Bayesova statisika

Seminarska naloga

Gregor Vavdi

1/3/2020

1 Primerjava frekventističnega in Bayesovega pristopa pri linearni regersiji

1.1 Cilj

Linearna regerisja je eno najbolj osnovnih statističnih orodij. V zadnjem času prihaja do porasta tudi v Bayesovi statistiki. V tej nalogi bom simuliral različne vrste podatkov, pri čemer me je zanimala predvsem razlika v klasičnem (frekvetističnem) prisotpu in Bayesovem pristopu. Podatke sem generiral z namenom, da me bosta zanimala predvsem kako vrsta slučajne napakez vpliva na pristopa in korelacija med podatki.

1.2 Opis simulacije

Simulacijo sem si zamislil na naslednji način. Najprej generiramo podatke, nato pa na istih podatkih apliciramo oba pristopa linearne regresije. To sem ponovil 1000x in nato pa akumuliral na različne statistike.

V podatkih sem spreminjal dva dejavnika:

1. Slučajno napako - spremenljivka s 4 različnimi vrstami napak: $N(0,1)$, $N(0,100)$, χ^2_1 in χ^2_4
2. Korelacija med spremenljivkami- spremenljivka s 3 različnimi vrednostmi: r = $\{0, 0.4, 0.8\}$. S tem sem preverajal ali korelacija vpliva na ocenjevanje bet in kako je pristop robusten na multikolinearnost spremenljivk.

Podatke sem generiral iz multivariatne normalne porazdelitve s povprečji: $\mu = \{3, 3, 4, 1\}$ in variančno - kovariančno matriko Σ:

$$\Sigma = \begin{bmatrix} 1 & r & r & r \\ r & 1 & r & r \\ r & r & 1 & r \\ r & r & r & 1 \end{bmatrix}$$

pri čemer je r predstavljal spreminjajoč dejavnik - korelacija. Poleg teh podatkov se dodal še eno binarno spremenljivko, ki je bila popolnoma neodvisna od zgornjih podatkov. Generiral sem jo iz $Ber(0.5)$. Velikost vzorca je bila 100.

V okvir simuliranja podatkov sodi tudi teoretične vrednosti za β. Uporabil sem naslednje vrednosti:

$$\beta_0 = -4.5, \quad \beta_1 = 0.8, \quad \beta_2 = 0.6, \quad \beta_3 = 3, \quad \beta_4 = 0, \quad \beta_5 = 1.4$$

Odločil sem se, da je začetna vrednost absolutno nekoliko višja. Močan efekt sem dal spremenljivkama: X_5, X_3, šibek efekt: X_1, X_2 in spremenljivki X_4, nisem dodelil efekta.

Na takšen način sem definiral spremenljivko Y, ki pa sem ji dodal še slučajno napako, ki pa je slučajni dejavnik in ga je bilo moč spreminjati na 4 različnih stopnjah. Vse skupaj sem zapakiral v funkcijo, ki je vidna spodaj:

```
gen.beta.data <- function(n = 100,r=0, napaka ){
  mu <- c(3, 3, 4, 1)
  Sigma <- rbind(c(1, r, r, r),
                 c(r, 1.0, r, r),
                 c(r, r, 1.0, r),
                 c(r, r, r, 1.0))
  podatki <- mvrnorm(n = n, mu = mu, Sigma = Sigma)
  #dodam še binarno spremenljivko
  podatki <- cbind(podatki, rbinom(n, size =0:1, prob=0.5))
  colnames(podatki) <- paste("X", 1:(length(mu)+1), sep="")

  # generiramo ciljno spremenljivko ("odvisno spremenljivko")
  b0 <- -4.5
  b1 <- 0.8
  b2 <- 0.6
  b3 <- 3
  b4 <- 0
  b5 <- 1.4

  y <- b0 + podatki[, "X1"]*b1 + podatki[, "X2"]*b2 +
    podatki[, "X3"]*b3 + podatki[, "X4"]*b4 + podatki[, "X5"]*b5 + napaka
  # zdruzimo podatke
  # data.frame popravi konverzijo formata
  podatki <- data.frame(cbind(podatki, y))
  return(podatki)
}
```

V frekventističnem pristopu sem uporabljal funkcijo `lm`, v Bayesovem pristopu pa sem uporabljal `bayesx` iz paketa `R2BayesX`. Za vsak scenarij sem pogledal konvergenco, če smo izbrali dovolj velik burn-in (200) in pregledali avtokorelacije. Vse to lahko najdete prilogi. Pri vsaki kombinaciji spreminjajočih se dejavnikov sem shranil podatke o ocenjenih paramaterih bet in standardne napake ocen.

1.3 Simulacija

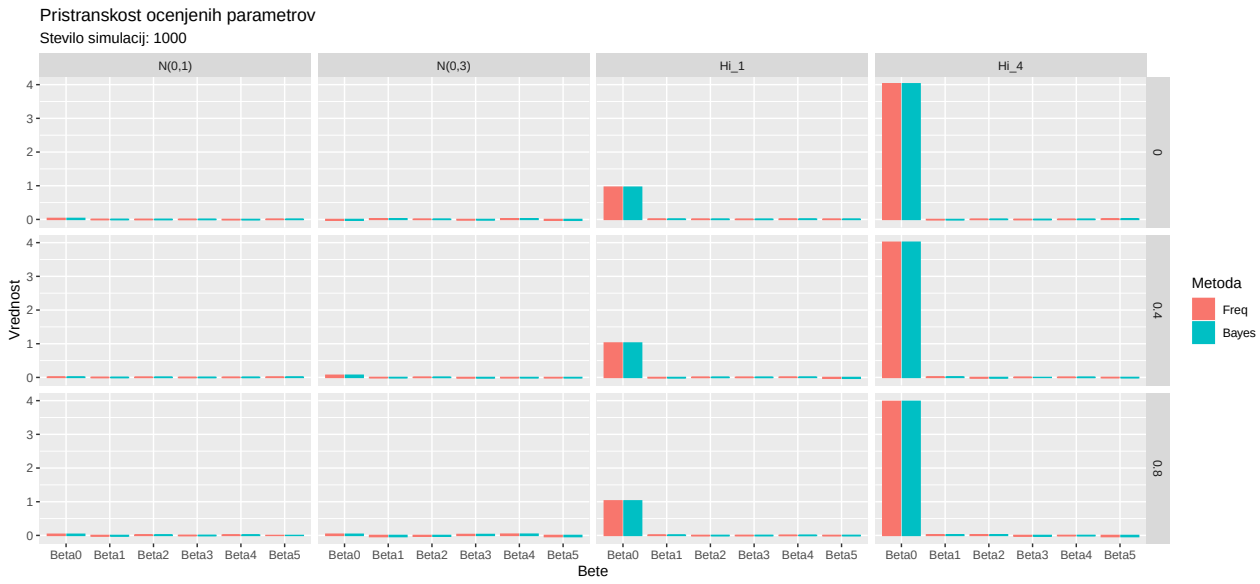
```
pon <- 1000
sampleSize <- 100
vrsta.napake <- c("N(0,1)", "N(0,3)", "Hi_1", "Hi_4")
korelacije <- c(0, 0.4, 0.8)
metoda <- c("Freq", "Bayes")
zasnova <- expand.grid(metoda, vrsta.napake,korelacije)
zasnova.lm <- do.call(rbind, replicate(pon, zasnova, simplify=FALSE)) %>%
  `colnames<-`(c("Metoda","Napaka", "Korelacija"))
zasnova.osnova <- do.call(rbind, replicate(pon,
                                           expand.grid(vrsta.napake,korelacije),
                                           simplify=FALSE)) %>%
  `colnames<-`(c("Napaka",  "Korelacija"))
skupaj.df <- as.data.frame(matrix(NA, nrow = nrow(zasnova.lm), ncol = 9))
table.sd <- as.data.frame(matrix(NA, nrow = nrow(zasnova.lm), ncol = 9))

start.time <- Sys.time()
i <- 1
j <- 1
while(i <= nrow(zasnova.osnova)){
  napaka.i <- as.character(zasnova.osnova[i, "Napaka"])
  if(napaka.i == "N(0,1)"){
    error <- rnorm(sampleSize, 0, 1)
  }
  else if(napaka.i == "N(0,3)"){
    error <- rnorm(sampleSize, 0, 3)
  }
  else if(napaka.i == "Hi_1"){
    error <- rchisq(sampleSize, df = 1)
  }
  else if(napaka.i == "Hi_4"){
    error <- rchisq(sampleSize, df = 4)
  }
  r.i <- zasnova.osnova[i, "Korelacija"]
  #Generiramo podakte
  data.i <- gen.beta.data(n = sampleSize, r= r.i, napaka = error)
  #FREKVENTISTIČNI PRISTOP####
  lm.model <- lm(y~., data = data.i)
  bete.hat <- summary(lm.model)$coef[,1]
  sd.bet <- summary(lm.model)$coef[,2]
  #BAYESOV PRISTOP#####
  bayesx.norm <- bayesx(y ~ X1+X2+X3+X4+X5 , data = data.i,
                        family = "gaussian", method = "MCMC")
```

```
  bayesx.mu <- attr(bayesx.norm$fixed.effects, "sample") #za vsako beto posebaj
  bayes.mu.mean.bet <- apply(bayesx.mu, 2, mean)
  bayes.mu.sd.bet <- apply(bayesx.mu, 2, sd)
  ###SHRANIMO REZULTATE#####
  index.i <- which(zasnova.lm$Napaka == napaka.i &
                     zasnova.lm$Korelacija==r.i& zasnova.lm$Metoda == "Freq")
  index.i.bay <- which(zasnova.lm$Napaka == napaka.i &
                         zasnova.lm$Korelacija==r.i& zasnova.lm$Metoda == "Bayes")
  skupaj.df[j, ] <- cbind(zasnova.lm[index.i.bay, ],
                          "Beta0" = bayes.mu.mean.bet[1],
                          "Beta1" = bayes.mu.mean.bet[2],
                          "Beta2" = bayes.mu.mean.bet[3],
                          "Beta3" = bayes.mu.mean.bet[4],
                          "Beta4" = bayes.mu.mean.bet[5],
                          "Beta5" = bayes.mu.mean.bet[6])
  table.sd[j, ] <- cbind(zasnova.lm[index.i.bay, ],
                         "Beta0" = bayes.mu.sd.bet[1],
                         "Beta1" = bayes.mu.sd.bet[2],
                         "Beta2" = bayes.mu.sd.bet[3],
                         "Beta3" = bayes.mu.sd.bet[4],
                         "Beta4" = bayes.mu.sd.bet[5],
                         "Beta5" = bayes.mu.sd.bet[6])
  skupaj.df[(j+1), ] <- cbind(zasnova.lm[index.i, ],
                          "Beta0" = bete.hat[1],
                          "Beta1" = bete.hat[2],
                          "Beta2" = bete.hat[3],
                          "Beta3" = bete.hat[4],
                          "Beta4" = bete.hat[5],
                          "Beta5" = bete.hat[6])
  table.sd[(j+1), ] <- cbind(zasnova.lm[index.i, ],
                             "Beta0" = sd.bet[1],
                             "Beta1" = sd.bet[2],
                             "Beta2" = sd.bet[3],
                             "Beta3" = sd.bet[4],
                             "Beta4" = sd.bet[5],
                             "Beta5" = sd.bet[6])
  j <- j + 2
  i <- i + 1
}
```

1.4 Rezultati

1.4.1 Pristranskost

```
#Podatki
bete.mean.est <- readRDS("data/simulation_I_bete_mean.RDS")
bete.sd.est <-readRDS("data/simulation_I_bete_sd.RDS")

original.beta <- list("Beta0" = -4.5, "Beta1"= 0.8 ,"Beta2" = 0.6,
                      "Beta3" = 3, "Beta4" = 0, "Beta5" = 1.4)

grupiraj.podatke <- bete.mean.est %>%
  group_by(Korelacija, Metoda, Napaka) %>%
  summarise_all(mean)

#Pristranskost
bias.data <- grupiraj.podatke %>%
  mutate_each(funs(. - original.beta$.), starts_with("Beta")) %>%
  gather(key = "Bete", "Vrednost", -c(Korelacija, Metoda, Napaka))
```



Pri standardno normalni napaki lahko vidimo, da je obe metode zelo dobro opravita nalogo in pristranskosti ni videti. Prav tako, pa tudi povečanje korelacije ne vpliva na pristranskost obeh metod. Tudi ko napaki povečamo standardni odklon na 3, se ne spremeni nič. (V eni od simulacij sem poskusil tudi standardni odklon povečati na 100 in videl, da metodi nista tako stabilni).

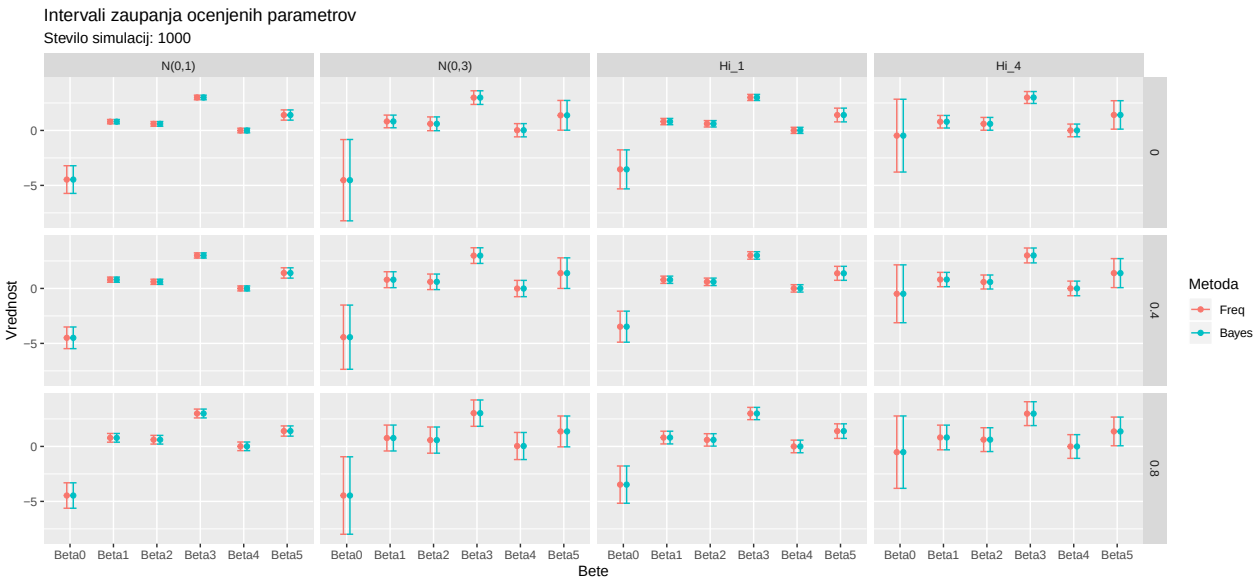
Pri napakah generiranih s χ_2 vidimo, da s povečevanjem parametra povečujemo pristranskost začetne vrednosti - s povečevanjem parametra precenjujemo β_0. Pravtako, pa korelacija ne vpliva na pristranskost metod.

Ko primerjamo obe metodi ocenjevanja parameterov dobimo zanimive in vsaj malo zaskr-

bljujoče podatke. Metodi sta si v vseh scenarijih praktično enako dobri. Manjša odstopanja so pri napaki N(0,100). V poročilu je nisem izbral, ker sem imel sum, da gre za preveliko napako,saj imamo povprečja med 1-10, z standardnim odklonom 100 pa je lahko dobimo zelo naključne rezultate.

1.4.2 Stanardna napaka

```
grupiraj.podatke.sd <- bete.mean.est %>%
  group_by(Korelacija, Metoda, Napaka) %>%
  summarise_all(sd) %>%
  gather(key = "Bete", "SE", -c(Korelacija, Metoda, Napaka))

grupiraj.podatke.mean <- bete.mean.est %>%
  group_by(Korelacija, Metoda, Napaka) %>%
  summarise_all(mean)%>%
  gather(key = "Bete", "Mean", -c(Korelacija, Metoda, Napaka))

podatki.se.mean.join <- grupiraj.podatke.mean %>% left_join(grupiraj.podatke.sd,by = c('
```



Standardne napake narejene na simulacije sem se odločil predstaviti v obliki intervalov zaupanja po normalni porazdelitve, saj sem predpostavil, da velja centralno limitni izrek. Na grafu so tako 95% intrvali zaupanja za povprečne vrednosti koeficientov β pri različni tipih slučajnih napak (stolpci) in korelaciji spremenljivk v podatkih (vrstice).

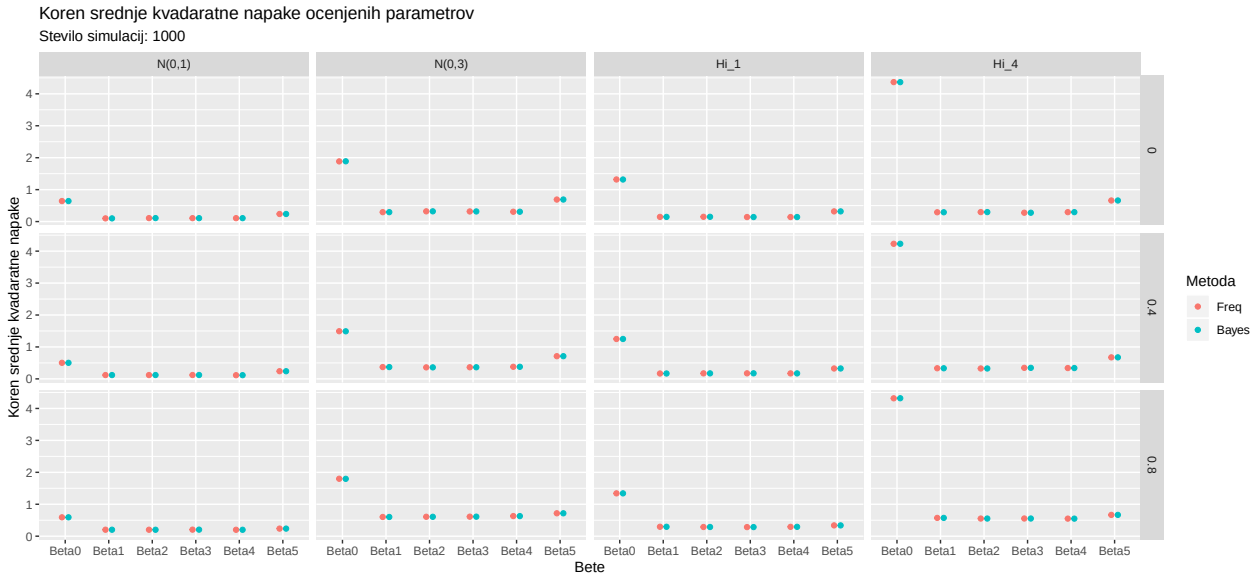
V prvem scenariju - slučajna napaka N(0,1), je opaziti, da so intervali izredno majhni pri vseh 4 povezanih spremenljivkah. Pri spremenljivki X_5 (dihotomna spremenljivka) je standardna napaka nekoliko večja kot pri ostalih. Največja standardna napaka je pri začetnem koeficientu β_0. S povečevanje korelacij oz. kolinearnosti med spremenljivkami X_1 do X_4 vidimo, da se intervali zaupanja nekoliko širšijo. Glede na razlike v prstopih razlik ni moč opaziti. S povečanjem standadnega odklona slučajne napaka, se intevali zaupanja nekoliko razširijo v

primerjavi s scenarijem 1. Še bolj pa k temu pripomore povečanje korelacijskega koeficienta. Standardna napaka je pri β_0 največja.

V drugem delu grafa imamo 2 scenarija χ^2 porazdelitve. Pri prvem imamo 1 stopnjo prostosti in se standardne napake obnašajo podobno kot pri slučajni napaki N(0,1). S povečanjem korelacij se povečuje tudi standardna napaka. Korelacijski koeficient ne vpliva na β_5, kar je pravilno. Pri 4 stopinjah prostosti pa so sicer standardne napake pri ostalih β majhne, vendar pri β_0 teoretična vrednost na zadane intervala zaupanja. Metodi (Bayes in Frekventistčni pristop) imata premajhne razlike, da bi bile opazne.

1.4.3 Koren srednje vrednosti

```
mean.square.error <- function(beta.original, beta.est){
  return(sum((beta.original - beta.est)^2) / length(beta.est))
}
bete.srednje <- bete.mean.est %>% group_by(Korelacija, Metoda, Napaka)%>%
  summarise(Beta0 = sqrt(mean.square.error(original.beta[[1]], Beta0)),
            Beta1 = sqrt(mean.square.error(original.beta[[2]], Beta1)),
            Beta2 = sqrt(mean.square.error(original.beta[[3]], Beta2)),
            Beta3 = sqrt(mean.square.error(original.beta[[4]], Beta3)),
            Beta4 = sqrt(mean.square.error(original.beta[[5]], Beta4)),
            Beta5 = sqrt(mean.square.error(original.beta[[6]], Beta5))) %>%
  gather(key = "Bete", "SrednjeVrednosti", -c(Korelacija, Metoda, Napaka))
```



Koren srednje kvadratne napake sem izračunal na naslednji način:

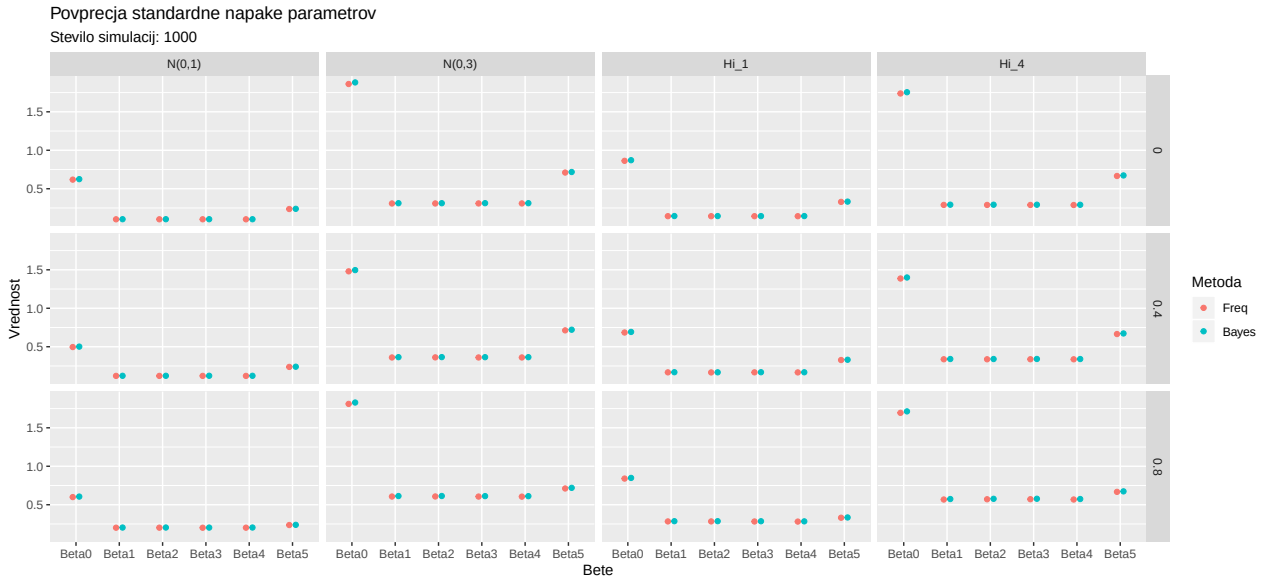
$$SMSE_{\beta_i} = \sqrt{\sum_{j=1}^{1000} (\hat{\beta_{ij}} - \beta_i)^2}$$

Tako sem za vsak koefeicient ter vsako kombinacijo simulacije izračunal SMSE in dobil zgornji

graf. Glavna opazka je, da razlik v pristopih ni in da je največja SMSE pri parametru $beta_0$, ta je še posebno visok pri scenariju z napako χ^2_4. Korelacija ne vpliva na napako SMSE. Pri scenariju z normalno slučajno se pozna vpliv povečanega standardnega odklona, kar se vidi na povečanjiu SMSE pri vseh parameterih. Vpliv neodvisne spremenljivke X_5 - dihotomna spremenljivka, je viden v odskoku vrednosti od ostalih pri scenariju N(0,3).

