> ########## ########## Simulation Blocker Beispiel mit Fixed Effects mit dem rjags package

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

>

>

> # Teil Simulation mit JAGS ------------------------------------------------------------------

>

>

> ##### Clear data

> rm(list=ls())

>

>

> ##### load libraries

> library(rjags)

> library(coda)

> library(random)

> library(matrixStats) # zusätzl Paket, berechnet Median

> load.module("glm")

> load.module("lecuyer")

> load.module("dic")

>

> ##### Sichergehen richtiger Working directory

> setwd("C:/Users/IvanB/Desktop/Masterarbeit/Statistische Programme und Gibbs Sampler/Programm JAGS/Nachrechnen TSD2-Dokument/Nachrechnen mit rjags/Blocker")

>

>

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

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

> head(data) # Shows the first six entries

V1 V2 V3 V4 V5 V6 V7

[1,] 3 3 39 38 1 2 2

[2,] 14 7 116 114 1 2 2

[3,] 11 5 93 69 1 2 2

[4,] 127 102 1520 1533 1 2 2

[5,] 27 28 365 355 1 2 2

[6,] 6 4 52 59 1 2 2

> data2 = read.table("Data\_Blocker\_Rest.txt")

> head(data2) # Shows the first six entries

V1 V2 V3

1 nt <- 2

2 ns <- 22

>

>

> ##### Values for simulation, prepare dat for JAGS (allocation values from data)

> ns <- nrow(data)

> # ns # check

> nt <- ncol(data[,5:6])

> #nt # check

>

> na <- data[,7]

> # na # check

> r <- data[,1:2]

> # r # Check

> n <- data[,3:4]

> # n # Check

> t <- data[,5:6]

> # t # Check

> dat <- list(ns=ns, nt=nt, na=na, r=r, n=n, t=t) # names list of numbers

>

>

> ##### read in inits with chains

> inits <- function(){

+ #chain 1

+ list(d=c( NA, 0),

+ sd=1,

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

+ .RNG.name="lecuyer::RngStream", .RNG.seed=1

+ )

+ #chain 2

+ list(d=c( NA, -1),

+ sd=4,

+ mu=c(-3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3, -3),

+ .RNG.name="lecuyer::RngStream", .RNG.seed=2

+ )

+ #chain 3

+ list(d=c( NA, 2),

+ sd=2,

+ mu=c(-3, 5, -1, -3, 7, -3, -4, -3, -3, 0, -3, -3,0, 3, 5, -3, -3, -1, -3, -7, -3, -3),

+ .RNG.name="lecuyer::RngStream", .RNG.seed=3

+ )

+ }

>

>

> ##### define JAGS model within R

> cat("model{ # \*\*\* PROGRAM STARTS

+ 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

+ logit(p[i,k]) <- 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[]) #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

+

+ # Provide estimates of treatment effects T[k] on the natural (probability) scale

+ # Given a Mean Effect, meanA, for 'standard' treatment 1, with precision (1/variance) precA

+

+ for (k in 1:nt) { logit(T[k]) <- A + d[k] }

+ A ~ dnorm(-2.2, 3.3) }",

+ file="Blocker\_Fixed.txt")

>

>

> ##### Set up the JAGS model and settings

> jags.m <- jags.model( file = "Blocker\_Random.txt", data=dat, inits=inits, n.chains=3, n.adapt=1500)

Compiling model graph

Resolving undeclared variables

Allocating nodes

Graph information:

Observed stochastic nodes: 44

Unobserved stochastic nodes: 47

Total graph size: 964

Initializing model

|++++++++++++++++++++++++++++++++++++++++++++++++++| 100%

>

>

> ##### Initialization

> update(jags.m, 10000) # burn in

|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| 100%

>

>

> ##### run JAGS and save posterior samples

> samps\_coda <- coda.samples( jags.m, variable.names=c("d[2]", "T[1]", "T[2]" ), n.iter=30000, DIC=T )

|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| 100%

> samps\_jags <- jags.samples( jags.m, variable.names=c("rhat", "dev", "totresdev", "deviance"), n.iter=20000, DIC=T )

|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| 100%

>

>

>

> # Ausgabe posteriore Werte, Berechnung Median und Berechnung DIC --------------------------

