> ########## ########## Simulation Blocker Beispiel mit fixed effects mit STAN ########## ##########

>

>

>

> ##### Clear data

> rm(list=ls())

>

>

>

> #### Setting working directory

> setwd("C:/Users/IvanB/Desktop/Masterarbeit/Statistische Programme und Gibbs Sampler/STAN/Nachrechnen TSD2/Blocker")

>

>

> #### Requiering stan

> library("rstan")

Lade nötiges Paket: StanHeaders

Lade nötiges Paket: ggplot2

rstan (Version 2.19.2, GitRev: 2e1f913d3ca3)

For execution on a local, multicore CPU with excess RAM we recommend calling

options(mc.cores = parallel::detectCores()).

To avoid recompilation of unchanged Stan programs, we recommend calling

rstan\_options(auto\_write = TRUE)

For improved execution time, we recommend calling

Sys.setenv(LOCAL\_CPPFLAGS = '-march=native')

although this causes Stan to throw an error on a few processors.

Warning messages:

1: Paket ‘rstan’ wurde unter R Version 3.6.1 erstellt

2: Paket ‘StanHeaders’ wurde unter R Version 3.6.1 erstellt

> library("rstantools")

This is rstantools version 1.5.1

Warning message:

Paket ‘rstantools’ wurde unter R Version 3.6.1 erstellt

> rstan\_options(auto\_write = TRUE)

> #options(mc.cores = parallel::detectCores())

> Sys.setenv(LOCAL\_CPPFLAGS = '-march=native')

>

>

>

> #### Read in data

> data = read.csv2("Blocker\_Data\_neu sortiert\_VI.csv", header=TRUE, sep = ";", quote = "\"", dec = ",", fill = TRUE, comment.char = "")

> #data

>

>

>

> #### Assignment data to stan

> NO =nrow(data)

> NT=max(data$ï..Treatment\_t )

> NS=max(data$Studie)

> n=data$Gesamtanzahl\_n

> r=data$Erfolge\_r

> t=data$ï..Treatment\_t

> s=data$Studie

> base=data$ï..Treatment\_t

>

> data\_list <- list(NO=NO, NT=NT, NS=NS, n=n, r=r, t=t, s=s,base=base)

>

>

>

> #### Read in inits

> inits1 <- function(chain\_id = 1) {

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

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

+ }

>

> inits2 <- function(chain\_id = 2) {

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

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

+ }

>

> inits3 <- function(chain\_id = 3) {

+ list(d=c( NA, 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))

+ }

>

> all.inits <- list(inits1, inits2, inits3)

> #all.inits

>

>

> # Compiling

> m <- stan\_model('Model\_Fixed\_final.stan')

> m <- stan\_model('Model\_Fixed\_final.stan')

>

> # Simulation

> stan\_samples <- sampling(m, data = data\_list, iter=40000, verbose=T, init ="all.inits", chain=4)

CHECKING DATA AND PREPROCESSING FOR MODEL 'Model\_Fixed\_final' NOW.

COMPILING MODEL 'Model\_Fixed\_final' NOW.

STARTING SAMPLER FOR MODEL 'Model\_Fixed\_final' NOW.

SAMPLING FOR MODEL 'Model\_Fixed\_final' NOW (CHAIN 1).

Chain 1:

Chain 1: Gradient evaluation took 0 seconds

Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.

Chain 1: Adjust your expectations accordingly!

Chain 1:

Chain 1:

Chain 1: Iteration: 1 / 40000 [ 0%] (Warmup)

Chain 1: Iteration: 4000 / 40000 [ 10%] (Warmup)

Chain 1: Iteration: 8000 / 40000 [ 20%] (Warmup)

Chain 1: Iteration: 12000 / 40000 [ 30%] (Warmup)

Chain 1: Iteration: 16000 / 40000 [ 40%] (Warmup)

Chain 1: Iteration: 20000 / 40000 [ 50%] (Warmup)

Chain 1: Iteration: 20001 / 40000 [ 50%] (Sampling)

Chain 1: Iteration: 24000 / 40000 [ 60%] (Sampling)

Chain 1: Iteration: 28000 / 40000 [ 70%] (Sampling)

Chain 1: Iteration: 32000 / 40000 [ 80%] (Sampling)

Chain 1: Iteration: 36000 / 40000 [ 90%] (Sampling)

Chain 1: Iteration: 40000 / 40000 [100%] (Sampling)

Chain 1:

Chain 1: Elapsed Time: 57.118 seconds (Warm-up)

Chain 1: 15.064 seconds (Sampling)

Chain 1: 72.182 seconds (Total)

Chain 1:

SAMPLING FOR MODEL 'Model\_Fixed\_final' NOW (CHAIN 2).