1.4.4 Povprečje ocenjenih standardnih napak koeficientov

```
grupiraj.podatke.sd <- bete.sd.est %>% group_by(Korelacija, Metoda, Napaka)%>%
  summarise_all(mean) %>%
  gather(key = "Bete", "Vrednost", -c(Korelacija, Metoda, Napaka))
```



Pričakovano podobno se obanašajo tudi povprelja standardnih napak ocenjenih β. Pri začetni vrednosti je povprečje standardne napake največeje in se spreminja z obliko slučajne napake. Pri scenariju N(0,3) se vidi nekolikošna razlika med pristopoma, saj ima Bayesov pristop nekoliko večjo povprečno standardno napako. Pri ostalih scenarijih tega iz grafa ni moč zaznati. Scenarija s χ^2 porazdelitvama imata podobne zaključke kot scenarija s normalno slučajno napako z razliko, da korelacija ne vpliva na velikost povepreči j. Ponovno se pozna efekt neodivsne dihotomne spremenljivke X_5, pri kateri je standardna napaka nekoliko večja kot pri ostalih spremenljivkah. Iz standardnih napak ni razvidno katera od spremenljivk naj bi imela visok ali zmeren efekt.

1.5 Zaključki

V simulaciji sem preveril nekatere statistike (pristranskost, MSME, standardna napaka) dveh različnih pristopov (Freq in Bayes) k ocenjevanju parametrov pri linearni regersiji, pri čemer sem imel dva spreminjajoča dejavnika: korelacijo med spremenljivkami (X_1 do X_4) in različne tipe slučajne napake. Razlik med frekventistčnim in Bayesovim pristopom ni bilo zaslediti. Sam menim, da je to zaradi tega, ker nismo v Bayesovem pristopu upoštevali

dodatnega podatka (prior). Različni tipi slučajnih napak so nakazali, da je to najbolj vpliva na začetno vrednost. S povečevanjem standardnega odklona slučajne napake pri normalni porazdelitvi sem ugotovil, da dokler je standardni odklon v okolici povprečji spremenljivk, linearna regerisja vredu deluje. S povečevanje korealcij oz. kolinearnosti pa povečujemo možnost napak. Pri slučajni napaki tipa χ^2 je opaziti, da korelacija ne vpliva na nobeno od statistik (pristranskost, SMSE, standardna napaka). Omeniti še velja, da pri neodivsni spremenljivki X_5, ki je binarna, SMSE in povprečje standardnih napak povečuje drugače od ostalih spremenljivk, ki so povezana s korelacijskim koeficientom.

2 Primerjava metod za izbor spremenljivk pri linearni regresiji

2.1 Opis

V drugem delu seminarske naloge, bom preveril več različnih metod za izbiro spremenljivk pri linearni regresiji. Predstavil bom 3 metode (metode sem zaradi lažje razumevanja pustil v angleškem zapisu). Prva metoda je *Reversible jump MCMC* s pomočjo programske knjižnice `nimble`. V drugi metodi imenovani Bootstrapped *Augmented Backward Elimination* z pomočjo programske knjižnice `abe`, sem s podobnimi simulacijami pregledoval deleže vrnjenih izbir spremenljivk za različne tipe slučajnih napak, korelacije in metriko (AIC, BIC, p-vrednost). Kot dodatno metoodo pa sem vpeljal še Spike - Slab metodo, pri kateri sem prav tako gledeal delež izbranih spremenljivk.

Podatke sem generiral na enak način kot v prvem delu. Dodal sem še en 5 spremenljivk, s povepreči $\mu = \{6, 17, 22, 12, 5)\}$ in enako variančno - kovariančno matriko kot v prvem delu. Efekt dodatnih spremenljivk sem dodal na 0, saj želimo dobiti bete, katere najbolj opišejo našo odvisno spremenljivko (Y). Vse spremenljivke X, razen binarne X_5 sem centriral, preden sem jo dodal v model.

$$\beta_0 = -4.5, \quad \beta_1 = 0.8, \quad \beta_2 = 0.6, \quad \beta_3 = 3, \quad \beta_4 = 0, \quad \beta_5 = 1.4, \quad \beta_6 = \beta_7 = \beta_8 = \beta_9 = \beta_{10} = 0$$

```
gen.beta.data.2 <- function(n = 100,r=0, napaka){
  mu <- c(3, 3, 4, 1)
  Sigma <- rbind(c(1, r, r, r),
                 c(r, 1.0, r, r),
                 c(r, r, 1.0, r),
                 c(r, r, r, 1.0))
  mu.brez.vpliva <- c(6, 17, 22, 12, 5)
  Sigma.brez.vpliva <- rbind(c(1, r, r, r, r),
                 c(r, 1.0, r, r, r),
                 c(r, r, 1.0, r, r),
                 c(r, r, r, 1.0, r),
                 c(r, r, r, r, 1.0))
```

```
  podatki <- mvrnorm(n = n, mu = mu, Sigma = Sigma)
  podatki <- cbind(podatki, rbinom(n, size =0:1, prob=0.5))
  podatki.brez.vpliva <-mvrnorm(n = n, mu = mu.brez.vpliva,
                                Sigma = Sigma.brez.vpliva)
  podatki <- cbind(podatki, podatki.brez.vpliva)
  colnames(podatki) <- paste("X", 1:ncol(podatki), sep="")
   # generiramo ciljno spremenljivko ("odvisno spremenljivko")
  b0 <- -4.5
  b1 <- 0.8
  b2 <- 0.6
  b3 <- 1.0
  b4 <- 0
  b5 <- 1.4
  b6 <- 0
  b7 <-0
  b8 <- 0
  b9 <-0
  b10 <- 0
  y <- b0 + podatki[, "X1"]*b1 + podatki[, "X2"]*b2 +
    podatki[, "X3"]*b3 + podatki[, "X4"]*b4 + podatki[, "X5"]*b5 +
    podatki[, "X6"]*b6 + podatki[, "X7"]*b7 + podatki[, "X8"]*b8 +
    podatki[, "X9"]*b9 + podatki[, "X10"]*b10 + napaka
  podatki <- data.frame(cbind(podatki,"Y"= y))
  return(podatki)
}
```

2.2 Reversible jump MCMC - NIMBLE

```
grafGG <- function(model, naslov = ""){
  shrani.rez <- as.data.frame(round(samplesSummary(model),2))
  shrani.rez$ime <- rownames(shrani.rez)
  z.izbrani <- shrani.rez %>%
    filter(grepl("z",ime)) %>%
    dplyr::select(Mean) %>%
    mutate(X = paste("X",1:10, sep =""))
  z.izbrani$X <- factor(z.izbrani$X, levels = paste("X", 1:10, sep = ""))
  gg <- ggplot(z.izbrani, aes(x = X, y = Mean)) + geom_bar(stat = "identity")+
    ggtitle(paste("Izbira spremenljivk pri", naslov))
  return(gg)
}

get.rjMCMC <- function(selectX, Y, p=10){
  codeSelect <- nimbleCode({
```

```
    sigma ~ dunif(0, 20)
    psi ~ dunif(0,1)
    beta0 ~ dnorm(0, sd=100)
    for(i in 1:p) {
      z[i] ~ dbern(psi) #indikator za vsak koeficient
      beta[i] ~ dnorm(0, sd = 100)
      zbeta[i] <- z[i] * beta[i]
    }
    for(i in 1:N) {
      y[i] ~ dnorm(beta0 + inprod(X[i, 1:p], zbeta[1:p]), sd = sigma)
    }
  })

  N <- dim(selectX)[1]
  p <- dim(selectX)[2]
  constantsSelect <- list(N = N, p = p)
  initsSelect <- list(sigma = 1, psi = 0.5, beta0 = 0,
                      beta = rnorm(p),
                      z = sample(c(0, 1), p, replace = TRUE))
  dataSelect <- list(y = Y, X = selectX)
  RmodelRJ <- nimbleModel(code = codeSelect, constants = constantsSelect, #isto
                          inits = initsSelect, data = dataSelect)
  confRJ <- configureMCMC(RmodelRJ) #isto
  confRJ$addMonitors('z') #isto
  configureRJ(confRJ,
              targetNodes = 'beta',
              indicatorNodes = 'z',
              control = list(mean = 0, scale = .2))
  RmcmcRJ <- buildMCMC(confRJ)
  CmodelRJ <- compileNimble(RmodelRJ)
  CmcmcRJ <- compileNimble(RmcmcRJ, project = CmodelRJ)
  samplesRJ <- runMCMC(CmcmcRJ, niter = 12000, nburnin = 2000)
  return(samplesRJ)}
```

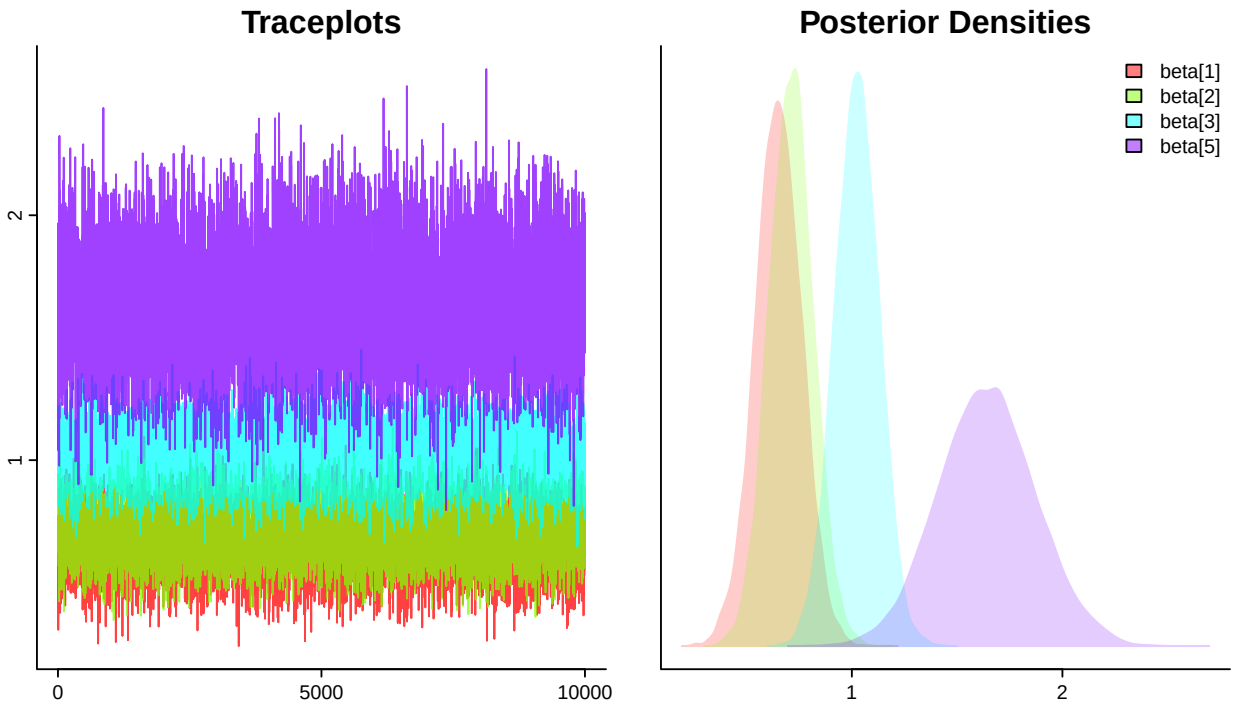


Figure 1: Konvergenca in posteriorne porazdelitve za neničelne bete

2.2.1 Rezultati

Za vsak scenarij bom pregledal konvergenco in posteriorno porazdelitev in pogledal katere spremenljivke bi dodali v model. Konvergenco in posteriorno porazdelitev sem narisal na dveh grafih - tisti, ki imajo ničelni koeficient in tiste, ki imajo neničelne koeficiente na drug graf.

2.2.1.1 Slučajna napaka: N(0,1)

```
data.NIBLE.N1 <- gen.beta.data.2(100, napaka = rnorm(100))
data.NIBLE.N1.selectY <- data.NIBLE.N1$Y
data.NIBLE.N1.selectX <- sweep(data.NIBLE.N1[,-5], 2, colMeans(data.NIBLE.N1[,-5]), FUN=
data.NIBLE.N1.selectX$X5 <- data.NIBLE.N1$X5
data.NIBLE.N1.selectX <- data.NIBLE.N1.selectX[, c(1,2,3,4,10,5,6,7,8,9)]
mod.N1 <- get.rjMCMC(data.NIBLE.N1.selectX, data.NIBLE.N1.selectY)
```

```
## thin = 1: sigma, psi, beta0, beta, z
## |-------------|-------------|-------------|-------------|
## |-----------------------------------------------------|
```

Konvergence so v skladu z pričakovanji in glede na grafičen prikaz za vse parametre model skonvergira. Posteriorne porazdelitve so pri neničelnih betah pravilno porazdelje s povprečji, ki se skladajo s teoretičnimi vrednostmi.

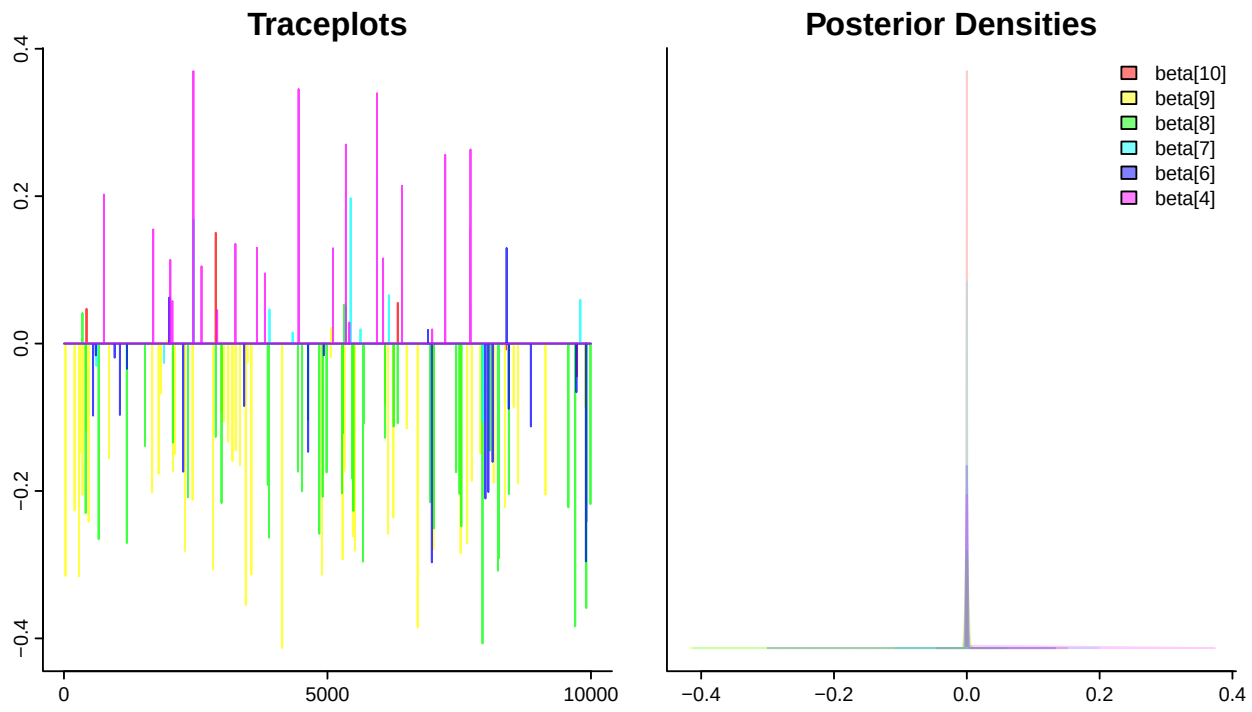
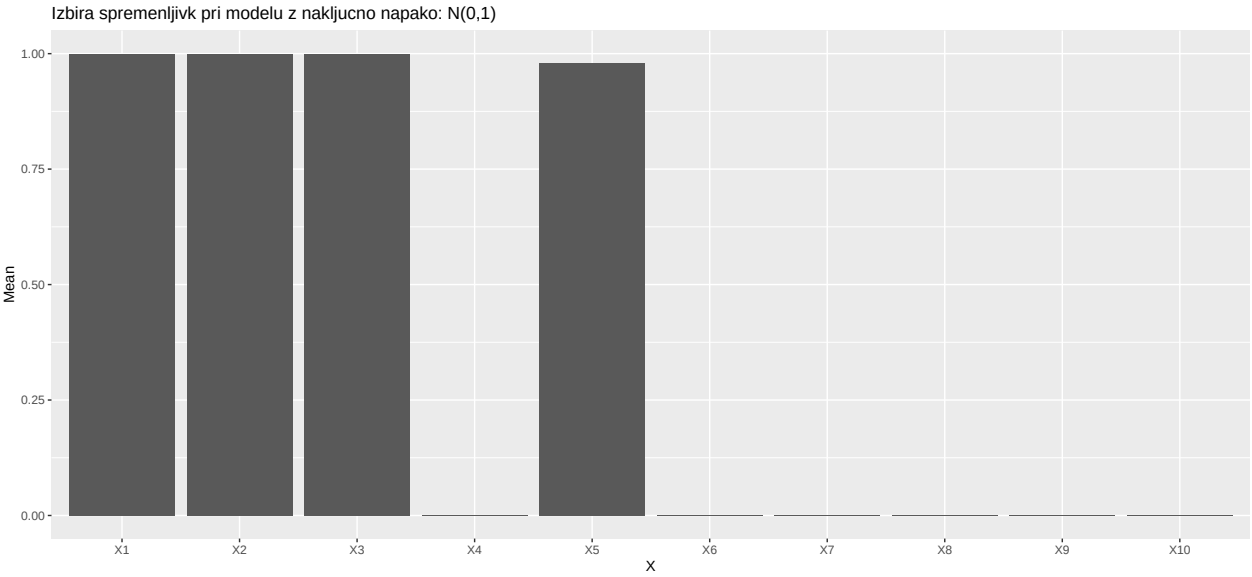


Figure 2: Konvergenca in posteriorne porazdelitve za ničelne bete



Model pravilno izbere vse 4 spremenljivke z zelo visokim deležem verjetja. Tudi ko sem poskusil s povečanjem korealacij, ga to ni zmotilo in se grafa nista razlikovala.

2.2.1.2 Slučajna napaka: N(0,3)

```
data.NIBLE.N3 <- gen.beta.data.2(100, napaka = rnorm(100))
data.NIBLE.N3.selectY <- data.NIBLE.N3$Y
```

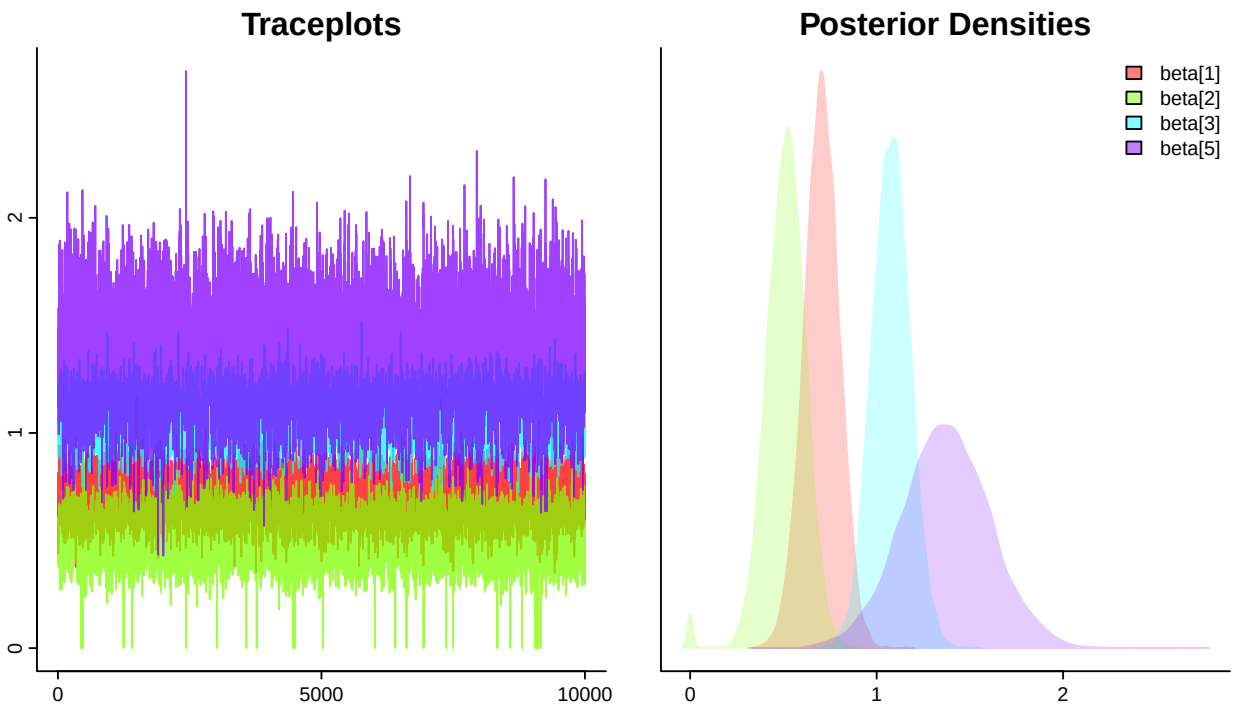
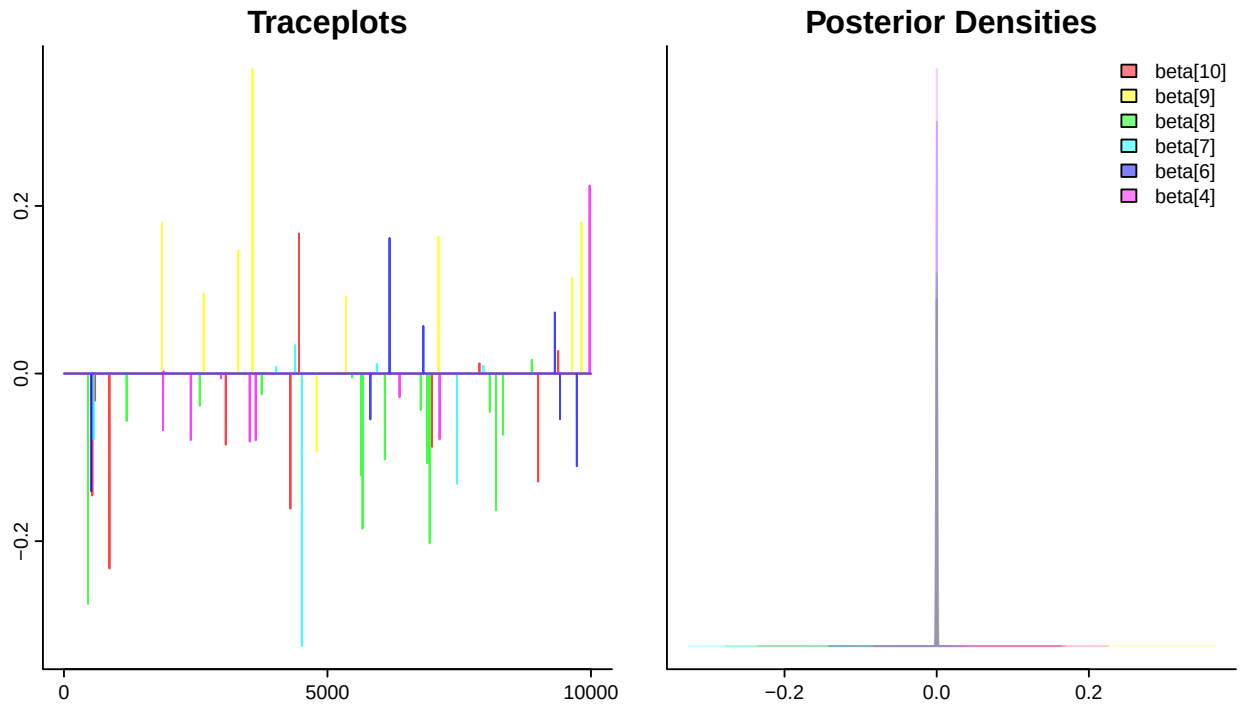