>

>

> ##### summarize posterior samples # funktioniert nur mit coda.samples

> summary(window(samps\_coda, start=10001)) # burnin = 10000

Iterations = 11501:41500

Thinning interval = 1

Number of chains = 3

Sample size per chain = 30000

1. Empirical mean and standard deviation for each variable,

plus standard error of the mean:

Mean SD Naive SE Time-series SE

T[1] 0.1111 0.05492 0.0001831 0.0001831

T[2] 0.0894 0.04612 0.0001537 0.0001601

d[2] -0.2505 0.06513 0.0002171 0.0008655

2. Quantiles for each variable:

2.5% 25% 50% 75% 97.5%

T[1] 0.03638 0.07145 0.10029 0.1389 0.2481

T[2] 0.02830 0.05625 0.07986 0.1118 0.2051

d[2] -0.37410 -0.29322 -0.25283 -0.2086 -0.1169

Warning messages:

1: In FUN(X[[i]], ...) : start value not changed

2: In FUN(X[[i]], ...) : start value not changed

3: In FUN(X[[i]], ...) : start value not changed

> # Anmerkung: mean für totresdev ist D\_res

>

>

> #### Trace Monitore

> #plot(samps\_coda)

>

>

> #### Median-Berechnung

> # Median für T1

> median(as.matrix(samps\_coda[,1]))

[1] 0.1002863

> # Median für T2

> median(as.matrix(samps\_coda[,2]))

[1] 0.07985646

> # Median für d[2]

> median(as.matrix(samps\_coda[,3]))

[1] -0.2528315

>

>

>

> ##### Berechnung DIC und pD

> D\_res <- summary(samps\_jags$totresdev[])

> D\_res

Min. 1st Qu. Median Mean 3rd Qu. Max.

15.73 36.18 41.39 41.80 47.02 79.72

> pD <- var(samps\_jags$deviance)/2

> pD

[1] 32.81885

> DIC = D\_res + pD

> DIC

Min. 1st Qu. Median Mean 3rd Qu. Max.

48.55 69.00 74.21 74.62 79.84 112.54

>

>

>

>

>

> # Berechnung Daten für leverage plot --------------------------------------

>

>

>

> #### Berechnung w\_ik

> samps\_jags[["dev"]]

mcarray:

[,1] [,2]

[1,] 0.5704196 0.5837087

[2,] 1.0598899 0.9395781

[3,] 0.8058415 0.5291042

[4,] 0.7878088 0.7219621

[5,] 1.0035127 1.1121387

[6,] 0.7445475 0.5629872

[7,] 1.3975836 1.4015318

[8,] 0.8540912 0.9117650

[9,] 0.7893099 0.6998284

[10,] 0.8805679 0.8194120

[11,] 0.6747846 0.7363280

[12,] 0.8853054 0.9241845

[13,] 0.8955622 0.7318270

[14,] 2.4051989 2.6628031

[15,] 0.6539477 0.5911788

[16,] 0.7068218 0.6869815

[17,] 0.9841288 1.0722220

[18,] 0.9238361 1.1052414

[19,] 0.8735042 1.0941265

[20,] 0.6597004 0.5886377

[21,] 1.2186652 1.1608722

[22,] 1.2155250 1.1692643

Marginalizing over: iteration(20000),chain(3)

> out\_dev <- capture.output(samps\_jags[["dev"]])

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

> Hilf = read.table("Hilf.txt", sep = "", header=F, skip=3, nrows=22, colClasses=c("NULL",NA,NA))

> Hilf

V2 V3

1 0.5706513 0.5815560

2 1.0558593 0.9388328

3 0.8015667 0.5246854

4 0.7725797 0.7154999

5 1.0014321 1.1030216

6 0.7375351 0.5629156

7 1.3883270 1.3844334

8 0.8491170 0.9121404

9 0.7904761 0.6910634

10 0.8869090 0.8128582

11 0.6752515 0.7425765

12 0.8924509 0.9312074

13 0.9068698 0.7340634

14 2.3986768 2.6320531

15 0.6557401 0.5925528

16 0.7092796 0.6873408

17 0.9873701 1.0763314

18 0.9307312 1.1088711

19 0.8727445 1.0995119

20 0.6629114 0.5939255

21 1.2067310 1.1441666

22 1.2225503 1.1593378

> cbind(Hilf, total = rowMeans(Hilf))

