> ########## ########## Simulation Blocker Beispiel mit Random 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/Blocker")

>

>

>

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

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

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

>

>

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

> # help(readBUGSmodel) # additionelle Infos

> readBUGS\_Model <- readBUGSmodel(model='Blocker\_Random\_Model\_Nimble\_II.bug', data = 'Blocker\_Data\_Nimble.R',

+ inits = 'Blocker\_Inits\_Nimble\_II.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: lifted\_d1\_over\_sqrt\_oPtaud\_oBi\_comma\_k\_cB\_cP\_L12, md, taud, sw. 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 = 20000, nburnin = 10000,

+ summary = TRUE, WAIC = F,

+ monitors = c("rhat",'dev', "totresdev", "T", "d", "sd"))

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] 0.11090976 0.09995965 0.05501707 3.631531e-02 0.2470577

T[2] 0.08925611 0.07974181 0.04622358 2.830026e-02 0.2051184

d[1] 0.00000000 0.00000000 0.00000000 0.000000e+00 0.0000000

d[2] -0.24999564 -0.25093450 0.06682976 -3.804287e-01 -0.1151238

dev[1, 1] 0.56686612 0.26109199 0.80161263 6.232319e-04 2.8441134

dev[2, 1] 1.06807948 0.53923128 1.37610371 1.379555e-03 4.8375997

dev[3, 1] 0.80889199 0.38646923 1.13948928 7.108529e-04 4.0468272

dev[4, 1] 0.79365794 0.37876762 1.10231671 8.086887e-04 4.0067349

dev[5, 1] 0.96990701 0.47851653 1.29148694 1.324566e-03 4.5475996

dev[6, 1] 0.73009306 0.34400302 1.00067394 7.125967e-04 3.5987279

dev[7, 1] 1.28671375 0.64905157 1.67340087 1.387152e-03 6.0862029

dev[8, 1] 0.83570298 0.38730847 1.16725497 7.430842e-04 4.0858681

dev[9, 1] 0.80165017 0.36741850 1.12454266 8.505397e-04 4.1333561

dev[10, 1] 0.92721129 0.42563997 1.32416635 9.605394e-04 4.6904012

dev[11, 1] 0.67084859 0.30085414 0.96997000 5.198256e-04 3.4540435

dev[12, 1] 0.89338681 0.41484405 1.22356168 1.066572e-03 4.3228681

dev[13, 1] 0.90393664 0.45063301 1.21285944 8.896154e-04 4.3374763

dev[14, 1] 2.25915611 1.50599970 2.35863163 4.748447e-03 8.3643743

dev[15, 1] 0.66636936 0.30811170 0.94548215 5.661706e-04 3.3684851

dev[16, 1] 0.70273129 0.32453503 0.97463534 5.854463e-04 3.4644986

dev[17, 1] 1.02618684 0.59518298 1.21308595 1.704256e-03 4.2266598

dev[18, 1] 0.91002998 0.45362955 1.19684797 1.236799e-03 4.2823355

dev[19, 1] 0.87076219 0.46152665 1.09722618 1.042260e-03 3.8962972

dev[20, 1] 0.69155632 0.31191845 0.97512414 6.897827e-04 3.4952420

dev[21, 1] 1.15554821 0.59833507 1.45961188 1.142584e-03 5.2488091

dev[22, 1] 1.20100241 0.62954040 1.54242966 1.750813e-03 5.4759903

dev[1, 2] 0.58041889 0.26658502 0.80324873 6.100068e-04 2.8523750

dev[2, 2] 0.92229151 0.48485975 1.15822345 1.414006e-03 4.1692997

dev[3, 2] 0.54317833 0.24726406 0.75997697 5.510353e-04 2.7223675

dev[4, 2] 0.74320781 0.33224740 1.07080602 6.543849e-04 3.8266285

