4. sklop: Algoritem Metropolis-Hastings

Nina Ruzic Gorenjec

Imamo nek model, za katerega poznamo verjetje $L(\theta \mid x)$ (θ je lahko vektor parametrov, podatki x pa matrika). Odlocimo se za apriorno porazdelitev $\pi(\theta)$, njeno formulo torej poznamo.

Izracunamo torej *le se* aposteriorno porazdelitev $\pi(\theta \mid x)$ preko Bayesove formule

$$\pi(\theta \mid x) \propto L(\theta \mid x)\, \pi(\theta).$$

Preko aposteriorne porazdelitve izracunamo *vse, kar nam srce pozeli*: za tockovno oceno vzamemo povprecje ali pa mediano, za interval zaupanja izracunamo primerne kvantile, izracunamo verjetnosti nasih domnev.

Kaj je tu problem?

1 MCMC (Markov Chain Monte Carlo)

Zelimo aposteriorno porazdelitev za parameter θ, natancneje vzorec iz aposteriorne porazdelitve.

Ideja algoritma:

1. Izberemo si zacetno vrednost $\theta^{(0)}$.
2. *Najdemo primerno* porazdelitev $p(\cdot|\cdot)$, tako da bomo lahko na vsakem koraku i dobili $\theta^{(i+1)}$ kot simulacijo iz porazdelitve $p(\theta^{(i+1)}|\theta^{(i)})$.
3. Po n korakih dobimo realizacijo $\theta^{(0)}, \theta^{(1)}, \ldots, \theta^{(n)}$. Postopek ponavljamo *dokler ne dosezemo zeljene konvergence*, pri cemer v koncnem vzorcu izpustimo zacetnih nekaj vrednosti (*burn-in*, npr. m).
4. Izbrani vzorec $\theta^{(m+1)}, \theta^{(m+2)}, \ldots, \theta^{(n)}$ je vzorec iz aposteriorne porazdelitve.
5. Preizkusimo razlicne zacetne vrednosti in primerjamo dobljene rezultate.

(6. Preizkusimo razlicne apriorne porazdelitve in primerjamo dobljene rezultate.)

2 Algoritem Metropolis-Hastings

Algoritem Metropolis-Hastings je eden izmed MCMC algoritmov.

Najprej si izberemo *proposal distribution* $q(\cdot|\cdot)$, ki je *primerne* oblike.

En korak Metropolis-Hastings algoritma, ko imamo iz prejsnjega koraka na voljo $\theta^{(i)}$:

1. Simuliramo kandidata θ^* iz porazdelitve $q(\cdot|\theta^{(i)})$.
2. Izracunamo verjetnost sprejetja (*acceptance probability*)

$$\alpha = \min\left\{1, \frac{L(\theta^*|x)\ \pi(\theta^*)\ \ q(\theta^{(i)}|\theta^*)}{L(\theta^{(i)}|x)\pi(\theta^{(i)})q(\theta^*|\theta^{(i)})}\right\}.$$

3. Simuliramo u iz enakomerne zvezne porazdelitve na [0,1].
4. Ce je $u \leq \alpha$, izberemo kandidata (*accept*) in postavimo $\theta^{(i+1)} = \theta^*$.
   Ce je $u > \alpha$, zavrnemo kandidata (*reject*) in postavimo $\theta^{(i+1)} = \theta^{(i)}$.

3 Algoritem Metropolis-Hastings za primer normalnega modela z znano varianco

Uporabili bomo algoritem Metropolis-Hastings za primer iz 3. sklopa, kjer so bili nasi podatki vzorec visin (metri) studentov moskega spola:

```
x <- c(1.91, 1.94, 1.68, 1.75, 1.81, 1.83, 1.91, 1.95, 1.77, 1.98,
       1.81, 1.75, 1.89, 1.89, 1.83, 1.89, 1.99, 1.65, 1.82, 1.65,
       1.73, 1.73, 1.88, 1.81, 1.84, 1.83, 1.84, 1.72, 1.91, 1.63)
```

Privzeli smo normalni model $N(\theta, \sigma^2 = 0.1^2)$ in zeleli oceniti θ.

Verjetje tega modela je enako

$$L(\theta \mid x) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\,0.1} e^{-\frac{(x_i-\theta)^2}{2\cdot(0.1)^2}}.$$

Za apriorno porazdelitev smo si izbrali normalno porazdelitev (konjugirana v tem modelu) s povprecjem $\mu_0 = 1.78$ in varianco $\sigma_0^2 = 0.2^2$.

Za aposteriorno porazdelitev smo zato dobili normalno porazdelitev s parametroma μ_n in σ_n^2, kjer je

$$\frac{1}{\sigma_n^2} = \frac{1}{\sigma_0^2} + \frac{n}{\sigma^2},$$

$$\mu_n = \frac{1/\sigma_0^2}{1/\sigma_0^2 + n/\sigma^2}\,\mu_0 + \frac{n/\sigma^2}{1/\sigma_0^2 + n/\sigma^2}\,\bar{x}.$$

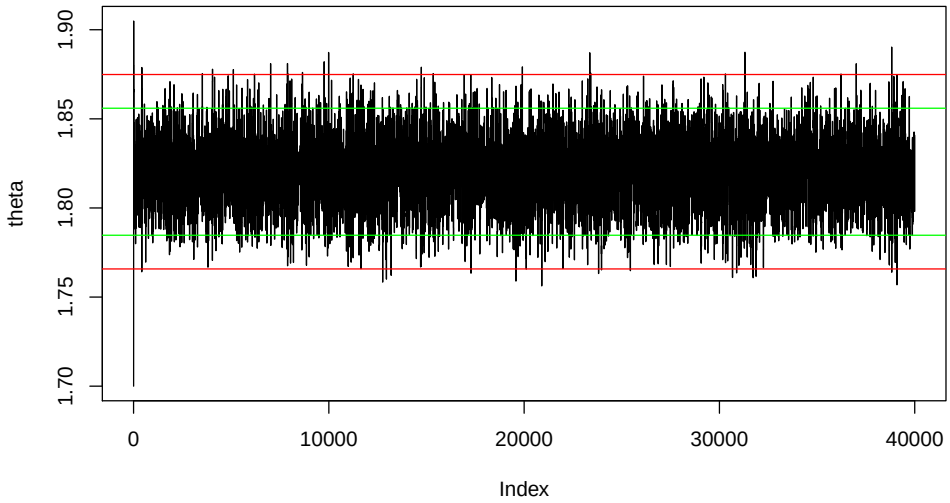
(Zapisemo lahko tudi preko *precision*=1/varianca.)

Pravo aposteriorno porazdelitev torej poznamo.

Sedaj jo bomo aproksimirali s pomocjo Metropolis-Hastings algoritma.

Najprej si moramo smiselno izbrati *proposal distribution* $q(\cdot|\theta^{(i)})$. Katero bi si izbrali? Normalno s povprecjem $\theta^{(i)}$ in nekim standardnim odklonom.

Primer zaporedja iz aposteriorne porazdelitve, ki *izgleda dobro*:



Na sliki smo za boljso predstavo oznacili 95% referencni interval prave aposterirone porazdelitve (zeleni crti) in odmik od povprecja aposteriorne porazdelitve za 3 standardne odklone, tj. 99.7% referencni interval (rdeci crti). **Pozor:** S tem smo uporabili vedenje o pravi aposteriorni porazdelitvi, ki v realni situaciji ni znana (ravno zato jo z MCMC metodami tudi ocenjujemo).

Pomembno: Cilj konvergence je porazdelitev, ne pa stevilo.

3.1 Druga domaca naloga

Za primer iz 3. sklopa (uporabite zgornje podatke, model z $\sigma^2 = 0.1^2$ in zgornjo apriorno porazdelitev z $\mu_0 = 1.78$ in $\sigma_0^2 = 0.2^2$ – ti parametri so fiksni) aproksimirajte aposteriorno porazdelitev s pomocjo algoritma Metropolis-Hastings, kjer sledite spodnjim korakom.

1. Sami v R-u sprogramirajte algoritem Metropolis-Hastings za primer ocenjevanja enega parametra oz. za nas primer. Kljucno je, da ga sprogramirate sami, pri cemer splosnost kode in efektivnost implementacije nista pomembni. (Za ta preprost primer boste npr. 40000 iteracij dobili v zelo kratkem casu, ne glede na izbor parametrov v spodnjih tockah ali efektivnost implementacije.)
2. Preizkusite ga na nasem primeru, kjer si sami izberite neko smiselno zacetno vrednost in varianco *proposal distribution*. Rezultate predstavite na naslednji nacin:
   - Narisite celotno dobljeno zaporedje $\theta^{(i)}$ (glede na iteracije i).
   - Narisite le prvih 500 ali pa 5000 clenov.
   - Narisite celotno zaporedje, kjer uporabite ustrezen *burn-in*.
   - Za tako izbrano zaporedje graficno predstavite aposteriorno porazdelitev in jo graficno primerjajte s pravo aposteriorno porazdelitvijo.
   - Ocenite parameter in 95% interval zaupanja za parameter iz izbranega zaporedja ter primerjajte z ocenami iz prave aposterirone porazdelitve.
3. Pozenite vas algoritem pri neki nesmiselni zacetni vrednosti. **Pozor:** ce boste α implementirali po formuli iz str. 2, potem algoritem pri zelo nesmiselnih zacetnih vrednostih ne bo deloval – zato je potrebno implementirati na ravni logaritma (primerno prilagodite korake algoritma). Rezultate predstavite:
   - Opisite, zakaj konkretno so se pojavile tezave, ce ste uporabili zelo nesmiselno zacetno vrednost in osnovno verzijo algoritma (brez logaritmiranja).
   - Narisite celotno dobljeno zaporedje $\theta^{(i)}$ (glede na iteracije i).
   - Narisite le prvih 500 ali pa 5000 clenov.
   - Narisite celotno zaporedje, kjer uporabite ustrezen *burn-in*.
4. Pri neki smiselni zacetni vrednosti pozenite algoritem pri nekaj razlicnih variancah za *proposal distribution*. Pri izboru pretiravajte v obe smeri (spomnite se, kaksni so po velikosti nasi podatki), tako da boste graficno opazili razlike na prvih npr. 500 iteracijah. Rezultate predstavite:
   - Za vsak primer narisite prvih nekaj (nekje med 500 in 5000) clenov in se celotno zaporedje.
   - Komentirajte razlike in zakaj do njih pride. Kaj in zakaj vas moti pri izbranih primerih?
   - **Bonus vprasanje:** Kaksen bi bil v splosnem (ne vezano na nas vzorec) vas predlog glede izbora variance *proposal distribution* oz. kaksen bi bil predlog za izbor koncnega zaporedja?