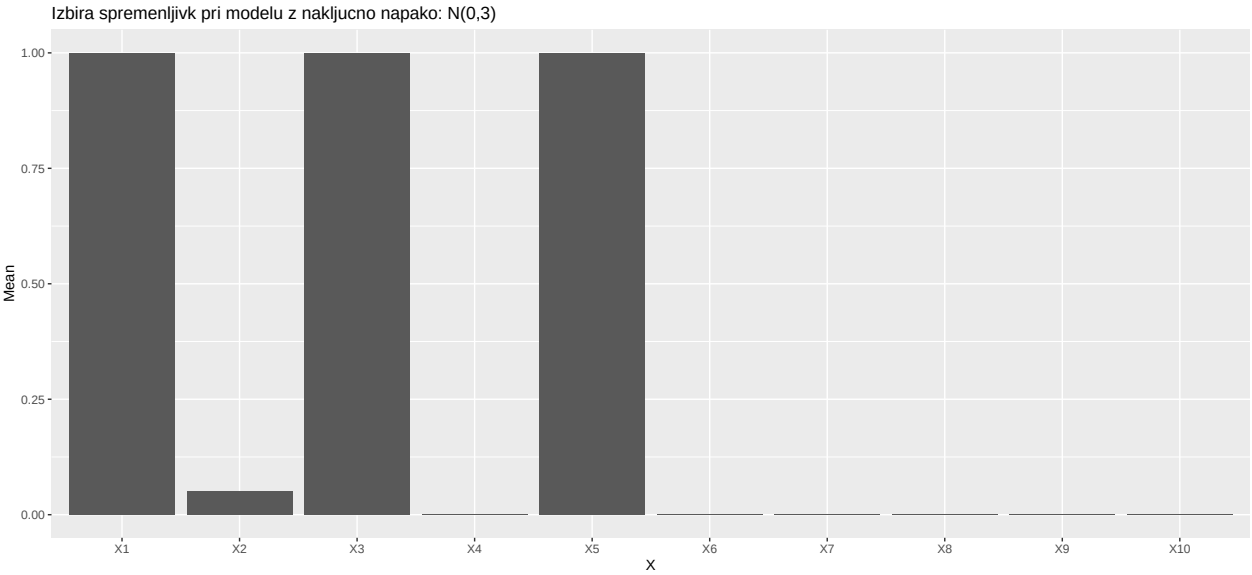
Figure 3: Konvergenca in posteriorne porazdelitve za neničelne bete

```
data.NIBLE.N3.selectX <- sweep(data.NIBLE.N3[,-5], 2, colMeans(data.NIBLE.N3[,-5]), FUN=
data.NIBLE.N3.selectX$X5 <- data.NIBLE.N3$X5
data.NIBLE.N3.selectX <- data.NIBLE.N3.selectX[, c(1,2,3,4,10,5,6,7,8,9)]
mod.N3 <- get.rjMCMC(data.NIBLE.N3.selectX, data.NIBLE.N3.selectY)
```

```
## thin = 1: sigma, psi, beta0, beta, z
## |-------------|-------------|-------------|-------------|
## |-----------------------------------------------------|
```



Konvergence so za ničelne v skladu z pričakovanji, kot tudi posteriorne porazdelitve z zelo visokim verjetjem okoli 0. Težav s konvergenco ni zaslediti tudi pri neničelnih parametrih β.



S povečanjem standardnega odklona slučajne napake se pri izboru spremenljivk ni spremenilo. Izbrane spremenljivke so bile še vedno izbrane pravilno in z visoko vrednostjo verjetja.

2.2.1.3 Slučajna napaka: χ^2_1

```
data.NIBLE.H1 <- gen.beta.data.2(100, napaka = rchisq(100, 1))
data.NIBLE.H1.selectY <- data.NIBLE.H1$Y
data.NIBLE.H1.selectX <- sweep(data.NIBLE.H1[,-5], 2,
                               colMeans(data.NIBLE.H1[,-5]),
```

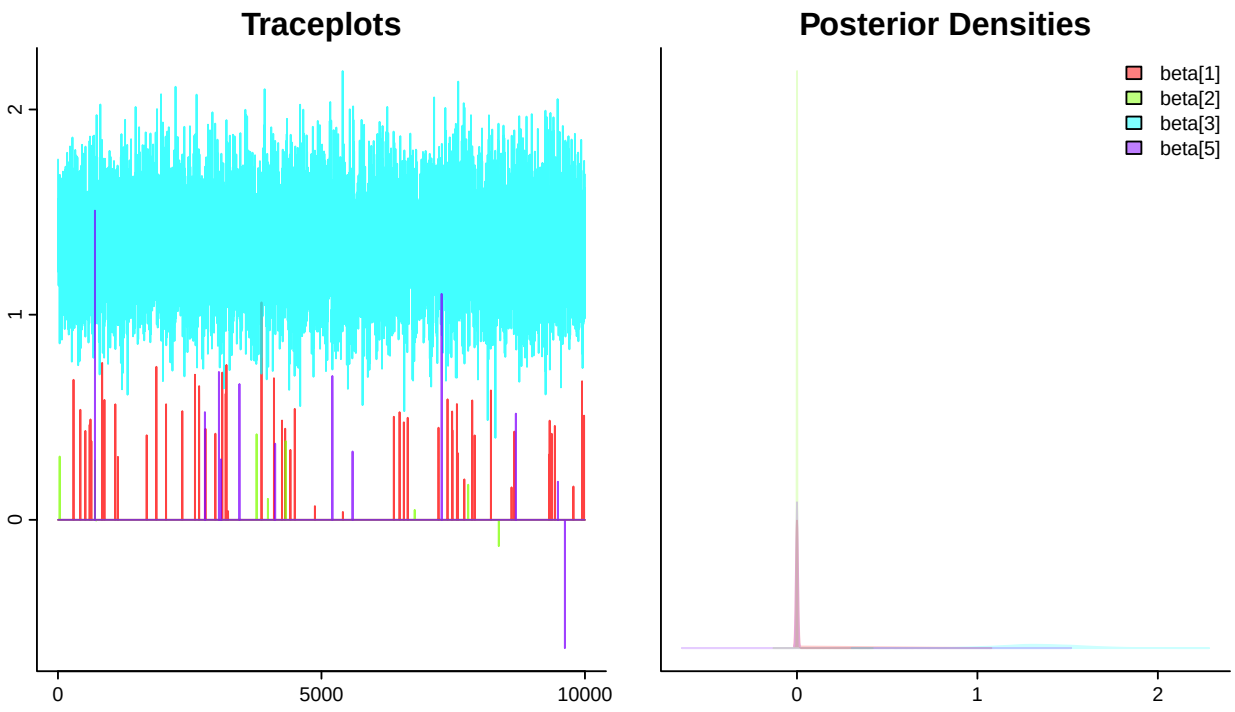


Figure 4: Konvergenca in posteriorne porazdelitve za neničelne bete

```
                                FUN="-")[,-10] #centriramo
data.NIBLE.H1.selectX$X5 <- data.NIBLE.H1$X5
data.NIBLE.H1.selectX <- data.NIBLE.H1.selectX[, c(1,2,3,4,10,5,6,7,8,9)]
mod.H1 <- get.rjMCMC(data.NIBLE.H1.selectX, data.NIBLE.H1.selectY)
```

```
## thin = 1: sigma, psi, beta0, beta, z
## |-------------|-------------|-------------|-------------|
## |-----------------------------------------------------|
```

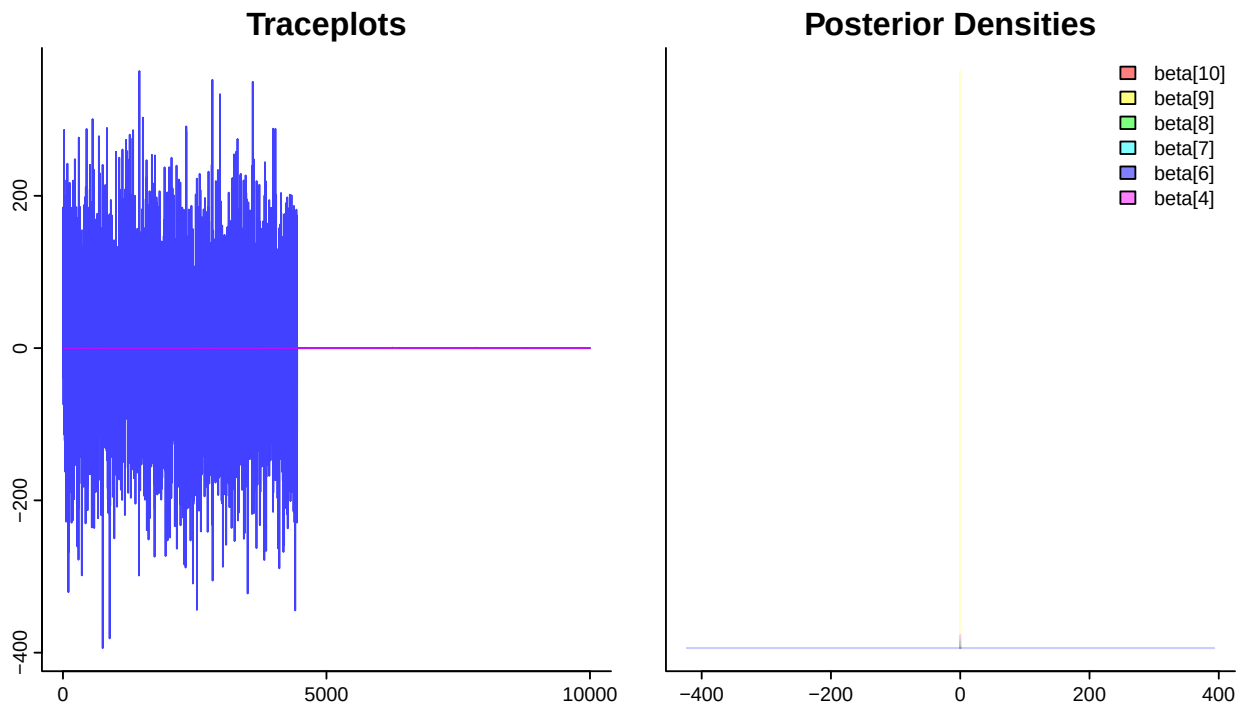
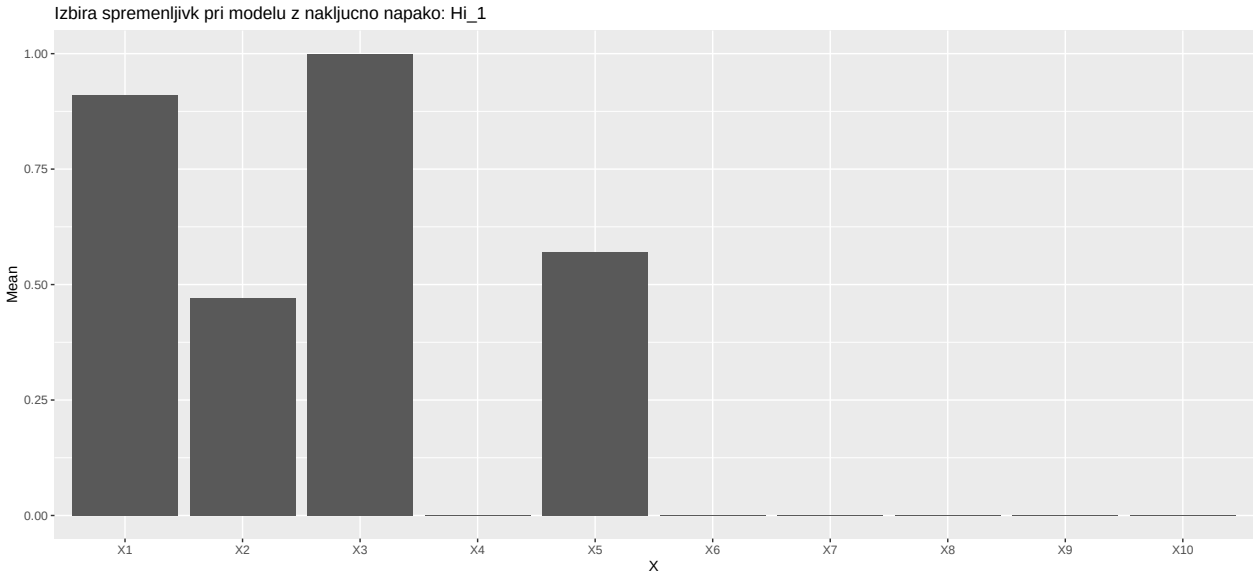


Figure 5: Konvergenca in posteriorne porazdelitve za ničelne bete



Ko spremenimo porazdelitev na χ_1^2 vidimo, da model ni več tako stabilen. Konvergenca je sploh pri spremenljivki X_6 zelo problematična, prav tako pa tudi pri ničenih parametrih spremenljivke X_3. To se pozna tudi na izbiri spremenljivk, ki je zato nepravilna in zelo netočna. Model namreč izbere samo spremenljivko X_3, ki pa ima že prej omenjeno težavo s konvergenco.

2.2.1.4 Slučajna napaka: χ_4^2

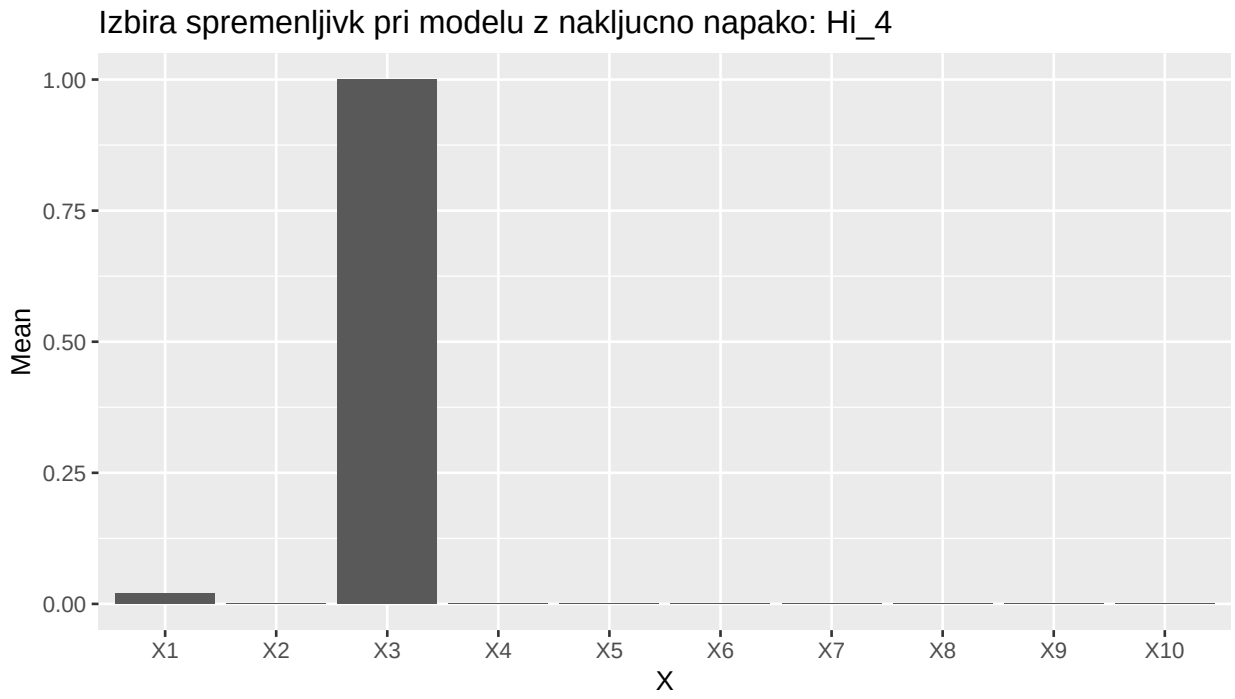


Figure 6: Konvergenca in posteriorne porazdelitve za neničelne bete

```
data.NIBLE.H4 <- gen.beta.data.2(100, napaka = rchisq(100, 4))
data.NIBLE.H4.selectY <- data.NIBLE.H4$Y
data.NIBLE.H4.selectX <- sweep(data.NIBLE.H4[,-5], 2,
                               colMeans(data.NIBLE.H4[,-5]),
                               FUN="-")[,-10] #centriramo
data.NIBLE.H4.selectX$X5 <- data.NIBLE.H4$X5
data.NIBLE.H4.selectX <- data.NIBLE.H4.selectX[, c(1,2,3,4,10,5,6,7,8,9)]
mod.H4 <- get.rjMCMC(data.NIBLE.H4.selectX, data.NIBLE.H4.selectY)
```

```
## thin = 1: sigma, psi, beta0, beta, z
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

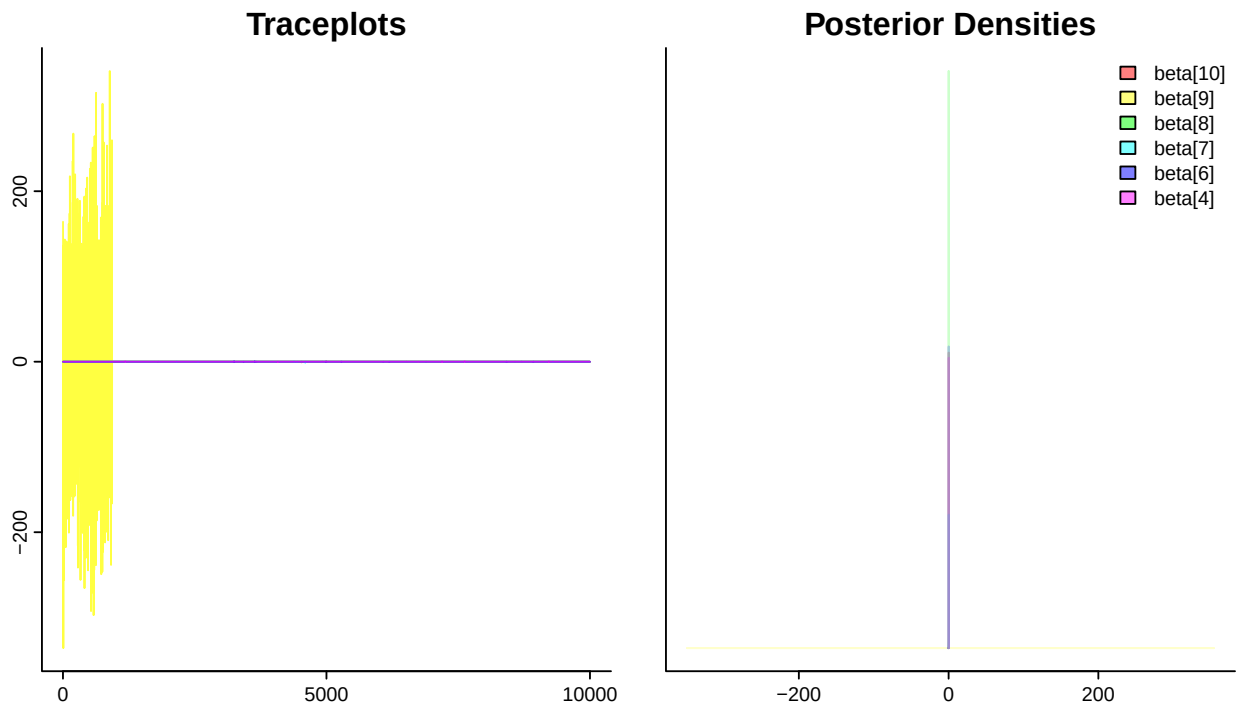
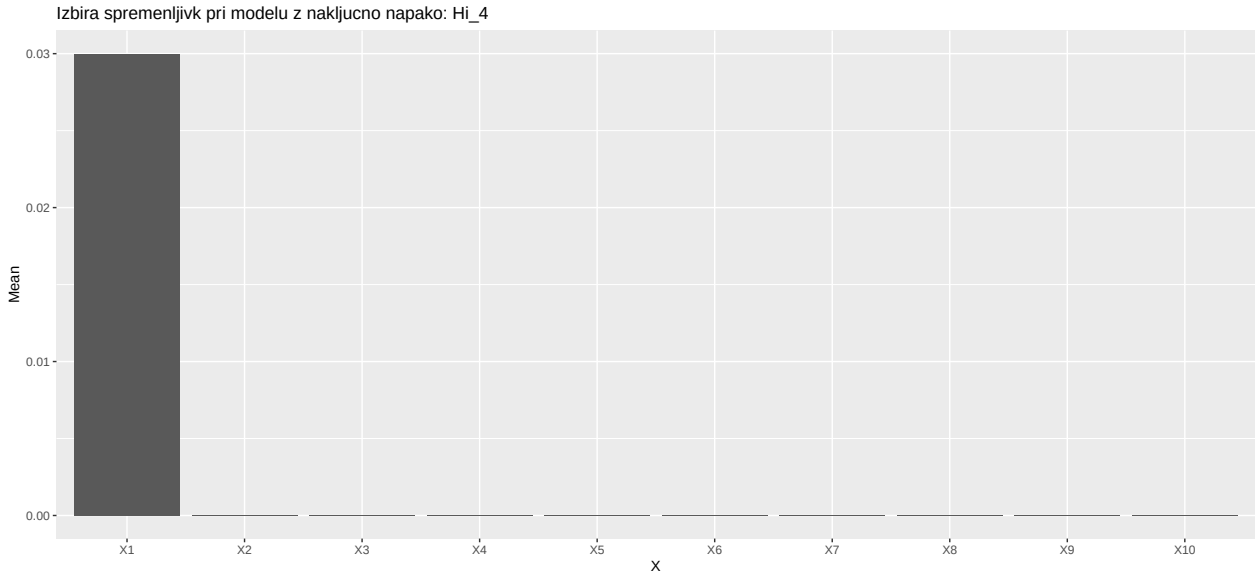


Figure 7: Konvergenca in posteriorne porazdelitve za ničelne bete



Podobna zgodba kot že pri prejšnem primeru, le da je za otenek večja verjetnost pri spremenljivki X_1, kar pa ne popravi kakovosti izbire modela. Pri tem modelu je problematična konvergenca spremenljivke X_9. Model slabo izbere spremenljivke.

2.2.2 Zaključek

Pri metodi Reverse Jump MCMC s programsko knjižnico `nimble`, je bilo moč opaziti, da pri slučajni napaki, ki obsega normalno porazdelitev, model dobro napoveduje izbiro spremenljivk. S tem ko pa spremenimo vrsto porazdelitve slučajne napake, pa metoda ni tako robustna. Razlog zato vidim v tem, saj smo izbrali veliko število hiperparametrov s normalno porazdelitvijo, kar daje prednost slučajnim napakam s normalno porazdelitivijo. Iz tega tudi izhaja, da pri slučajni napaki χ^2_1 dobimo povsem slabo ocenjen model, s vprašljivo konvergenco.

2.3 Bootstrapped Augmented Backward Elimination - ABE

Za potrebe simulacije sem naredil funkcijo, ki je vrnila model `abe`, imela pa je naslednje vhodne podatke: spremenljivke X, Y, metriko s katero smo določali, kateri model je najboljši, število bootstrap ponovitev.

