> ########## ########## 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/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\_Fixed\_Model\_II.bug', data = 'Blocker\_Data\_Nimble.R',

+ inits = 'Blocker\_Inits\_Nimble\_Fixed.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...

model building finished.

> readBUGS\_Model$initializeInfo()

All model variables are initialized.

>

>

>

> ##### Simulation

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

+ nchains = 3, niter = 20000, nburnin = 10000,

+ summary = TRUE, WAIC = F,

+ monitors = c("rhat",'dev', "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] 0.11034717 0.09941986 0.05499766 3.578758e-02 0.2470625

T[2] 0.08795571 0.07826558 0.04567648 2.760341e-02 0.2036134

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

d[2] -0.26007944 -0.26025053 0.04956191 -3.567145e-01 -0.1619077

dev[1, 1] 0.57544983 0.25285298 0.81928698 6.792370e-04 2.9053270

dev[2, 1] 1.08213739 0.55643605 1.38132929 1.339765e-03 4.9464223

dev[3, 1] 0.78717122 0.37208993 1.08146311 6.992429e-04 3.8669133

dev[4, 1] 0.61227873 0.27197692 0.86619089 6.269715e-04 3.0718307

dev[5, 1] 1.14982084 0.63672768 1.41419228 1.680804e-03 5.0280312

dev[6, 1] 0.71066495 0.33946287 0.97382597 6.642655e-04 3.5196392

dev[7, 1] 2.18358771 1.59081264 2.13233884 8.454134e-03 7.7891802

dev[8, 1] 0.90151975 0.47231257 1.13363108 1.083360e-03 4.0360325

dev[9, 1] 0.77958013 0.36638303 1.06476134 8.358113e-04 3.7696488

dev[10, 1] 0.83046648 0.38850685 1.13621702 1.024690e-03 3.9995408

dev[11, 1] 0.51862102 0.23773686 0.76163032 5.191664e-04 2.6871269

dev[12, 1] 0.95489700 0.49741920 1.22590006 1.199024e-03 4.4053647

dev[13, 1] 0.91206360 0.44705625 1.21095906 1.154763e-03 4.3411282

dev[14, 1] 3.77259631 3.17555990 2.84299293 1.182847e-01 10.8487339

dev[15, 1] 0.55698237 0.24784164 0.80360200 4.962051e-04 2.8523102

dev[16, 1] 0.65825702 0.30207209 0.92830814 5.884958e-04 3.2538456

dev[17, 1] 1.12975317 0.75772372 1.19582529 2.764489e-03 4.3026409

dev[18, 1] 0.96859438 0.52204332 1.21055955 1.055872e-03 4.3866976

dev[19, 1] 0.91696804 0.51101132 1.11117416 1.483767e-03 4.0004514

dev[20, 1] 0.58119579 0.26753184 0.83838561 5.218427e-04 2.9571276

dev[21, 1] 1.37421667 0.82992706 1.57022690 2.285848e-03 5.7147320

dev[22, 1] 1.36929716 0.81018223 1.59219743 2.083514e-03 5.7702701

dev[1, 2] 0.58926352 0.26586818 0.82889433 6.003557e-04 2.9924069

dev[2, 2] 0.95527528 0.57717181 1.10073655 1.679610e-03 3.9123719

dev[3, 2] 0.49591967 0.23229689 0.68361206 5.487771e-04 2.4375933

dev[4, 2] 0.53096825 0.24036062 0.75824276 5.215634e-04 2.6524196

dev[5, 2] 1.27622598 0.82281460 1.40108975 2.521285e-03 5.0184898

dev[6, 2] 0.54984075 0.24597148 0.78141372 4.670829e-04 2.8294571

dev[7, 2] 2.28493875 1.77424533 2.03417547 1.574613e-02 7.5405400

dev[8, 2] 0.92904596 0.47692999 1.18010327 1.118660e-03 4.2426628

dev[9, 2] 0.65313017 0.31307511 0.88542308 7.739122e-04 3.1896571

dev[10, 2] 0.74684699 0.35982724 1.01380847 8.440909e-04 3.5808791

dev[11, 2] 0.60389883 0.27497113 0.86730395 6.376615e-04 3.0476226

dev[12, 2] 1.02512728 0.58298753 1.22425007 1.426617e-03 4.3367594

dev[13, 2] 0.71904730 0.36521177 0.92773740 1.008439e-03 3.3762541