3.2 Resitve

Fiksni parametri nasega modela:

```
sigma <- 0.1

mu0 <- 1.78
sigma0 <- 0.2
```

Parametri prave aposteriorne porazdelitve:

```
n <- length(x)
prec <- 1/sigma^2
prec0 <- 1/sigma0^2

prec.n <- prec0 + n*prec
sigma.n <- sqrt(1/prec.n)

mu.n <- prec0/prec.n * mu0 + n*prec/prec.n * mean(x)
```

Porazdelitve, ki bodo nastopale v algoritmu Metropolis-Hastings (podane izven algoritma) – implementacija na logaritemski skali (najbolj preprosto uporabiti `log = TRUE`):

```
#Logaritem verjetja
loglik <- function(x, theta, sigma = sigma) {
   sum(dnorm(x, mean = theta, sd = sigma, log = TRUE))
}

#Apriorna porazdelitev (log)
logprior <- function(theta, mu0 = mu0, sigma0 = sigma0) {
  dnorm(theta, mean = mu0, sd = sigma0, log = TRUE)
}

#Predlagalna porazdelitev - vzorcenje
rq <- function(theta, sigma.q) {
  rnorm(1, mean = theta, sd = sigma.q)
}

#Predlagalna porazdelitev - gostota (log)
logdq <- function(theta.arg, theta.cond, sigma.q) {
  dnorm(theta.arg, mean = theta.cond, sd = sigma.q, log = TRUE)
}
```

Algoritem Metropolis-Hastings:

```
mh <- function(n.iter=40500, theta.init=1.7, sigma.q=0.1,
               x=x, sigma=sigma, mu0 = mu0, sigma0 = sigma0) {
  theta <- rep(NA, n.iter)
  theta[1] <- theta.init

  for(i in 2:n.iter) {
    new.theta <- rq(theta[i - 1], sigma.q=sigma.q)

    #Log-Accepance probability
    logacc.prob <- loglik(x, theta=new.theta, sigma=sigma) +
      logprior(theta=new.theta, mu0 = mu0, sigma0 = sigma0) +
      logdq(theta[i - 1], new.theta, sigma.q=sigma.q) -
      loglik(x, theta=theta[i - 1], sigma=sigma) -
      logprior(theta=theta[i - 1], mu0 = mu0, sigma0 = sigma0) -
      logdq(new.theta, theta[i - 1], sigma.q=sigma.q)
    logacc.prob <- min(0, logacc.prob) #0 = log(1)

    if(log(runif(1)) < logacc.prob) {
      #Accept
      theta[i] <- new.theta
    } else {
      #Reject
      theta[i] <- theta[i - 1]
    }
  }
  return(theta)
}
```

Zakaj smo ga implementirali na logaritemski skali?

Ob nesmiselni zacetni vrednosti bi bili vrednosti verjetja in apriorne porazdelitve v zacetni vrednosti in naslednjem kandidatu tako majhne, da bi dobili v stevcu in imenovalcu 0 in s tem *acceptance probability* ne bi bil definiran (NaN). Preizkusimo:

```
#Verjetje
lik <- function(x, theta, sigma = sigma) {
   sum(dnorm(x, mean = theta, sd = sigma))
}

#Apriorna porazdelitev
prior <- function(theta, mu0 = mu0, sigma0 = sigma0) {
  dnorm(theta, mean = mu0, sd = sigma0)
}
```

```
#Predlagalna porazdelitev - gostota
dq <- function(theta.arg, theta.cond, sigma.q) {
  dnorm(theta.arg, mean = theta.cond, sd = sigma.q)
}

theta.init=10
sigma.q=0.1
n.iter=40500

theta <- rep(NA, n.iter)
theta[1] <- theta.init

i = 2
new.theta <- rq(theta[i - 1], sigma.q=sigma.q)

(stevec <- lik(x, theta=new.theta, sigma=sigma)*
   prior(theta=new.theta, mu0 = mu0, sigma0 = sigma0)*
   dq(theta[i - 1], new.theta, sigma.q=sigma.q))
```

```
## [1] 0
```

```
(imenovalec <- lik(x, theta=theta[i - 1], sigma=sigma)*
   prior(theta=theta[i - 1], mu0 = mu0, sigma0 = sigma0)*
   dq(new.theta, theta[i - 1], sigma.q=sigma.q))
```

```
## [1] 0
```

```
(acc.prob <- stevec/imenovalec)
```

```
## [1] NaN
```

Ali bi lahko pri izbrani predlagalni porazdelitvi algoritem poenostavili?

Zaradi simetricnosti normalne porazdelitve je $q(\theta^{(i)}|\theta^*) = q(\theta^*|\theta^{(i)})$, ta clen bi torej lahko pri tem izboru vrste predlagalne porazdelitve izpustili. Preizkusimo:

```
dq(1.7, 1.8, sigma.q)
```

```
## [1] 2.419707
```

```
dq(1.8, 1.7, sigma.q)
```

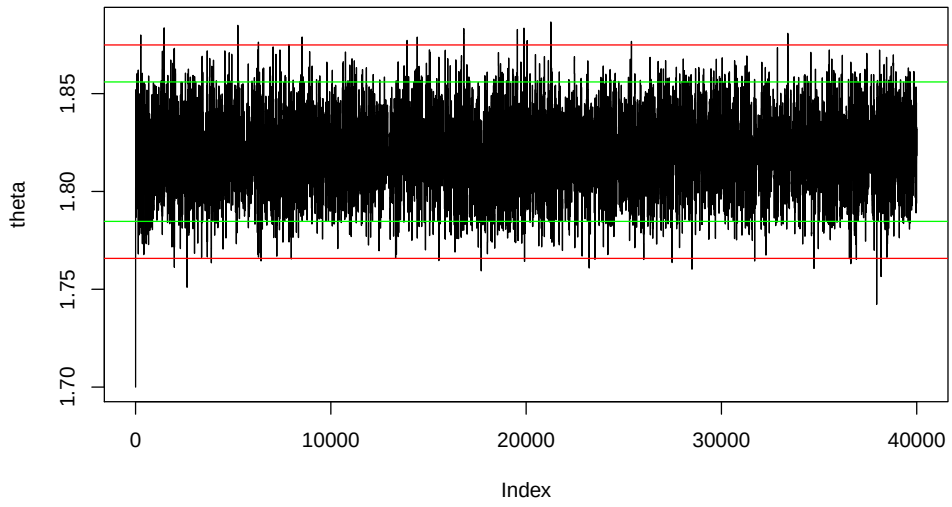
```
## [1] 2.419707
```

3.2.1 Primer s smiselno zacetno vrednostjo in "ustrezno" varianco predlagalne porazdelitve

Uporabimo zacetno vrednost 1.7 metra (smiselno za visino, povprecje vzorca 1.8) in standardni odklon (SO) predlagalne porazdelitve 0.1. V naslednjem koraku se bomo torej lahko oddaljili najvec do nekje $\pm 3 \cdot 0.1 = \pm 0.3$ metra. To je po eni strani gotovo dovolj, da raziscemo celotno obmocje aposteriorne porazdelitve (razpon vzorca od 1.6 do 2, ocenjujemo povprecje), po drugi strani pa nas ne omeji na premajhno obmocje okoli prejsnje vrednosti $\theta^{(i)}$. Seveda bi bila tudi marsikatera druga vrednost za SO predlagalne porazdelitve ustrezna.

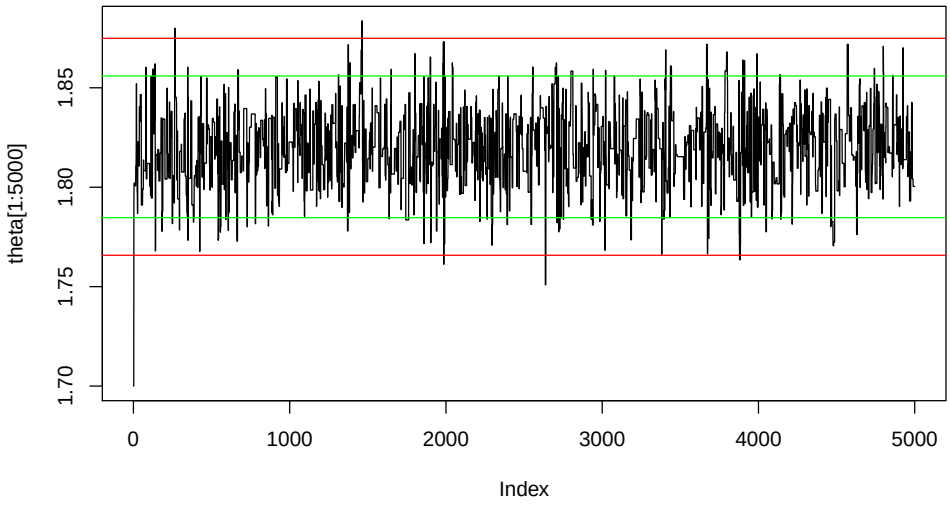
Narisemo dobljeni vzorec:

```
theta <- mh(n.iter=40000, theta.init=1.7, sigma.q=0.1,
            x=x, sigma=sigma, mu0 = mu0, sigma0 = sigma0)

plot(theta, type="l")
abline(h=mu.n-1.96*sigma.n, col="green")
abline(h=mu.n+1.96*sigma.n, col="green")
abline(h=mu.n-3*sigma.n, col="red")
abline(h=mu.n+3*sigma.n, col="red")
```

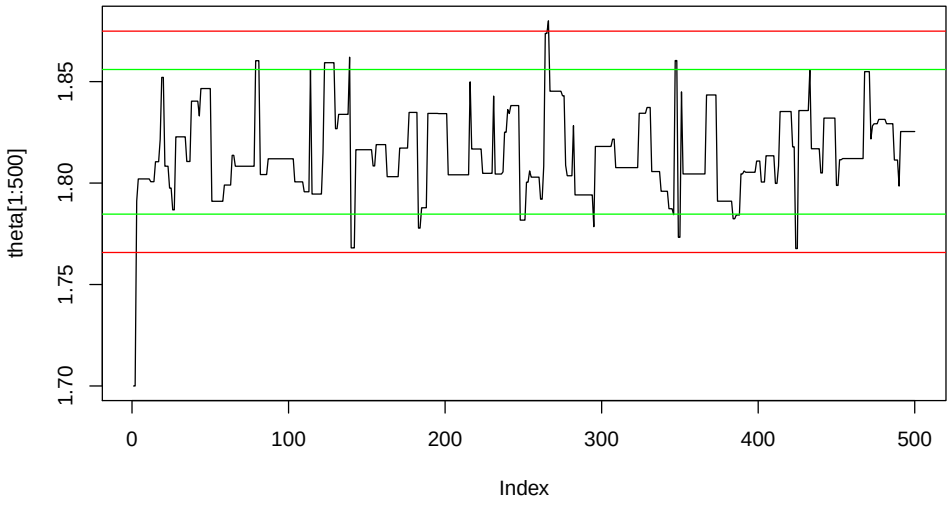


Na sliki smo za boljso predstavo oznacili 95% referencni interval prave aposterirone porazdelitve (zeleni crti) in odmik od povprecja aposteriorne porazdelitve za 3 standardne odklone, tj. 99.7% referencni interval (rdeci crti). **Pozor:** S tem smo uporabili vedenje o pravi aposteriorni porazdelitvi, ki v realni situaciji ni znana (ravno zato jo z MCMC metodami tudi ocenjujemo).