Chain 2:

Chain 2: Gradient evaluation took 0 seconds

Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.

Chain 2: Adjust your expectations accordingly!

Chain 2:

Chain 2:

Chain 2: Iteration: 1 / 40000 [ 0%] (Warmup)

Chain 2: Iteration: 4000 / 40000 [ 10%] (Warmup)

Chain 2: Iteration: 8000 / 40000 [ 20%] (Warmup)

Chain 2: Iteration: 12000 / 40000 [ 30%] (Warmup)

Chain 2: Iteration: 16000 / 40000 [ 40%] (Warmup)

Chain 2: Iteration: 20000 / 40000 [ 50%] (Warmup)

Chain 2: Iteration: 20001 / 40000 [ 50%] (Sampling)

Chain 2: Iteration: 24000 / 40000 [ 60%] (Sampling)

Chain 2: Iteration: 28000 / 40000 [ 70%] (Sampling)

Chain 2: Iteration: 32000 / 40000 [ 80%] (Sampling)

Chain 2: Iteration: 36000 / 40000 [ 90%] (Sampling)

Chain 2: Iteration: 40000 / 40000 [100%] (Sampling)

Chain 2:

Chain 2: Elapsed Time: 52.047 seconds (Warm-up)

Chain 2: 15.081 seconds (Sampling)

Chain 2: 67.128 seconds (Total)

Chain 2:

SAMPLING FOR MODEL 'Model\_Fixed\_final' NOW (CHAIN 3).

Chain 3:

Chain 3: Gradient evaluation took 0 seconds

Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.

Chain 3: Adjust your expectations accordingly!

Chain 3:

Chain 3:

Chain 3: Iteration: 1 / 40000 [ 0%] (Warmup)

Chain 3: Iteration: 4000 / 40000 [ 10%] (Warmup)

Chain 3: Iteration: 8000 / 40000 [ 20%] (Warmup)

Chain 3: Iteration: 12000 / 40000 [ 30%] (Warmup)

Chain 3: Iteration: 16000 / 40000 [ 40%] (Warmup)

Chain 3: Iteration: 20000 / 40000 [ 50%] (Warmup)

Chain 3: Iteration: 20001 / 40000 [ 50%] (Sampling)

Chain 3: Iteration: 24000 / 40000 [ 60%] (Sampling)

Chain 3: Iteration: 28000 / 40000 [ 70%] (Sampling)

Chain 3: Iteration: 32000 / 40000 [ 80%] (Sampling)

Chain 3: Iteration: 36000 / 40000 [ 90%] (Sampling)

Chain 3: Iteration: 40000 / 40000 [100%] (Sampling)

Chain 3:

Chain 3: Elapsed Time: 58.963 seconds (Warm-up)

Chain 3: 15.14 seconds (Sampling)

Chain 3: 74.103 seconds (Total)

Chain 3:

SAMPLING FOR MODEL 'Model\_Fixed\_final' NOW (CHAIN 4).

Chain 4:

Chain 4: Gradient evaluation took 0 seconds

Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.

Chain 4: Adjust your expectations accordingly!

Chain 4:

Chain 4:

Chain 4: Iteration: 1 / 40000 [ 0%] (Warmup)

Chain 4: Iteration: 4000 / 40000 [ 10%] (Warmup)

Chain 4: Iteration: 8000 / 40000 [ 20%] (Warmup)

Chain 4: Iteration: 12000 / 40000 [ 30%] (Warmup)

Chain 4: Iteration: 16000 / 40000 [ 40%] (Warmup)

Chain 4: Iteration: 20000 / 40000 [ 50%] (Warmup)

Chain 4: Iteration: 20001 / 40000 [ 50%] (Sampling)

Chain 4: Iteration: 24000 / 40000 [ 60%] (Sampling)

Chain 4: Iteration: 28000 / 40000 [ 70%] (Sampling)