```
selection.ABE <- function(dataSetSelectX, Y, p = 10, metrika = "AIC", alpha=0.2, num.boo
  data.Skupaj <- cbind(dataSetSelectX, "Y" = Y)
  fit <- lm(Y ~ . , data = data.Skupaj, x = TRUE, y = TRUE)
  if(metrika %in% c("AIC", "BIC")){
    abe.fit.boot <- abe.boot(fit, criterion = metrika, data = dataSetSelectX,
                             tau = Inf, exp.beta = FALSE, num.boot = num.boot,
                             type.boot = "bootstrap")
  }
  else{
    abe.fit.boot <- abe.boot(fit, criterion = metrika, alpha = alpha,
                             data = dataSetSelectX, tau = Inf, exp.beta = FALSE,
                             num.boot = num.boot, type.boot = "bootstrap")
  }
  return(abe.fit.boot)
}
```

2.3.1 Opis simulacije

Za simulacije sem se odločil, saj me je zanimalo odstopanje različnih metod, glede na korelacijo in vrsto slučajne napake. Zato sem se odločil, da za vse 3 metrike naredim 100 ponovitev vseh kombinacij korelacije (r = 0, 0.8) in vrste napak (enake kot v prvem delu).

```
pon <- 100
sampleSize <- 100
vrsta.napake <- c("N(0,1)", "N(0,3)", "Hi_1", "Hi_4")
#vrsta.metode <- c("AIC", "BIC", "alpha")
korelacije <- c(0, 0.8)
zasnova <- expand.grid(korelacije, vrsta.napake)
zasnova <- do.call(rbind, replicate(pon, zasnova, simplify=FALSE)) %>%
  `colnames<-`(c("Korelacija","Napaka"))
```

```r
matrika.rez <- matrix(NA, nrow = 3*nrow(zasnova), ncol = 14)

i = 1
j = 1
while(i < nrow(zasnova)){
  vrsta.napake.i <- zasnova[i, "Napaka"]
    if(vrsta.napake.i == "N(0,1)"){
      error <- rnorm(sampleSize, 0, 1)
    }
    else if(vrsta.napake.i == "N(0,3)"){
      error <- rnorm(sampleSize, 0, 3)
    }
    else if(vrsta.napake.i == "Hi_1"){
      error <- rchisq(sampleSize, df = 1)
    }
    else if(vrsta.napake.i == "Hi_4"){
      error <- rchisq(sampleSize, df = 4)
    }
    r.i <- zasnova[i, "Korelacija"]

  data.ABE <- gen.beta.data.2(sampleSize, r = r.i, napaka = error)
  data.ABE.selectY <- data.ABE$Y
  data.ABE.selectX <- sweep(data.ABE[,-5], 2, colMeans(data.ABE[,-5]), FUN="-")[,-10] #
  data.ABE.selectX$X5 <- data.ABE$X5
  data.ABE.selectX <- data.ABE.selectX[, c(1,2,3,4,10,5,6,7,8,9)]
  abe.AIC <-selection.ABE(data.ABE.selectX, Y =data.ABE.selectY,
                              p = 10, metrika = "AIC",num.boot = 500)
  abe.BIC <-selection.ABE(data.ABE.selectX, Y =data.ABE.selectY,
                            p = 10, metrika = "BIC",num.boot = 500)
  abe.alpha <-selection.ABE(data.ABE.selectX, Y =data.ABE.selectY,
                            p = 10, metrika = "alpha",num.boot = 500)

  rez.aic <- summary(abe.AIC)$var.rel.frequencies
  rez.bic <- summary(abe.BIC)$var.rel.frequencies
  rez.alpha <- summary(abe.alpha)$var.rel.frequencies
  matrika.rez[j,] <- c("R" = r.i,  "Metoda" = 1, "Napaka" = vrsta.napake.i, rez.aic)
  matrika.rez[j+1,] <- c("R" = r.i,  "Metoda" = 2, "Napaka" = vrsta.napake.i, rez.bic)
  matrika.rez[j+2,] <- c("R" = r.i,  "Metoda" = 3, "Napaka" = vrsta.napake.i, rez.alpha)
  j <- j + 3
  i <- i + 1
}

rez.df <- na.omit(as.data.frame(matrika.rez))
rez.df.nov <- as.data.frame(apply(rez.df, 2, as.numeric))
```

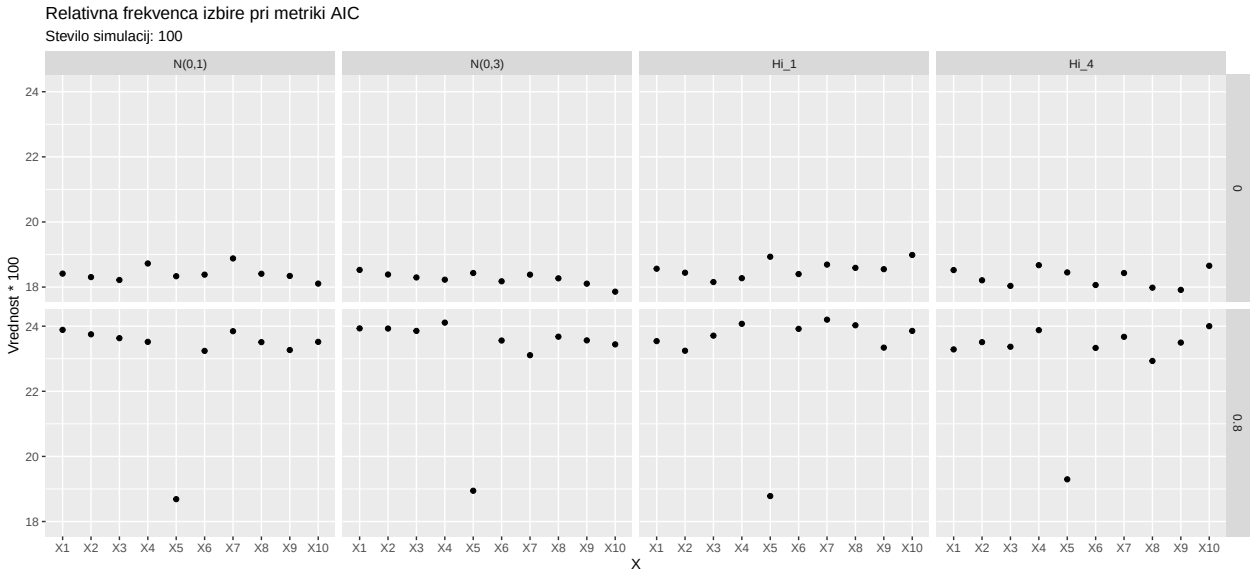
```
rez.df.nov$Metoda <- factor(rez.df.nov$V2, labels = c("AIC", "BIC", "alpha"))
rez.df.nov$Napaka <- factor(rez.df.nov$V3,labels = vrsta.napake)
rez.df.nov$Korelacija <- rez.df.nov$V1
abe.rez <- rez.df.nov[,4:17]
colnames(abe.rez) <- c( paste("X", 0:10, sep=""),"Metoda", "Napaka","Korelacija")
```

2.3.2 Rezultati

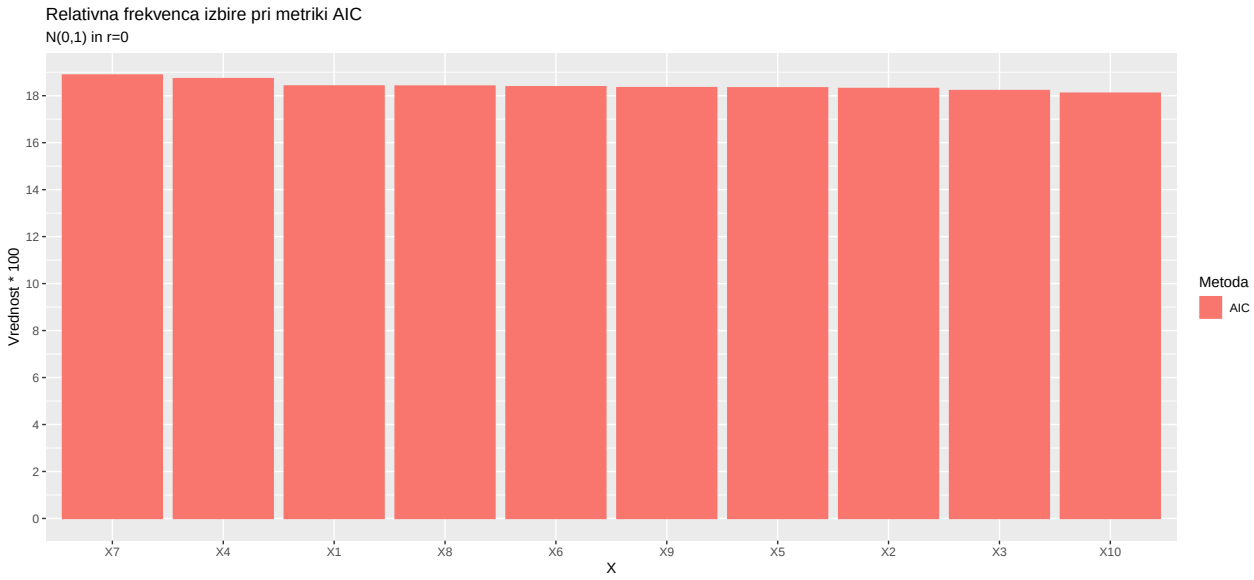
```
abe.data <- readRDS("data/abe_simulation.RDS")

aba.data.sim <- abe.data %>%
  group_by(Korelacija, Metoda, Napaka)%>%
  summarise_all(mean) %>%
  gather(key = "X", "Vrednost", -c(Korelacija, Metoda, Napaka))  %>%
  filter(X != "X0")

aba.data.sim$X <- factor(aba.data.sim$X, levels = paste("X", 1:10, sep = ""))
```

Pri predstavitvi rezultatov bom izpustil parameter β_0, ki je v prisoten v vsakem modelu.

2.3.2.1 AIC

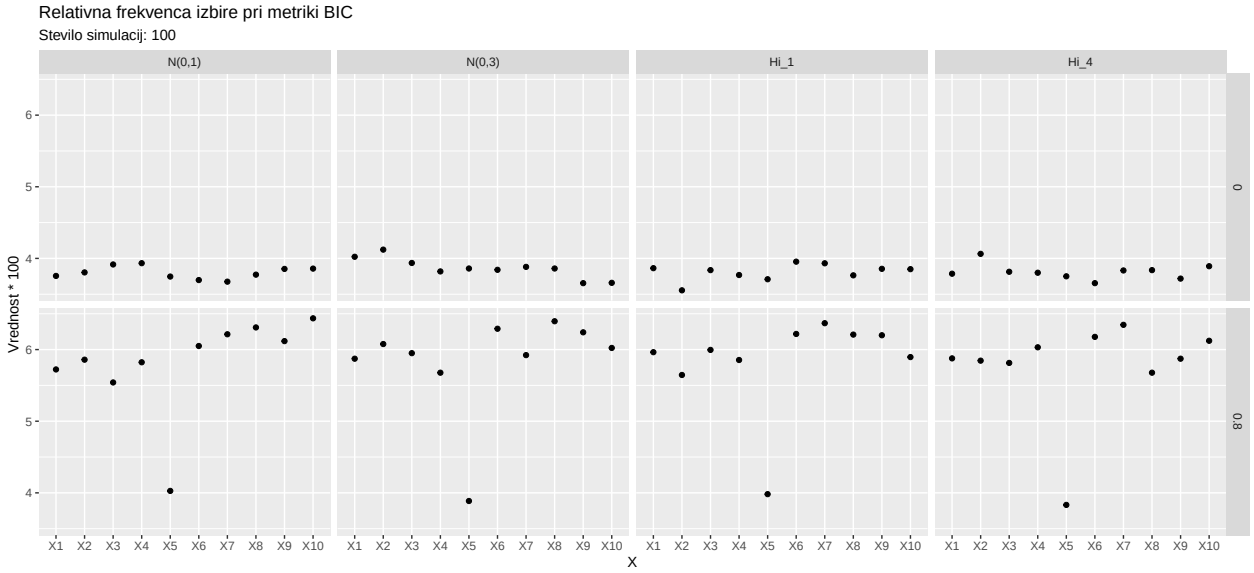


Kot vidimo, so razlike v deležih spremenljivk zelo majhne. Sklepam, da je to posledica boostrapa 500, saj za vsako kombinacijo in vsako metriko naredimo 100 ponovitev po 500 bootstrap vzorcev, kar daje enakomerno porazdelitev vsem spremenljivkam (ali pa je samo slab model). Omeniti še velja, da se pri povečanju korelacije deleži vključevanja v model povečujejo in spremenljivka X_5 se začne vesti drugače - porast vključevanja v model bistveno pade. Zaradi zelo majhnih razlik si poglejmo posebaj primer N(0,1) in r = 0.

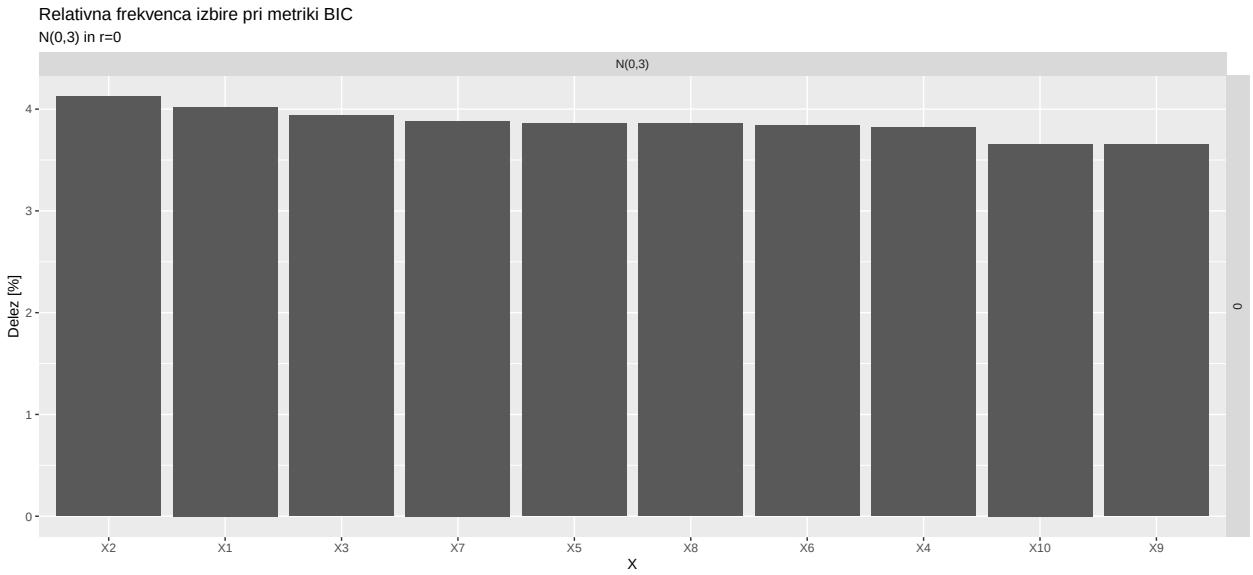


Ko pogledamo deleže, vidimo, da imata najvišji deleže spremenljivke, ki imajo $\beta = 0$, kar nakazuje, da v tem našem primeru model ne bi pravilno izbiral. Med najvišjimi deleži je kar 5 spremenljivk, ki imajo β fiksirano na 0. Podobno je tudi pri drugih scenarijih.

2.3.2.2 BIC

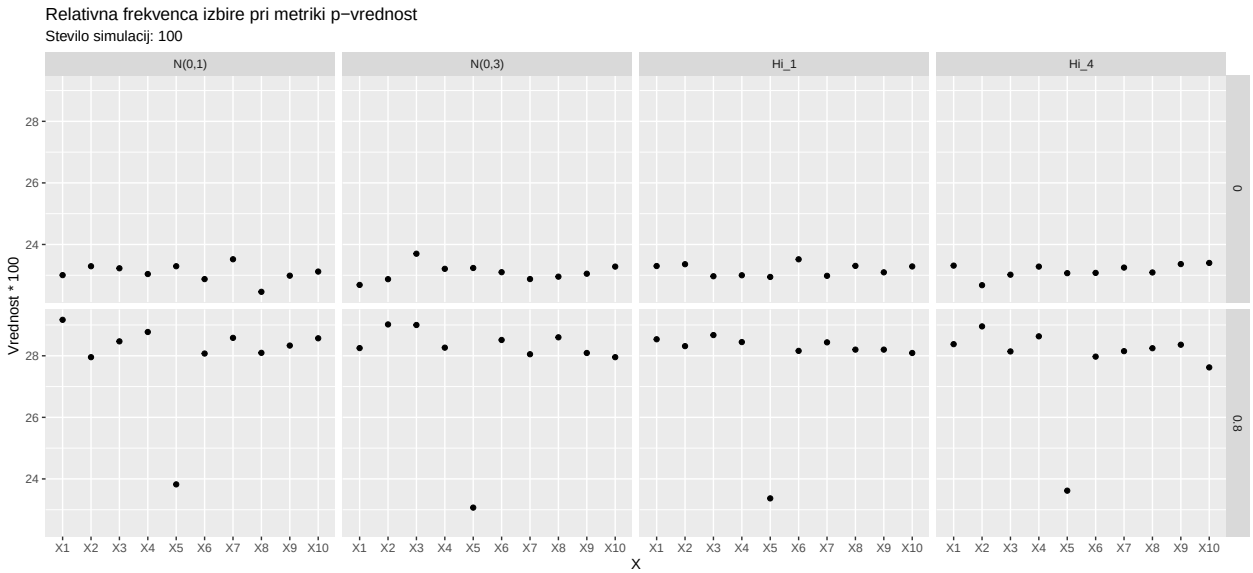


Tako kot pri metriki AIC, so tudi pri BIC razlike v deležih zelo majhne. Razlika, ki jo je moč opaziti je, da imajo spremenljivke X_6 do X_10 nekoliko manjše deleže pri normalni slučajni napaki in korelaciji r = 0. Pri slučajnih napak iz porazdelitve χ^2, se stvari obnašajo čudno in nepravilno (največje deleže imajo spremenljivke s $\beta = 0$). S tem ko povečamo korelacijo med spremenljivkami se nam zgodi, da binarna spremenljivka ne opisje več dobro modela, zato je njen delež daleč najslabši. Še posebno pri slučajni napaki N(0,3) in r = 0 so spremenljivke pravilno selekcionirane, ampak ko povečamo korelacijo, selekcija ni več pravilna, zato ugotavljam, da je korelacija oz. kolinearnost eden od močnih vplivov na izbiro spremenljivk. Vseeno si poglejmo od bližje rezulatate pri slučajni napaki N(0,3) in r = 0.

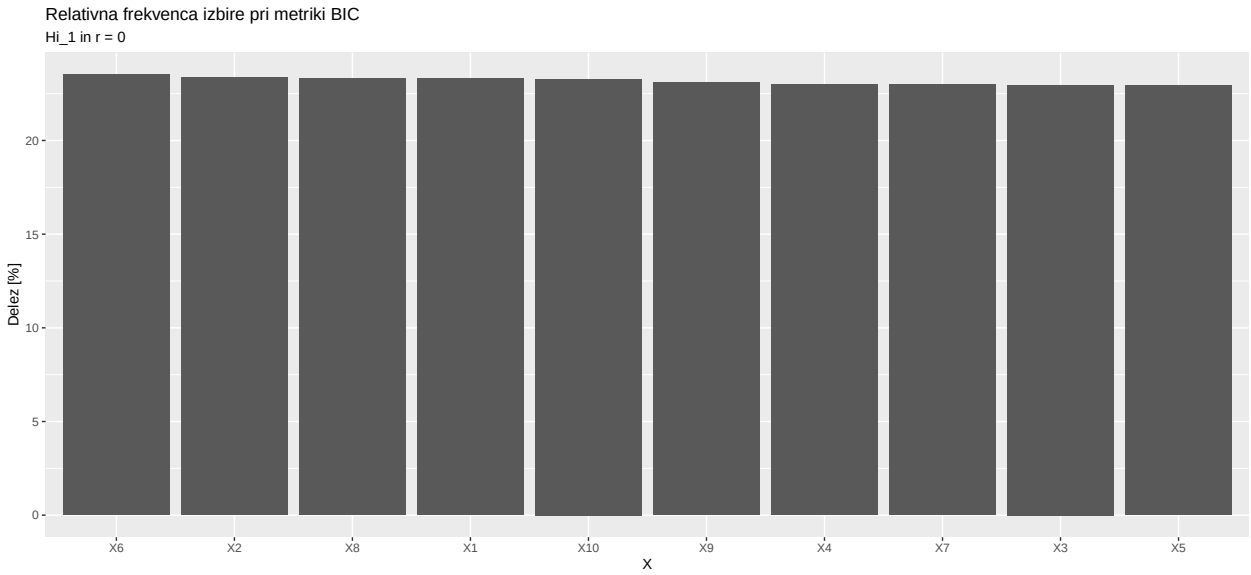


Kot vidimo je vrstni red pravilen, le da gre pri X_7 in X_5 za minimalne razlike.

2.3.2.3 p-vrednost



Pri metriki p-vrednosti sem alpha fiksiral na 0.2, dobimo zanimive rezultate. Pri scenarijih z slučajno napako iz normalne porazdelitve so selekcije spremenljivk boljše pri večjem standardne odklonu, pri korealcijskih koeficientu r = 0. Ko povečamo korealcijski koefiecient se delež pravilno napovedanih spremenljivk znatno zmanjša. Pri scenarijih iz χ^2 porazdelitve, da spremenljivke z največjim deležem izbire imajo teoroetičen koeficien enak 0. S povečanjem korealcij se ponovno opazi razliko na spremenljivki X_5. Tokrat si podrobneje oglejmo χ^2_1 pri korelaciji r = 0.



Dejstvo je, da so razlike tako majhne, da se delež največkrat izbrane spremenljivke v primerjavi z najmanjšim deležem izbrane spremenljivke razlikuje za 0.06. Težko rečemo, da bi v tem primeru izbrali pravilne spremenljivke.

2.3.3 Zaključek

Ko pogledamo vse tri različne metrike, lahko rečemo, da so razlike pri nekaterih metrikah tako majhne, da se nam hitro lahko zgodi da vzmemo nevplivno spremenljivko. Simulacija je vseeno pokazala, da je BIC metrika, dajala najbolj zanesljive rezultate. Rezultati so bili dokaj natančni samo za slučajno napako normalne porazdelitve, pri hi-kvadrat porazdelitvi, so bile pravilne spremenljivke redkeje izbrane. Korelacija je pomemnbno vplivala na izbor spremenljvik, tako, da v tem primeru priororčam izbor glede na metriko p-vrednosti.

2.4 Spike and Slab - BoomSpikeSlab

Spike and Slab regresija se Bayesovi statistiki uporablja predvsem za selekcijo spremenljivk, ko imamo več spremenljivk kot statisitčnih enot. Bralec si lahko več prebere **tukaj** ali **tukaj**

2.4.1 Predstavitev simulacije

```
selection.spike.slab <- function(dataSetSelectX, Y){
  dataX <- cbind(rep(1, nrow(dataSetSelectX)),
                 as.matrix(dataSetSelectX))
  prior <- IndependentSpikeSlabPrior(dataX, Y,
                                     prior.df = .0)
  dataX <- dataX[,-1]
  model <- lm.spike(Y ~ dataX,  niter = 1000, prior = prior)
  return(model)
}
```

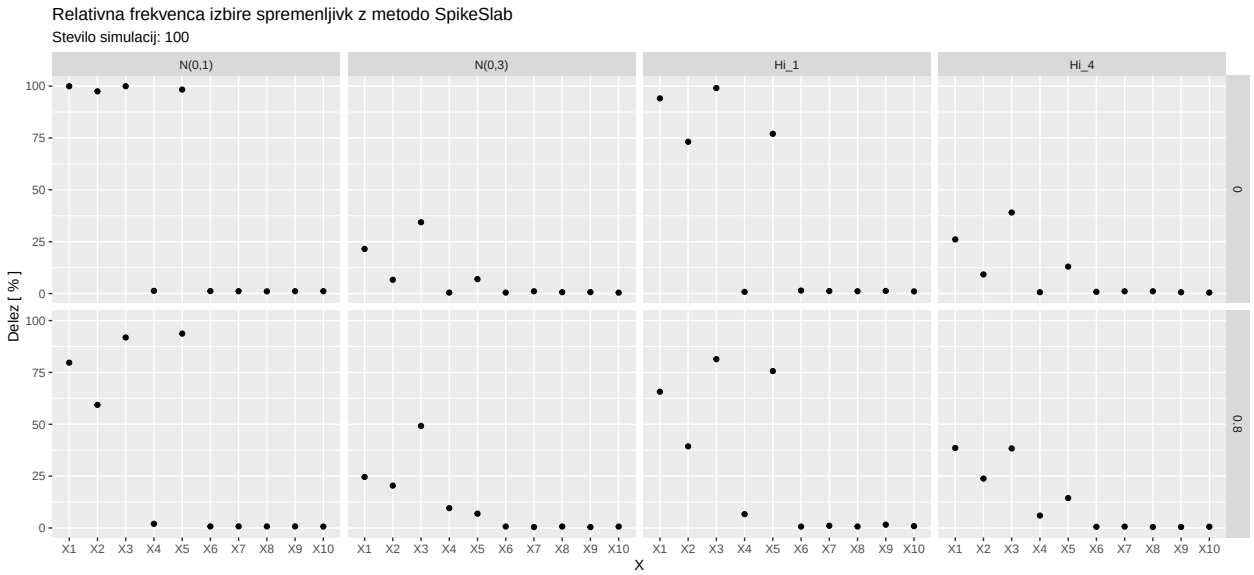
V enostavni simulaciji bom naredil 100 ponovitev zgoraj opisane funkcije, pri čemer me je zanimala frekvenca in izbor izbranih spremenljivk. To sem naredil za vse vse 4 vrste napak in dve različni vrednosti korelacije.

