1. DN - Bayesova Statistika: Algoritem Metropolis-Hastings

Jan Črne

1 Primer: normalni model z znano varianco

Uporabili bomo algoritem Metropolis-Hastings za primer iz 3. sklopa, kjer so bili naši podatki število ur, ki so jih dijaki potrebovali za pripravo domače naloge.:

```
library(glue)
```

```
## Warning: package 'glue' was built under R version 3.6.3
```

```
x <- c(2.11, 9.75, 13.88, 11.3, 8.93, 15.66, 16.38, 4.54, 8.86, 11.94,
  12.47, 11.11, 11.65, 14.53, 9.61, 7.38, 3.34, 9.06, 9.45, 5.98,
  7.44, 8.5, 1.55, 11.45, 9.73)
```

Privzemimo normalni model z znano varianco $\sigma^2 = 4$, torej $(X_i|\theta) \sim N(\theta, \sigma^2 = 4)$, medtem ko naj bo apriorna porazdelitev $\theta \sim N(\theta_0 = 6, \tau_0^2 = 9)$. Zanimala nas je aposteriorna porazdelitev $(\theta|X)$, ki vemo, da je porazdeljena normalno $N(\mu_1, \tau_1^2)$, kjer sta parametra enaka:

$$\mu_1 = \frac{\tau_0^2}{\frac{\sigma^2}{n} + \tau_0^2}\bar{X} + \frac{\frac{\sigma^2}{n}}{\frac{\sigma^2}{n} + \tau_0^2}\mu_0$$

$$\tau_1^2 = \frac{\frac{\sigma^2}{n} \cdot \tau_0^2}{\frac{\sigma^2}{n} + \tau_0^2}$$

Pravo aposteriorno porazdelitev torej poznamo.

V nalogi jo bomo aproksimirali s pomočjo algoritma Metropolis-Hastings.

2 Naloge za vaje, ki so hkrati domača naloga

Za primer iz 3. sklopa aproksimirajte aposteriorno porazdelitev s pomočjo algoritma Metropolis-Hastings, kjer sledite spodnjim korakom.

1. Sami v R-u sprogramirajte algoritem Metropolis-Hastings za naš primer. Izberite smiselno *predlagalno jedro* $q(\cdot|\theta^{(n-1)})$ (npr: $q(\cdot|\theta^{(n-1)}) \sim N(\theta^{(n-1)}, \sigma_q^2 = 0.1^2)$; lahko tudi izberete drugo porazdelitev). Ključno je, da algoritem sprogramirate sami, pri čemer splošnost kode in efektivnost implementacije nista pomembni. Opomba: $f(X|\theta)$ je verjetje za $X = (X_1, ..., X_n)$, kjer so X_i normalno porazdeljeni: $f(X|\theta) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \cdot \exp\Big(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(X_i - \theta)^2\Big)$.

```
n <- length(x)
pv <- mean(x)


sigma.vz <- 2 # standardni odklon vzorčne porazdelitve

omega0 <- 6 # upanje apriorne normalne porazdelitve
tau0 <- 3 # standardni odklon apriorne normalne porazdelitve

sigma.q <- 0.1 # standardni odklon normalnega predlagalnega jedra

mi1 <- tau0^2 / (sigma.vz^2 / n + tau0^2) * pv +
   (sigma.vz^2 / n) / ( sigma.vz^2 / n + tau0^2) * omega0 # teoretično
                                                        # upanje apost. porazdelitve

tau1 <- sqrt((sigma.vz^2 / n * tau0^2) / (sigma.vz^2 / n + tau0^2))
# stand. odklon teoretične apost. porazdelitve
```

