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

>

>

>

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

> data = as.matrix(read.table("Diabetes\_Data\_neu sortiert.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='Diabetes\_Fixed\_Model\_Nimble.bug', data = 'Diabetes\_Data\_Nimble.R',

+ inits = 'Diabetes\_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, 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 = readBUGS\_Model,

+ nchains = 3, niter = 100000, nburnin = 50000,

+ 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...

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

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

>

>

> readBUGS\_Model$r

[,1] [,2] [,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

>

>

> # 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] 0.06505016 0.04390924 0.06728028 0.006986185 0.24952272

T[2] 0.05173703 0.03447970 0.05517223 0.005442009 0.20105629

T[3] 0.06180223 0.04158465 0.06445339 0.006562932 0.23809126

T[4] 0.05142456 0.03430899 0.05485128 0.005399496 0.19995447

T[5] 0.04656620 0.03090954 0.05023212 0.004861527 0.18208024

T[6] 0.04262057 0.02822392 0.04644364 0.004426803 0.16728025

d[1] 0.00000000 0.00000000 0.00000000 0.000000000 0.00000000

d[2] -0.24685938 -0.24640174 0.05586532 -0.356709581 -0.13713586

d[3] -0.05662001 -0.05664473 0.05502280 -0.165504986 0.05192311

d[4] -0.25306608 -0.25314078 0.05311533 -0.358088095 -0.14928328

d[5] -0.35832056 -0.35788529 0.05266334 -0.462361531 -0.25583808

d[6] -0.45237547 -0.45195720 0.06268767 -0.577041690 -0.33059409

totresdev 78.17403668 77.53543054 7.34019396 65.746496387 94.23392533

>

>

>

> #### 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%

0.006993048 0.249402917

>

>

> ## 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%

0.005436553 0.201314406

>

>

> ## 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%

0.006560735 0.237915841

>

>

> ## 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%

0.005403032 0.200101585

>

>

> ## 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%

0.004866276 0.182281293

>

>

> ## 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%

0.00442634 0.16721163

>

>

>

> #### 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%

-0.3570478 -0.1372467

>

>

> ## 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%

-0.16548268 0.05131457

>

>

> ## 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%

-0.3580863 -0.1496428

>

>

> ## 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%

-0.4618899 -0.2559702

>

>

> ## 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%

-0.5766215 -0.3304438

>

>

>

>

>

> # 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("Hilf3.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

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

>

>

>

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