```
pon <- 100
sampleSize <- 100
vrsta.napake <- c("N(0,1)", "N(0,3)", "Hi_1", "Hi_4")
#vrsta.metode <- c("AIC", "BIC", "alpha")
korelacije <- c(0, 0.8)
zasnova <- expand.grid(korelacije, vrsta.napake)
zasnova <- do.call(rbind, replicate(pon, zasnova, simplify=FALSE)) %>%
  `colnames<-`(c("Korelacija","Napaka"))
matrika.rez <- matrix(NA, nrow = nrow(zasnova), ncol = 13)
i = 1

while(i <= nrow(zasnova)){
  vrsta.napake.i <- zasnova[i, "Napaka"]
  if(vrsta.napake.i == "N(0,1)"){
    error <- rnorm(sampleSize, 0, 1)
  }
  else if(vrsta.napake.i == "N(0,3)"){
    error <- rnorm(sampleSize, 0, 3)
  }
  else if(vrsta.napake.i == "Hi_1"){
    error <- rchisq(sampleSize, df = 1)
  }
  else if(vrsta.napake.i == "Hi_4"){
    error <- rchisq(sampleSize, df = 4)
  }
  r.i <- zasnova[i, "Korelacija"]

  data.Spike <- gen.beta.data.2(sampleSize, r = r.i, napaka = error)
  data.Spike.selectY <- data.Spike$Y
  data.Spike.selectX <- sweep(data.Spike[,-5], 2, colMeans(data.Spike[,-5]), FUN="-")[,-
  data.Spike.selectX$X5 <- data.Spike$X5
  data.Spike.selectX <- data.Spike.selectX[, c(1,2,3,4,10,5,6,7,8,9)]

  mod.spikeSlab <- selection.spike.slab(dataSetSelectX = data.Spike.selectX, Y =data.Spi
  inc.prob <- summary(mod.spikeSlab)$coef[,"inc.prob"]
  inc.prob.urejen <- inc.prob[order(factor(names(inc.prob), levels = paste("dataXX", 0:1

  matrika.rez[i,] <- c("R" = r.i,"Napaka" = vrsta.napake.i, inc.prob.urejen)
  i <- i + 1
}
```

```
spikeSlab.df <- as.data.frame(matrika.rez[,-13])
spikeSlab.df$V2 <- factor(spikeSlab.df$V2,labels = vrsta.napake)
colnames(spikeSlab.df) <- c("Korelacija", "Napaka", paste("X", 1:10, sep =""))
```

2.4.2 Rezultati

```
spikeSlab.data <- readRDS("data/spikeSlab_simulation.RDS")

spike.data.sim <- spikeSlab.data %>%
  group_by(Korelacija, Napaka)%>%
  summarise_all(mean) %>%
  gather(key = "X", "Vrednost", -c(Korelacija, Napaka))
spike.data.sim$X <- factor(spike.data.sim$X, levels = paste("X", 1:10, sep = ""))
```



Metoda SpikeSlab se je izkazala kot najbolj uspešna metoda v iskanju najboljši spremenljivk za opis spremenljvike Y. Pri korealcijskem koeficientu r = 0, namreč za vse vrste slučajnih napak pravilno napove največji delež za spremenljvke, ki imajo neničelni koeficient β. Če primerjamo po posameznih porazdelitvah vidimo, da pri normalni sluačjni napaki, večji standardni odklon nekoliko pokvari rezultate, vendar nikoli ni vključil modela, ki bi vseboval ničelni koeficient β. Pri hi-kvadrat porazdelitvi s eno prostostno stopnjo imajo teoretično pravilne spremenljivke visok delež izbire, s povečavanjem prostostnih stopenj na štiri, pa se ta delež nekoliko zmanjša. Metoda pravilno zazna tudi binarno spremenljivko X_5. S povečavo korelacijskega koeficienta (r = 0.8) se deleži nekoliko zmanjšajo, vendar to nespremeni bistveno rezultatov, razen pri slučajni napaki - N(0,3), kjer je delež za spremenljivko X_4 ($\beta_4 = 0$) nekoliko večji kot X_5 ($\beta_5 \neq 0$).

2.5 Zaključek

Kot najboljša metoda se mi je zdela SpikeSlab metoda, saj je delovala najbolj robustno, tudi na slučajnih napakah porazdeljene iz χ_2. Na drugo mesto bi postavil Reverse jump MCMC metodo, ki je za normalne porazdelitve delovala zelo dobro, zmanjkalo jo je pri drugi vrsti porazdelitve. Pri bacward bootstrap elimination se mi zdi, da je vseeno najboljše izbira bila BIC kriterij. Možno, je da je kakšna napaka, saj so deleži preveč enakomerno porazdeljeni po spremenljvkah, tudi v primerih ko je šlo za zelo nizko slučajno napako.

3 Praktična uporava Bayesove statistike

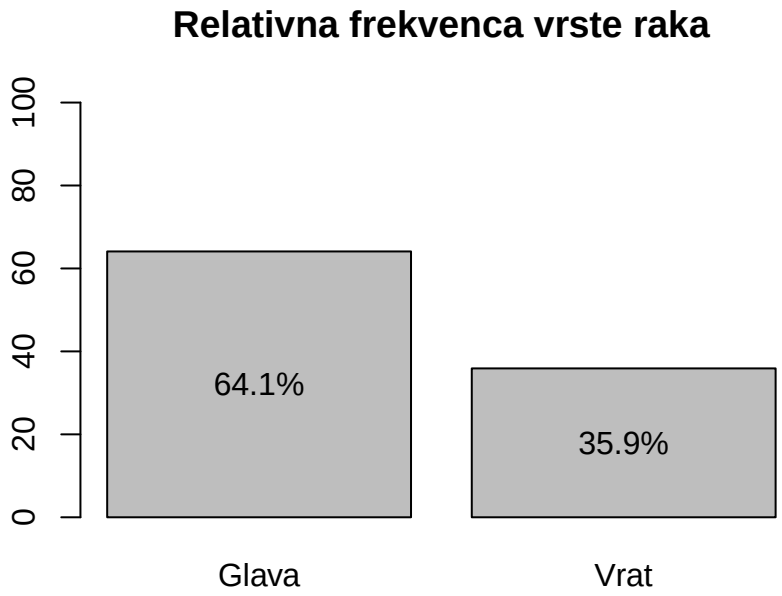
3.1 Podatki

Podatke sem pridobil za namene Statističnega svetovanja, ki ga opravljam na Onkološkem inštitu v Ljubljani. Raziskovalca zanima varnostni pas pri obsevanju z radioterapijo. Vsak pacient je deležen svojega obsevalnega načrta, ki zajema različno število frakcij (obsevanje). Teh frakcij je lahko do 35. Pri obsevanju se pacient postavi na mizo, kjer ga s pomočjo slik skalibirajo na teoretično pravilen položaj. Ker pa človek ni togo telo, se vseskozi premika (dihanje, napete mišice, itd.). Zato v ta namen gledajo premike v x, y in z smeri, ki so se zgodili v času ene frakcije od teoretične postavitve, ki bi jo moral pacient dosegati. Ti premiki po oseh določajo varnostni pas obsevanja, da pacientov tumor vseeno v celoti obsevan. Imenujemo jih interfrakcijski razmiki.

V prvem delu se bom osredotočil kaj vpliva na translacijske premike po y-osi (`Lng`). Neodvisne spremenljivke, ki jih bom vključil v model sta: vrsta raka in število frakcij. Model se mi zdi smiseln, saj me zanima ali vrsta raka dejansko pomeni večje translacijske premike pri radioterapiji (pri kateri vrsti raka, se pacienti bolj premikajo) in ali število obsevanj vpliva na napako. Translacij `Vrt` in `Lat`, ter rotacije v model nisem dodajal v model, saj ne gre za neodvisne spremenljivke. Predpostavljam, da se pacient ne more premakniti samo po eni osi.

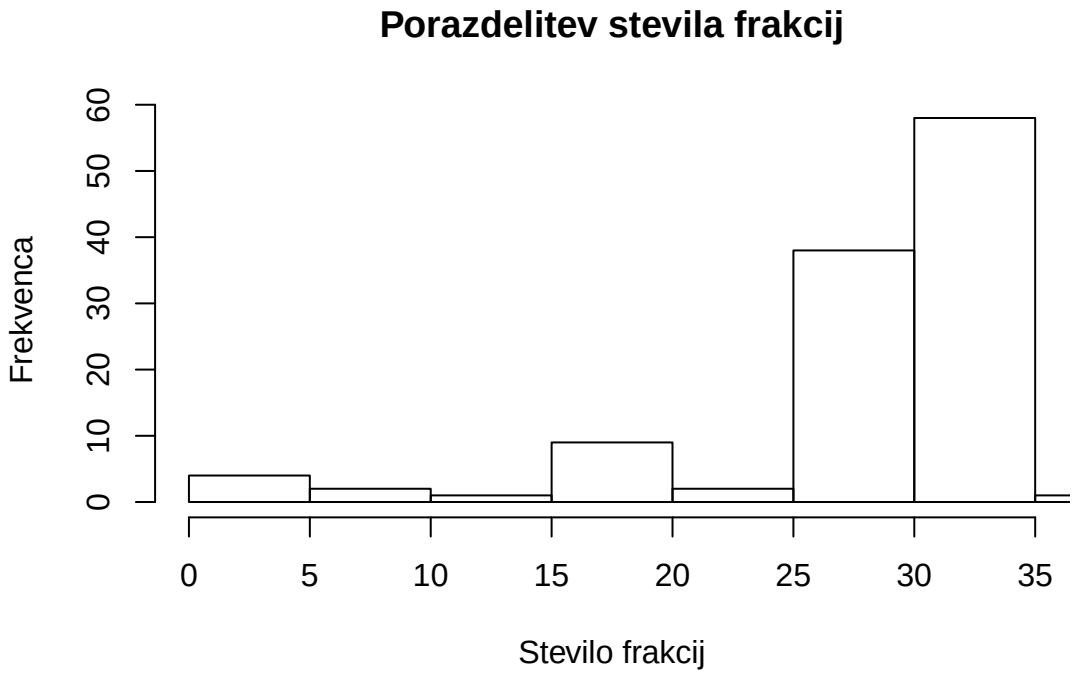
V1	AnonId	PlanId	RefIso	TreatDate	Vrt	Lng	Lat	Rtn
0	Patient000	Plan1	Glava	2014-03-25	-0.5	0.0	0.5	1.0
1	Patient000	Plan1	Glava	2014-03-26	-0.4	-0.1	0.2	0.0
2	Patient000	Plan1	Glava	2014-03-27	0.0	0.3	0.2	0.0
3	Patient000	Plan1	Glava	2014-03-28	-0.2	0.1	0.4	0.1
4	Patient000	Plan1	Glava	2014-03-31	-0.1	0.1	0.3	0.0
5	Patient000	Plan1	Glava	2014-04-01	-0.1	0.1	0.3	0.0

3.1.1 Vrsta raka



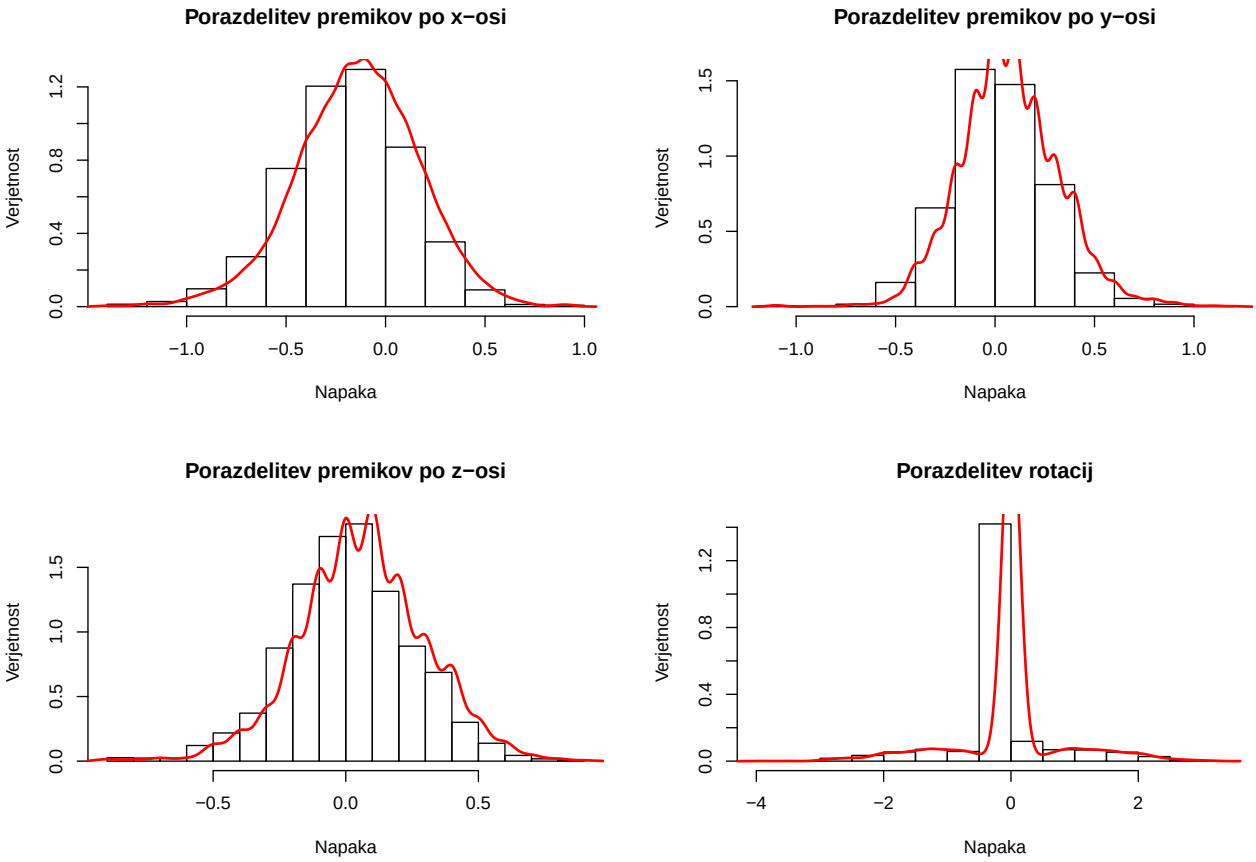
V podatkovju imamo 115 pacientov, ki je skupaj opravilo 3393 obsevanj z radioterapijo. 64 % je imelo raka v glavi, ostali pa na vratu. Obsevanje je potekalo od septembra 2012 do marca 2015.

3.1.2 Število frakcij



Za porazdelitev števila frakcij med pacienti, ki velja za ključno v mojem problemu, je na vzorcu vidna ena velika skupina, ki obsega 84 % pacientov, ki ima med 25 in 35 frakcij. Skoraj 10 % pacientov ima število obsevanj med 15 in 25, medtem ko ima le 6 % pacientov od 1 do 15 obsevanj.

3.1.3 Translacije in rotacije pacientov



	n	mean	sd	median	min	max	skew	kurtosis
Vrt	3393	-0.14	0.30	-0.1	-1.4	0.9	-0.20	0.44
Lng	3393	0.07	0.25	0.1	-1.1	1.2	0.19	0.95
Lat	3393	0.06	0.24	0.1	-0.9	0.9	-0.10	0.57
Rtn	3393	-0.01	0.75	0.0	-3.9	3.2	-0.15	4.12

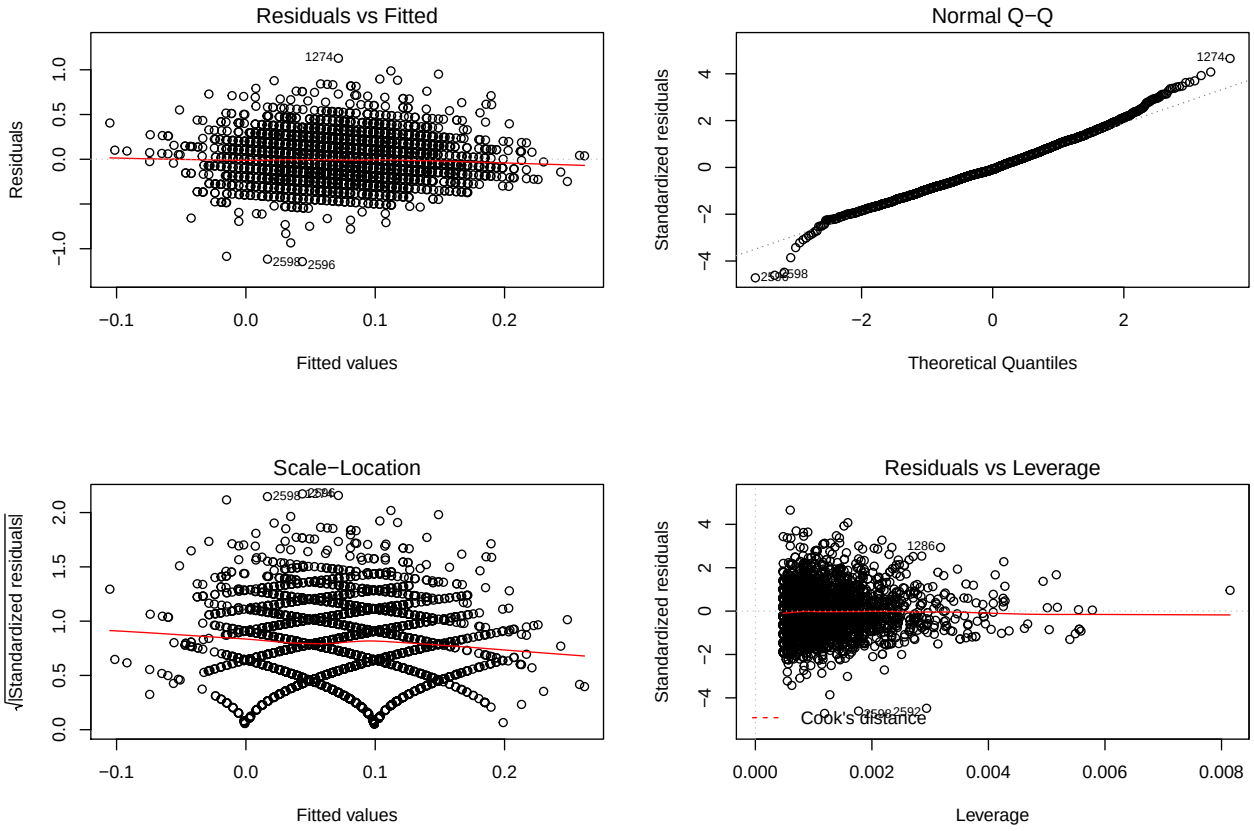
Porazdelitve po oseh so normalno porazdeljene s precej podobnim povprečji. Pri porazdelitvi rotacij ne moremo trditi, da je spremenljivka normalno porazdeljena, saj je prevelik del vrednosti okoli 0, ostale vrednosti pa so minimalno prisotne v negativno in pozitivno smer.

3.1.4 Frekventistični model

```
lm.mod <- lm(Lng ~ Vrt + Lat + RefIso, data = podatki.st.frakcij)
summary(lm.mod)
```

```
##
## Call:
## lm(formula = Lng ~ Vrt + Lat + RefIso, data = podatki.st.frakcij)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.14363 -0.16204 -0.01716  0.15107  1.12851
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.048813   0.005630   8.671  < 2e-16 ***
## Vrt         -0.136919   0.013691 -10.001  < 2e-16 ***
## Lat         -0.089825   0.017491  -5.135 2.97e-07 ***
## RefIsoVrat   0.031188   0.008681   3.593 0.000332 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2425 on 3389 degrees of freedom
## Multiple R-squared:  0.03843,    Adjusted R-squared:  0.03758
## F-statistic: 45.15 on 3 and 3389 DF,  p-value: < 2.2e-16
```



Predpostavka o konstantni varianci je izpolnjena, problematični so morda ostanki, ki ne kažejo, da so normalno porazdeljeni. Vseeno bom nadaljeval z analizo.

Pregledam še kolinearnost obeh spremenljivk in vidim, da kolinearnost ni prisotna.