dev[14, 2] 4.33858387 3.84829687 2.89338443 2.685558e-01 11.3390986

dev[15, 2] 0.49531876 0.22557355 0.70193333 4.492670e-04 2.5277278

dev[16, 2] 0.62697508 0.29277798 0.86712078 6.861328e-04 3.0560380

dev[17, 2] 1.15671062 0.61147254 1.45380017 1.289820e-03 5.1641760

dev[18, 2] 1.16195741 0.69860476 1.34190411 2.101898e-03 4.8781944

dev[19, 2] 1.12484099 0.61304594 1.36473920 1.341993e-03 4.8847694

dev[20, 2] 0.49072942 0.22570245 0.69921086 4.518002e-04 2.4290282

dev[21, 2] 1.36308821 0.89938890 1.45850687 2.801363e-03 5.1961470

dev[22, 2] 1.34973426 0.91029721 1.41575129 4.309137e-03 5.1156791

rhat[1, 1] 3.37884257 3.23584427 1.31445100 1.248677e+00 6.3775114

rhat[2, 1] 11.80649201 11.67825858 2.41146167 7.460991e+00 16.8400649

rhat[3, 1] 10.06359217 9.89207872 2.37822783 5.934679e+00 15.2246212

rhat[4, 1] 127.70980399 127.62416841 8.44198077 1.113710e+02 144.4401861

rhat[5, 1] 31.16568242 31.00477770 4.08330093 2.376862e+01 39.6390412

rhat[6, 1] 5.27109713 5.11654807 1.57748223 2.606898e+00 8.7613339

rhat[7, 1] 138.34093887 138.13521923 8.36885584 1.222908e+02 155.0008300

rhat[8, 1] 52.47832623 52.32403207 4.81301140 4.349545e+01 62.2515930

rhat[9, 1] 34.80101299 34.67983734 4.23988506 2.697801e+01 43.5575923

rhat[10, 1] 182.49296971 182.31319084 10.21084799 1.631755e+02 203.3044471

rhat[11, 1] 53.28455560 53.11814185 4.89934709 4.434122e+01 63.5422826

rhat[12, 1] 51.11708501 51.06051645 4.84394088 4.195031e+01 60.9449272

rhat[13, 1] 14.01807598 13.86118602 2.71009873 9.178204e+00 19.9408185

rhat[14, 1] 57.83659156 57.60340781 5.75946626 4.718313e+01 69.8096195

rhat[15, 1] 30.34014023 30.25576169 3.55454822 2.352150e+01 37.4641622

rhat[16, 1] 39.78943407 39.68206841 4.31393033 3.162023e+01 48.5289290

rhat[17, 1] 15.17547522 15.05431913 2.27285911 1.105499e+01 19.9484632

rhat[18, 1] 7.91958467 7.75255307 2.04210828 4.355280e+00 12.4115193

rhat[19, 1] 4.48364203 4.33084349 1.46368214 2.096270e+00 7.7641503

rhat[20, 1] 40.61299297 40.45208989 4.35682922 3.253295e+01 49.6364637

rhat[21, 1] 37.87760751 37.70362294 4.36416262 2.974742e+01 46.8420540

rhat[22, 1] 34.08645089 33.90081893 4.31265008 2.611475e+01 43.0075443

rhat[1, 2] 2.59743491 2.47668854 1.03168156 9.460079e-01 4.9594139

rhat[2, 2] 9.17003908 9.06538090 1.92692471 5.739476e+00 13.2112672

rhat[3, 2] 5.91277991 5.80134648 1.44054031 3.433382e+00 9.0403070

rhat[4, 2] 101.27433362 101.10473997 7.01806950 8.793447e+01 115.3478126

rhat[5, 2] 23.84783698 23.66747186 3.21242148 1.802042e+01 30.5204703

rhat[6, 2] 4.72978625 4.58586209 1.44854165 2.322239e+00 8.0084413

rhat[7, 2] 111.11394301 110.94905728 7.14316345 9.774295e+01 125.7094515

rhat[8, 2] 55.72389886 55.52344910 5.23646334 4.598350e+01 66.3815418

rhat[9, 2] 27.23347204 27.10035650 3.42712435 2.092758e+01 34.3630404

rhat[10, 2] 143.47780873 143.38125601 8.43928548 1.275359e+02 160.3536731

rhat[11, 2] 62.83305808 62.65549660 5.77023027 5.211815e+01 74.7227338

rhat[12, 2] 40.77942809 40.64859547 4.06828018 3.326167e+01 49.0478296

rhat[13, 2] 10.86011849 10.72011786 2.13886734 7.072615e+00 15.5103423

rhat[14, 2] 44.00015607 43.79442801 4.50256447 3.566641e+01 53.3160835

rhat[15, 2] 25.74274872 25.66828024 3.19475468 1.971520e+01 32.2448700

