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

> ########## Verwendung nimbleModel

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

>

>

>

>

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

>

>

>

> ##### Clear data

> rm(list=ls())

>

>

>

> ##### Sichergehen richtiger Working directory

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

>

>

>

> ##### load libraries

> library(car)

> library(nimble)

> library(coda)

> #library(igraph)

>

>

>

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

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

> #head(data) # Shows the first six entries

> data2 = read.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

>

>

>

> ##### Zuordnen der Argumente für NIBMLE

>

> ### Zuordnen Konstanten

> Nimble\_constants = list(ns=ns, nt=nt, na=na, t=t, E=E)

> Nimble\_constants

$ns

[1] 10

$nt

[1] 3

$na

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

$t

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

$E

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

>

>

> ### Zuordnen data

> Nimble\_data = list(r=r) # , dev=dev

> Nimble\_data

$r

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

>

>

> ### Zuordnen Inits

> Nimble\_inits = list(d=c( NA, 0, 0),

+ mu=c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0),

+ # zusätzlich noch einzufügen:

+ A=0)

>

>

> ### Create Model Code

> Code\_Model<- nimbleCode( {

+ 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[1:10]) #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] }

+ })

>

>

>

> # nimbleModel prozessiert BUGS-Modellcode und optionale Konstanten, Daten und Initialwerte. Liefert ein NIMBLE-Modell zurück.

> # dieser Schritt ist bei den hier getesteten BUGS Modellen nicht nötig

> Model\_Nimble <- nimbleModel(code = Code\_Model, name = "ProcessedModel", constants = Nimble\_constants,

+ data = Nimble\_data, inits = Nimble\_inits)

defining 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.

>

> Model\_Nimble$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 = Code\_Model, constants = Nimble\_constants,

+ data = Nimble\_data, inits = Nimble\_inits,

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

+ summary = TRUE, WAIC = F,

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

defining 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).

checking model calculations...

model building finished.

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...

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

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

> #help(buildMCMC)

>

>

>

>

>

> # 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] 6.579177e-02 0.048934251 0.05750478 1.159132e-02 2.234312e-01

T[2] 6.504454e-02 0.048545229 0.05693238 1.140490e-02 2.193413e-01

T[3] 2.035579e+162 0.008406905 Inf 9.655927e-88 1.065140e+84

d[1] 0.000000e+00 0.000000000 0.00000000 0.000000e+00 0.000000e+00

d[2] -1.232728e-02 -0.013397081 0.05194555 -1.176833e-01 9.250301e-02

d[3] -1.397586e+00 -1.761801217 99.99448334 -1.964188e+02 1.968820e+02

totresdev 2.234226e+01 21.743587647 4.68742252 1.517548e+01 3.308814e+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%

0.01165093 0.22228907

>

>

> ## 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%

0.01141241 0.21824418

>

>

> ## 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%

-0.11597832 0.09061256

>

>

>

>

>

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

>

>

> #Model\_Nimble$dev

>

>

> out\_lePlo <- capture.output( Model\_Nimble$dev)

> cat("Hilf\_pD", out\_lePlo, file="Hilf2.txt", sep="\n", append=TRUE)

>

>

> Hilf\_data = read.table("Hilf2.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

> Model\_Nimble$plotGraph() # Anweisung geht nicht bei nimbleMCMC

>

>

>

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