Bolj podrobno si ogledamo prvih 5000 iteracij:

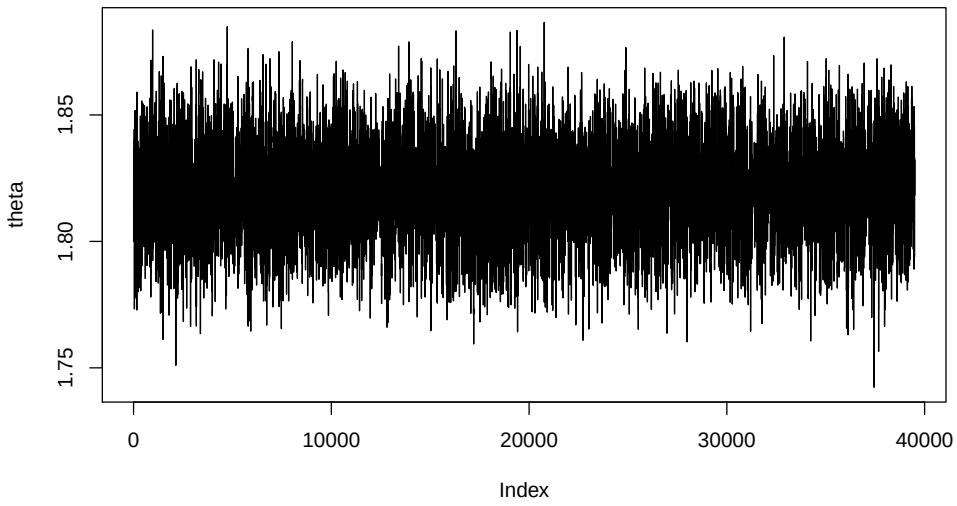


Bolj podrobno si ogledamo prvih 500 iteracij:



Potreben burn-in je zelo majhen, ker pa imamo veliko iteracij, vseeno vzamemo za burn-in kar 500:

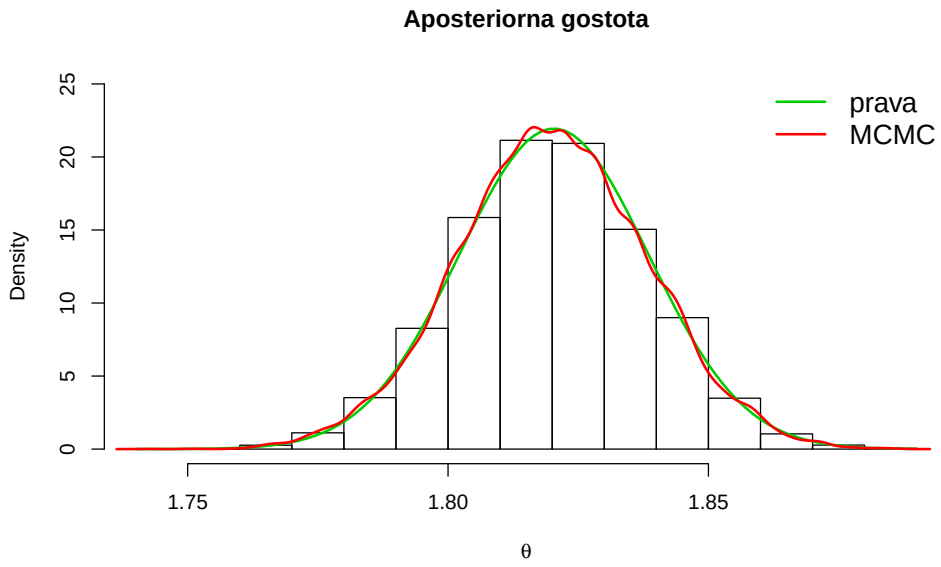
```
theta <- theta[-c(1:500)]
plot(theta, type="l")
```



Veriga torej izgleda primerno – skace gor in dol po nekem omejenem območju, "precesava" porazdelitev.

Primerjamo s pravo aposteriorno **porazdelitvijo – cilj konvergence je cela porazdelitev** (dobimo podobno):

```
### Narisemo
hist(theta, prob=T, main = "Aposteriorna gostota", xlab = expression(theta),
     ylim=c(0,25))
curve(dnorm(x, mean=mu.n, sd=sigma.n), add=T, col="green3", lwd=2)
lines(density(theta), col="red", lwd=2)
legend("topright",  lty = 1, lwd=2,
       c("prava", "MCMC"), col = c("green3","red"), bty = "n", cex = 1.3)
```



Primerjamo dobljeni tockovni oceni (dobimo podobno):

```
mean(theta) #MCMC
```

```
## [1] 1.820035
```

```
mu.n #teoreticno
```

```
## [1] 1.820331
```

Primerjamo dobljena kredibilna intervala (dobimo podobno):

```
library(HDInterval)
hdi(theta, credMass = 0.95) #MCMC
```

```
##    lower    upper
## 1.782682 1.854699
## attr(,"credMass")
## [1] 0.95
```

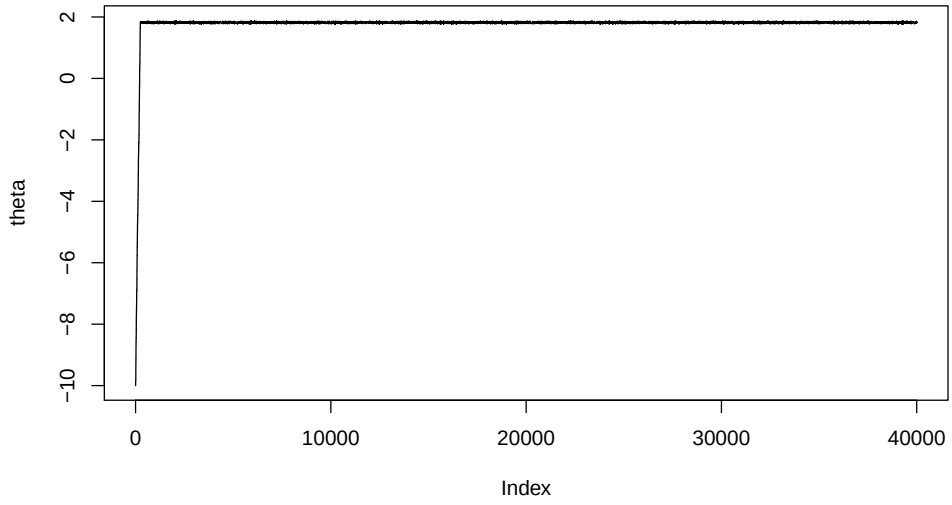
```
qnorm(c(0.025, 0.975), mean = mu.n, sd = sigma.n) #teoreticno
```

```
## [1] 1.784695 1.855966
```

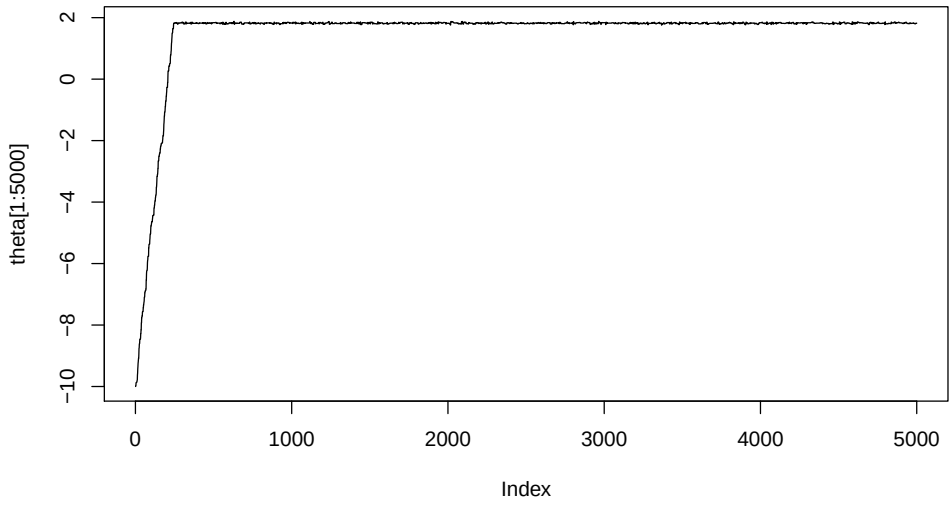
3.2.2 Primer z nesmiselno zacetno vrednostjo (varianca predlagalne porazdelitve se zmeraj "ustrezna")

Drasticno preizkusimo -10 kot zacetno vrednost – algoritem se zmeraj deluje, ker smo ga implementirali na logaritemski skali.

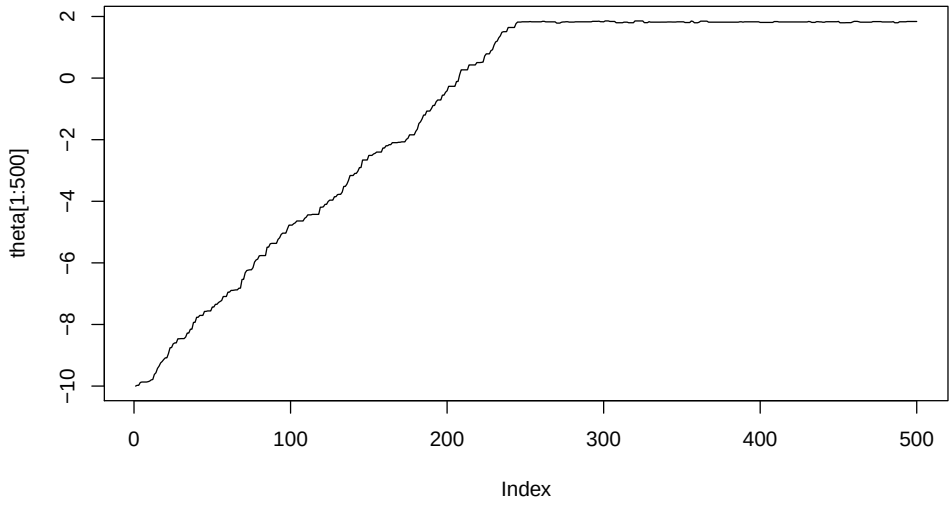
```
theta <- mh(n.iter=40000, theta.init=-10, sigma.q=0.1,
            x=x, sigma=sigma, mu0 = mu0, sigma0 = sigma0)
plot(theta, type="l")
```



Bolj podrobno si ogledamo prvih 5000 iteracij:

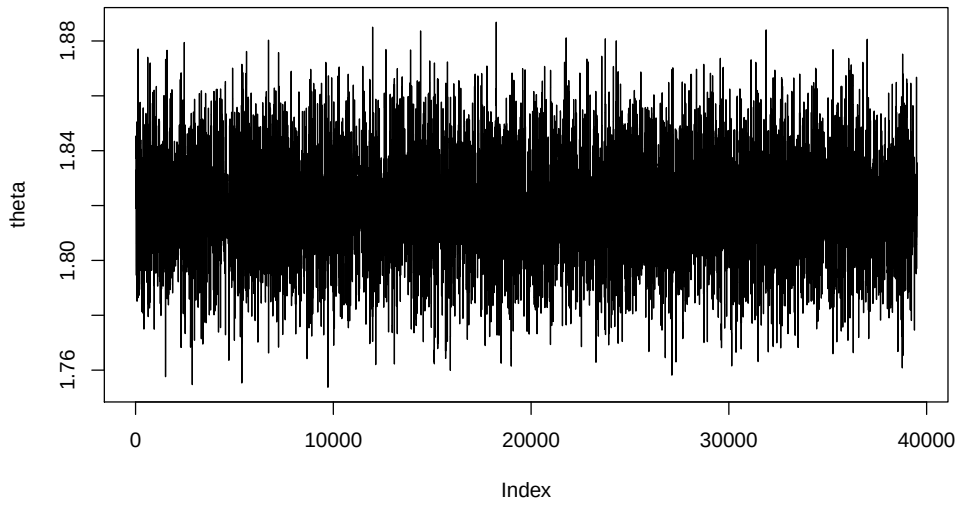


Bolj podrobno si ogledamo prvih 500 iteracij:



Odstranimo burn-in – je nujno, vendar kljub zelo nesmiselni zacetni vrednosti le-ta ni velik, saj imamo zelo preprost model z efektivnim vzorcevalnikom (*sampler*) :

```
theta <- theta[-c(1:500)]
plot(theta, type="l")
```



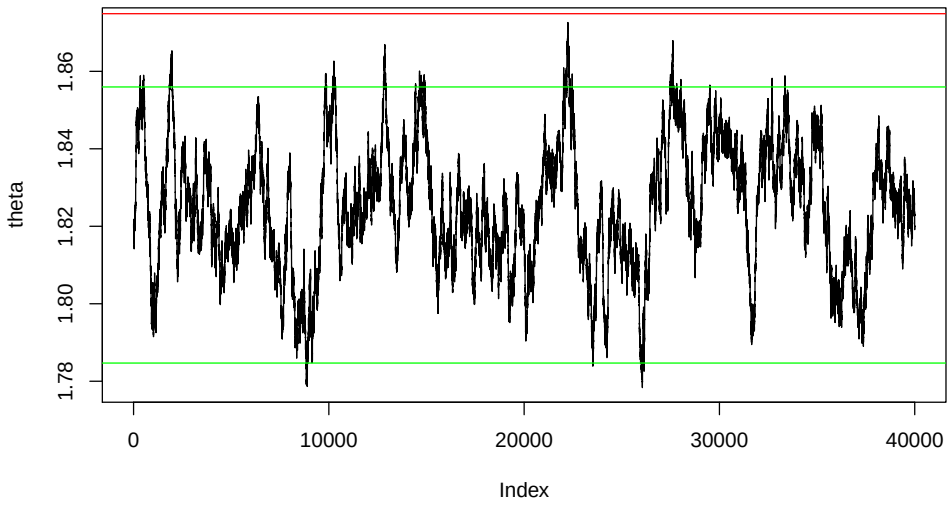
Taksna veriga izgleda primerno – **nas model in vzorcevalnik nista obcutljiva na izbor razlicnih zacetnih vrednosti** (pri bolj kompleksnih modelih seveda ne bi preizkusali tako cudnih zacetnih vrednostih, kot smo jih tu).

3.2.3 Primer z zelo majhno varianco predlagalne porazdelitve (pri tem je zacetna vrednost smiselna)

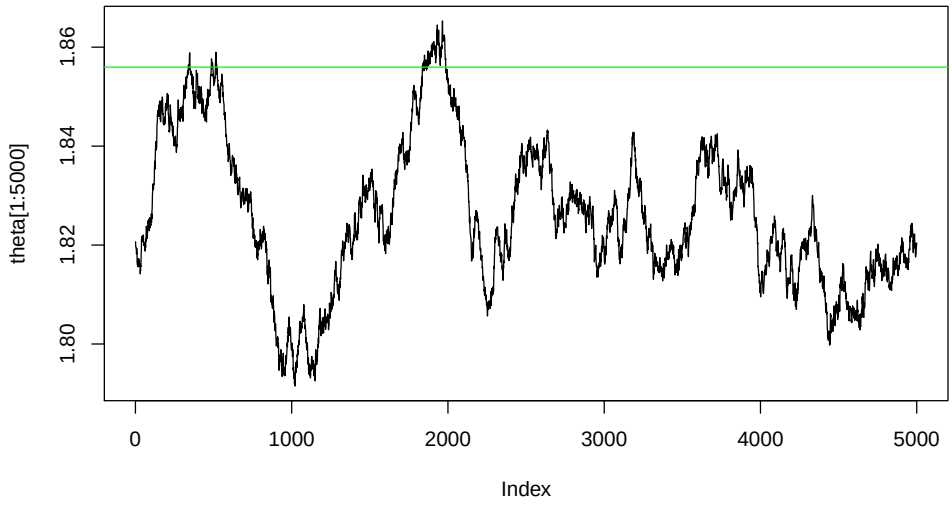
Za zacetno vrednost vzamemo povprecje vzorca, burn-in zato niti ne bomo potrebovali (v praksi vedno vzamemo burn-in, saj v kompleksnejsih modelih ponavadi zacnemo pri vsaj nekoliko prevec oddaljenih vrednostih). Za SO predlagalne porazdelitve vzamemo 0.001. V naslednjem koraku se bomo torej lahko oddaljili najvec do nekje $\pm 3 \cdot 0.001 = \pm 0.003$, tj. 3 mm — omejimo se na zelo majhno obmocje okoli prejsnje $\theta^{(i)}$. Ko smo torej na nekem obmocju aposteriorne porazdelitve, bomo lahko presli na drugi konec porazdelitve v zelo kratkih korakih in bomo zato aposterirono porazdelitev raziskovali zelo pocasi.

Narisemo dobljeni vzorec:

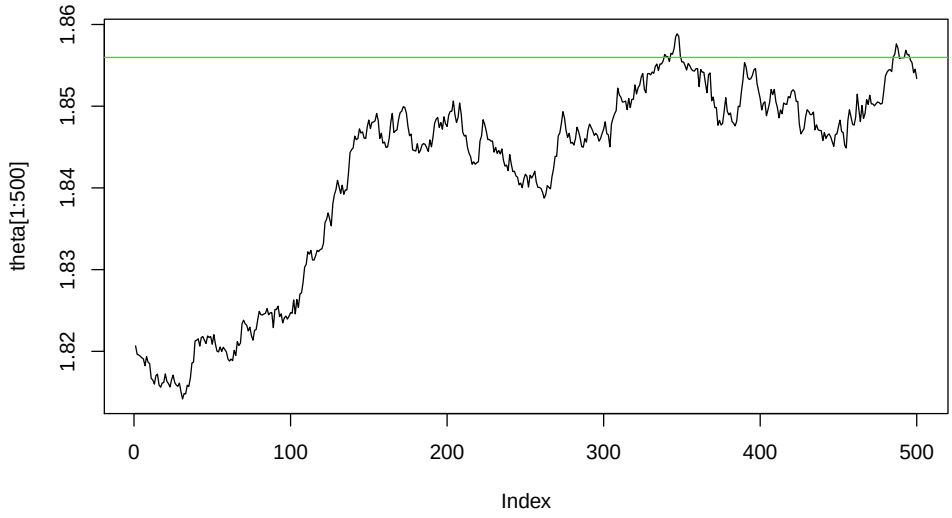
```
theta <- mh(n.iter=40000, theta.init=mean(x), sigma.q=0.001,
            x=x, sigma=sigma, mu0 = mu0, sigma0 = sigma0)

plot(theta, type="l")
abline(h=mu.n-1.96*sigma.n, col="green")
abline(h=mu.n+1.96*sigma.n, col="green")
abline(h=mu.n-3*sigma.n, col="red")
abline(h=mu.n+3*sigma.n, col="red")
```



Bolj podrobno si ogledamo prvih 5000 iteracij:

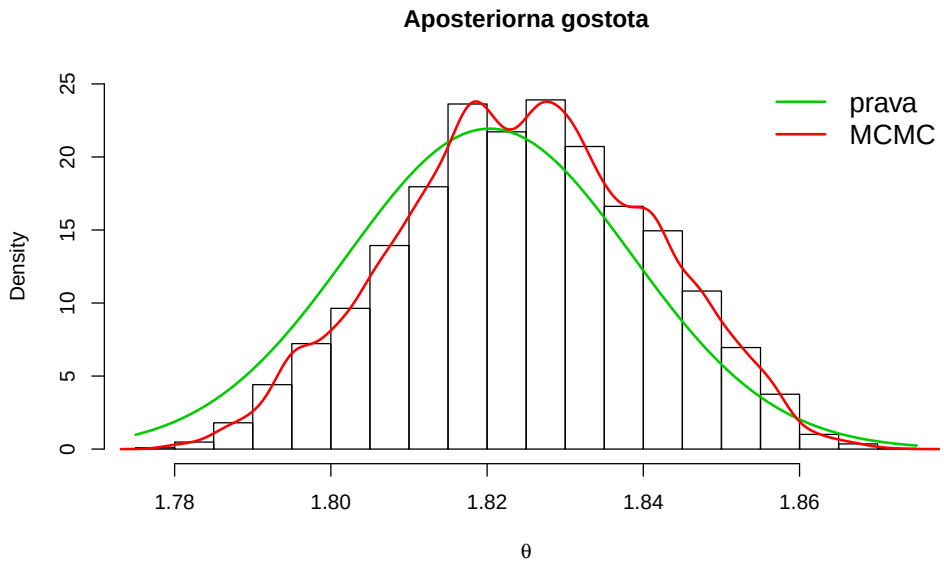


Bolj podrobno si ogledamo prvih 500 iteracij:

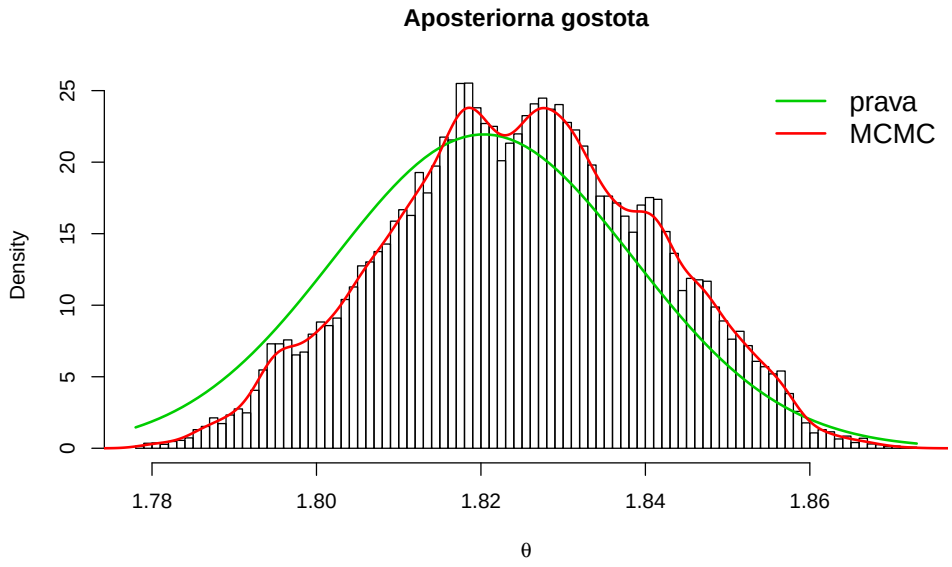


Nase razmisljanje o problemih ob premajhni varianci predlagalne porazdelitve se je torej potrdilo – **porazdelitev prepocasi raziskujemo, tj. iteracije so med seboj prevec korelirane** (vec o tem spodaj).

