Plotes und des pD notwendig

>

>

>

> ##### Definierung Model Code, seiner Konstanten, Daten, und initialen Werte für MCMC.

> # help(readBUGSmodel) # additionelle Infos

> readBUGS\_Model <- readBUGSmodel(model='DietaryFat\_Fixed\_Model\_Nimble.bug', data = 'DietaryFat\_Data\_Nimble.R',

+ inits = 'DietaryFat\_Inits\_Fixed\_Nimble.R' )

defining model...

Detected r as data within 'constants'.

Adding r as data for building model.

building model...

setting data and initial values...

running calculate on model (any error reports that follow may simply reflect missing values in model variables) ...

checking model sizes and dimensions... This model is not fully initialized. This is not an error. To see which variables are not initialized, use model$initializeInfo(). For more information on model initialization, see help(modelInitialization).

model building finished.

> readBUGS\_Model$initializeInfo()

Missing values (NAs) or non-finite values were found in model variables: r, theta, lambda, dev. This is not an error, but some or all variables may need to be initialized for certain algorithms to operate properly. For more information on model initialization, see help(modelInitialization).

>

>

>

> ##### Simulation

> mcmc.out <- nimbleMCMC(code = readBUGS\_Model,

+ nchains = 3, niter = 21000, nburnin = 19000,

+ summary = TRUE, WAIC = F,

+ monitors = c("totresdev", "T", "d"))

compiling... this may take a minute. Use 'showCompilerOutput = TRUE' to see C++ compilation details.

compilation finished.

running chain 1...

|-------------|-------------|-------------|-------------|

|-------------------------------------------------------|

running chain 2...

|-------------|-------------|-------------|-------------|

|-------------------------------------------------------|

running chain 3...

|-------------|-------------|-------------|-------------|

|-------------------------------------------------------|

>

>

>

>

>

> # Teil 2: Anzeigen Ergebnisse der Simulation ------------------------------

>

>

>

>

> #### Zusammenfassung posterioreer Werte

> mcmc.out[["summary"]][["all.chains"]]

Mean Median St.Dev. 95%CI\_low 95%CI\_upp

T[1] 6.771218e-02 0.051296277 6.106022e-02 1.173832e-02 2.222911e-01

T[2] 6.746394e-02 0.050937651 6.152215e-02 1.160111e-02 2.194120e-01

T[3] 1.498920e+144 0.046403650 1.160552e+146 7.388369e-87 2.931913e+84

d[1] 0.000000e+00 0.000000000 0.000000e+00 0.000000e+00 0.000000e+00

d[2] -5.628143e-03 -0.004543555 5.326746e-02 -1.119996e-01 9.625544e-02

d[3] 7.825477e-01 -0.073767881 9.888184e+01 -1.948944e+02 1.973561e+02

totresdev 2.211390e+01 21.454174151 4.626502e+00 1.509956e+01 3.298050e+01

>

>

>

> #### Berechnung der CrI

>

> ## Berechnung CrI von T[1]

> T1\_1 <- quantile(mcmc.out$samples[["chain1"]][,"T[1]"] , c(0.025, 0.975))

> T1\_2 <- quantile(mcmc.out$samples[["chain2"]][,"T[1]"] , c(0.025, 0.975))

> T1\_3 <- quantile(mcmc.out$samples[["chain3"]][,"T[1]"] , c(0.025, 0.975))

>

> # CrI von T[1]

> (T1\_1 + T1\_2 + T1\_3)/3

2.5% 97.5%

0.01179803 0.22214072

>

>

> ## Berechnung CrI von T[2]

> T2\_1 <- quantile(mcmc.out$samples[["chain1"]][,"T[2]"] , c(0.025, 0.975))

> T2\_2 <- quantile(mcmc.out$samples[["chain2"]][,"T[2]"] , c(0.025, 0.975))

> T2\_3 <- quantile(mcmc.out$samples[["chain3"]][,"T[2]"] , c(0.025, 0.975))

>

> # CrI von T[2]

> (T2\_1 + T2\_2 + T2\_3)/3

2.5% 97.5%

0.01161149 0.21906904

>

>

> ## Berechnung CrI von d[2]

> d2\_1 <- quantile(mcmc.out$samples[["chain1"]][,"d[2]"] , c(0.025, 0.975))

> d2\_2 <- quantile(mcmc.out$samples[["chain2"]][,"d[2]"] , c(0.025, 0.975))

> d2\_3 <- quantile(mcmc.out$samples[["chain3"]][,"d[2]"] , c(0.025, 0.975))

>

> # CrI von d[2]

> (d2\_1 + d2\_2 + d2\_3)/3

2.5% 97.5%

-0.11299556 0.09905119

>

>

>

>

>

> # Teil 3: Nachträgliche Berechnung von pD und Erzeugung DAG des Modelles -----------------

>

>

> #Model\_Nimble$dev

>

>

>

> out\_lePlo <- capture.output( readBUGS\_Model$dev)

> cat("Hilf\_pD", out\_lePlo, file="Hilf4.txt", sep="\n", append=TRUE)

>

>

> Hilf\_data = read.table("Hilf4.txt", sep = "", header=F, skip=2)

> #Hilf\_data

>

>

> Hilf\_dev <- c (Hilf\_data[,2], Hilf\_data[,3], Hilf\_data[2,4])

> #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] 30253708

>

>

>

> #### Plot of model

> #directed acyclic graph

> #durch igraph

> readBUGS\_Model$plotGraph() # Anweisung geht nicht bei nimbleMCMC

>

>

>

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