> ########## ########## Simulation Blocker Beispiel mit Random Effects mit dem runjags package

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

>

>

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

>

>

> ##### Clear data

> rm(list=ls())

>

>

> ##### load libraries

> library(rjags)

Lade nötiges Paket: coda

Linked to JAGS 4.3.0

Loaded modules: basemod,bugs

> library(runjags)

> library(random)

> library(coda)

> load.module("glm")

module glm loaded

> load.module("lecuyer")

module lecuyer loaded

> load.module("dic")

module dic loaded

>

>

> ##### Sichergehen richtiger Working directory

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

> data2 = read.table("Data\_Blocker\_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

> 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

>

>

> ##### read in inits with chains

> inits1 <- 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)

>

> inits2 <- 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)

>

> inits3 <- 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 )

>

> all.inits <- list(inits1, inits2, inits2)

>

>

> ##### define JAGS model within R

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

+ for(i in 1:ns){ # LOOP THROUGH STUDIES

+ w[i,1] <- 0 # adjustment for multi-arm trials is zero for control arm

+ delta[i,1] <- 0 # treatment effect is zero for control arm

+ 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] + delta[i,k] # 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

+ for (k in 2:na[i]) { # LOOP THROUGH ARMS

+ delta[i,k] ~ dnorm(md[i,k],taud[i,k]) # trial-specific LOR distributions

+ md[i,k] <- d[t[i,k]] - d[t[i,1]] + sw[i,k] # mean of LOR distributions (with multi-arm trial correction)

+ taud[i,k] <- tau \*2\*(k-1)/k # precision of LOR distributions (with multi-arm trial correction)

+ w[i,k] <- (delta[i,k] - d[t[i,k]] + d[t[i,1]]) # adjustment for multi-arm RCTs

+ sw[i,k] <- sum(w[i,1:(k-1)])/(k-1) # cumulative adjustment for multi-arm trials

+ }

+ }

+ totresdev <- sum(resdev[]) #Total Residual Deviance

+ d[1] <- 0 # treatment effect is zero for reference treatment

+ sd ~ dunif(0,5) # vague prior for between-trial SD.

+ tau <- pow(sd,-2) # between-trial precision = (1/between-trial variance)

+

+ # 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 2:nt){ d[k] ~ dnorm(0,0.0001) }

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

+ A ~ dnorm(-2.2, 3.3)

+ } ",

+ file="Blocker\_Random.txt")

>

>

>

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

> jags.m <- run.jags(model="Blocker\_Random.txt", monitor=c("d[2]", "T[1]", "T[2]", "sd" , "totresdev", "deviance", "pd", "dic" , "full.pd" ),

+ data=list("ns"=ns, "nt"=nt, "na"=na, "r"=r, "n"=n, "t"=t) , n.chains=3, inits=all.inits, burnin = 10000, sample = 30000, adapt = 1500)

Compiling rjags model...

Calling the simulation using the rjags method...

Adapting the model for 1500 iterations...

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

Burning in the model for 10000 iterations...

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

Running the model for 30000 iterations...

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

Extending 30000 iterations for pD/DIC estimates...

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

Simulation complete

Calculating summary statistics...

Calculating the Gelman-Rubin statistic for 6 variables....

Finished running the simulation

>

>

> #### optional, falls nicht konvergiert:

> #jags.m <- autorun.jags(model="Blocker\_Random.txt", monitor=c("d[2]", "T[1]", "T[2]", "sd" , "totresdev", "deviance", "pd", "dic" , "full.pd" ),

> # data=list("ns"=ns, "nt"=nt, "na"=na, "r"=r, "n"=n, "t"=t) , n.chains=3,

> # inits=all.inits, burnin = 10000, sample = 30000, adapt = 1500, , max.time="1hr")

>

>

> # Ausgabe posteriore Werte --------------------------------------

>

>

> print(jags.m)

JAGS model summary statistics from 90000 samples (chains = 3; adapt+burnin = 11500):

Lower95 Median Upper95 Mean SD Mode MCerr MC%ofSD SSeff AC.10 psrf

d[2] -0.37106 -0.25107 -0.11356 -0.2507 0.064935 -- 0.00098199 1.5 4373 0.28946 1.0005