Chain 4: Iteration: 32000 / 40000 [ 80%] (Sampling)

Chain 4: Iteration: 36000 / 40000 [ 90%] (Sampling)

Chain 4: Iteration: 40000 / 40000 [100%] (Sampling)

Chain 4:

Chain 4: Elapsed Time: 56.405 seconds (Warm-up)

Chain 4: 14.966 seconds (Sampling)

Chain 4: 71.371 seconds (Total)

Chain 4:

>

>

>

> #### Plotting and summarizing the posterior distribution

> stan\_samples # = print(stan\_samples)

Inference for Stan model: Model\_Fixed\_final.

4 chains, each with iter=40000; warmup=20000; thin=1;

post-warmup draws per chain=20000, total post-warmup draws=80000.

mean se\_mean sd 2.5% 25% 50% 75% 97.5% n\_eff Rhat

d[1] -0.01 0.01 2.13 -4.22 -1.44 -0.01 1.42 4.19 180317 1

d[2] 0.00 0.01 2.13 -4.16 -1.44 0.00 1.44 4.18 178688 1

mu[1] -2.54 0.00 0.44 -3.48 -2.82 -2.51 -2.23 -1.75 150035 1

mu[2] -2.32 0.00 0.23 -2.79 -2.47 -2.31 -2.15 -1.88 171795 1

mu[3] -2.24 0.00 0.27 -2.79 -2.41 -2.23 -2.05 -1.74 168901 1

mu[4] -2.51 0.00 0.07 -2.65 -2.56 -2.51 -2.47 -2.38 165698 1

mu[5] -2.50 0.00 0.14 -2.79 -2.59 -2.50 -2.40 -2.23 169105 1

mu[6] -2.35 0.00 0.34 -3.05 -2.57 -2.34 -2.12 -1.74 154838 1

mu[7] -1.88 0.00 0.07 -2.01 -1.92 -1.88 -1.83 -1.75 171547 1

mu[8] -2.22 0.00 0.10 -2.43 -2.29 -2.22 -2.16 -2.03 173932 1

mu[9] -2.09 0.00 0.14 -2.36 -2.18 -2.09 -2.00 -1.83 172606 1

mu[10] -2.38 0.00 0.06 -2.49 -2.42 -2.38 -2.34 -2.27 175551 1

mu[11] -2.45 0.00 0.10 -2.64 -2.51 -2.45 -2.38 -2.26 172720 1

mu[12] -1.56 0.00 0.11 -1.79 -1.64 -1.56 -1.48 -1.34 179324 1

mu[13] -3.12 0.00 0.21 -3.54 -3.26 -3.12 -2.98 -2.74 163258 1

mu[14] -2.78 0.00 0.10 -2.99 -2.85 -2.78 -2.71 -2.58 180447 1

mu[15] -1.48 0.00 0.15 -1.78 -1.58 -1.48 -1.38 -1.20 173117 1

mu[16] -1.60 0.00 0.13 -1.86 -1.68 -1.60 -1.51 -1.35 172327 1

mu[17] -2.13 0.00 0.17 -2.47 -2.24 -2.12 -2.01 -1.81 172092 1

mu[18] -3.06 0.00 0.28 -3.64 -3.25 -3.05 -2.87 -2.56 157468 1

mu[19] -3.55 0.00 0.35 -4.28 -3.77 -3.53 -3.31 -2.93 152299 1

mu[20] -1.60 0.00 0.13 -1.86 -1.69 -1.60 -1.51 -1.35 164815 1

mu[21] -2.29 0.00 0.13 -2.54 -2.37 -2.28 -2.20 -2.05 174238 1

mu[22] -3.06 0.00 0.13 -3.32 -3.15 -3.06 -2.97 -2.81 172459 1

A[1] -2.20 0.00 0.30 -2.79 -2.41 -2.20 -1.99 -1.60 171722 1

A[2] -2.20 0.00 0.30 -2.80 -2.40 -2.20 -1.99 -1.60 165473 1

d\_II[1] 0.00 NaN 0.00 0.00 0.00 0.00 0.00 0.00 NaN NaN

d\_II[2] 0.00 0.01 2.13 -4.16 -1.44 0.00 1.44 4.18 178688 1