V2 V3 total

1 0.5706513 0.5815560 0.5761036

2 1.0558593 0.9388328 0.9973460

3 0.8015667 0.5246854 0.6631261

4 0.7725797 0.7154999 0.7440398

5 1.0014321 1.1030216 1.0522268

6 0.7375351 0.5629156 0.6502253

7 1.3883270 1.3844334 1.3863802

8 0.8491170 0.9121404 0.8806287

9 0.7904761 0.6910634 0.7407698

10 0.8869090 0.8128582 0.8498836

11 0.6752515 0.7425765 0.7089140

12 0.8924509 0.9312074 0.9118291

13 0.9068698 0.7340634 0.8204666

14 2.3986768 2.6320531 2.5153649

15 0.6557401 0.5925528 0.6241465

16 0.7092796 0.6873408 0.6983102

17 0.9873701 1.0763314 1.0318508

18 0.9307312 1.1088711 1.0198011

19 0.8727445 1.0995119 0.9861282

20 0.6629114 0.5939255 0.6284185

21 1.2067310 1.1441666 1.1754488

22 1.2225503 1.1593378 1.1909441

> Hilf2 <- cbind(Hilf, total = rowMeans(Hilf))

> w\_ik <- sqrt(Hilf2[,3])

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

> fertige\_Daten\_w\_ik <- cbind(Hilf2, w\_ik\_neg, w\_ik)

> fertige\_Daten\_w\_ik

V2 V3 total w\_ik\_neg w\_ik

1 0.5706513 0.5815560 0.5761036 -0.7590149 0.7590149

2 1.0558593 0.9388328 0.9973460 -0.9986721 0.9986721

3 0.8015667 0.5246854 0.6631261 -0.8143255 0.8143255

4 0.7725797 0.7154999 0.7440398 -0.8625774 0.8625774

5 1.0014321 1.1030216 1.0522268 -1.0257811 1.0257811

6 0.7375351 0.5629156 0.6502253 -0.8063655 0.8063655

7 1.3883270 1.3844334 1.3863802 -1.1774465 1.1774465

8 0.8491170 0.9121404 0.8806287 -0.9384182 0.9384182

9 0.7904761 0.6910634 0.7407698 -0.8606798 0.8606798

10 0.8869090 0.8128582 0.8498836 -0.9218913 0.9218913

11 0.6752515 0.7425765 0.7089140 -0.8419703 0.8419703

12 0.8924509 0.9312074 0.9118291 -0.9548975 0.9548975

13 0.9068698 0.7340634 0.8204666 -0.9057961 0.9057961

14 2.3986768 2.6320531 2.5153649 -1.5859902 1.5859902

15 0.6557401 0.5925528 0.6241465 -0.7900294 0.7900294

16 0.7092796 0.6873408 0.6983102 -0.8356496 0.8356496

17 0.9873701 1.0763314 1.0318508 -1.0158005 1.0158005

18 0.9307312 1.1088711 1.0198011 -1.0098520 1.0098520

19 0.8727445 1.0995119 0.9861282 -0.9930399 0.9930399

20 0.6629114 0.5939255 0.6284185 -0.7927285 0.7927285

21 1.2067310 1.1441666 1.1754488 -1.0841812 1.0841812

22 1.2225503 1.1593378 1.1909441 -1.0913038 1.0913038

>

>

> #### Berechnung leverage\_ik

> samps\_jags[["rhat"]]

mcarray:

[,1] [,2]

