> ########## ########## Simulation Diabetes 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/Ex3 Diabetes")

>

>

>

>

> ##### load libraries

> library(car)

> library(nimble)

> library(coda)

> #library(igraph)

>

>

>

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

> #data = read.table("Diabetes\_Data\_neu sortiert.txt", sep = "", header=F)

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

> head(data) # Shows the first six entries

time.. t..1. t..2. t..3. r..1. r..2. r..3. n..1. n..2. n..3. na..

[1,] 5.8 1 2 3 43 34 37 1081 2213 1102 3

[2,] 4.7 1 2 NA 29 20 NA 416 424 NA 2

[3,] 3.0 1 2 NA 140 118 NA 1631 1578 NA 2

[4,] 3.8 1 3 NA 75 86 NA 3272 3297 NA 2

[5,] 4.0 1 4 5 302 154 119 6766 3954 4096 3

[6,] 3.0 1 4 NA 176 136 NA 2511 2508 NA 2

> #data2 = as.data.frread.table("Diabetes\_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] 22

> nt <- 6

> nt # check

[1] 6

> na <- data[,11]

> na # check

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

> r <- data[,5:7]

> r # Check

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

[1,] 43 34 37

[2,] 29 20 NA

[3,] 140 118 NA

[4,] 75 86 NA

[5,] 302 154 119

[6,] 176 136 NA

[7,] 200 138 NA

[8,] 8 1 NA

[9,] 154 177 NA

[10,] 489 449 NA

[11,] 155 102 NA

[12,] 399 335 NA

[13,] 202 163 NA

[14,] 115 93 NA

[15,] 70 32 45

[16,] 97 95 93

[17,] 799 567 NA

[18,] 251 216 NA

[19,] 665 569 NA

[20,] 380 337 NA

[21,] 320 242 NA

[22,] 845 690 NA

> time <- data[,1]

> time # Check

[1] 5.8 4.7 3.0 3.8 4.0 3.0 4.1 1.0 3.3 3.0 4.5 4.8 3.1 3.7 3.8 4.0 5.5 4.5 4.0 6.1 4.8 4.2

> t <- data[,2:4]

> t # Check

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

[1,] 1 2 3

[2,] 1 2 NA

[3,] 1 2 NA

[4,] 1 3 NA

[5,] 1 4 5

[6,] 1 4 NA

[7,] 1 5 NA

[8,] 1 6 NA

[9,] 2 4 NA

[10,] 2 5 NA

[11,] 2 5 NA

[12,] 2 5 NA

[13,] 2 6 NA

[14,] 2 6 NA

[15,] 3 4 5

[16,] 3 4 5

[17,] 3 4 NA

[18,] 3 4 NA

[19,] 3 4 NA

[20,] 3 5 NA

[21,] 3 6 NA

[22,] 4 6 NA

> n <- data[,8:10]

> n

n..1. n..2. n..3.