rhat[1] 3.06 0.00 1.19 1.16 2.19 2.92 3.78 5.77 164781 1

rhat[2] 10.62 0.01 2.22 6.70 9.05 10.48 12.05 15.34 175870 1

rhat[3] 9.21 0.01 2.17 5.41 7.67 9.06 10.58 13.92 175779 1

rhat[4] 114.03 0.02 7.19 100.36 109.08 113.90 118.82 128.45 166081 1

rhat[5] 27.89 0.01 3.62 21.21 25.37 27.75 30.22 35.44 169906 1

rhat[6] 4.71 0.00 1.40 2.35 3.70 4.59 5.59 7.79 166169 1

rhat[7] 124.62 0.02 7.33 110.62 119.62 124.51 129.47 139.37 171213 1

rhat[8] 46.15 0.01 4.20 38.25 43.24 46.03 48.92 54.67 174480 1

rhat[9] 31.23 0.01 3.73 24.28 28.66 31.12 33.67 38.87 175165 1

rhat[10] 163.22 0.02 8.64 146.70 157.31 163.07 168.98 180.61 175539 1

rhat[11] 46.47 0.01 4.13 38.73 43.62 46.37 49.18 54.92 173462 1

rhat[12] 46.29 0.01 4.37 38.04 43.27 46.20 49.19 55.17 180899 1

rhat[13] 12.58 0.01 2.45 8.23 10.85 12.42 14.14 17.81 172052 1

rhat[14] 51.77 0.01 4.98 42.46 48.31 51.61 55.06 61.94 181646 1

rhat[15] 27.36 0.01 3.29 21.24 25.08 27.27 29.54 34.09 174867 1

rhat[16] 36.02 0.01 3.91 28.71 33.32 35.90 38.61 44.01 172965 1

rhat[17] 13.10 0.00 1.95 9.52 11.75 13.02 14.37 17.18 172819 1

rhat[18] 7.10 0.00 1.84 3.92 5.78 6.95 8.25 11.09 172581 1

rhat[19] 3.94 0.00 1.29 1.83 3.02 3.81 4.72 6.81 166838 1

rhat[20] 36.77 0.01 3.94 29.37 34.04 36.66 39.38 44.76 165773 1

rhat[21] 33.77 0.01 3.82 26.63 31.13 33.65 36.27 41.66 174548 1

rhat[22] 30.38 0.01 3.78 23.42 27.76 30.24 32.86 38.23 175109 1

rhat[23] 2.99 0.00 1.16 1.13 2.13 2.84 3.68 5.62 164781 1

rhat[24] 10.44 0.01 2.18 6.59 8.90 10.30 11.84 15.08 175870 1

rhat[25] 6.84 0.00 1.61 4.01 5.69 6.72 7.85 10.33 175779 1

rhat[26] 115.01 0.02 7.25 101.22 110.02 114.87 119.84 129.55 166081 1

rhat[27] 27.12 0.01 3.52 20.63 24.67 26.99 29.39 34.47 169906 1

rhat[28] 5.35 0.00 1.59 2.66 4.20 5.21 6.35 8.83 166169 1

rhat[29] 125.42 0.02 7.38 111.32 120.39 125.31 130.30 140.26 171213 1

rhat[30] 61.93 0.01 5.63 51.33 58.03 61.77 65.64 73.36 174480 1

rhat[31] 30.79 0.01 3.68 23.94 28.25 30.68 33.19 38.32 175165 1

rhat[32] 162.79 0.02 8.62 146.32 156.90 162.65 168.54 180.14 175539 1

rhat[33] 69.59 0.01 6.18 57.99 65.32 69.43 73.65 82.24 173462 1

rhat[34] 45.77 0.01 4.32 37.61 42.78 45.68 48.64 54.55 180899 1

rhat[35] 12.49 0.01 2.43 8.18 10.77 12.34 14.05 17.69 172052 1

rhat[36] 50.30 0.01 4.84 41.26 46.94 50.15 53.50 60.19 181646 1

rhat[37] 28.67 0.01 3.45 22.26 26.27 28.56 30.94 35.72 174867 1