T[1] 0.025723 0.10022 0.22006 0.1111 0.054711 -- 0.00030663 0.6 31837 0.00052266 1.0003

T[2] 0.019703 0.07978 0.18 0.089368 0.045956 -- 0.00026946 0.6 29087 0.0033885 1.0002

sd 0.000058089 0.12139 0.28493 0.12935 0.086752 -- 0.0028505 3.3 926 0.72505 1.0057

totresdev 26.568 41.64 58.095 42.013 8.0927 -- 0.10737 1.3 5681 0.10556 1.0005

deviance 244 259.07 275.53 259.44 8.0927 -- 0.10737 1.3 5681 0.10556 1.0005

Model fit assessment:

DIC = 278.0544 (range between chains: 277.9607 - 278.3601)

[PED not available from the stored object]

Estimated effective number of parameters: pD = 18.75494

Total time taken: 2.2 minutes

> summary(jags.m$mcmc) # für 2.5 - 97.5 CrI

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

d[2] -0.25070 0.06496 0.0002165 0.0010081

T[1] 0.11037 0.05430 0.0001810 0.0001785

T[2] 0.08874 0.04551 0.0001517 0.0001521

sd 0.12926 0.08658 0.0002886 0.0026439

totresdev 42.04147 8.08402 0.0269467 0.0944694

deviance 259.47202 8.08402 0.0269467 0.0944694

2. Quantiles for each variable:

2.5% 25% 50% 75% 97.5%

d[2] -0.375118 -0.29406 -0.25129 -0.2101 -0.1177

T[1] 0.036319 0.07088 0.09967 0.1381 0.2453

T[2] 0.028286 0.05594 0.07938 0.1110 0.2030

sd 0.002742 0.06141 0.12095 0.1831 0.3213

totresdev 27.283672 36.45198 41.66907 47.2124 58.9342

deviance 244.714216 253.88252 259.09961 264.6430 276.3647

>

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

>

> #### erneute Simulation für die Erzeugung von dev und rhat

> jags.m\_levPlot <- run.jags(model="Blocker\_Random.txt", monitor=c("dev", "rhat"),

+ data=list("ns"=ns, "nt"=nt, "na"=na, "r"=r, "n"=n, "t"=t) ,

+ n.chains=3, inits=all.inits, burnin = 10000, sample = 20000, adapt = 1000)

Compiling rjags model...

Calling the simulation using the rjags method...

Adapting the model for 1000 iterations...

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

Burning in the model for 10000 iterations...

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

Running the model for 20000 iterations...

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

Simulation complete

Note: Summary statistics were not produced as there are >50 monitored variables

[To override this behaviour see ?add.summary and ?runjags.options] FALSEFinished running the simulation

>

>

> jags.m\_levPlot\_summary <- summary(jags.m\_levPlot$mcmc)

> #jags.m\_levPlot\_summary

>

>

> out\_lePlo <- capture.output( jags.m\_levPlot\_summary)

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

> Hilf\_data = read.table("Hilf.txt", sep = "", header=F, skip=11, nrows=88)

> #Hilf\_data

>

>

> #### Berechnung w\_ik

>

> Hilf\_dev <- Hilf\_data[1:44,2]

> Hilf\_dev

[1] 0.5763 1.0830 0.8079 0.7788 1.0044 0.7395 1.4300 0.8240 0.7994 0.8785 0.6417 0.8922 0.9256 2.3897 0.6596 0.6948 0.9806 0.9077 0.8667

[20] 0.6738 1.2273 1.2377 0.5873 0.9375 0.5221 0.7292 1.1133 0.5733 1.4735 0.9033 0.7006 0.8264 0.7176 0.9295 0.7495 2.6498 0.5901 0.6679

[39] 1.0670 1.1124 1.1014 0.6180 1.1771 1.1723

> 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

>

> # 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] 19.9866

>

> #### Berechnung leverage\_ik

>

> Hilf\_rhat <- Hilf\_data[45:88,2]

> #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

>

> 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

>

>

> #### Erzeugen leverage plot

> library(car)

Lade nötiges Paket: carData

> 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 )

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