```
kable(vif(lm.mod), "markdown", col.names = "VIF")
```

	VIF
Vrt	1.000854
Lat	1.001205
RefIso	1.000387

Pregledali smo osnovne karakteristike linearnega modela, ki jih smatram, da jih moramo narediti, tudi če se odlčamo za Bayesovo statistiko.

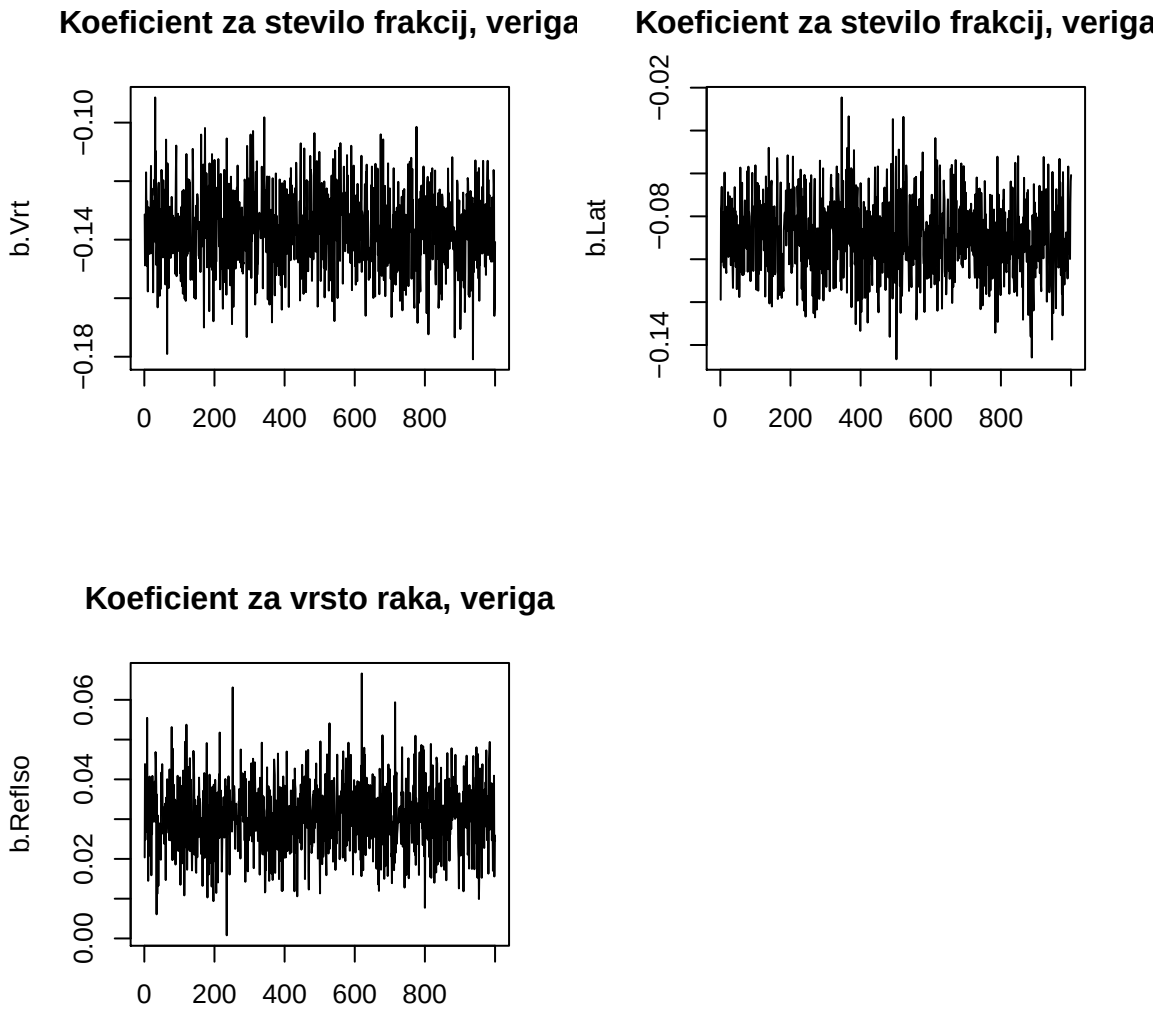
3.1.5 Bayesev model

```
fit2.bayesx <- bayesx(Lng ~ Vrt+ Lat + RefIso ,
                      data = podatki.st.frakcij,
                      family = "gaussian", method = "MCMC", iterations = 12000,
                      burnin = 2000)

b.Vrt <- attr(fit2.bayesx$fixed.effects, "sample")[,2]
b.Lat <- attr(fit2.bayesx$fixed.effects, "sample")[,3]
b.RefIso <- attr(fit2.bayesx$fixed.effects, "sample")[,4]

summary(fit2.bayesx)
```

```
## Call:
## bayesx(formula = Lng ~ Vrt + Lat + RefIso, data = podatki.st.frakcij,
##     family = "gaussian", method = "MCMC", iterations = 12000,
##     burnin = 2000)
##
## Fixed effects estimation results:
##
## Parametric coefficients:
##                Mean      Sd    2.5%     50%   97.5%
## (Intercept)  0.0490  0.0058  0.0377  0.0489  0.0599
## Vrt         -0.1366  0.0134 -0.1631 -0.1365 -0.1109
## Lat         -0.0896  0.0176 -0.1240 -0.0895 -0.0559
## RefIsoVrat   0.0306  0.0087  0.0135  0.0305  0.0475
##
## Scale estimate:
##          Mean     Sd   2.5%    50% 97.5%
## Sigma2 0.0589 0.0015 0.0560 0.0588 0.062
##
## N = 3393  burnin = 2000  method = MCMC  family = gaussian
## iterations = 12000  step = 10
```

3.2 Konvergenca

```
par(mfrow = c(2, 2))
plot(b.Vrt, type = "l", main = "Koeficient za stevilo frakcij, veriga",
     xlab = "")
plot(b.Lat, type = "l", main = "Koeficient za stevilo frakcij, veriga",
     xlab = "")
plot(b.RefIso, type = "l", main = "Koeficient za vrsto raka, veriga",
     xlab = "")
```

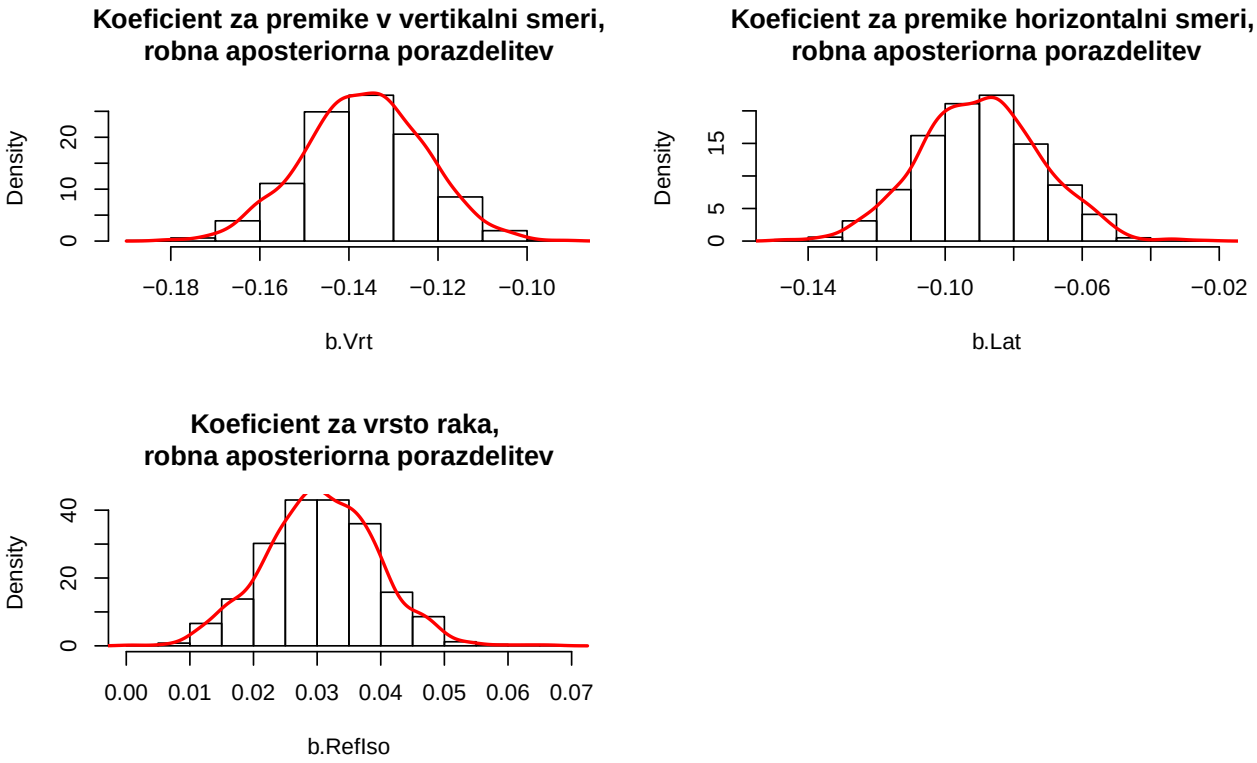


Konvergence za vse parametre je v skladu z dovoljenim.

3.3 Interpretacija

```
par(mfrow = c(2, 2))
hist(b.Vrt, prob = T, main = "Koeficient za premike v vertikalni smeri,\nrobna aposterio
lines(density(b.Vrt), col = "red", lwd = 2)
hist(b.Lat, prob = T, main = "Koeficient za premike horizontalni smeri, \nrobna aposteri
```

```
lines(density(b.Lat), col = "red", lwd = 2)
hist(b.RefIso, prob = T, main = "Koeficient za vrsto raka, \nrobna aposteriorna porazdel
lines(density(b.RefIso), col = "red", lwd = 2)
```



Levi graf zgoraj prikazuje porazdelitev za koeficient β_1, ki določa efekt za spremenljivko `Vrt`. Povprečje porazdelitve je -0.1372, 95 % interval pa je med -0.1634 in -0.1110. Spremenljivka `Lat`, ki je skrita pod koeficient β_2 ima normalno porazdelitev s povprečjem -0.0895, 95 % interval med -0.1237 in -0.0571. Spremenljivka `RefIso`, ki določajo vrsto raka. Njen pripadajoči parameter β_3 je porazdeljen normalno, s povprečjem 0.0313, njen 95 % interval pa je med 0.0146 - 0.0479.

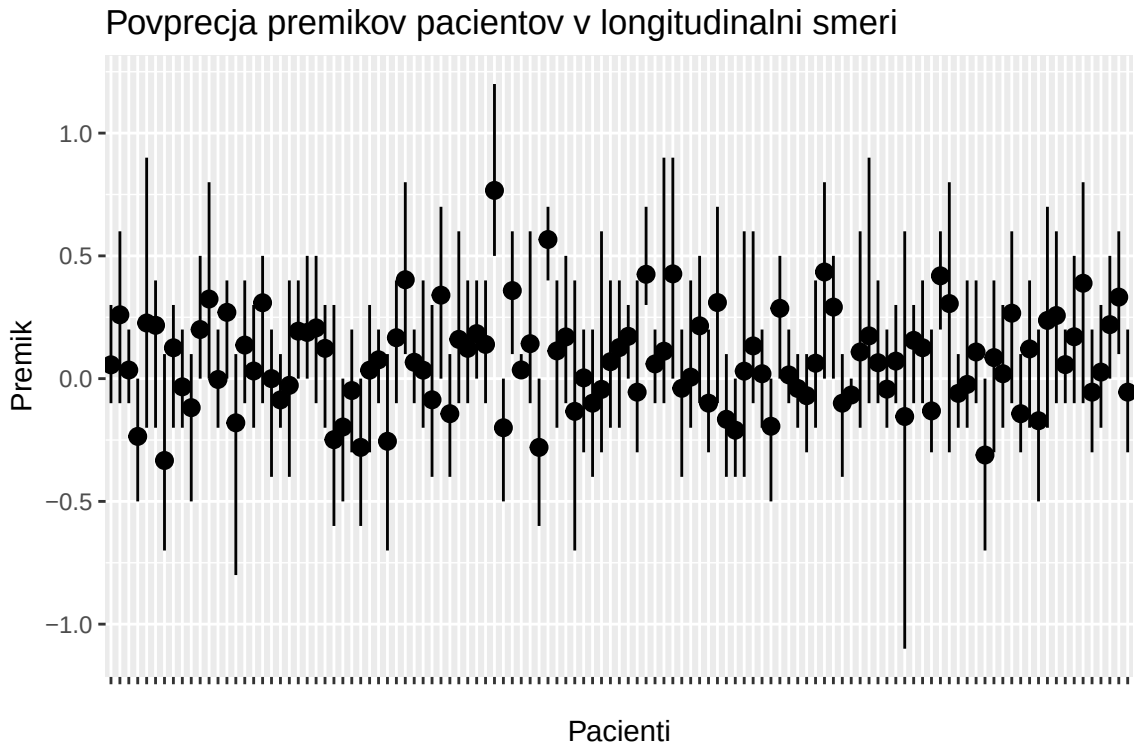
Ob upoštevanju `Vrt` in `Lat` v modelu je `Lng` za paciente z rakom na vratu za 0.0313 enote, kot tiste z rakom v glavi. Ob upoštevanju vrednosti z `RefIso` in `Lat` velja: če se premik v vertikalni smeri poveča za 1 se `Lng` v povprečju zmanjša za 0.1372. Ob upoštevanju vrednosti z `RefIso` in `Vrt` velja: če se premik v horizontalni smeri (Lat) poveča za 1 se `Lng` v povprečju zmanjša za 0.0895.

3.4 Hierarhični model

Hierarhičen model sem definiral na naslednji način. Zanimala me bo spremenljivka `Lng`, glede na paciente, in kako se razlikuje med njimi. Pri tem bom naredil pogumno predpostavko o tem, da je varianca med posameznimi pacienti enaka. Podatke imamo za 115 pacientov, vsak od njih pa ima do 35 merjenj.

```r
pod.Lng <- dt %>%
  group_by(AnonId) %>%
  summarise(povprecje = mean(Lng), n=length(Lng), varianca = var(Lng))
```

```r
ggplot(dt, aes(x = AnonId, y = Lng, group = AnonId))+
  stat_summary(fun.ymin = min, fun.ymax = max, fun.y = mean) +
  theme(axis.text.x =element_text(color = "white"))+
  labs(x = "Pacienti", y = "Premik" )+
  ggtitle("Povprecja premikov pacientov v longitudinalni smeri")
```




```r
m <- length(pod.Lng$AnonId)
n <- pod.Lng$n
ime.unique <- levels(dt$AnonId)

xMatrix <- matrix(NA, ncol = m, nrow = max(n))
for (j in 1:m) {
  xMatrix[1:n[j],j] <- dt[dt$AnonId == ime.unique[j],]$Lng - mean(dt[dt$AnonId == ime.ur
}
```

Določil sem tudi parametre hiperapriornih porazdelitev:

$$\begin{aligned}\sigma^2 &\sim \text{Inv-Gama}(\nu_0/2, \sigma_0^2\nu_0/2),\\ \mu &\sim \mathcal{N}(\mu_0, \tau_0^2),\\ \eta^2 &\sim \text{Inv-Gama}(\kappa_0/2, \eta_0^2\kappa_0/2).\end{aligned}$$

```
code <- nimbleCode({
  mu ~ dnorm(0, sd = 10);
  eta ~ dunif(0, 100)
  sigma ~ dunif(0, 100)

  for (j in 1:m) {
    muGroups[j] ~ dnorm(mu, sd = eta)
    for (i in 1:n[j]) {
      y[i, j] ~ dnorm(muGroups[j], sd = sigma);
    }
  }
})
```

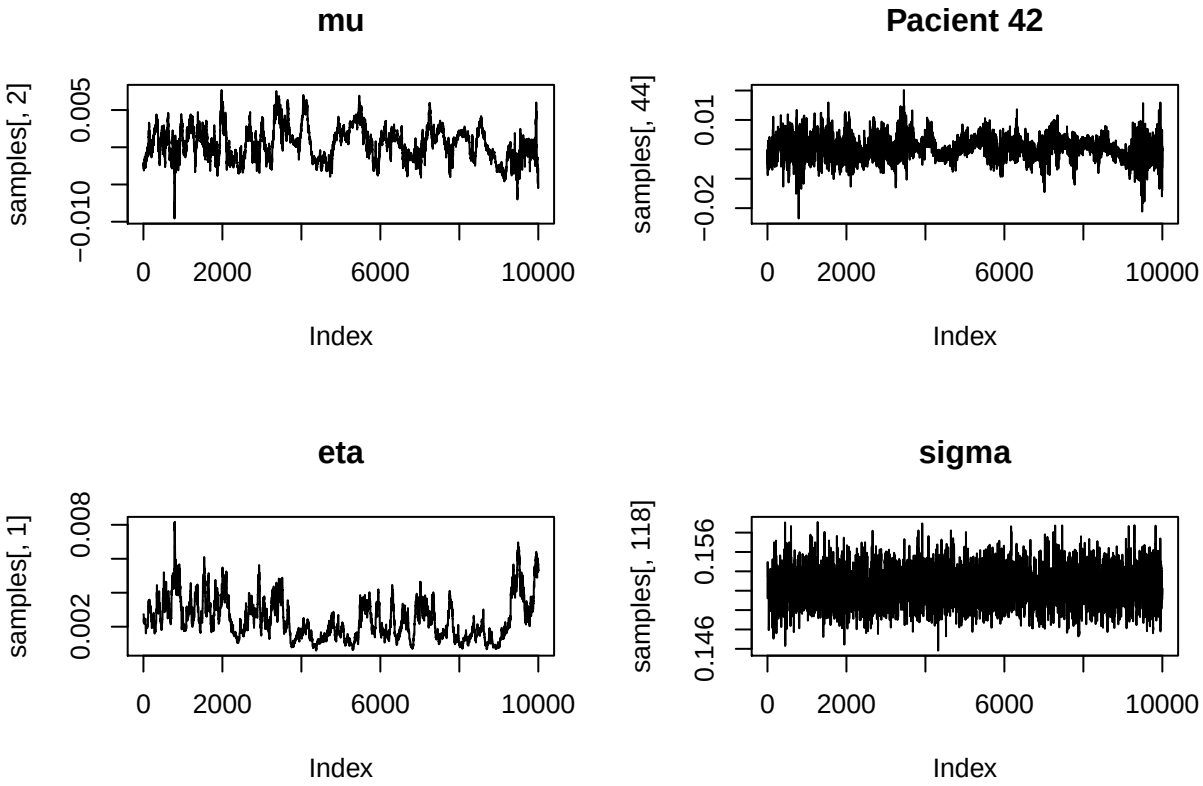
Ker je želja, da bi bili premiki med opazovanjem čim manjši, kar pomeni, da je pričakovana vrednost radioterapije enaka 0. Za standardni odklon sem preizkusil več vrednosti, nato sem se odločil za 1. Med podatki ni nikoli vrednosti višje od 2, vednar sem želel biti previden in si nisem želel preveč omejevati. Parametra η in σ sem vzorčil iz enakomerne porazdelitve 0 - 10. Pri tem sem poskusil, tudi širše intervale, vendar so se mi tukaj rezultati zdeli najbolj optimalni.

```
constants <- list(m = m, n = n)

inits <- list(mu = mean(pod.Lng$povprecje),
              eta = sd(pod.Lng$povprecje),
              sigma = mean(sqrt(pod.Lng$varianca)),
              muGroups = pod.Lng$povprecje)

data <- list(y = xMatrix)

Rmodel <- nimbleModel(code = code, constants = constants,
                      inits = inits, data = data)
Rmodel$initializeInfo()

conf <- configureMCMC(Rmodel)
#conf$printSamplers()
#conf$printMonitors()
conf$addMonitors('muGroups')

Rmcmc <- buildMCMC(conf)
Cmodel <- compileNimble(Rmodel)
Cmcmc <- compileNimble(Rmcmc, project = Cmodel)
samples <- runMCMC(Cmcmc, niter = 12000, nburnin = 2000 )
#saveRDS(samples, "data/HieMod.RDS")
```

3.4.1 Konvergenca

Najprej sem preučil konvergenco parametrov in naključno izbranega pacienta.

```
par(mfrow = c(2,2))
plot(samples[,2], type = "l", main = "mu")
plot(samples[,44], type = "l", main = "Pacient 42")
plot(samples[,1], type = "l", main = "eta")
plot(samples[,118], type = "l", main = "sigma")
```

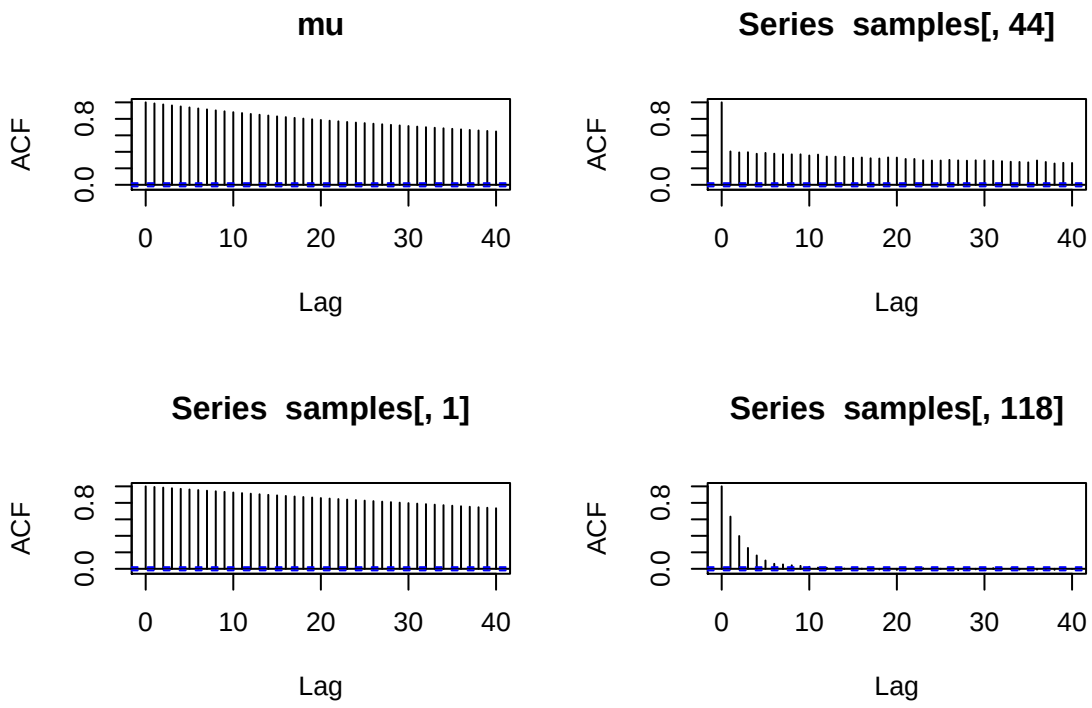


Z konvergenco izgleda vse ok, saj so vrednosti na y-osi dovolj male. Pozorni moramo biti, saj so vrednosti merjenja majhne in da ne pride do prevelikih odstopanj.

3.4.2 Thinning

Ker v MCMC verigah prevladuje visoka stopnja avtokorelacije, zato je potrebna analiza tudi v mojem primeru.

```
par(mfrow = c(2,2))
acf(samples[,2], main = "mu")
acf(samples[,44])
acf(samples[,1])
acf(samples[,118])
```



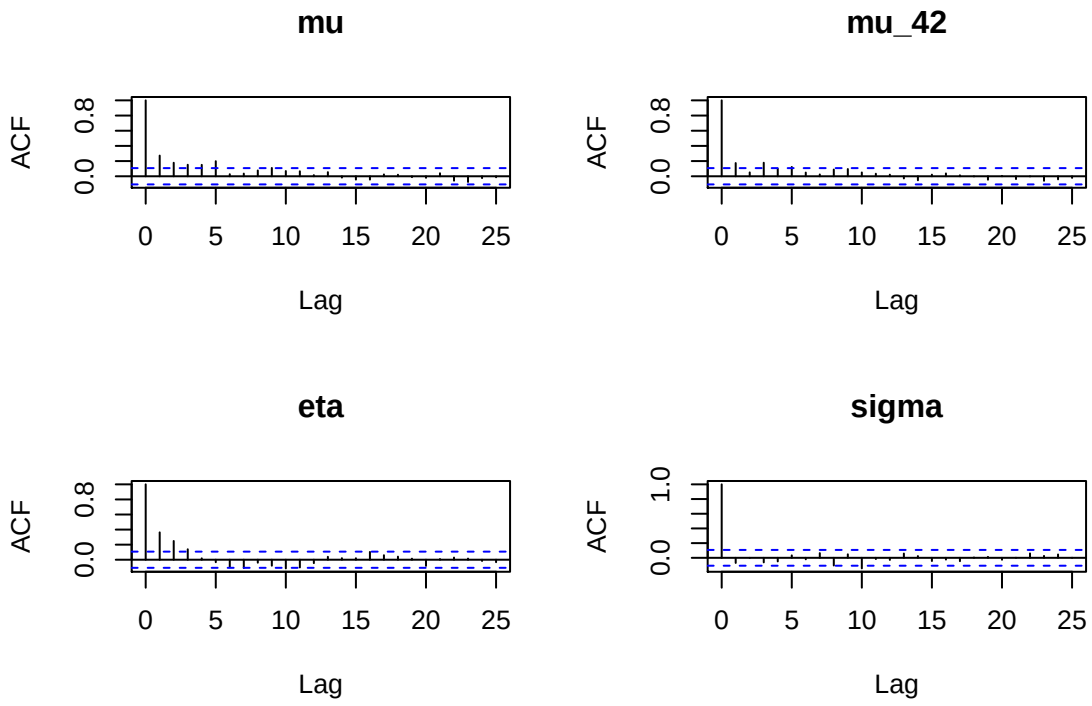
Kot vidimo so podatki v MCMC verigi visoko korelirani, zato moramo uporabiti thinning. Koliko vrednosti bomo spusitli vmes je odvisno od podatkov, zato sem to storil s poskušanjem. Na koncu sem se odločil za 300 in se s tem rešil avtokorelacije. (Ne vem ali je to prevelika številka v praksi in bi moral drugače postopati).

Na novo definiramo model in temu primerno povečamo število iteracij in burn-in.