rhat[38] 35.01 0.01 3.80 27.91 32.38 34.89 37.52 42.77 172965 1

rhat[39] 26.96 0.01 4.02 19.58 24.17 26.79 29.57 35.34 172819 1

rhat[40] 6.96 0.00 1.81 3.85 5.66 6.81 8.09 10.88 172581 1

rhat[41] 5.12 0.00 1.67 2.38 3.92 4.95 6.13 8.84 166838 1

rhat[42] 35.25 0.01 3.78 28.16 32.63 35.15 37.76 42.92 165773 1

rhat[43] 36.28 0.01 4.10 28.61 33.44 36.14 38.96 44.75 174548 1

rhat[44] 30.65 0.01 3.81 23.63 28.00 30.51 33.15 38.58 175109 1

p\_II[1] 0.08 0.00 0.03 0.03 0.06 0.07 0.10 0.15 164781 1

p\_II[2] 0.09 0.00 0.02 0.06 0.08 0.09 0.10 0.13 175870 1

p\_II[3] 0.10 0.00 0.02 0.06 0.08 0.10 0.11 0.15 175779 1

p\_II[4] 0.08 0.00 0.00 0.07 0.07 0.07 0.08 0.08 166081 1

p\_II[5] 0.08 0.00 0.01 0.06 0.07 0.08 0.08 0.10 169906 1

p\_II[6] 0.09 0.00 0.03 0.05 0.07 0.09 0.11 0.15 166169 1

p\_II[7] 0.13 0.00 0.01 0.12 0.13 0.13 0.14 0.15 171213 1

p\_II[8] 0.10 0.00 0.01 0.08 0.09 0.10 0.10 0.12 174480 1

p\_II[9] 0.11 0.00 0.01 0.09 0.10 0.11 0.12 0.14 175165 1

p\_II[10] 0.08 0.00 0.00 0.08 0.08 0.08 0.09 0.09 175539 1

p\_II[11] 0.08 0.00 0.01 0.07 0.07 0.08 0.08 0.09 173462 1

p\_II[12] 0.17 0.00 0.02 0.14 0.16 0.17 0.18 0.21 180899 1

p\_II[13] 0.04 0.00 0.01 0.03 0.04 0.04 0.05 0.06 172052 1

p\_II[14] 0.06 0.00 0.01 0.05 0.05 0.06 0.06 0.07 181646 1

p\_II[15] 0.19 0.00 0.02 0.14 0.17 0.19 0.20 0.23 174867 1

p\_II[16] 0.17 0.00 0.02 0.13 0.16 0.17 0.18 0.21 172965 1

p\_II[17] 0.11 0.00 0.02 0.08 0.10 0.11 0.12 0.14 172819 1

p\_II[18] 0.05 0.00 0.01 0.03 0.04 0.05 0.05 0.07 172581 1

p\_II[19] 0.03 0.00 0.01 0.01 0.02 0.03 0.04 0.05 166838 1

p\_II[20] 0.17 0.00 0.02 0.13 0.16 0.17 0.18 0.21 165773 1

p\_II[21] 0.09 0.00 0.01 0.07 0.09 0.09 0.10 0.11 174548 1

p\_II[22] 0.05 0.00 0.01 0.03 0.04 0.04 0.05 0.06 175109 1

p\_II[23] 0.08 0.00 0.03 0.03 0.06 0.07 0.10 0.15 164781 1

p\_II[24] 0.09 0.00 0.02 0.06 0.08 0.09 0.10 0.13 175870 1

p\_II[25] 0.10 0.00 0.02 0.06 0.08 0.10 0.11 0.15 175779 1

p\_II[26] 0.08 0.00 0.00 0.07 0.07 0.07 0.08 0.08 166081 1

p\_II[27] 0.08 0.00 0.01 0.06 0.07 0.08 0.08 0.10 169906 1

p\_II[28] 0.09 0.00 0.03 0.05 0.07 0.09 0.11 0.15 166169 1

[ reached getOption("max.print") -- omitted 64 rows ]

Samples were drawn using NUTS(diag\_e) at Sun Aug 18 17:24:00 2019.

For each parameter, n\_eff is a crude measure of effective sample size,

and Rhat is the potential scale reduction factor on split chains (at

convergence, Rhat=1).