[1,] 1081 2213 1102

[2,] 416 424 NA

[3,] 1631 1578 NA

[4,] 3272 3297 NA

[5,] 6766 3954 4096

[6,] 2511 2508 NA

[7,] 2826 2800 NA

[8,] 196 196 NA

[9,] 4870 4841 NA

[10,] 2646 2623 NA

[11,] 2883 2837 NA

[12,] 3472 3432 NA

[13,] 2721 2715 NA

[14,] 2175 2167 NA

[15,] 405 202 410

[16,] 1960 1965 1970

[17,] 7040 7072 NA

[18,] 5059 5095 NA

[19,] 8078 8098 NA

[20,] 5230 5183 NA

[21,] 3979 4020 NA

[22,] 5074 5087 NA

>

>

>

> ##### Zuordnen der Argumente für NIBMLE

>

>

> ### Zuordnen Konstanten

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

> Nimble\_constants

$ns

[1] 22

$nt

[1] 6

$na

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

$t

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

[1,] 1 2 3

[2,] 1 2 NA

[3,] 1 2 NA

[4,] 1 3 NA

[5,] 1 4 5

[6,] 1 4 NA

[7,] 1 5 NA

[8,] 1 6 NA

[9,] 2 4 NA

[10,] 2 5 NA

[11,] 2 5 NA

[12,] 2 5 NA

[13,] 2 6 NA

[14,] 2 6 NA

[15,] 3 4 5

[16,] 3 4 5

[17,] 3 4 NA

[18,] 3 4 NA

[19,] 3 4 NA

[20,] 3 5 NA

[21,] 3 6 NA

[22,] 4 6 NA

$time

[1] 5.8 4.7 3.0 3.8 4.0 3.0 4.1 1.0 3.3 3.0 4.5 4.8 3.1 3.7 3.8 4.0 5.5 4.5 4.0 6.1 4.8 4.2

$n

n..1. n..2. n..3.

[1,] 1081 2213 1102

[2,] 416 424 NA

[3,] 1631 1578 NA

[4,] 3272 3297 NA

[5,] 6766 3954 4096

[6,] 2511 2508 NA

[7,] 2826 2800 NA

[8,] 196 196 NA

[9,] 4870 4841 NA

[10,] 2646 2623 NA

[11,] 2883 2837 NA

[12,] 3472 3432 NA

[13,] 2721 2715 NA

[14,] 2175 2167 NA

[15,] 405 202 410

[16,] 1960 1965 1970

[17,] 7040 7072 NA

[18,] 5059 5095 NA

[19,] 8078 8098 NA

[20,] 5230 5183 NA

[21,] 3979 4020 NA

[22,] 5074 5087 NA

>

>

> ### Zuordnen data

> Nimble\_data = list(r=r)

> Nimble\_data

$r

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

[1,] 43 34 37

[2,] 29 20 NA

[3,] 140 118 NA

[4,] 75 86 NA

[5,] 302 154 119

[6,] 176 136 NA

[7,] 200 138 NA

[8,] 8 1 NA

[9,] 154 177 NA

[10,] 489 449 NA

[11,] 155 102 NA

[12,] 399 335 NA

[13,] 202 163 NA

[14,] 115 93 NA

[15,] 70 32 45

[16,] 97 95 93

[17,] 799 567 NA

[18,] 251 216 NA

[19,] 665 569 NA

[20,] 380 337 NA

[21,] 320 242 NA

[22,] 845 690 NA

>

>

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

>

> ### Zuordnen Inits

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

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

+ 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] ~ dbin(p[i,k],n[i,k]) # Binomial likelihood

+ cloglog(p[i,k]) <- log(time[i]) + mu[i] + d[t[i,k]] - d[t[i,1]] # model for linear predictor

+ rhat[i,k] <- p[i,k] \* n[i,k] # expected value of the numerators

+ dev[i,k] <- 2 \* (r[i,k] \* (log(r[i,k])-log(rhat[i,k])) + (n[i,k]-r[i,k]) \* (log(n[i,k]-r[i,k]) - log(n[i,k]-rhat[i,k]))) #Deviance contribution

+ }

+ resdev[i] <- sum(dev[i,1:na[i]]) # summed residual deviance contribution for this trial

+ }

+ totresdev <- sum(resdev[1:22]) #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

+ A ~ dnorm(-4.2,1.11)

+ for (k in 1:nt) {

+ cloglog(T[k]) <- log(3) + 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, p, rhat, 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 = 100000, nburnin = 50000,

+ 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] 0.06527043 0.04399181 0.06752423 0.007022250 0.2522554