Najbolj pomembno je seveda, kaj dobimo za rezultat, tj. aposteriorno porazdelitev:



Za boljsi prikaz povecamo stevilo stolpcev, tako da je bolj jasna oblika zglajene gostote (rdeca krivulja):



Nas priblizek za aposteriorno porazdelitev je torej obcutno slabsi kot pri prvem primeru (`sigma.q = 0.001`) ob enakem stevilu iteracij, tj. ob enaki casovni in prostorski zahtevnosti algoritma.

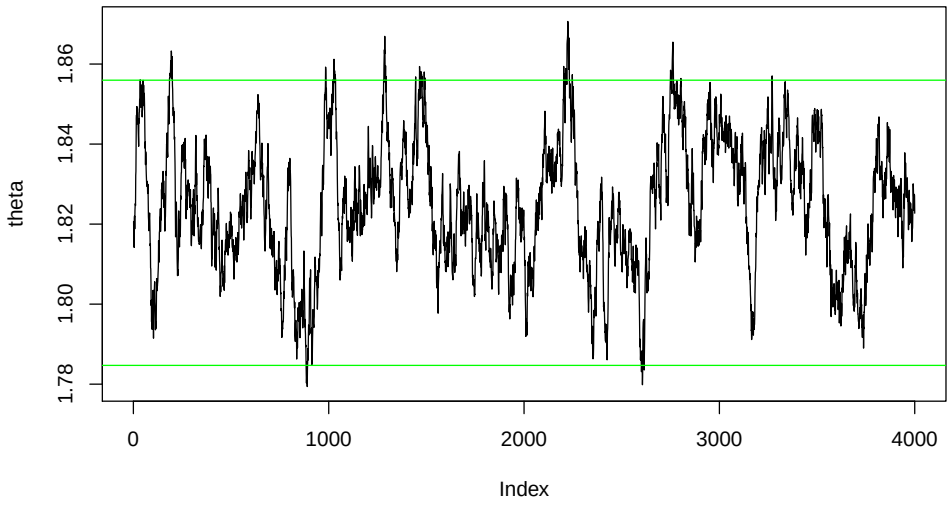
Ali *thinning* pomaga?

O tem bomo sicer podrobneje govorili na naslednjih vajah. Na kratko: v koncni vzorec vzamemo vsakega k-tega. Ideja je, da se s tem izognemo korelaciji med cleni verige, tj. avtokorelaciji. Le-ta je pri MCMC metodah vedno prisotna zaradi same narave vzorcenja, avtokoreliran vzorec pa ne moremo imeti za slucajni vzorec (neodvisne enako porazdeljene, IID) iz porazdelitve.

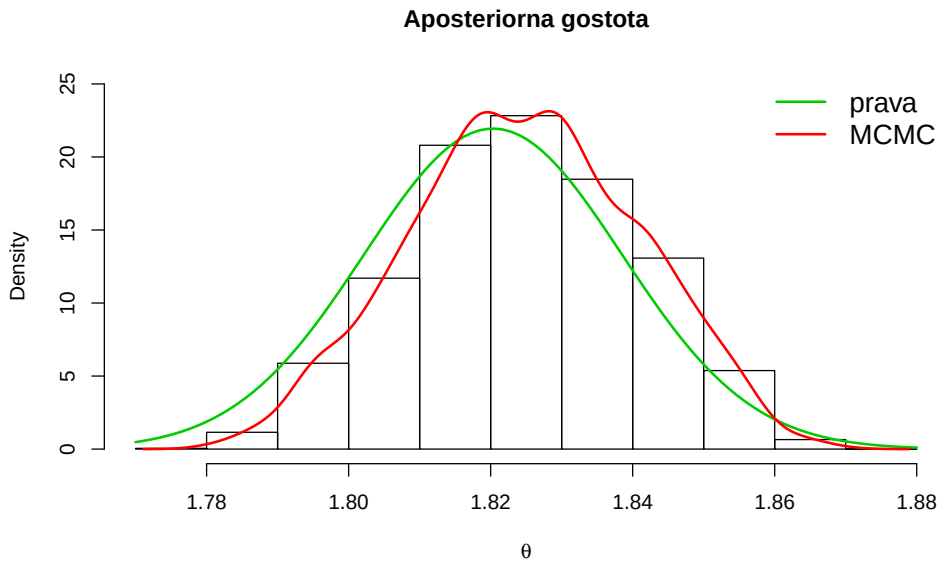
V vzorec vzamemo vsakega 10-tega (velik faktor):

```
theta <- theta[seq(1, length(theta), by = 10)]
```

Razredcena veriga – malo bolje, vendar se zmeraj kar mocna avtokorelacija, saj smo vzeli res zelo majhen SO predlagalne porazdelitve:



Dobljena aposteriorna porazdelitev:



Zavedati se moramo predvsem, da smo dobesedno vrgli stran 90 % ze izracunanih clenov, casovna zahtevnost torej ostaja enaka ob bistveno manjsem koncnem vzorcu. Prihranili bi lahko le pri prostorski zahtevnosti, saj bi lahko algoritem implementirali tako, da bi si po desetih izracunanih iteracijah shranili le zadnjega, preostale pa pozabili. Ob casovni zahtevnosti na eni strani in velikosti vzorca iz aposteriorne porazdelitve na drugi strani (pa ceprav je avtokoreliran), je prostorska zahtevnost pravzaprav edina prednost *thinninga*, kar pa v dobi modernih raucnalnikov ni vec taksen problem. **Redcenje verig (*thinning*) se torej v praksi opusca, porocamo pa t.i. *effective sample size***, tj. stevilo neodvisnih opazovanj, ki se izracuna tako, da se ob stevilu iteracij upposteva avtokorelacije vzorca – vec o tem v 6. sklopu.

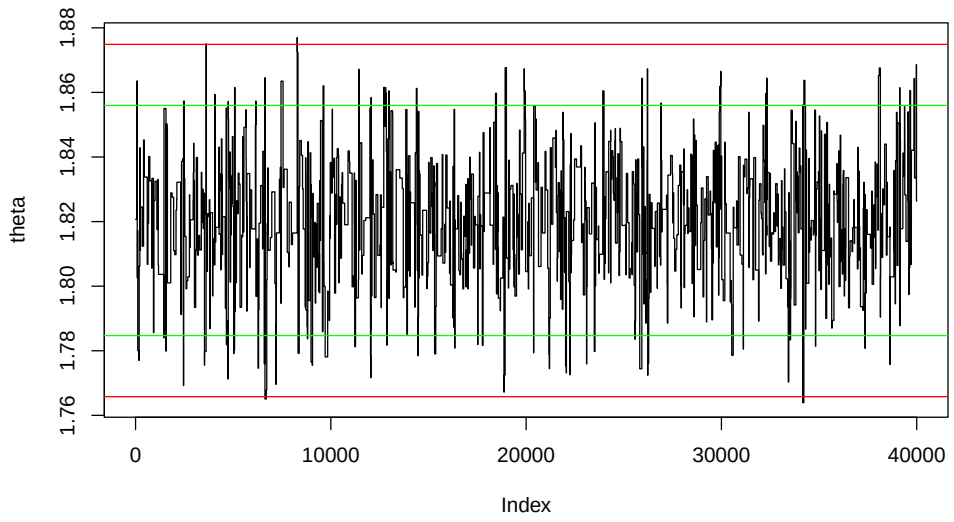
Seveda pa je prvi korak ob taksni verigi, da vzamemo bolj primeren vzorcevalnik oz. razmislimo o smiselnosti modela, ce se tezave zopet pojavijo.

3.2.4 Primer z zelo veliko varianco predlagalne porazdelitve (pri tem je zacetna vrednost smiselna)

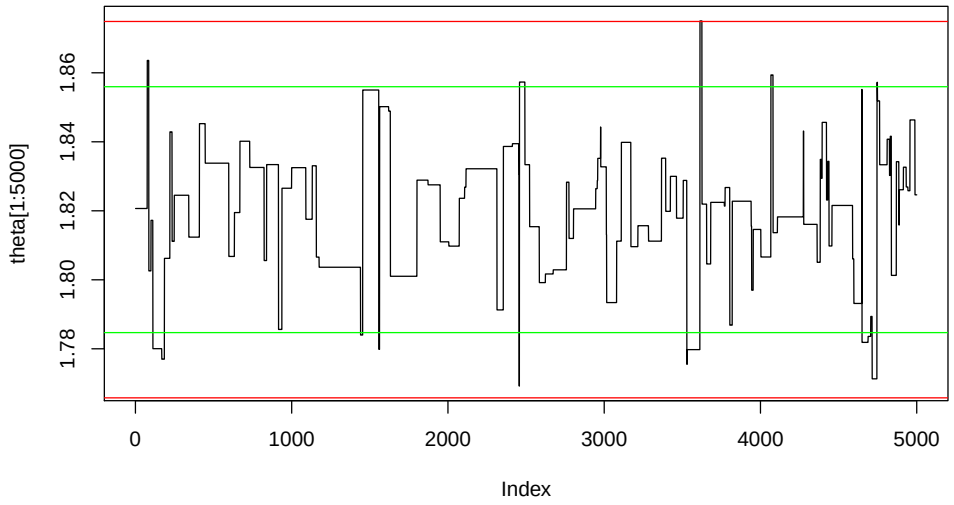
Spet vzamemo za zacetno vrednost povprecje vzorca, za SO predlagalne porazdelitve pa 1. V naslednjem koraku se bomo torej lahko oddaljili najvec do nekje $\pm 3 \cdot 1 = \pm 3$ metre – kandidati za novo vrednost θ^* lahko torej nekontrolirano skacejo okoli prejsnje $\theta^{(i)}$, zaradi cesar so z zelo veliko verjetnostjo neprimerni, manj primerni od prejsnjega. Ko smo torej na neki dokaj smiselni vrednosti iz aposteriorne porazdelitve, se bomo z zelo majhno verjetnostjo premaknili nekam drugam – **tratimo cas, medtem ko ne dobivamo novih opazovanj iz aposteriorne porazdelitve**.