> plot(stan\_samples)

'pars' not specified. Showing first 10 parameters by default.

ci\_level: 0.8 (80% intervals)

outer\_level: 0.95 (95% intervals)

> Stan\_summary <- summary(stan\_samples, pars = c("d[2]", "d\_II[2]", "T[1]", "T[2]", "totalresdev"), probs = c(0.025, 0.975))$summary

> Stan\_summary

mean se\_mean sd 2.5% 97.5% n\_eff Rhat

d[2] 0.003436854 5.031261e-03 2.12678999 -4.164392888 4.1759081 178688.07 0.9999909

d\_II[2] 0.003436854 5.031261e-03 2.12678999 -4.164392888 4.1759081 178688.07 0.9999909

T[1] 0.103144175 7.264852e-05 0.02831761 0.057675039 0.1679229 151935.69 1.0000175

T[2] 0.213328332 8.312607e-04 0.25179298 0.001643335 0.8818865 91751.42 0.9999722

totalresdev 73.085977709 3.817809e-02 6.62510736 62.059944876 87.9664714 30113.24 1.0000844

>

>

>

> #### Additional Lines for Median

> Median\_d2 <- median(as.matrix(stan\_samples, pars = c("d\_II[2]")))

> Median\_d2

[1] -0.001420987

> Median\_T1 <- median(as.matrix(stan\_samples, pars = c("T[1]")))

> Median\_T1

[1] 0.09974621

> Median\_T2 <- median(as.matrix(stan\_samples, pars = c("T[2]")))

> Median\_T2

[1] 0.09985972

>

>

>

>

>

>

> # Section for Convergence Diagnostic --------------------------------------------------

>

>

> # konkreter Vergleich mit BUGS nicht möglich ->kein pD in dem Sinne

> # DIC (und damit pD) ist veraltet

> # => loo()-Fkt und WAIC

> #(pD Code für Python)

> #allerdings andere Konvergenz - Diagnostika:

> # Diagnostik mir rstan Paket

> sampler\_params <- get\_sampler\_params(stan\_samples, inc\_warmup = TRUE)

> summary(do.call(rbind, sampler\_params), digits = 2)

accept\_stat\_\_ stepsize\_\_ treedepth\_\_ n\_leapfrog\_\_ divergent\_\_ energy\_\_

Min. :0.00 Min. : 0.01 Min. : 0.0 Min. : 1 Min. :0e+00 Min. : 5973

1st Qu.:0.76 1st Qu.: 0.54 1st Qu.: 3.0 1st Qu.: 7 1st Qu.:0e+00 1st Qu.: 5986

Median :0.88 Median : 0.58 Median : 3.0 Median : 7 Median :0e+00 Median : 5989

Mean :0.84 Mean : 0.63 Mean : 3.2 Mean : 19 Mean :4e-04 Mean : 5991

3rd Qu.:0.97 3rd Qu.: 0.70 3rd Qu.: 3.0 3rd Qu.: 7 3rd Qu.:0e+00 3rd Qu.: 5993

Max. :1.00 Max. :14.39 Max. :10.0 Max. :1023 Max. :1e+00 Max. :16487

> # each chain separately

> lapply(sampler\_params, summary, digits = 2)

[[1]]

accept\_stat\_\_ stepsize\_\_ treedepth\_\_ n\_leapfrog\_\_ divergent\_\_ energy\_\_

Min. :0.00 Min. : 0.014 Min. : 0.0 Min. : 1 Min. :0.00000 Min. : 5973

1st Qu.:0.77 1st Qu.: 0.536 1st Qu.: 3.0 1st Qu.: 7 1st Qu.:0.00000 1st Qu.: 5986

Median :0.89 Median : 0.536 Median : 3.0 Median : 7 Median :0.00000 Median : 5990

Mean :0.84 Mean : 0.622 Mean : 3.2 Mean : 19 Mean :0.00047 Mean : 5990

3rd Qu.:0.97 3rd Qu.: 0.707 3rd Qu.: 3.0 3rd Qu.: 7 3rd Qu.:0.00000 3rd Qu.: 5993

Max. :1.00 Max. :14.386 Max. :10.0 Max. :1023 Max. :1.00000 Max. :13743

[[2]]

accept\_stat\_\_ stepsize\_\_ treedepth\_\_ n\_leapfrog\_\_ divergent\_\_ energy\_\_