[1,] 3.362692 2.628965

[2,] 12.027522 9.026363

[3,] 10.134464 5.892523

[4,] 127.264053 101.786402

[5,] 30.270931 24.775766

[6,] 5.268262 4.688813

[7,] 144.375787 105.529956

[8,] 51.035807 57.056029

[9,] 35.239201 26.858888

[10,] 184.987895 140.977477

[11,] 52.742663 63.184131

[12,] 49.939016 42.004184

[13,] 14.294268 10.734940

[14,] 53.960448 47.999972

[15,] 30.318611 25.653777

[16,] 39.364079 31.696099

[17,] 14.657820 25.302587

[18,] 7.765791 6.238981

[19,] 4.394884 4.625982

[20,] 40.405882 31.594761

[21,] 38.978420 31.065322

[22,] 35.066271 25.926837

Marginalizing over: iteration(20000),chain(3)

> out\_rhat <- capture.output(samps\_jags$rhat)

> out\_rhat

[1] "mcarray:" " [,1] [,2]"

[3] " [1,] 3.362692 2.628965" " [2,] 12.027522 9.026363"

[5] " [3,] 10.134464 5.892523" " [4,] 127.264053 101.786402"

[7] " [5,] 30.270931 24.775766" " [6,] 5.268262 4.688813"

[9] " [7,] 144.375787 105.529956" " [8,] 51.035807 57.056029"

[11] " [9,] 35.239201 26.858888" "[10,] 184.987895 140.977477"

[13] "[11,] 52.742663 63.184131" "[12,] 49.939016 42.004184"

[15] "[13,] 14.294268 10.734940" "[14,] 53.960448 47.999972"

[17] "[15,] 30.318611 25.653777" "[16,] 39.364079 31.696099"

[19] "[17,] 14.657820 25.302587" "[18,] 7.765791 6.238981"

[21] "[19,] 4.394884 4.625982" "[20,] 40.405882 31.594761"

[23] "[21,] 38.978420 31.065322" "[22,] 35.066271 25.926837"

[25] "" "Marginalizing over: iteration(20000),chain(3) "

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

> Hilf\_rhat2 = read.table("Hilf\_rhat.txt", sep = "", header=F, skip=3, nrows=22, colClasses=c("NULL",NA,NA))

> Hilf\_rhat2

V2 V3

1 3.364254 2.630551

2 12.027998 9.027631

3 10.126095 5.888140

4 127.292721 101.782507

5 30.285768 24.801898

6 5.280163 4.698170

7 144.416760 105.477633

8 51.015441 57.063491

9 35.246533 26.843760

10 185.008071 140.860166

11 52.757869 63.193893

12 50.002340 42.027341

13 14.279538 10.724532

14 53.953381 48.044468

15 30.335874 25.636792

16 39.362260 31.702319

17 14.654284 25.293966

18 7.769623 6.238681

19 4.389864 4.619811

20 40.419056 31.596941

21 38.990134 31.026440

22 35.079379 25.881697

>

>

> #### Berechnung leverage

> dev\_tilde\_erst\_k1 <- data[,1]\*log(data[,1]/Hilf\_rhat2[,1])

> #dev\_tilde\_erst\_k1

> dev\_tilde\_zweit\_k1 <- (data[,3]-data[,1])\*log((data[,3]-data[,1])/(data[,3]-Hilf\_rhat2[,1]))

> #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] 0.5259964 0.7112558 0.7189794 0.7718445 0.5991617 0.6324308 0.9243244 0.6456608 0.6921651 0.8336231 0.6632295 0.6669178 0.6967962

[14] 0.7267762 0.6375179 0.6509210 0.4095309 0.4717019 0.3622811 0.6575635 0.7581189 0.7757266

>

> dev\_tilde\_erst\_k2 <- data[,2]\*log(data[,2]/Hilf\_rhat2[,2])

> #dev\_tilde\_erst\_k2

> dev\_tilde\_zweit\_k2 <- (data[,4]-data[,2])\*log((data[,4]-data[,2])/(data[,4]-Hilf\_rhat2[,2]))

> #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] 0.5280677 0.4059728 0.3709878 0.7150024 0.6762279 0.4446670 0.7749236 0.7485300 0.5479195 0.7497795 0.7315345 0.6855652 0.4299766

[14] 0.9583637 0.5734491 0.6253025 0.7640619 0.6312274 0.7117447 0.5878893 0.5525718 0.5226721

>

>

> #### Erzeugen leverage plot

> library(car)

> 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 )

> 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)

>

>

>

>

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