```
samples.thinning <- samples <- runMCMC(Cmcmc, niter = 120000, nburnin = 20000, thin =300
#saveRDS(samples.thinning, "data/HieMod_thinning.RDS")
```

in najprej pogledamo thinning:

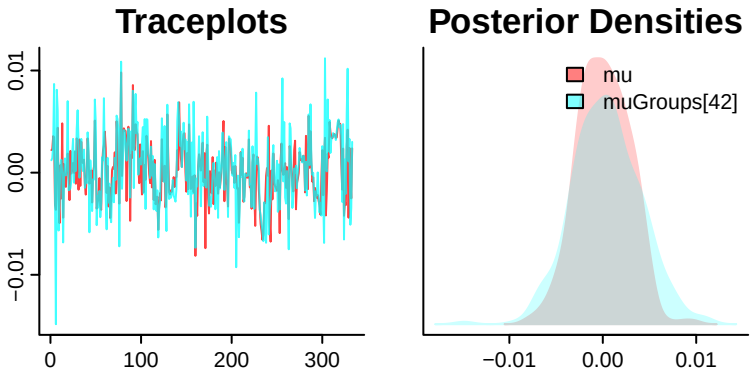
```
par(mfrow = c(2,2))
acf(samples.thinning[,2], main = "mu")
acf(samples.thinning[,44], main = "mu_42")
acf(samples.thinning[,1], main = "eta")
acf(samples.thinning[,118], main = "sigma")
```



Še vedno ni videti vredu, saj nekateri saj so ostanki po lagih še vedno večji od 95 % intervala zaupanja, ki je narisan s črtkano črto. Vseeno nadaljujem z analizo.

Še enkrat sem pogledal konvergenco za končni model (s thinningom):

```
samplesPlot(samples.thinning, var = c("mu","muGroups[42]"))
```

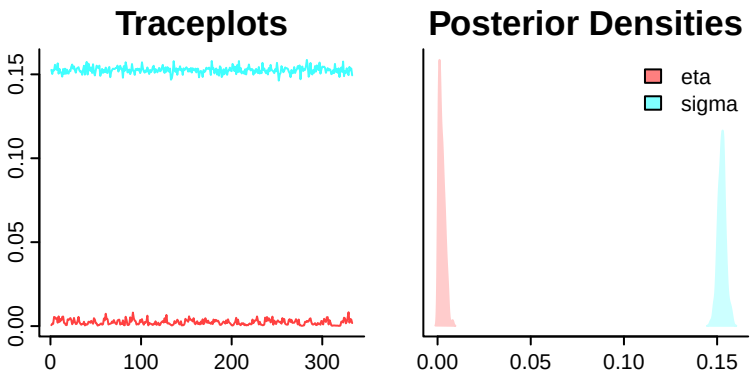



```
kable(samplesSummary(samples.thinning)[c(2, 44), ], "markdown")
```

	Mean	Median	St.Dev.	95%CI_low	95%CI_upp
mu	0.0001099	0.0000950	0.0027880	-0.0051606	0.0050117
muGroups[42]	0.0003875	0.0002561	0.0036447	-0.0066618	0.0075813

Glede na skalo, ki je na y-osi, bi rekel, da je konvergenca spremenljiva, čeprav graf od daleč zgleda da zelo variara.

```
samplesPlot(samples.thinning, var = c("eta","sigma"))
```




```
kable(samplesSummary(samples.thinning)[c(1, 118), ], "markdown")
```

	Mean	Median	St.Dev.	95%CI_low	95%CI_upp
eta	0.0023202	0.0019823	0.0015631	0.0002947	0.0056823
sigma	0.1525250	0.1525196	0.0020404	0.1483986	0.1569827

Za parametra σ in η je konvergenca vredu. Posteriorne porazdelitve η dosegajo vrednosti zelo blizu, medtem ko porazdelitev parametra σ malce odmaknjena od 0, s povprečjem 0.15.

3.4.3 Efective size in standardna napaka

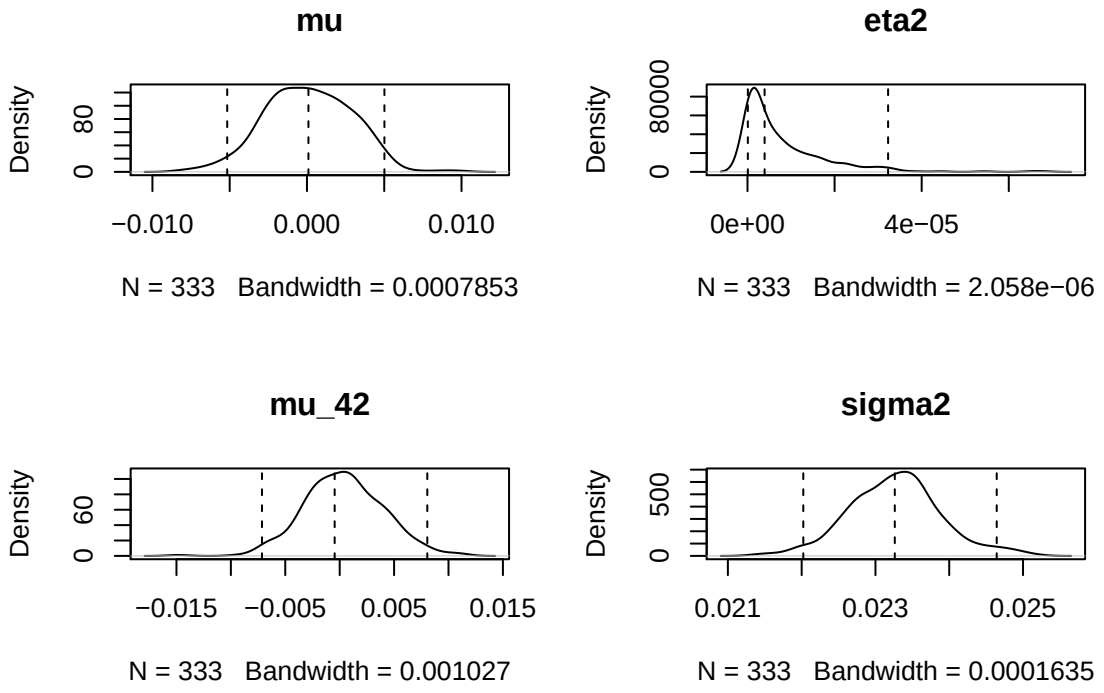
```
efektivni.vzorec<- effectiveSize(samples.thinning)
moj.efect.vzorec <- efektivni.vzorec[c(1,2,44,118)]
sd.vzorec <- apply(samples.thinning[,c(1,2,44,118)], 2, sd)
standardne.napake <- sapply(1:4, function(i){sd.vzorec[i]/sqrt(moj.efect.vzorec[i])})

kable(data.frame("Effective size" = moj.efect.vzorec, "Standardna napaka" = standardne.n
```

	Effective.size	Standardna.napaka
eta	117.89036	0.0001440
mu	98.10931	0.0002815
muGroups[42]	157.32631	0.0002906
sigma	333.00000	0.0001118

Efektivna velikost vzorca se giblje okoli 100 za posameznega pacienta. Za hiper parameter σ pa okoli 333. Standardne napake so za vse parametre zelo majhne, kar je dober znak za model.

3.4.4 Marginalne aposteriorne porazdelitve

```
par(mfrow=c(2, 2))
plot(density(samples.thinning[ , 2]), type = "l", main = "mu")
abline(v = quantile(samples.thinning[ , 2], prob=c(0.025, 0.5, 0.975)), lty = 2)
plot(density(samples.thinning[ , 1]**2), type = "l", main = "eta2")
abline(v = quantile(samples.thinning[ , 1]**2, prob=c(0.025, 0.5, 0.975)), lty = 2)
plot(density(samples.thinning[ , 44]), type = "l", main = "mu_42")
abline(v = quantile(samples.thinning[ , 3], prob=c(0.025, 0.5, 0.975)), lty = 2)
plot(density(samples.thinning[ , 118]**2), type = "l", main = "sigma2")
abline(v = quantile(samples.thinning[ , 118]**2, prob=c(0.025, 0.5, 0.975)), lty = 2)
```

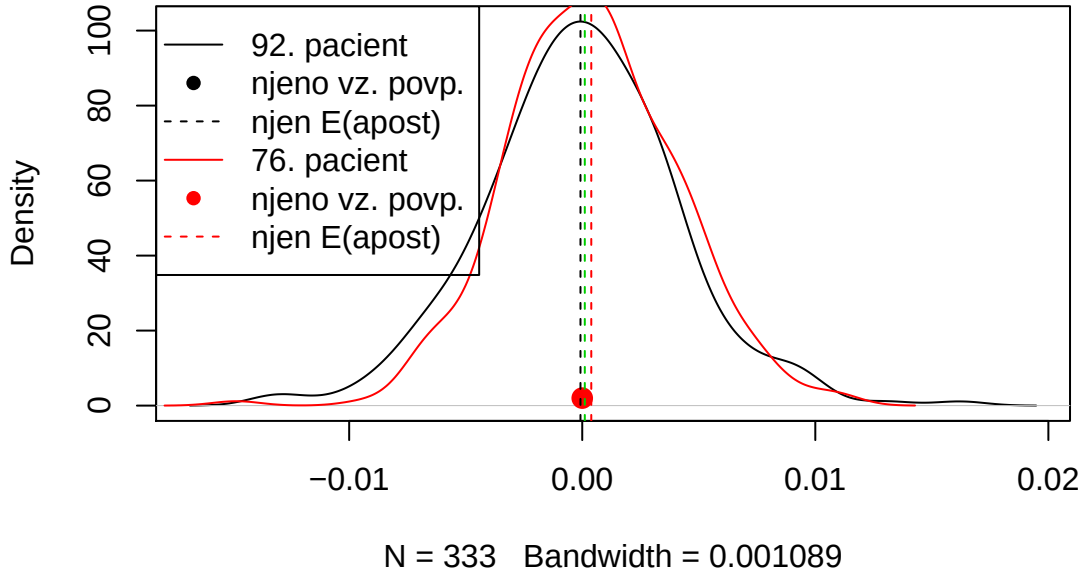


Model za parameter μ daje zelo optimistične napovedi, saj pravi, da bo skupno povprečje vseh pacientov znašalo zelo blizu 0. Paramater η je zelo blizu 0. 95 % interval zaupnja za σ^2 je 0.022 in 0.248, kar kaže na to, da bodo odstopanja od povprečja zelo majhne pri vseh pacientih v modelu. Tako se izkaže tudi pri primeru enega od pacientov, ki ima povprečje pri 0.0003.

Posebaj sem pogledal primer za dva pacienta. Izbrana sta pacienta, ki sta imela največjo in najmanjšo razliko med vzorčnim povprečj in 0. To sta: največjo (76. pacient) in najmanjšo (92.pacient).

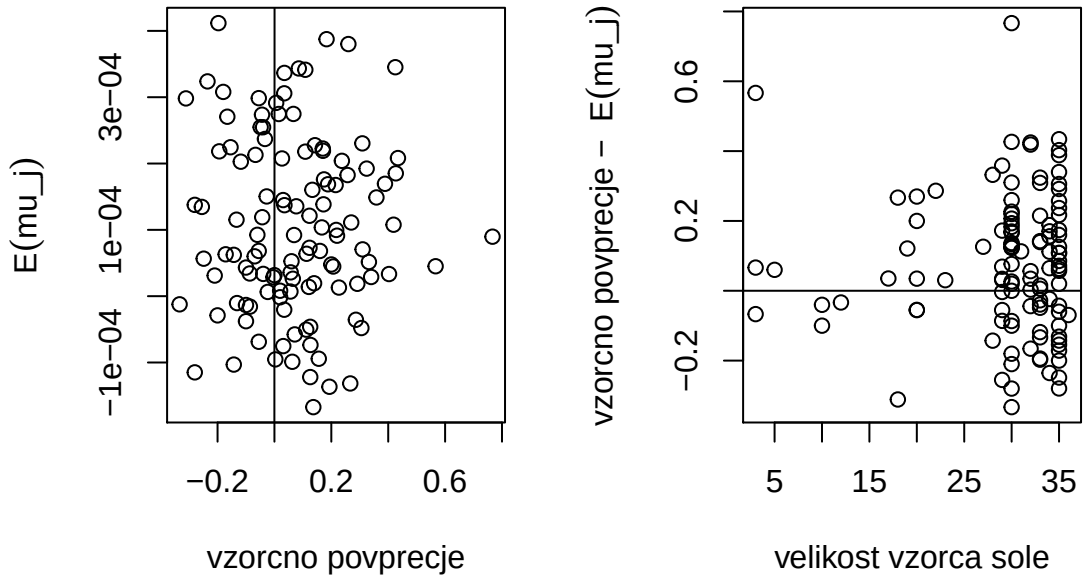
```
plot(density(samples.thinning[,19]), type="l", main="")
points(pod.Lng[19,]$povprecje, 2, pch=16, cex=1.5, col="red")
abline(v = mean(samples.thinning[,19]), lty=2)
lines(density(samples.thinning[,44]), type="l", col="red")
points(pod.Lng[44,]$povprecje, 2, pch=16, cex=1.5, col="red")
```

```
abline(v = mean(samples.thinning[,44]), lty=2, col="red")
abline(v = mean(samples.thinning[,2]), lty=2, col="green3")
legend("topleft", c("92. pacient", "njeno vz. povp.", "njen E(apost)",
                    "76. pacient", "njeno vz. povp.", "njen E(apost)"),
       col=c("black","black","black","red","red","red"), lty=c(1,NA,2,1,NA,2),
       pch=c(NA,16,NA,NA,16,NA))
```



Posteriorni porazdelitvi se nekoliko razlikjueta, njuna modelska povprečja $\mu_4 3$ in μ_6 omejujeta modelsko skupno povepreč̌je (zelena črtkana črta). Vzorčno povprečje za $\mu_4 3 = 0.7$ je tako daleč stran, modelske napovedi, da ju na grafu ni mogoče narisati.

```
pod.Lng$EMuGroup <- colMeans(samples.thinning[,3:117])

par(mfrow=c(1,2))
plot(pod.Lng$povprecje, pod.Lng$EMuGroup,
     xlab = "vzorcno povprecje", ylab = expression(E(mu_j)))
abline(a = 0, b = 1)

plot(pod.Lng$n, pod.Lng$povprecje - pod.Lng$EMuGroup,
     xlab = "velikost vzorca sole",
     ylab = expression(paste("vzorcno povprecje - "," ",E(mu_j), sep="")))
abline(h = 0)
```



Na levi sliki je predstavljeno pričakovana vrednost model za vsakega pacienta v primerjavi z vzorčnim povprečjem vsakega pacienta. Sam menim, da model ne predstavlja dobro podatkov, saj o ocenjuje napako zelo blizu 0 (povečanje apriorne porazdelitve ne pomaga kaj dosti), vzorčna pa se raztezajo od -0.2 do 0.8. Črta predstavlja kako dobro se vzorčna povprečja ujemajo z modelom.

Desna slika predstavlja vpliv velikosti vzorca na razliko vzorčnege povepрečja j-tega pacienta z njegovo pričakovano vrednostjo modela. Vidimo, da je število obsevanj ne vpliva na minimiziranje razlik med podatki in modelom. To je seveda logična posledica, saj so merjenja med seboj popolnoma neodivsna in več merjenj ne bo dalo manjših premikov pacienta. Po drugi strani pa pacienta med obsevanji večkrat slikajo in nato prilagodijo njegov novi položaj, kar posledično pomeni, da bi se napaka morala zmanjševat. Vendar podatkov o tem, kdaj mu na novo izračunajo položaj nimam.

Zakaj je prišlo do tega? Po mojem mnenju zato, ker sem model gradil na osnovni predpostavki, da so variance med pacienti enake. Mislim, da je to glavni razlog zakaj se modelske napovedi ne ujemajo z vzorčnimi povprečji.

4 Priloga

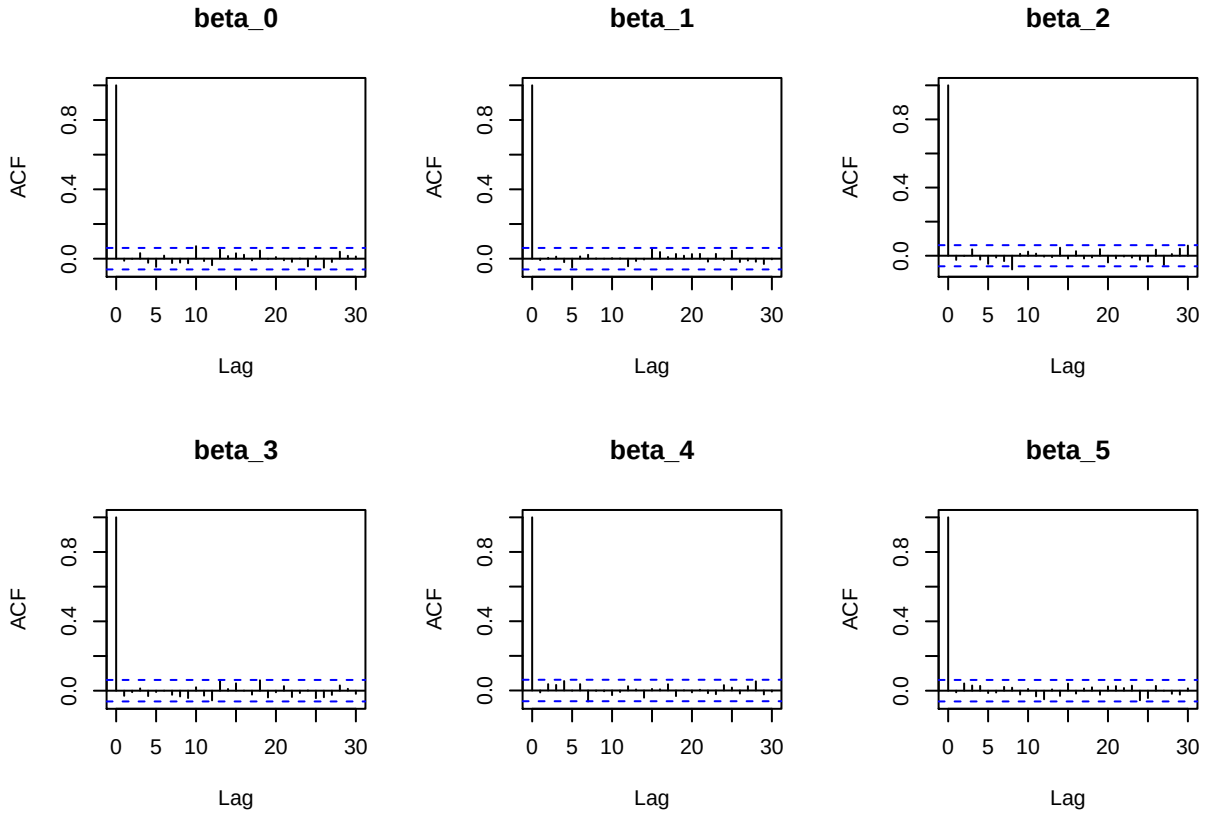
V prilogi so grafi in opombe za vsak simulacijski scenarij, kjer smo uporabili bayesx funkcijo.

4.1 Scenarij I

```
data.N1 <- gen.beta.data(100, napaka = rnorm(100))
bayesx.norm <- bayesx(y ~ X1+X2+X3+X4+X5 , data = data.N1,
                      family = "gaussian", method = "MCMC")
bayesx.mu <- attr(bayesx.norm$fixed.effects, "sample")
```

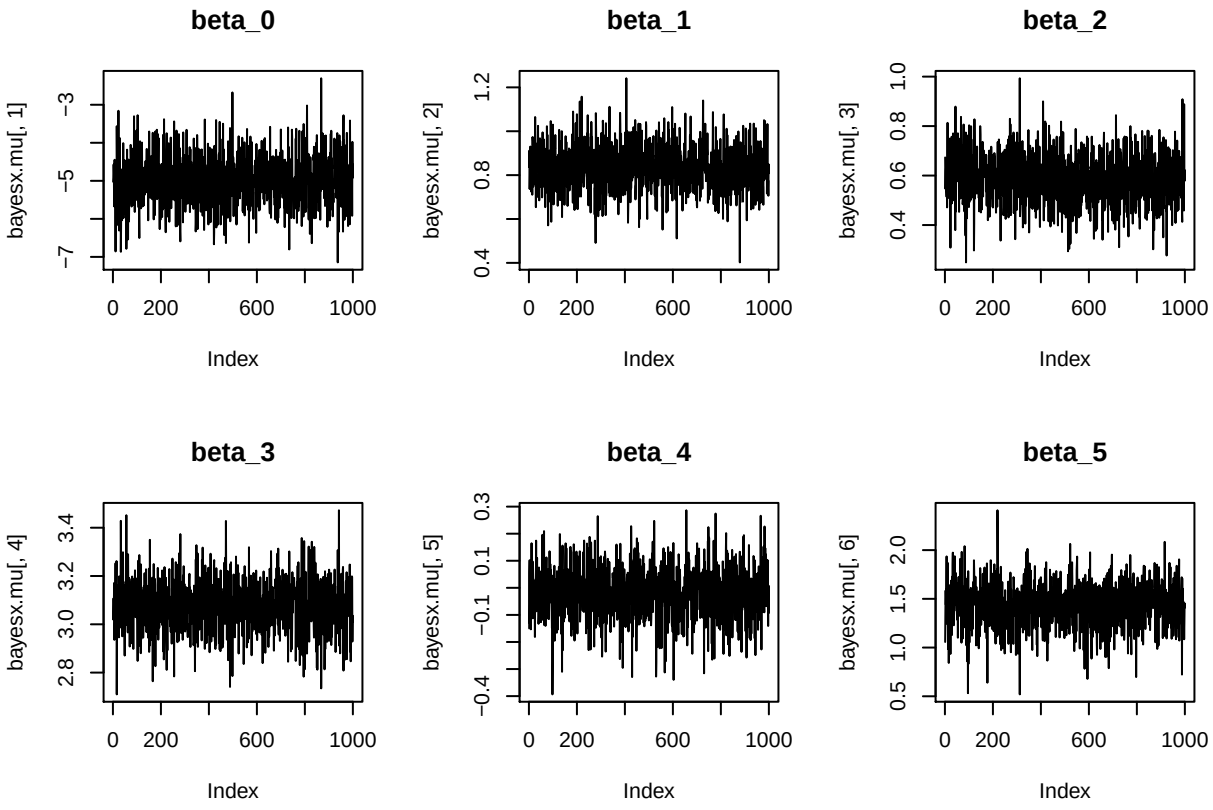
4.1.1 Thinning

```
par(mfrow= c(2,3))
acf(bayesx.mu[,1], main = "beta_0")
acf(bayesx.mu[,2],main = "beta_1")
acf(bayesx.mu[,3],main = "beta_2")
acf(bayesx.mu[,4],main = "beta_3")
acf(bayesx.mu[,5],main = "beta_4")
acf(bayesx.mu[,6],main = "beta_5")
```



4.1.2 Burn-in