Min. :0.00 Min. : 0.01 Min. : 0.0 Min. : 1 Min. :0.00000 Min. : 5974

1st Qu.:0.75 1st Qu.: 0.58 1st Qu.: 3.0 1st Qu.: 7 1st Qu.:0.00000 1st Qu.: 5986

Median :0.88 Median : 0.58 Median : 3.0 Median : 7 Median :0.00000 Median : 5989

Mean :0.84 Mean : 0.64 Mean : 3.2 Mean : 18 Mean :0.00035 Mean : 5992

3rd Qu.:0.97 3rd Qu.: 0.70 3rd Qu.: 3.0 3rd Qu.: 7 3rd Qu.:0.00000 3rd Qu.: 5993

Max. :1.00 Max. :14.39 Max. :10.0 Max. :1023 Max. :1.00000 Max. :15570

[[3]]

accept\_stat\_\_ stepsize\_\_ treedepth\_\_ n\_leapfrog\_\_ divergent\_\_ energy\_\_

Min. :0.00 Min. : 0.015 Min. : 0.0 Min. : 1 Min. :0.00000 Min. : 5973

1st Qu.:0.76 1st Qu.: 0.542 1st Qu.: 3.0 1st Qu.: 7 1st Qu.:0.00000 1st Qu.: 5986

Median :0.89 Median : 0.542 Median : 3.0 Median : 7 Median :0.00000 Median : 5989

Mean :0.84 Mean : 0.622 Mean : 3.2 Mean : 19 Mean :0.00032 Mean : 5991

3rd Qu.:0.97 3rd Qu.: 0.700 3rd Qu.: 3.0 3rd Qu.: 7 3rd Qu.:0.00000 3rd Qu.: 5993

Max. :1.00 Max. :14.386 Max. :10.0 Max. :1023 Max. :1.00000 Max. :16487

[[4]]

accept\_stat\_\_ stepsize\_\_ treedepth\_\_ n\_leapfrog\_\_ divergent\_\_ energy\_\_

Min. :0.00 Min. : 0.012 Min. : 0.0 Min. : 1 Min. :0.00000 Min. : 5974

1st Qu.:0.75 1st Qu.: 0.577 1st Qu.: 3.0 1st Qu.: 7 1st Qu.:0.00000 1st Qu.: 5986

Median :0.88 Median : 0.577 Median : 3.0 Median : 7 Median :0.00000 Median : 5989

Mean :0.84 Mean : 0.639 Mean : 3.2 Mean : 19 Mean :0.00045 Mean : 5990

3rd Qu.:0.97 3rd Qu.: 0.697 3rd Qu.: 3.0 3rd Qu.: 7 3rd Qu.:0.00000 3rd Qu.: 5993

Max. :1.00 Max. :14.386 Max. :10.0 Max. :1023 Max. :1.00000 Max. :12207

> Stan\_summary\_lp\_\_ <- summary(stan\_samples, pars = c("lp\_\_"), probs = c(0.025, 0.975))$summary # sigmasq\_delta entspricht sd, nachher ändern

> Stan\_summary\_lp\_\_

mean se\_mean sd 2.5% 97.5% n\_eff Rhat

lp\_\_ -5976.825 0.02078279 3.613668 -5984.794 -5970.692 30233.53 1.00004

> # weitere Möglichkeit: Package 'shinystan'

>

>

>

>

>

> # Section for leverage plot -----------------------------------------------

>

>

>

> #### Read in single values for dev and rhat

>

> SingeValues\_dev <- summary(stan\_samples, pars = c("dev[1]", "dev[2]", "dev[3]", "dev[4]", "dev[5]", "dev[6]", "dev[7]", "dev[8]", "dev[9]", "dev[10]", "dev[11]", "dev[12]", "dev[13]", "dev[14]", "dev[15]", "dev[16]", "dev[17]", "dev[18]", "dev[19]", "dev[20]", "dev[21]", "dev[22]", "dev[23]", "dev[24]", "dev[25]", "dev[26]", "dev[27]", "dev[28]", "dev[29]", "dev[30]", "dev[31]", "dev[32]", "dev[33]", "dev[34]", "dev[35]", "dev[36]", "dev[37]", "dev[38]", "dev[39]", "dev[40]", "dev[41]", "dev[42]", "dev[43]", "dev[44]"