dev[5, 2] 1.07462350 0.59156902 1.32755000 1.402449e-03 4.6515498

dev[6, 2] 0.58600529 0.26246934 0.85512452 6.134727e-04 3.0026011

dev[7, 2] 1.33558845 0.70436292 1.66932446 1.788057e-03 5.8681473

dev[8, 2] 0.89816571 0.42311137 1.23665123 1.201211e-03 4.4073636

dev[9, 2] 0.69911000 0.31985445 0.98319526 7.428031e-04 3.5298051

dev[10, 2] 0.83764327 0.37697066 1.19199915 8.283739e-04 4.1898495

dev[11, 2] 0.75421059 0.35164919 1.04688900 8.387941e-04 3.7561343

dev[12, 2] 0.91742332 0.43983606 1.23923938 7.979779e-04 4.4807500

dev[13, 2] 0.73550591 0.36530254 1.00120983 8.965620e-04 3.5565546

dev[14, 2] 2.47109011 1.73203599 2.49156992 5.192285e-03 8.9223093

dev[15, 2] 0.60710719 0.27128549 0.87345390 6.043498e-04 3.1447719

dev[16, 2] 0.68353509 0.31944794 0.94009729 6.749372e-04 3.3343803

dev[17, 2] 1.03780948 0.49753012 1.39652866 1.137661e-03 4.9398284

dev[18, 2] 1.10380266 0.62308028 1.32416908 1.437884e-03 4.8241928

dev[19, 2] 1.12487745 0.59688864 1.40832111 1.527759e-03 4.9378889

dev[20, 2] 0.60689642 0.27630577 0.85591928 5.107545e-04 3.0577786

dev[21, 2] 1.09761625 0.59965435 1.36056758 1.920197e-03 4.8390768

dev[22, 2] 1.11583489 0.60410371 1.36710165 1.274515e-03 4.8562619

rhat[1, 1] 3.36519319 3.21136781 1.30691663 1.273905e+00 6.2993227

rhat[2, 1] 12.02784384 11.85831646 2.59555228 7.520954e+00 17.5784332

rhat[3, 1] 10.14888173 9.94350396 2.46200961 5.857414e+00 15.4845721

rhat[4, 1] 127.22527334 126.94879210 9.62649488 1.096554e+02 146.6060227

rhat[5, 1] 30.07725064 29.91018225 4.30828252 2.222037e+01 38.8977033

rhat[6, 1] 5.31274747 5.14182135 1.63993956 2.609549e+00 9.0919055

rhat[7, 1] 144.99671474 144.72480163 10.26161797 1.255611e+02 166.3781837

rhat[8, 1] 50.75919905 50.57749601 5.60448027 4.013055e+01 62.0446250

rhat[9, 1] 35.26552922 35.07971314 4.56001854 2.670480e+01 44.7389826

rhat[10, 1] 185.41239952 184.93240661 12.12294837 1.626330e+02 210.2840190

rhat[11, 1] 52.74228617 52.52696034 5.65601829 4.221346e+01 64.7175910

rhat[12, 1] 49.97353935 49.84240749 5.33016049 3.987174e+01 60.7676400

rhat[13, 1] 14.30206922 14.05898421 2.92257268 9.177364e+00 20.6329599

rhat[14, 1] 53.56820870 53.38142200 6.48918488 4.115633e+01 66.4308426

rhat[15, 1] 30.39747214 30.21618227 3.92221727 2.315051e+01 38.3312382

rhat[16, 1] 39.24031258 39.02772271 4.62020934 3.063897e+01 48.5509309

rhat[17, 1] 14.69131030 14.59981747 2.55398898 9.973187e+00 19.8683506

rhat[18, 1] 7.71468416 7.51678789 2.09087927 4.150532e+00 12.3182875

rhat[19, 1] 4.37281904 4.20878341 1.48503594 1.962423e+00 7.6839092

rhat[20, 1] 40.27159055 40.11842988 4.75022892 3.132093e+01 49.9511752

rhat[21, 1] 39.17319989 38.93134243 4.91210023 3.025138e+01 49.6003120

rhat[22, 1] 35.22961675 34.90380180 4.89782913 2.641947e+01 45.6912523

rhat[1, 2] 2.63859488 2.50397149 1.05670529 9.872808e-01 5.0859232

rhat[2, 2] 8.98644551 8.81615500 2.05204734 5.429265e+00 13.4565176