Narisemo dobljeni vzorec:

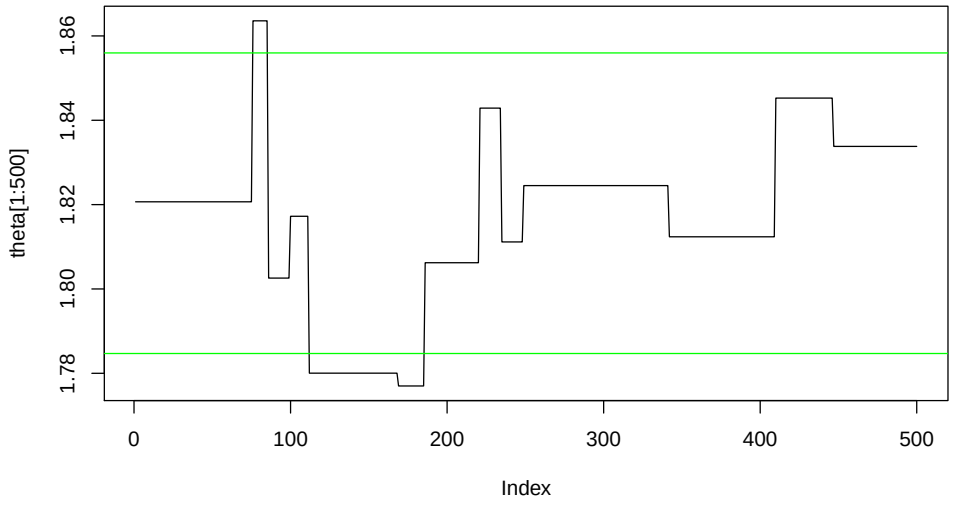
```
theta <- mh(n.iter=40000, theta.init=mean(x), sigma.q=1,
            x=x, sigma=sigma, mu0 = mu0, sigma0 = sigma0)

plot(theta, type="l")
abline(h=mu.n-1.96*sigma.n, col="green")
abline(h=mu.n+1.96*sigma.n, col="green")
abline(h=mu.n-3*sigma.n, col="red")
abline(h=mu.n+3*sigma.n, col="red")
```



Bolj podrobno si ogledamo prvih 5000 iteracij:

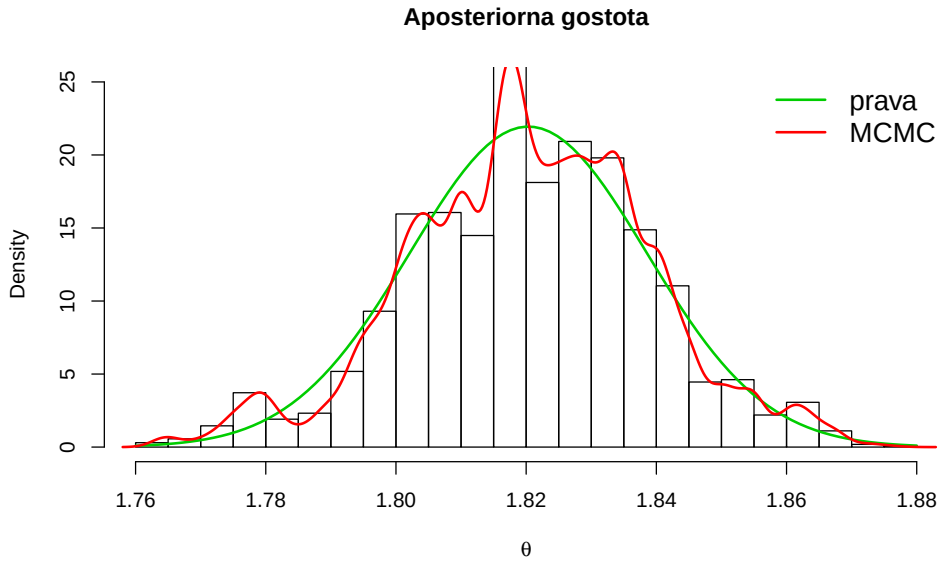


Bolj podrobno si ogledamo prvih 500 iteracij:

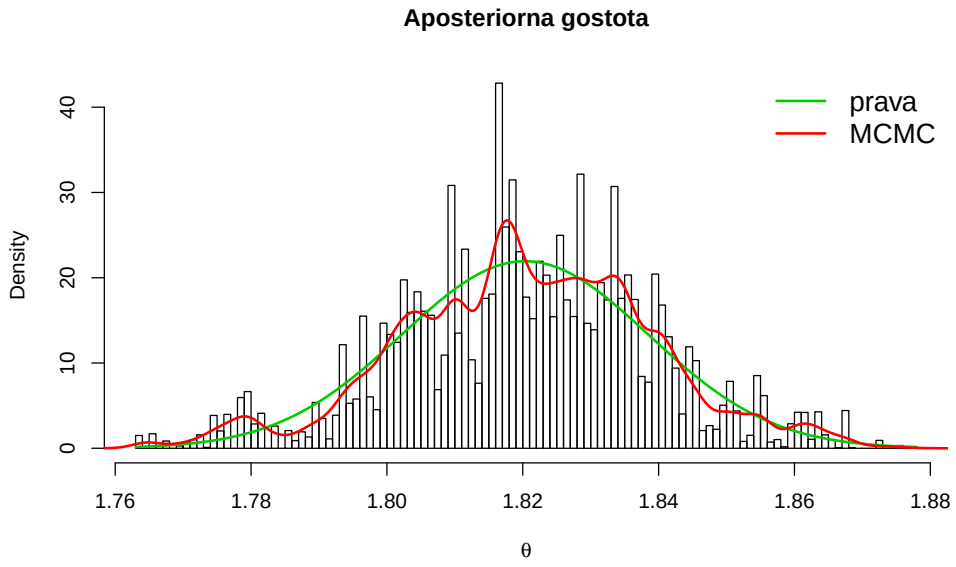


Nase razmisljanje o problemih ob preveliki varianci predlagalne porazdelitve se je torej potrdilo – veriga je velikokrat konstantna za veliko stevilo iteracij.

Dobljena aposteriorna porazdelitev:



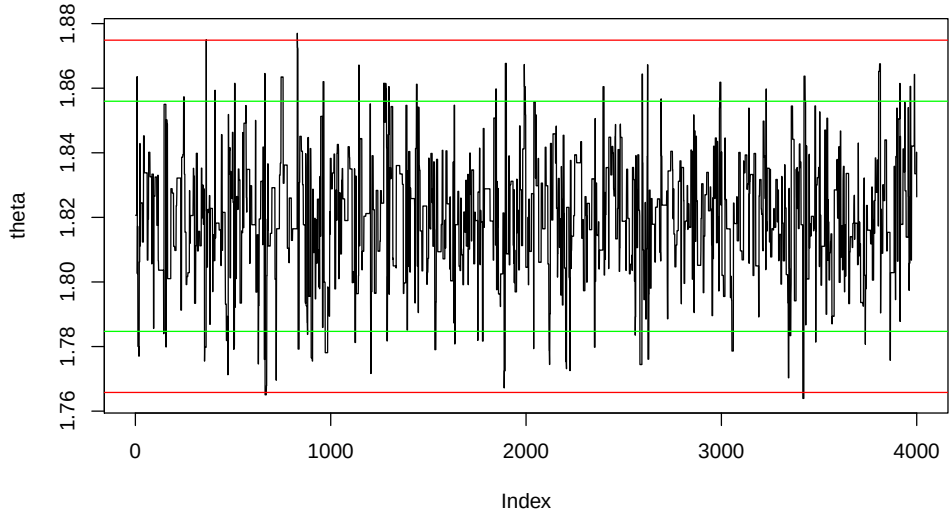
Za boljsi prikaz povecamo stevilo stolpcev – tako so postanejo se bolj jasni vrhovi zglajene gostote (rdeca krivulja), ki se primerijo tam, kjer je veriga konstantna za veliko stevilo iteracij:

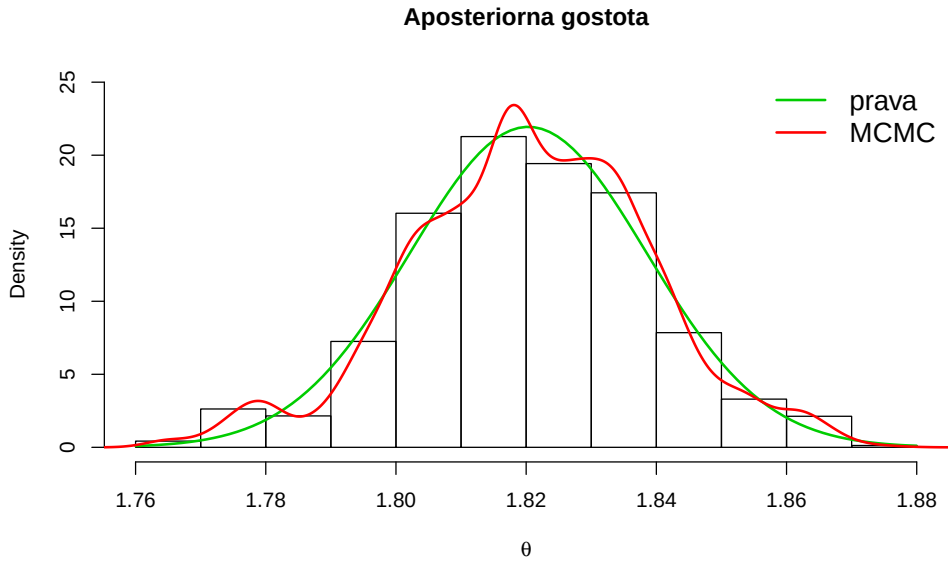


Nas priblizek za aposteriorno porazdelitev je torej tudi tu obcutno slabsi kot pri prvem primeru (`sigma.q = 0.001`) ob enaki casovni in prostorski zahtevnosti algoritma.

Ali *thinning* pomaga?

```
theta <- theta[seq(1, length(theta), by = 10)]
```





Podoben komentar kot prej – navidezno pomaga, saj je veriga bolj primerna, vendar imamo majhen vzorec ob veliki casovni zahtevnosti.

3.2.5 Kako izbrati primeno varianco predlagalne porzdelitve?

Algoritmi vzorcenja, ki so implementirani v knjiznicah, morajo seveda dobro delovati na sirokem naboru modelov oz. spremenljivk, predvsem bi bilo absurdno, da algoritem ne bi deloval ob spremembi enot spremenljivke – primerna velikost variance predlagalne porazdelitve je namrec odvisna od samih vrednosti spremenljivke.

Problem pri zgornjih dveh primerih je bil:

- Pri premajhni varianci je algoritem sprejel za novo vrednost skoraj vsakega kandidata θ^*, tj. delez sprejetih (*acceptance rate*) je bil prevelik.
- Pri preveliki varianci je algoritem malokrat sprejel kandidata θ^* za novo vrednost, tj. delez sprejetih je bil premajhen.

Resitev je *adaptive* algoritem, kjer vsakih npr. 200 korakov izracunamo, koliksen delez kandidatov smo v zadnjih 200 iteracijah sprejeli. Ce je delez premajhen ali prevelik, potem varianco predlagalne porazdelitve zmanjsamo oz. povecamo.

Kaj pomeni premajhen/prevelik delez sprejetih oz. kaksen delez sprejetih je optimalen? Tipicno je to 0.234 (teoreticno dokazano v nekih okvirih; vseeno ni vedno res najbolje). Fascinantno natancna stevilka za splosno uporabo, ki sicer presenetljivo ni 0.42 :)

4 Metropolis-Hastings za dvoparametricni model

Za primer vzamemo normalni model z dvema parametroma $\theta = (\mu, \tau)$, kjer je $\tau = 1/\sigma^2$, za nas vzorec x (glejte 5. sklop), kjer za apriorno porazdelitev vzamemo

$$\pi(\mu, \tau) = \pi(\mu) \cdot \pi(\tau), \quad \mu \sim N(0, \tau_0 = 0.001), \quad \tau \sim \text{Gama}(0.01, 0.01).$$

Vzeli smo sibko informativno porazdelitev (neodvisnost parametrov kot pri Jeffreyevi apriorni, vendar vzamemo pravi porazdelitvi za vsak parameter).

Za predlagalno porazdelitev si moramo izbrati dvorazsezno porazdelitev. Nov predlog za θ bomo vzorcili tako, da bomo neodvisno vzorcili nova predloga za μ in τ, sprejeli ali zavrnili pa ju bomo skupaj.