T[2] 0.05188143 0.03451450 0.05528516 0.005460582 0.2038311

T[3] 0.06193572 0.04156866 0.06462594 0.006612226 0.2409116

T[4] 0.05156159 0.03427383 0.05495641 0.005438248 0.2027354

T[5] 0.04669106 0.03092599 0.05028790 0.004891119 0.1841471

T[6] 0.04269978 0.02820979 0.04639921 0.004447853 0.1692315

d[1] 0.00000000 0.00000000 0.00000000 0.000000000 0.0000000

d[2] -0.24755087 -0.24764861 0.05618864 -0.357225738 -0.1372423

d[3] -0.05798122 -0.05816890 0.05537586 -0.165746939 0.0534563

d[4] -0.25394617 -0.25428494 0.05305951 -0.356934463 -0.1490549

d[5] -0.35915654 -0.35937065 0.05318085 -0.462986995 -0.2542119

d[6] -0.45381575 -0.45449350 0.06247464 -0.574618158 -0.3308609

totresdev 78.21050262 77.53802667 7.32478091 65.794694392 94.3400669

>

>

>

>

>

>

> #### Berechnung der CrI für T

>

>

> ## 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%

0.007017682 0.252307731

>

>

> ## 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%

0.005468719 0.204008907

>

>

> ## Berechnung CrI von T[3]

> T3\_1 <- quantile(mcmc.out$samples[["chain1"]][,"T[3]"] , c(0.025, 0.975))

> T3\_2 <- quantile(mcmc.out$samples[["chain2"]][,"T[3]"] , c(0.025, 0.975))

> T3\_3 <- quantile(mcmc.out$samples[["chain3"]][,"T[3]"] , c(0.025, 0.975))

>

> # CrI von T[3]

> (T3\_1 + T3\_2 + T3\_3)/3

2.5% 97.5%

0.00661076 0.24140738

>

>

> ## Berechnung CrI von T[4]

> T4\_1 <- quantile(mcmc.out$samples[["chain1"]][,"T[4]"] , c(0.025, 0.975))

> T4\_2 <- quantile(mcmc.out$samples[["chain2"]][,"T[4]"] , c(0.025, 0.975))

> T4\_3 <- quantile(mcmc.out$samples[["chain3"]][,"T[4]"] , c(0.025, 0.975))

>

> # CrI von T[4]

> (T4\_1 + T4\_2 + T4\_3)/3

2.5% 97.5%

0.005432509 0.202734842

>

>

> ## Berechnung CrI von T[5]

> T5\_1 <- quantile(mcmc.out$samples[["chain1"]][,"T[5]"] , c(0.025, 0.975))

> T5\_2 <- quantile(mcmc.out$samples[["chain2"]][,"T[5]"] , c(0.025, 0.975))

> T5\_3 <- quantile(mcmc.out$samples[["chain3"]][,"T[5]"] , c(0.025, 0.975))

>

> # CrI von T[5]

> (T5\_1 + T5\_2 + T5\_3)/3

2.5% 97.5%

0.004894879 0.184219227

>

>

> ## Berechnung CrI von T[6]

> T6\_1 <- quantile(mcmc.out$samples[["chain1"]][,"T[6]"] , c(0.025, 0.975))

> T6\_2 <- quantile(mcmc.out$samples[["chain2"]][,"T[6]"] , c(0.025, 0.975))

> T6\_3 <- quantile(mcmc.out$samples[["chain3"]][,"T[6]"] , c(0.025, 0.975))

>

>

> # CrI von T[6]

> (T6\_1 + T6\_2 + T6\_3)/3

2.5% 97.5%

0.004447494 0.169177622

>

>

>

> #### Berechnung der CrI für d

>

>

> ## 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%

-0.3571960 -0.1373538

>

>

> ## Berechnung CrI von d[3]

> d3\_1 <- quantile(mcmc.out$samples[["chain1"]][,"d[3]"] , c(0.025, 0.975))

> d3\_2 <- quantile(mcmc.out$samples[["chain2"]][,"d[3]"] , c(0.025, 0.975))

> d3\_3 <- quantile(mcmc.out$samples[["chain3"]][,"d[3]"] , c(0.025, 0.975))