rhat[3, 2] 5.92795834 5.79278820 1.53167425 3.290018e+00 9.3080238

rhat[4, 2] 101.85403433 101.64111733 8.39067259 8.584413e+01 119.2522427

rhat[5, 2] 24.87359251 24.61374470 3.73157602 1.835313e+01 32.9495626

rhat[6, 2] 4.73347116 4.57574852 1.51326156 2.252893e+00 8.1375220

rhat[7, 2] 104.98716926 104.98655153 8.87145198 8.766130e+01 122.1394972

rhat[8, 2] 57.22410201 56.91921910 6.13057664 4.604979e+01 70.0563586

rhat[9, 2] 26.75638043 26.61788292 3.80594732 1.965030e+01 34.6544783

rhat[10, 2] 140.62927028 140.57746646 10.17041605 1.210363e+02 160.9500989

rhat[11, 2] 63.27471921 63.02529215 6.57758631 5.109774e+01 76.8741692

rhat[12, 2] 42.16773347 41.96617650 4.83207128 3.319707e+01 52.2993700

rhat[13, 2] 10.70243451 10.54510967 2.30201937 6.614249e+00 15.6983676

rhat[14, 2] 48.48115636 47.98477303 6.19822009 3.775800e+01 62.1692403

rhat[15, 2] 25.66074203 25.53676302 3.56198836 1.898181e+01 32.9959655

rhat[16, 2] 31.76029302 31.57556575 4.03765793 2.435825e+01 40.1210097

rhat[17, 2] 25.54962418 25.36562824 4.06155066 1.823893e+01 34.0593613

rhat[18, 2] 6.26952943 6.10087232 1.75033130 3.337431e+00 10.1555514

rhat[19, 2] 4.60867231 4.43965381 1.57885734 2.090136e+00 8.2041266

rhat[20, 2] 31.67056361 31.50978905 3.99562938 2.420710e+01 39.9457025

rhat[21, 2] 30.85518073 30.72141433 4.18534474 2.290153e+01 39.4032990

rhat[22, 2] 25.70993089 25.51947172 3.88700214 1.854678e+01 33.7123554

sd 0.15013213 0.14358042 0.08226187 2.437873e-02 0.3333607

totresdev 41.21623064 40.79990305 8.16688136 2.657425e+01 58.5268887

>

>

>

> #### 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%

0.03635657 0.24667367

>

>

> ## 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%

0.02829323 0.20466499

>

>

> ## 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%

-0.3794674 -0.1154829

>

>

> ## Berechnung CrI von sd

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

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

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

>

> # CrI von sd

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

2.5% 97.5%

0.03285981 0.33270922

>

>

>

>

>

> # Teil 3: Leverage Plot und nachträgliche Berechnung von pD -----------------

>

>

>

>

> out\_lePlo <- capture.output( mcmc.out$summary)

> cat("Hilf\_lePlo", out\_lePlo, file="Hilf3.txt", sep="\n", append=TRUE)

>

>

> Hilf\_data = read.table("Hilf3.txt", sep = "", header=F, skip=298, nrows=88)

> #Hilf\_data

>

>

>

> #### Berechnung w\_ik

>

> Hilf\_dev <- Hilf\_data[1:44,3]

> #Hilf\_dev

> Hilf\_dev\_k1 <- Hilf\_dev[1:22]

> #Hilf\_dev\_k1

> Hilf\_dev\_k2 <- Hilf\_dev[23:44]

> #Hilf\_dev\_k2

> Hilf\_dev\_II <- cbind(Hilf\_dev\_k1, Hilf\_dev\_k2)

> #Hilf\_dev\_II

> Hilf\_dev\_III <- cbind(Hilf\_dev\_II, total = rowMeans(Hilf\_dev\_II))

> #Hilf\_dev\_III

> w\_ik <- sqrt(Hilf\_dev\_III[,3])

> w\_ik\_neg <- -w\_ik

> fertige\_Daten\_w\_ik <- cbind(Hilf\_dev\_III, w\_ik\_neg, w\_ik)