```
par(mfrow= c(2,3))
plot(bayesx.mu[,1], type = "l", main = "beta_0")
plot(bayesx.mu[,2], type = "l", main = "beta_1")
plot(bayesx.mu[,3], type = "l", main = "beta_2")
plot(bayesx.mu[,4], type = "l", main = "beta_3")
plot(bayesx.mu[,5], type = "l", main = "beta_4")
plot(bayesx.mu[,6], type = "l", main = "beta_5")
```

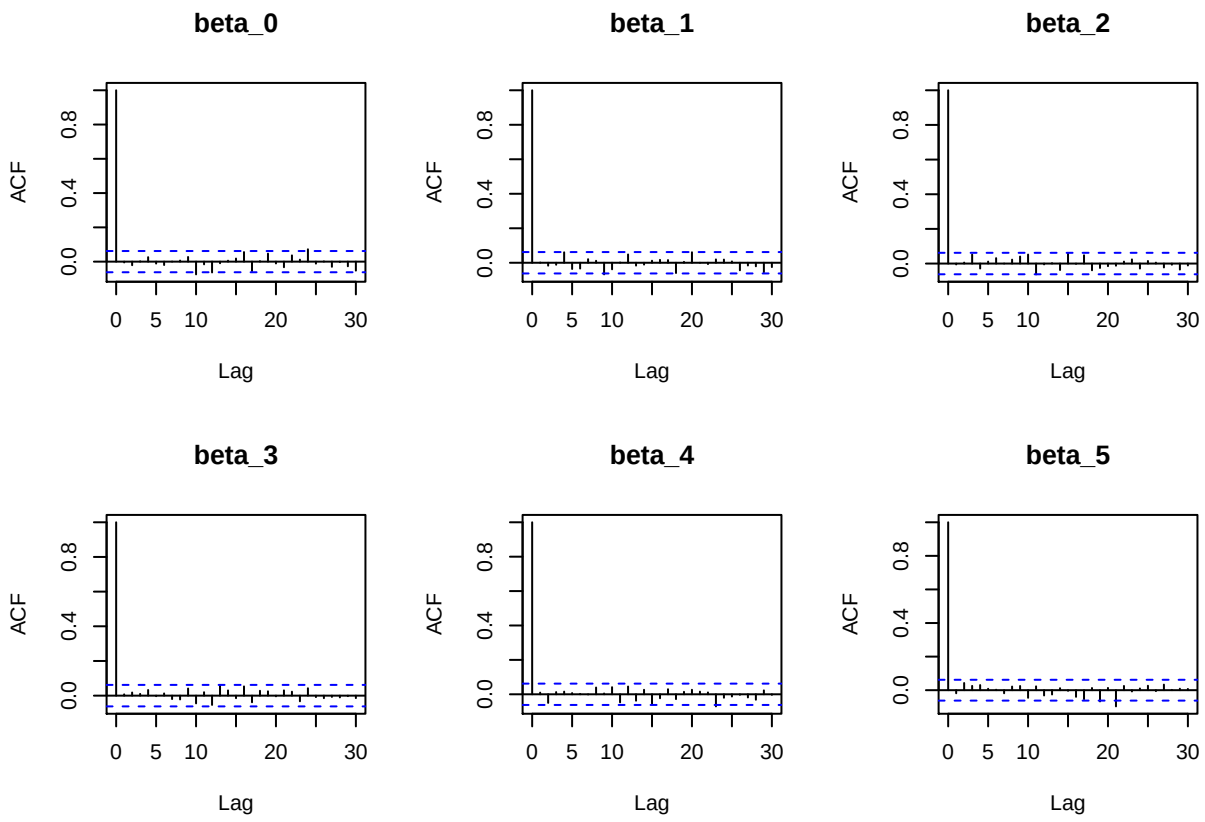


4.2 Scenarij II

```
data.N3 <- gen.beta.data(100, napaka = rnorm(100,0,3))
bayesx.norm <- bayesx(y ~ X1+X2+X3+X4+X5 , data = data.N3,
                      family = "gaussian", method = "MCMC")
bayesx.mu <- attr(bayesx.norm$fixed.effects, "sample")
```

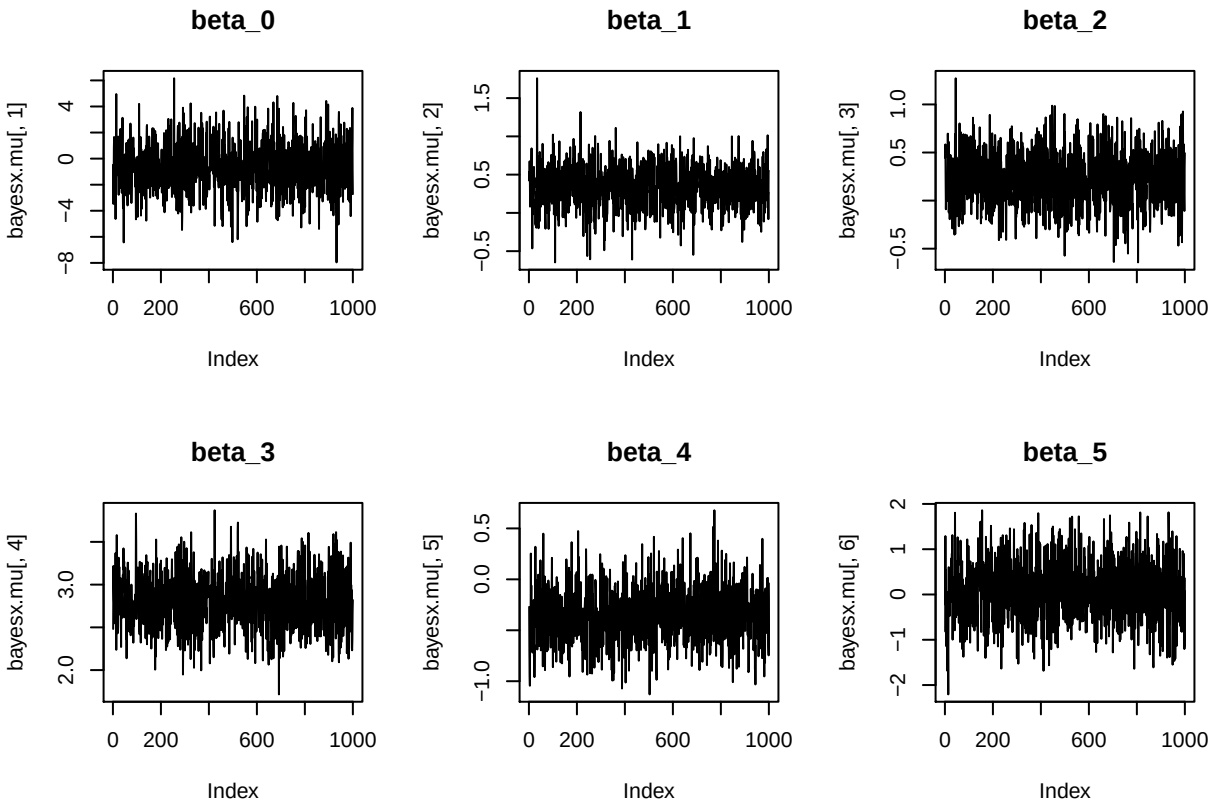
4.2.1 Thinning

```
par(mfrow= c(2,3))
acf(bayesx.mu[,1], main = "beta_0")
acf(bayesx.mu[,2],main = "beta_1")
acf(bayesx.mu[,3],main = "beta_2")
acf(bayesx.mu[,4],main = "beta_3")
acf(bayesx.mu[,5],main = "beta_4")
acf(bayesx.mu[,6],main = "beta_5")
```



4.2.2 Burn-in

```
par(mfrow= c(2,3))
plot(bayesx.mu[,1], type = "l", main = "beta_0")
plot(bayesx.mu[,2], type = "l", main = "beta_1")
plot(bayesx.mu[,3], type = "l", main = "beta_2")
plot(bayesx.mu[,4], type = "l", main = "beta_3")
plot(bayesx.mu[,5], type = "l", main = "beta_4")
plot(bayesx.mu[,6], type = "l", main = "beta_5")
```

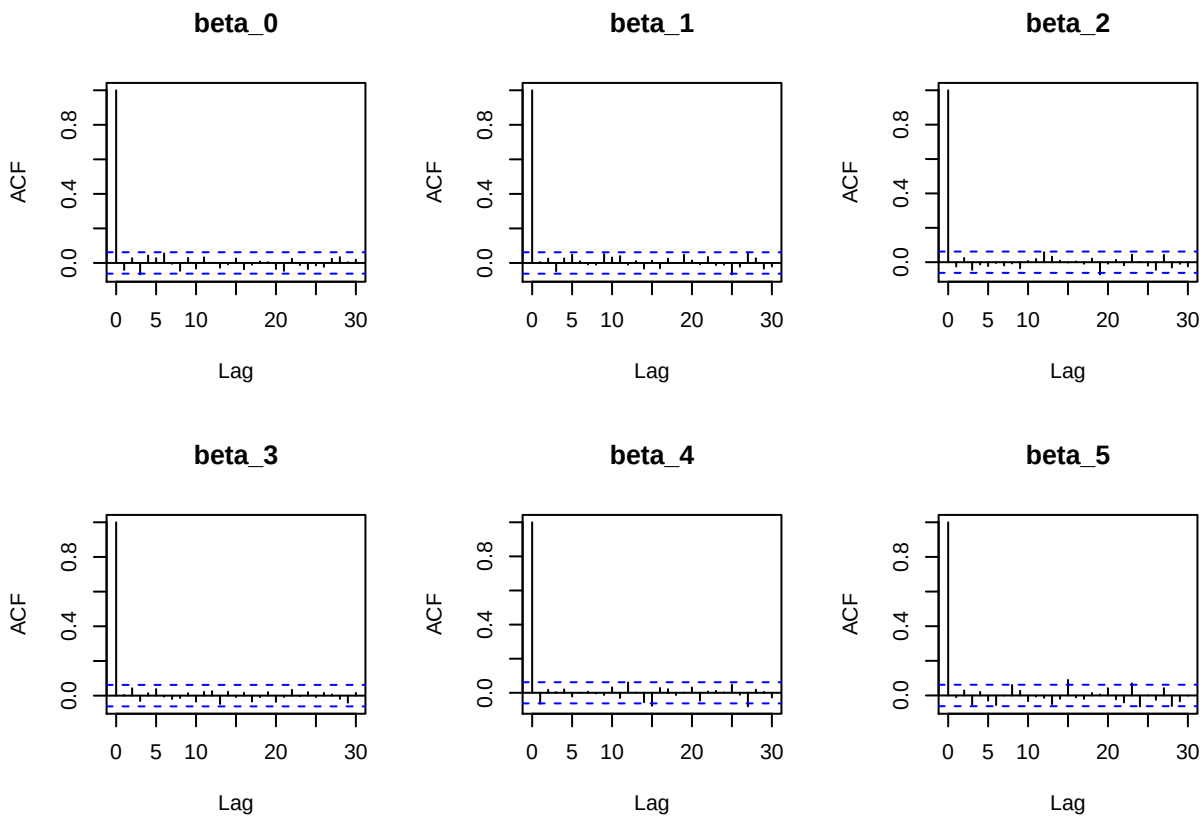


4.3 Scenarij III

```
data.H1 <- gen.beta.data(100, napaka = rchisq(100, 1))
bayesx.norm <- bayesx(y ~ X1+X2+X3+X4+X5 , data = data.H1,
                      family = "gaussian", method = "MCMC")
bayesx.mu <- attr(bayesx.norm$fixed.effects, "sample")
```

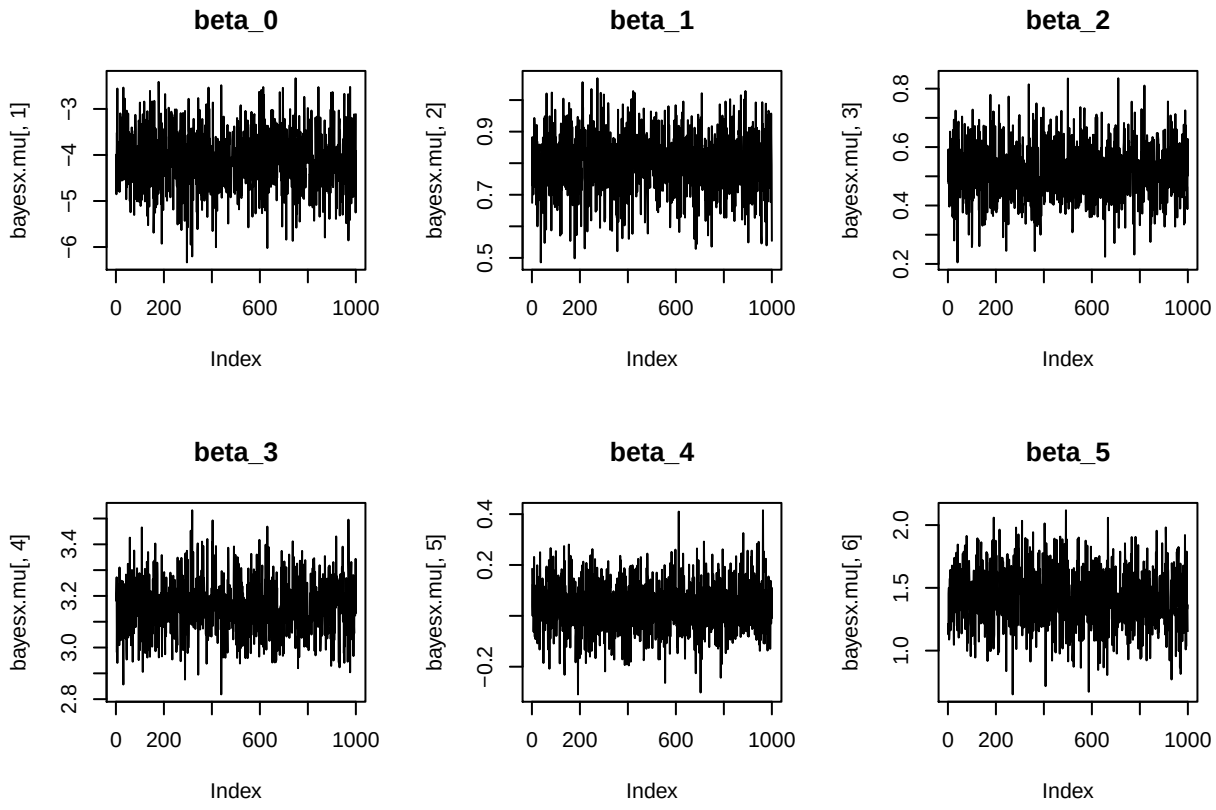
4.3.1 Thinning

```
par(mfrow= c(2,3))
acf(bayesx.mu[,1], main = "beta_0")
acf(bayesx.mu[,2],main = "beta_1")
acf(bayesx.mu[,3],main = "beta_2")
acf(bayesx.mu[,4],main = "beta_3")
acf(bayesx.mu[,5],main = "beta_4")
acf(bayesx.mu[,6],main = "beta_5")
```



4.3.2 Burn-in

```
par(mfrow= c(2,3))
plot(bayesx.mu[,1], type = "l", main = "beta_0")
plot(bayesx.mu[,2], type = "l", main = "beta_1")
plot(bayesx.mu[,3], type = "l", main = "beta_2")
plot(bayesx.mu[,4], type = "l", main = "beta_3")
plot(bayesx.mu[,5], type = "l", main = "beta_4")
plot(bayesx.mu[,6], type = "l", main = "beta_5")
```

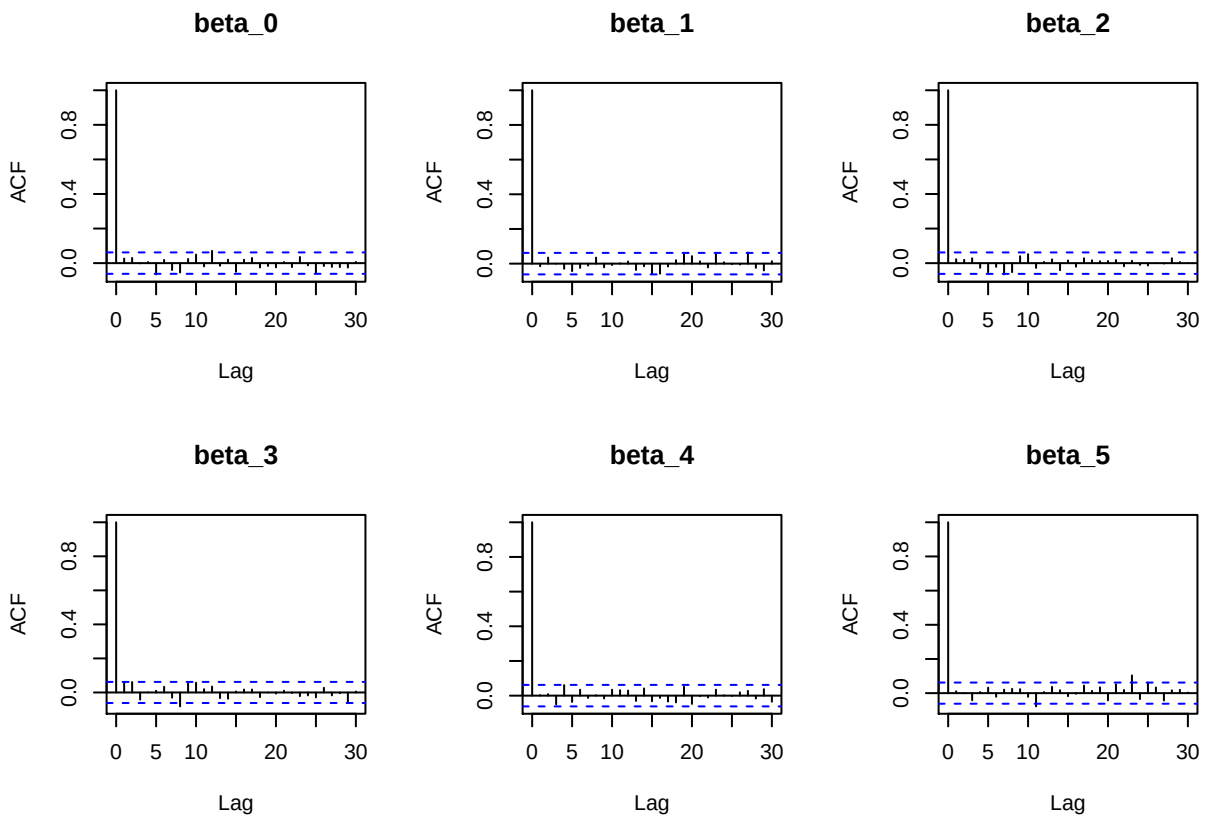


4.3.3 Scenarij IV

```
data.H4 <- gen.beta.data(100, napaka = rchisq(100, 4))
bayesx.norm <- bayesx(y ~ X1+X2+X3+X4+X5 , data = data.H4,
                      family = "gaussian", method = "MCMC")
bayesx.mu <- attr(bayesx.norm$fixed.effects, "sample")
```

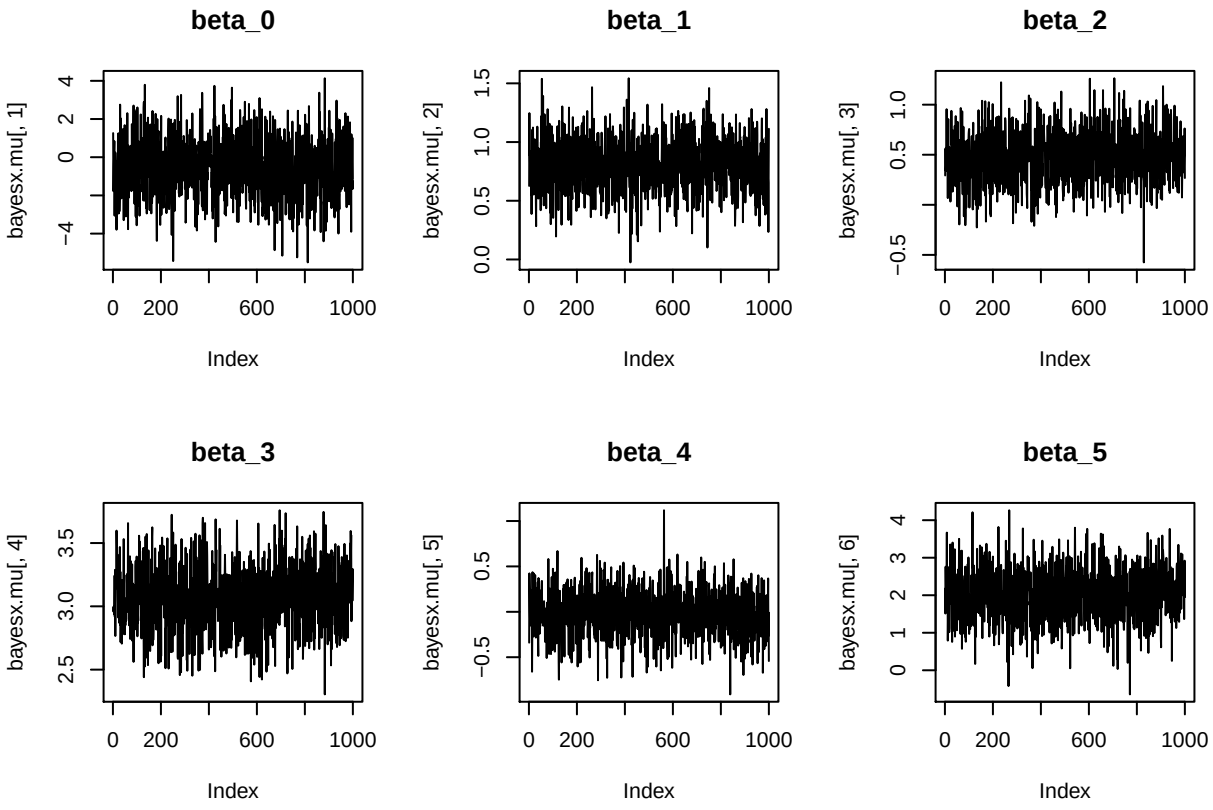
4.3.4 Thinning

```
par(mfrow= c(2,3))
acf(bayesx.mu[,1], main = "beta_0")
acf(bayesx.mu[,2],main = "beta_1")
acf(bayesx.mu[,3],main = "beta_2")
acf(bayesx.mu[,4],main = "beta_3")
acf(bayesx.mu[,5],main = "beta_4")
acf(bayesx.mu[,6],main = "beta_5")
```



4.3.5 Burn-in

```
par(mfrow= c(2,3))
plot(bayesx.mu[,1], type = "l", main = "beta_0")
plot(bayesx.mu[,2], type = "l", main = "beta_1")
plot(bayesx.mu[,3], type = "l", main = "beta_2")
plot(bayesx.mu[,4], type = "l", main = "beta_3")
plot(bayesx.mu[,5], type = "l", main = "beta_4")
plot(bayesx.mu[,6], type = "l", main = "beta_5")
```


beta_0
bayesx.mu[, 1]
−4
0
2
4
0
200
600
1000
Index
beta_1
bayesx.mu[, 2]
0.0
0.5
1.0
1.5
0
200
600
1000
Index
beta_2
bayesx.mu[, 3]
−0.5
0.5
1.0
0
200
600
1000
Index
beta_3
bayesx.mu[, 4]
2.5
3.0
3.5
0
200
600
1000
Index
beta_4
bayesx.mu[, 5]
−0.5
0.5
0
200
600
1000
Index
beta_5
bayesx.mu[, 6]
0
1
2
3
4
0
200
600
1000
Index