+ ))$summary

> #SingeValues\_dev

> SingeValues\_rhat <- summary(stan\_samples, pars = c("rhat[1]", "rhat[2]", "rhat[3]", "rhat[4]", "rhat[5]", "rhat[6]", "rhat[7]", "rhat[8]", "rhat[9]", "rhat[10]", "rhat[11]", "rhat[12]", "rhat[13]", "rhat[14]", "rhat[15]", "rhat[16]", "rhat[17]", "rhat[18]", "rhat[19]", "rhat[20]", "rhat[21]", "rhat[22]", "rhat[23]", "rhat[24]", "rhat[25]", "rhat[26]", "rhat[27]", "rhat[28]", "rhat[29]", "rhat[30]", "rhat[31]", "rhat[32]", "rhat[33]", "rhat[34]", "rhat[35]", "rhat[36]", "rhat[37]", "rhat[38]", "rhat[39]", "rhat[40]", "rhat[41]", "rhat[42]", "rhat[43]", "rhat[44]"

+ ))$summary

> #SingeValues\_rhat

>

> out\_lePlo <- capture.output( SingeValues\_dev)

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

> out\_lePlo <- capture.output( SingeValues\_rhat)

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

>

>

>

> Hilf\_dev\_I <- read.table("Hilf\_lePlo.txt", sep = "", header=F, skip=2, nrows=22)

> #Hilf\_dev\_I

> Hilf\_dev\_II <- read.table("Hilf\_lePlo.txt", sep = "", header=F, skip=24, nrows=22)

> #Hilf\_dev\_II

> Hilf\_rhat\_I <- read.table("Hilf\_lePlo.txt", sep = "", header=F, skip=48, nrows=22)

> #Hilf\_rhat\_I

> Hilf\_rhat\_II <- read.table("Hilf\_lePlo.txt", sep = "", header=F, skip=70, nrows=22)

> #Hilf\_rhat\_II

>

>

>

> #### Berechnung w\_ik

> Hilf\_dev\_1 <- cbind(Hilf\_dev\_I[,2], Hilf\_dev\_II[,2])

> #Hilf\_dev\_1

> Hilf\_dev\_2 <- cbind(Hilf\_dev\_1, total = rowMeans(Hilf\_dev\_1))

> #Hilf\_dev\_2

> w\_ik <- sqrt(Hilf\_dev\_2[,3])

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

>

> fertige\_Daten\_w\_ik <- cbind(Hilf\_dev\_2, w\_ik\_neg, w\_ik)

> #fertige\_Daten\_w\_ik

>

>

>

> #### Berechnung leverage\_ik

> dev\_tilde\_erst\_I <- data[1:22,2]\*log(data[1:22,2]/Hilf\_rhat\_I[,2])

> #dev\_tilde\_erst\_I

> dev\_tilde\_zweit\_I <- (data[1:22,3]-data[1:22,2])\*log((data[1:22,3]-data[1:22,2])/(data[1:22,3]-Hilf\_rhat\_I[,2]))

> #dev\_tilde\_zweit\_I

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

> #dev\_tilde\_gesamt\_I

>

> dev\_tilde\_erst\_II <- data[23:44,2]\*log(data[23:44,2]/Hilf\_rhat\_II[,2])

> #dev\_tilde\_erst\_II

> dev\_tilde\_zweit\_II <- (data[23:44,3]-data[23:44,2])\*log((data[23:44,3]-data[23:44,2])/(data[23:44,3]-Hilf\_rhat\_II[,2]))

> #dev\_tilde\_zweit\_II

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

> #dev\_tilde\_gesamt\_II

>

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

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

> #leverage\_I

> #leverage\_II

>

>

>

> #### 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\_I, leverage\_II), 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)

>

>

>

> # manuelle Berechnung von pD ----------------------------------------------

>

>

> #dev ist Std-Abweichung jedes einzelnen Werts

> # insg 22\*2 Wertepaare, also 44 Werte

> Hilf\_dev\_pD <- read.table("Hilf\_lePlo.txt", sep = "", header=F, skip=2, nrows=44)

> #Hilf\_dev\_pD

> Var\_manuell <- sum(Hilf\_dev\_pD[,2])^2/44

> pD\_manuell <- Var\_manuell/2

> pD\_manuell

[1] 19.2368

>

>

>

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