>

> # CrI von d[3]

> (d3\_1 + d3\_2 + d3\_3)/3

2.5% 97.5%

-0.16531348 0.05289337

>

>

> ## Berechnung CrI von d[4]

> d4\_1 <- quantile(mcmc.out$samples[["chain1"]][,"d[4]"] , c(0.025, 0.975))

> d4\_2 <- quantile(mcmc.out$samples[["chain2"]][,"d[4]"] , c(0.025, 0.975))

> d4\_3 <- quantile(mcmc.out$samples[["chain3"]][,"d[4]"] , c(0.025, 0.975))

>

> # CrI von d[4]

> (d4\_1 + d4\_2 + d4\_3)/3

2.5% 97.5%

-0.3564408 -0.1492000

>

>

> ## Berechnung CrI von d[5]

> d5\_1 <- quantile(mcmc.out$samples[["chain1"]][,"d[5]"] , c(0.025, 0.975))

> d5\_2 <- quantile(mcmc.out$samples[["chain2"]][,"d[5]"] , c(0.025, 0.975))

> d5\_3 <- quantile(mcmc.out$samples[["chain3"]][,"d[5]"] , c(0.025, 0.975))

>

>

> # CrI von d[5]

> (d5\_1 + d5\_2 + d5\_3)/3

2.5% 97.5%

-0.4629117 -0.2540727

>

>

> ## Berechnung CrI von d[6]

> d6\_1 <- quantile(mcmc.out$samples[["chain1"]][,"d[6]"] , c(0.025, 0.975))

> d6\_2 <- quantile(mcmc.out$samples[["chain2"]][,"d[6]"] , c(0.025, 0.975))

> d6\_3 <- quantile(mcmc.out$samples[["chain3"]][,"d[6]"] , c(0.025, 0.975))

>

> # CrI von d[6]

> (d6\_1 + d6\_2 + d6\_3)/3

2.5% 97.5%

-0.5744427 -0.3306026

>

>

>

>

> #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

V1 V2 V3 V4

1 [1,] 11679.4924 24925.1807 12030.327

2 [2,] 3427.9237 3636.7640 NA

3 [3,] 8005.1902 7933.1027 NA

4 [4,] 23585.9759 23610.5701 NA

5 [5,] 49254.8702 29104.1476 30743.694

6 [6,] 12753.0120 13188.6353 NA

7 [7,] 20095.0651 20733.1350 NA

8 [8,] 316.4911 378.3662 NA

9 [9,] 29770.2966 29276.9662 NA

10 [10,] 10459.1669 10688.5076 NA

11 [11,] 23347.7698 23738.5829 NA

12 [12,] 27030.6196 27541.6209 NA

13 [13,] 14197.2073 14604.4432 NA

14 [14,] 14349.7825 14584.6851 NA

15 [15,] 2176.2757 1116.8864 2492.312

16 [16,] 14135.3061 14202.6005 14270.006

17 [17,] 63676.6137 67610.6758 NA

18 [18,] 41280.5299 42127.6772 NA

19 [19,] 54733.7376 56134.0377 NA

20 [20,] 56447.2941 56629.0362 NA

21 [21,] 32905.0455 34443.5812 NA

22 [22,] 30978.9859 32916.8887 NA

>

>

> Hilf\_dev <- c (Hilf\_data[,2], Hilf\_data[,3], Hilf\_data[1,4], Hilf\_data[5,4], Hilf\_data[15,4], Hilf\_data[16,4])

> #Hilf\_dev

>

>

> # manuelle Berechnung von pD

> # dev ist Std-Abweichung jedes einzelnen Werts

> # insg 48 Werte

> Var\_manuell <- sum(Hilf\_dev)^2/48

> pD\_manuell <- Var\_manuell/2

> pD\_manuell

[1] 13854474745

>

>

>

> #### Plot of model

> #directed acyclic graph

> #durch igraph

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

>

>

>

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