Nov predlog za μ bomo vzorcili iz $N(\mu^{(i-1)}, \tau_{\text{predlog}} = 10)$, kjer je τ_{predlog} *precision* predlagalne porazdelitve, τ pa iz log-normalne z ustreznima parametroma.

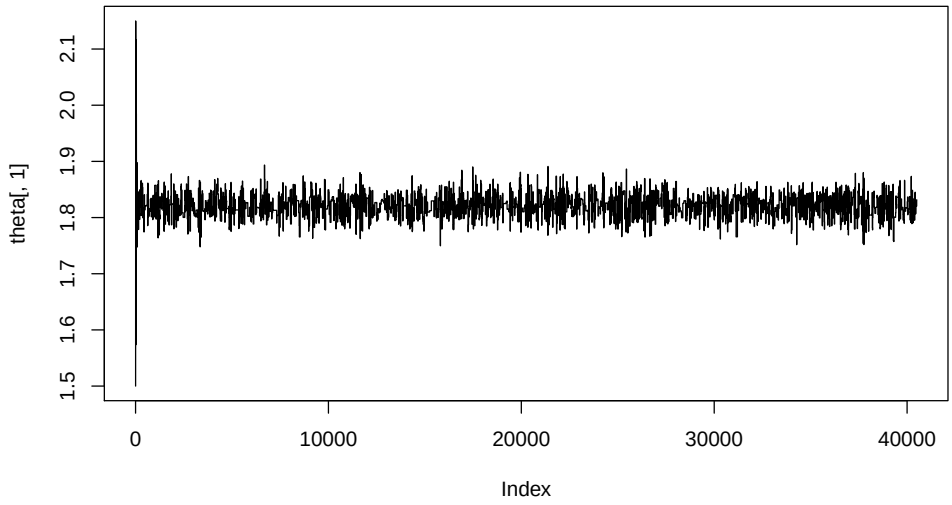
```
#Logaritem verjetja
loglik <- function(x, theta) {
   sum(dnorm(x, mean = theta[1], sd = sqrt(1 / theta[2]), log = TRUE))
}

#Apriorna porazdelitev (log)
logprior <- function(theta) {
  #na log skali sestejemo gostoti zaradi neodvisnosti
  dnorm(theta[1], 0, sd = sqrt(1 / 0.001), log = TRUE) +
    dgamma(theta[2], 0.01, 0.01, log = TRUE)
}

#Predlagalna porazdelitev - vzorcenje
rq <- function(theta) {
  rez <- c(NA, NA)
  #neodvisno vzorcimo vsak parameter posebej
  rez[1] <- rnorm(1, theta[1], sd = sqrt(1 / 10))
  rez[2] <- rlnorm(1, meanlog = log(theta[2]), sdlog = sqrt(1 / 10))
  return(rez)
}

#Predlagalna porazdelitev - gostota (log)
logdq <- function(theta.arg, theta.cond) {
  #na log skali sestejemo gostoti zaradi neodvisnosti
  dnorm(theta.arg[1], theta.cond[1], sd = sqrt(1 / 10), log = TRUE) +
    dlnorm(theta.arg[2], meanlog = log(theta.cond[2]), sdlog = sqrt(1 / 10), log = TRUE)
}
```

```r
#Enak algoritem kot prej,
#le da prilagodimo za racunanje in shranjevanje dveh parametrov.
#Ker uporabimo seznam (list), deluje spodnje za poljubno stevilo parametrov
#(doloceno z uporabljenimi funkcijami zgoraj).
mh2 <- function(n.iter=40500, theta.init=c(1.5, 1), x=x) {
  theta <- as.list(rep(NA, n.iter))
  theta[[1]] <- theta.init

  for(i in 2:n.iter) {
    new.theta <- rq(theta[[i - 1]])

    #Log-Accepance probability
    logacc.prob <- loglik(x, new.theta) +
      logprior(new.theta) +
      logdq(theta[[i - 1]], new.theta) -
      loglik(x, theta[[i - 1]]) -
      logprior(theta[[i - 1]]) -
      logdq(new.theta, theta[[i - 1]])
    logacc.prob <- min(0, logacc.prob)#0 = log(1)

    if(log(runif(1)) < logacc.prob) {
      #Accept
      theta[[i]] <- new.theta
    } else {
      #Reject
      theta[[i]] <- theta[[i - 1]]
    }
  }
  return(do.call(rbind, theta))
}
```

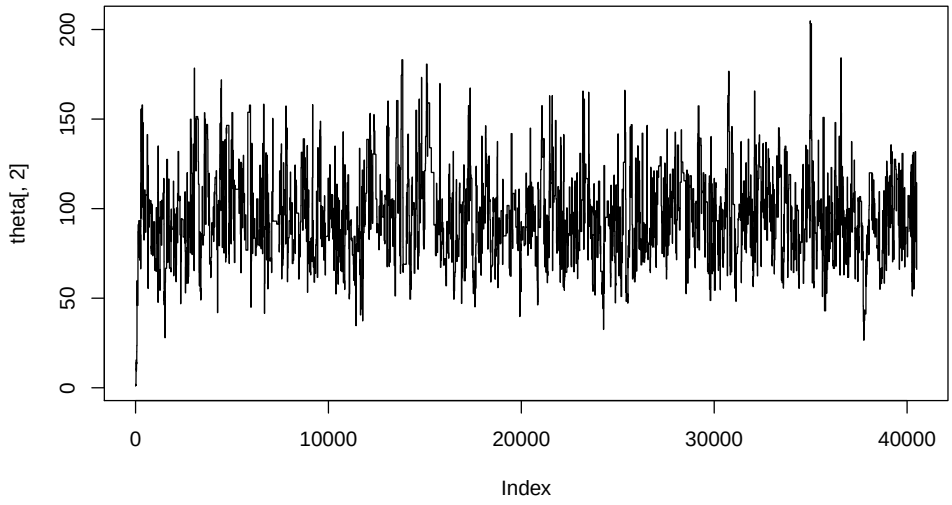
Pozenemo algoritem:

```r
theta <- mh2(n.iter=40500, theta.init=c(1.5, 1), x=x)
```

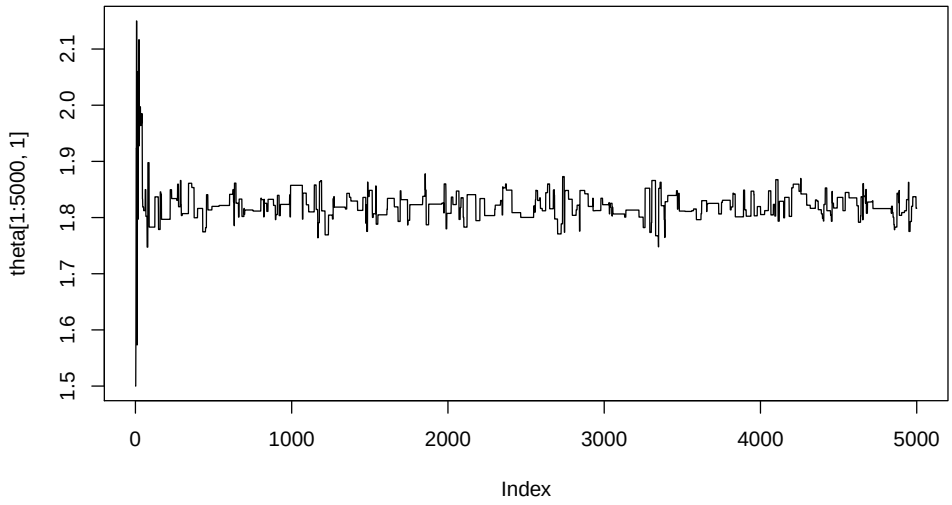
Veriga za μ:



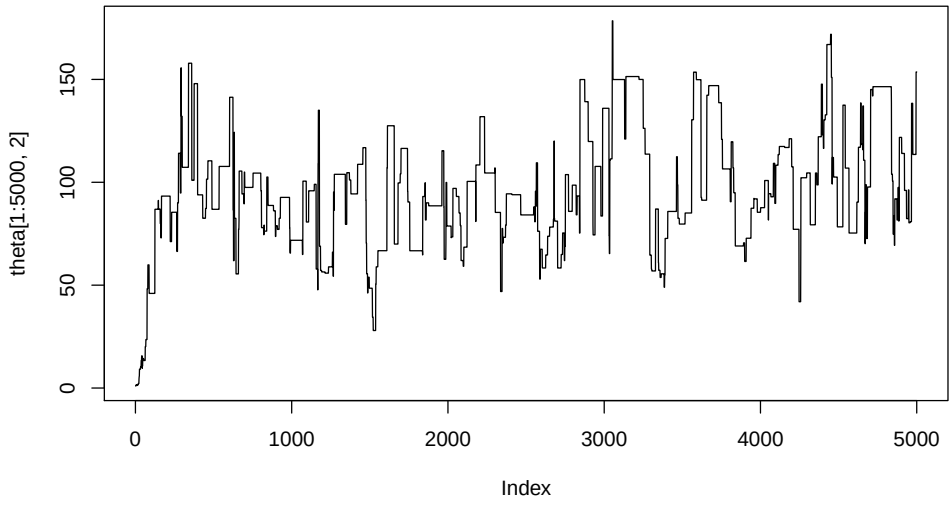
Veriga za τ:



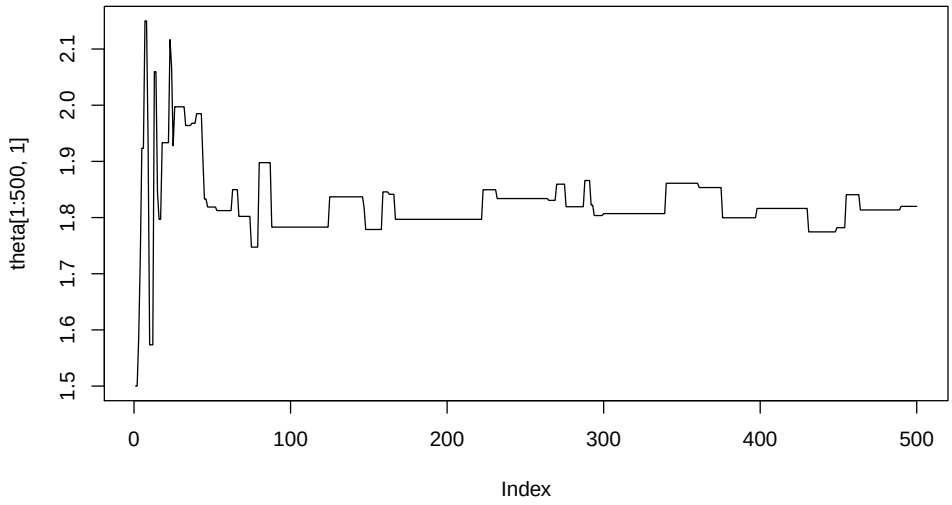
Veriga za μ – prvih 5000 iteracij:



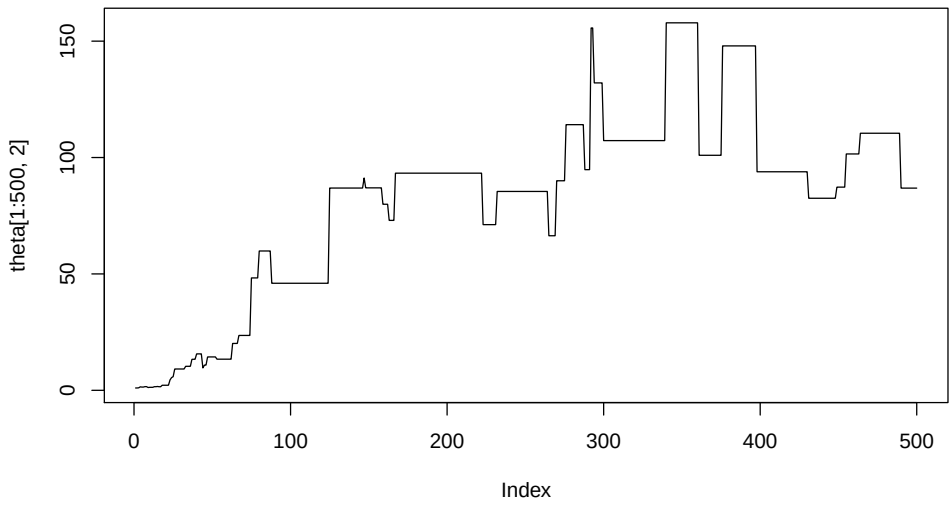
Veriga za τ – prvih 5000 iteracij:



Veriga za μ – prvih 500 iteracij:



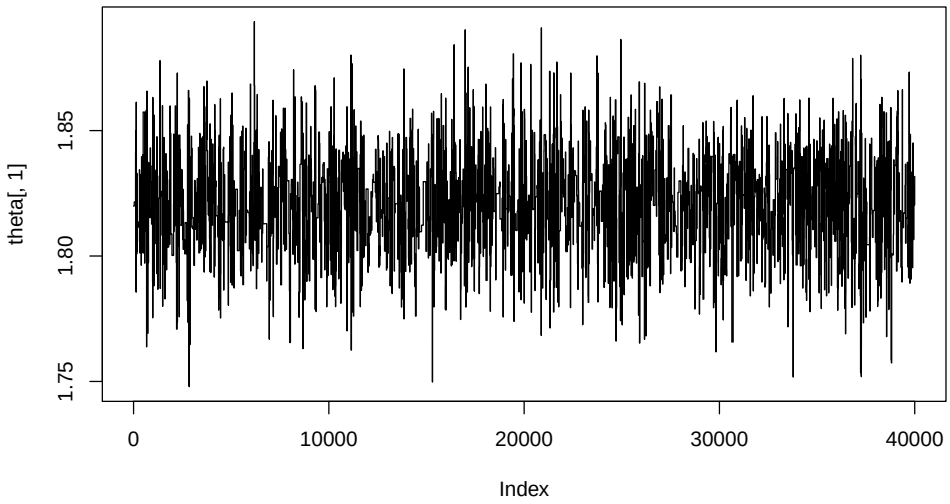
Veriga za τ – prvih 500 iteracij:



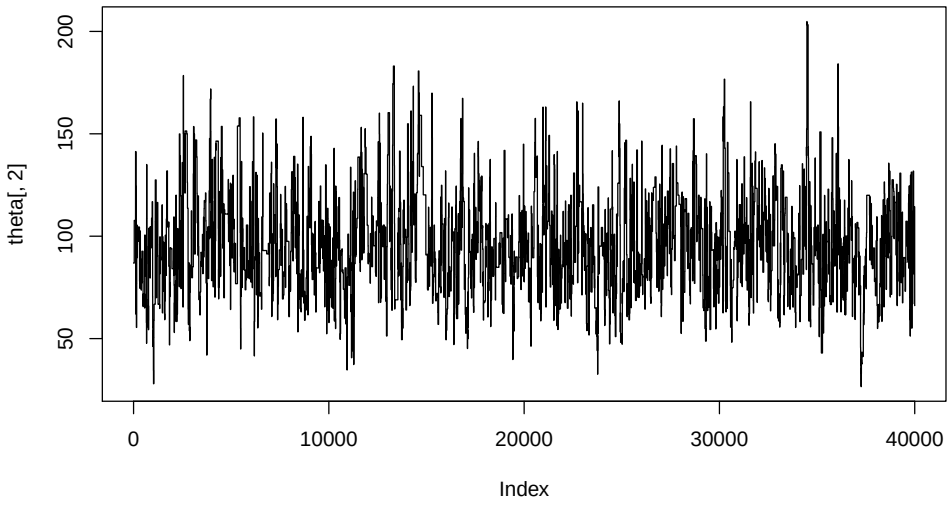
Odstranimo zacetnih nekaj clenov (*burn-in*):

```
theta <- theta[-c(1:500),]
```

Koncna veriga za μ:



Koncna veriga za τ:

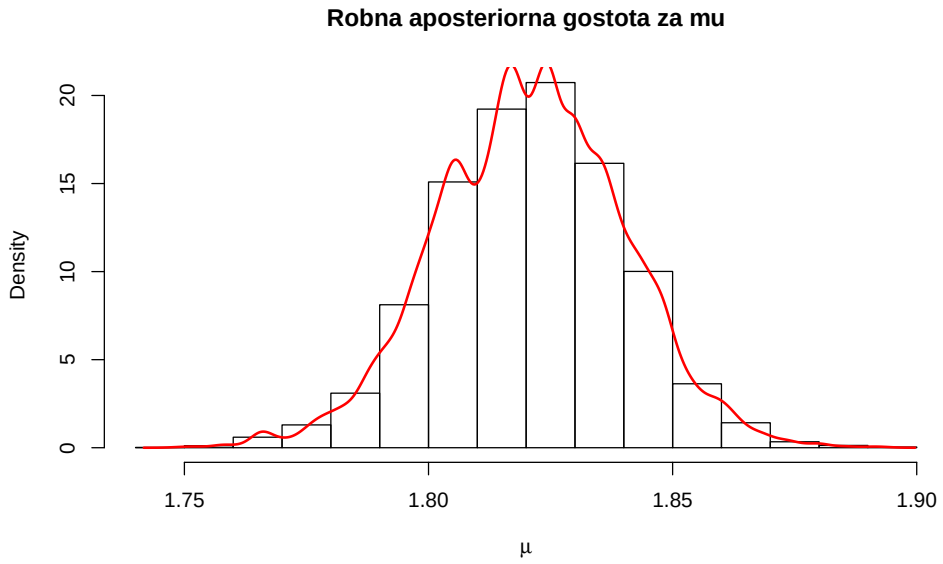


Dobljene robne aposteriorne porazdelitve:

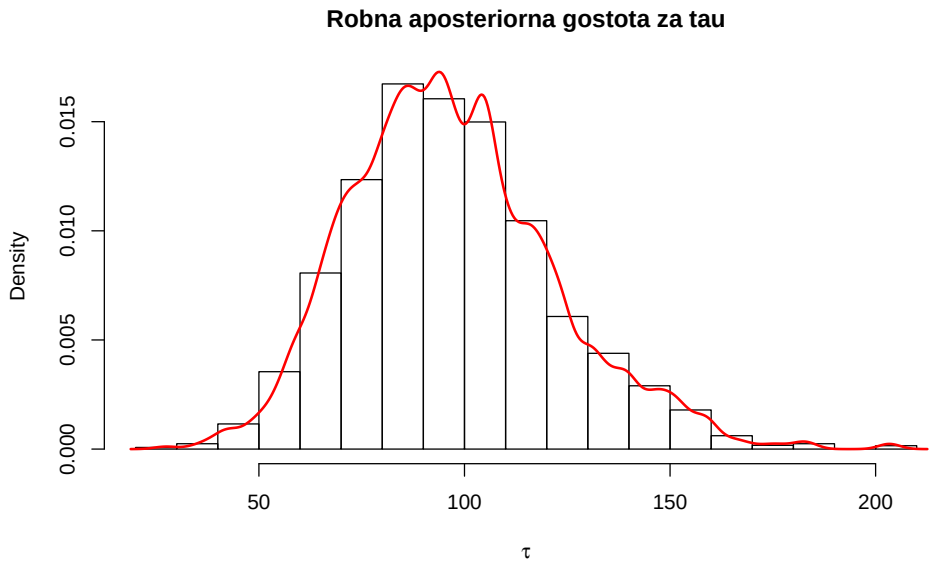
```
apply(theta, 2, summary)
```

```
##              [,1]      [,2]
## Min.    1.747887  26.54191
## 1st Qu. 1.807428  79.71434
## Median  1.821196  94.78347
## Mean    1.820875  96.92168
## 3rd Qu. 1.834077 111.66688
## Max.    1.893445 204.84960
```

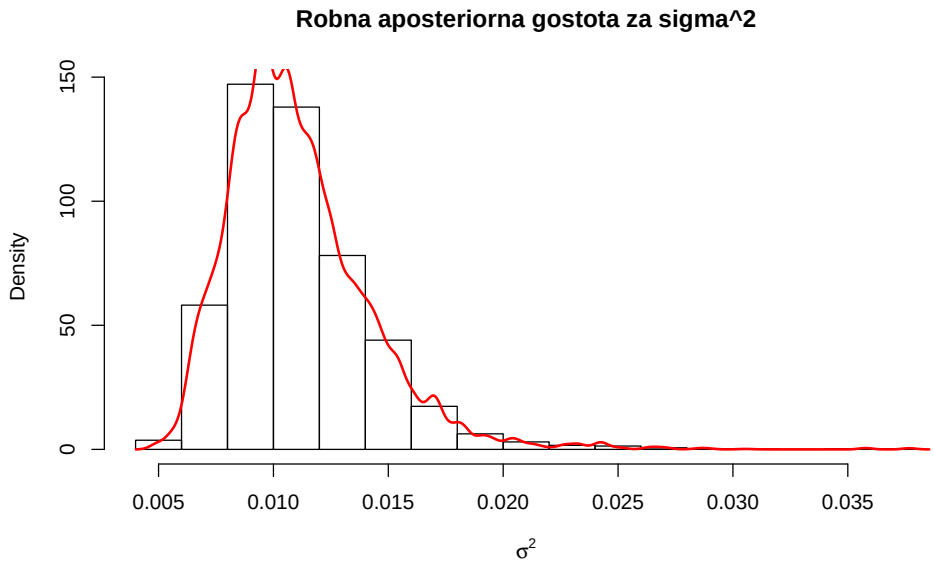
```
hist(theta[,1], prob=T, main = "Robna aposteriorna gostota za mu",
     xlab = expression(mu))
lines(density(theta[,1]), col="red", lwd=2)
```




```
hist(theta[,2], prob=T, main = "Robna aposteriorna gostota za tau",
     xlab = expression(tau))
lines(density(theta[,2]), col="red", lwd=2)
```




```
hist(1/theta[,2], prob=T, main = "Robna aposteriorna gostota za sigma^2",
     xlab = expression(sigma^2))
lines(density(1/theta[,2]), col="red", lwd=2)
```



Vzorec iz robne aposteriorne porazdelitve za σ^2 torej dobimo iz vzorcev za τ na preprost nacin (zadnji graf).