rhat[16, 2] 31.16454077 31.05571735 3.55243271 2.456454e+01 38.4703035

rhat[17, 2] 24.78715494 24.58740973 3.77062231 1.802046e+01 32.7681733

rhat[18, 2] 6.06448507 5.93455334 1.58745100 3.314228e+00 9.5529910

rhat[19, 2] 4.52761293 4.37823214 1.49242636 2.100600e+00 7.8925407

rhat[20, 2] 31.37537747 31.24940128 3.52789327 2.485815e+01 38.6722157

rhat[21, 2] 32.14599930 31.99721965 3.79638730 2.515995e+01 39.9236072

rhat[22, 2] 26.83346901 26.67406583 3.46308098 2.040848e+01 34.0646613

totresdev 46.79358690 46.14869349 6.84717804 3.539046e+01 62.0688728

>

>

>

> #### 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%

0.03576493 0.24742407

>

>

> ## 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%

0.02767132 0.20352924

>

>

> ## 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%

-0.3562945 -0.1614981

>

>

>

>

>

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

>

>

>

>

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

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

>

>

> Hilf\_data = read.table("Hilf4.txt", sep = "", header=F, skip=295, 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,] 0.5819499 0.5611673 0.5715586 -0.7560150 0.7560150

[2,] 1.1086951 0.9525510 1.0306230 -1.0151961 1.0151961

[3,] 0.7657756 0.5017997 0.6337876 -0.7961078 0.7961078

[4,] 0.6270303 0.5273036 0.5771670 -0.7597151 0.7597151

[5,] 1.1585135 1.2803679 1.2194407 -1.1042829 1.1042829

[6,] 0.6963009 0.5587981 0.6275495 -0.7921802 0.7921802

[7,] 2.1456722 2.3208272 2.2332497 -1.4944061 1.4944061

[8,] 0.8800034 0.9848648 0.9324341 -0.9656263 0.9656263

[9,] 0.7373075 0.6515783 0.6944429 -0.8333324 0.8333324

[10,] 0.8249832 0.7469174 0.7859503 -0.8865384 0.8865384

[11,] 0.4935526 0.6075663 0.5505594 -0.7419969 0.7419969

[12,] 0.9494872 1.0342638 0.9918755 -0.9959295 0.9959295

[13,] 0.8953372 0.7496398 0.8224885 -0.9069115 0.9069115

[14,] 3.8445346 4.2727788 4.0586567 -2.0146108 2.0146108

[15,] 0.5661682 0.4919127 0.5290405 -0.7273517 0.7273517

[16,] 0.6481206 0.6172527 0.6326866 -0.7954160 0.7954160

[17,] 1.1605047 1.1187594 1.1396320 -1.0675355 1.0675355

[18,] 0.9857189 1.1779321 1.0818255 -1.0401084 1.0401084

[19,] 0.8923617 1.1615135 1.0269376 -1.0133793 1.0133793

[20,] 0.5723833 0.4816460 0.5270146 -0.7259577 0.7259577

[21,] 1.3666741 1.3661936 1.3664338 -1.1689456 1.1689456

[22,] 1.3471369 1.3576035 1.3523702 -1.1629145 1.1629145

>

>

>

> # 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] 24.85873

>

>

>

> #### 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] 0.5146399 0.6694146 0.6802073 0.6198925 0.5217513 0.5976062 0.6585068 0.4732785 0.5928667 0.6433575 0.4641287 0.5373145 0.6426814

[14] 0.4791496 0.5439285 0.5442367 0.3147311 0.4484553 0.3453144 0.5601325 0.6293848 0.6534904

>

>

> 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] 0.5140408 0.3603175 0.3313686 0.5228412 0.5365556 0.4213011 0.4748927 0.5903627 0.4347549 0.5270113 0.5775503 0.5155600 0.3737067

[14] 0.6111146 0.4715299 0.4926147 0.7034193 0.5915413 0.6864655 0.4667768 0.4223726 0.3823650

>

>

>

> #### 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 fixed 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 ########## ########## ##########