> fertige\_Daten\_w\_ik

Hilf\_dev\_k1 Hilf\_dev\_k2 total w\_ik\_neg w\_ik

[1,] 0.5936358 0.5980192 0.5958275 -0.7718986 0.7718986

[2,] 1.0514127 0.9493989 1.0004058 -1.0002029 1.0002029

[3,] 0.7833645 0.5184524 0.6509085 -0.8067890 0.8067890

[4,] 0.7594096 0.6907076 0.7250586 -0.8515037 0.8515037

[5,] 0.9735719 1.1644114 1.0689916 -1.0339205 1.0339205

[6,] 0.7084203 0.5645927 0.6365065 -0.7978136 0.7978136

[7,] 1.4722321 1.4656086 1.4689203 -1.2119902 1.2119902

[8,] 0.8640163 0.9156080 0.8898121 -0.9432985 0.9432985

[9,] 0.8052527 0.6755125 0.7403826 -0.8604549 0.8604549

[10,] 0.9191850 0.8132523 0.8662186 -0.9307087 0.9307087

[11,] 0.6641325 0.7337779 0.6989552 -0.8360354 0.8360354

[12,] 0.9182274 0.9486498 0.9334386 -0.9661463 0.9661463

[13,] 0.9103919 0.7581663 0.8342791 -0.9133888 0.9133888

[14,] 2.5365535 2.7633890 2.6499713 -1.6278732 1.6278732

[15,] 0.6806393 0.6081285 0.6443839 -0.8027353 0.8027353

[16,] 0.7217434 0.7031936 0.7124685 -0.8440785 0.8440785

[17,] 1.0274452 1.0690149 1.0482301 -1.0238311 1.0238311

[18,] 0.9330749 1.1325145 1.0327947 -1.0162651 1.0162651

[19,] 0.8697201 1.1194598 0.9945900 -0.9972913 0.9972913

[20,] 0.6658870 0.5960710 0.6309790 -0.7943419 0.7943419

[21,] 1.2061918 1.1606888 1.1834403 -1.0878604 1.0878604

[22,] 1.2118018 1.1654034 1.1886026 -1.0902305 1.0902305

>

>

>

> # manuelle Berechnung von pD

> # dev ist Std-Abweichung jedes einzelnen Werts

> # insg 22 Wertepaare

> Var\_manuell <- sum(Hilf\_dev)^2/44

> pD\_manuell <- Var\_manuell/2

> pD\_manuell

[1] 20.41977

>

>

>

> #### Berechnung leverage\_ik

>

> Hilf\_rhat <- Hilf\_data[45:88,3]

> #Hilf\_rhat

> Hilf\_rhat\_k1 <- Hilf\_rhat[1:22]

> #Hilf\_rhat\_k1

> Hilf\_rhat\_k2 <- Hilf\_rhat[23:44]

> #Hilf\_rhat\_k2

>

>

> dev\_tilde\_erst\_k1 <- data[,1]\*log(data[,1]/Hilf\_rhat\_k1)

> #dev\_tilde\_erst\_k1

> dev\_tilde\_zweit\_k1 <- (data[,3]-data[,1])\*log((data[,3]-data[,1])/(data[,3]-Hilf\_rhat\_k1))

> #dev\_tilde\_zweit\_k1

> dev\_tilde\_gesamt\_k1 <- 2\*(dev\_tilde\_erst\_k1+dev\_tilde\_zweit\_k1)

> #dev\_tilde\_gesamt\_k1

> leverage\_k1 <- fertige\_Daten\_w\_ik[,1] - dev\_tilde\_gesamt\_k1

>

> leverage\_k1

[1] 0.5424243 0.7053207 0.6982241 0.7574987 0.5755389 0.6101020 0.9248132 0.6382187 0.6909673 0.8521708 0.6441838 0.6513025 0.7132380

[14] 0.6951535 0.6657428 0.6551919 0.4124601 0.4729919 0.3600620 0.6591888 0.7526965 0.7396578

>

>

> dev\_tilde\_erst\_k2 <- data[,2]\*log(data[,2]/Hilf\_rhat\_k2)

> #dev\_tilde\_erst\_k2

> dev\_tilde\_erst\_k2 <- data[,2]\*log(data[,2]/Hilf\_rhat\_k2)

> #dev\_tilde\_erst\_k2

> dev\_tilde\_zweit\_k2 <- (data[,4]-data[,2])\*log((data[,4]-data[,2])/(data[,4]-Hilf\_rhat\_k2))

> #dev\_tilde\_zweit\_k2

> dev\_tilde\_gesamt\_k2 <- 2\*(dev\_tilde\_erst\_k2+dev\_tilde\_zweit\_k2)

> #dev\_tilde\_gesamt\_k2

> leverage\_k2 <- fertige\_Daten\_w\_ik[,2] - dev\_tilde\_gesamt\_k2

>

> leverage\_k2

[1] 0.5477882 0.4071567 0.3667408 0.6899268 0.6600295 0.4372218 0.7674925 0.7379119 0.5389607 0.7511513 0.7216380 0.6816312 0.4329596

[14] 0.9476301 0.5885872 0.6230452 0.7627249 0.6336514 0.7065624 0.5877977 0.5488756 0.4904358

>

>

>

> #### Erzeugen leverage plot

>

> scatterplot(c(fertige\_Daten\_w\_ik[,"w\_ik\_neg"], fertige\_Daten\_w\_ik[,"w\_ik"]), c(leverage\_k1, leverage\_k2),

+ main="Leverage plot for the random effects model", xlim=c(-3,3), ylim=c(0,4.5), xlab=expression('w'[ik]),

+ ylab=expression('leverage'[ik]), regLine =F, smooth=F, boxplots=F, grid=TRUE )

> curve(-x^2+1, from=-3, to=3, col="red", lty="solid", add=T)

> curve(-x^2+2, from=-3, to=3, col="green", lty="dashed", add=T)

> curve(-x^2+3, from=-3, to=3, col="blueviolet", lty="dotted", add=T)

> curve(-x^2+4, from=-3, to=3, col="blue", lty="dotdash", add=T)

>

>

>

> #### Plot of model

> #directed acyclic graph

> #durch igraph

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

>

>

>

>

>

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