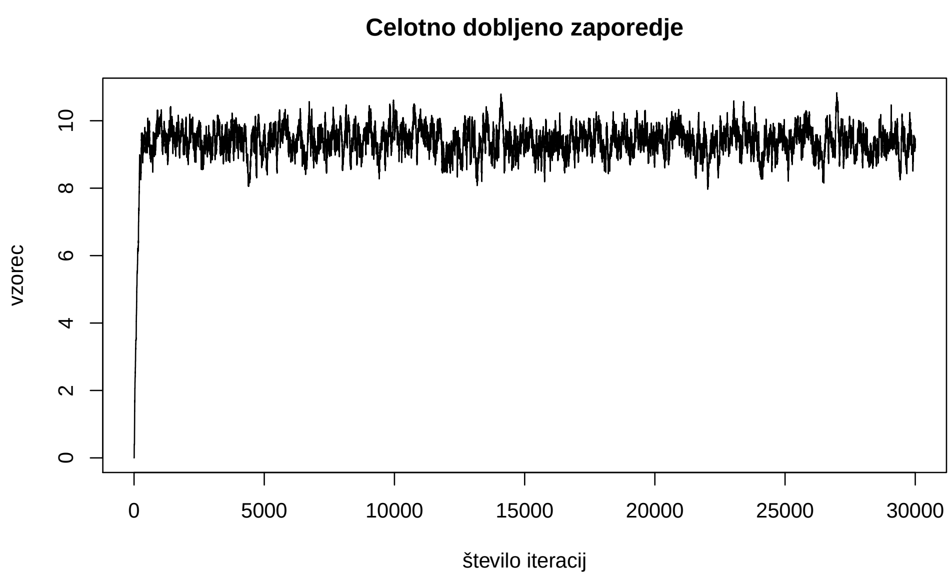
```
MetropolisHastings <- function(zac, st_iteracij, x, sigma.q, B = 1) {
   vzorec <- c(zac, rep(0, st_iteracij))
   u_ji <- runif(st_iteracij + 1, 0, 1)

   for (i in 2:st_iteracij + 1) {
      q.prej <- vzorec[i - 1] # prejsnja realizacija oz. pric. vr. nove
      y <- rnorm(1, q.prej, sigma.q)

      imenovalec <- log(
         1/(2 * pi * sigma.vz^2)^(length(x) / 2) *
         exp(-1/(2 * sigma.vz^2) * sum((x - q.prej)^2))
         ) +
         log(dnorm(q.prej, omega0, tau0)) +
```

```r
        log(dnorm(y, q.prej, sigma.q))


      stevec <-  log(
        1/(2 * pi * sigma.vz^2)^(length(x) / 2) *
        exp(-1/(2 * sigma.vz^2) * sum((x - y)^2))) +
        log(dnorm(y, omega0, tau0)) +
        log(dnorm(q.prej, y, sigma.q))


      verj_rho <- 0

      if (imenovalec != -Inf){
        verj_rho <- min(log(1), stevec - imenovalec)
      }
      else {
        verj_rho <- log(1)
      }
      # print(verj_rho)

      if (log(u_ji[i]) <= verj_rho) {
        vzorec[i] <- y
      }
      else {
        vzorec[i] <- q.prej
      }

  }

  vzorec[-c(1:B)]

  # vzorec

}
```

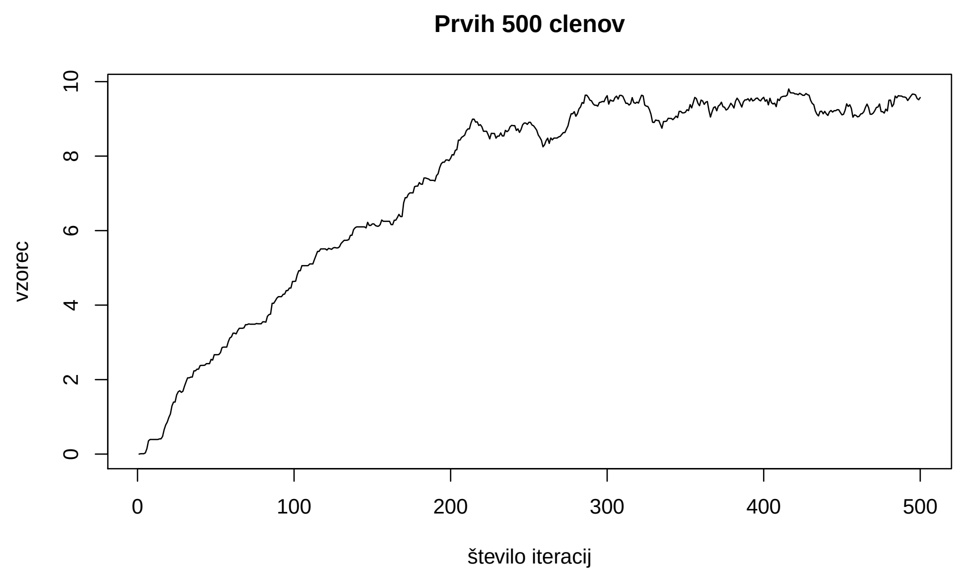
2. Preizkusite algoritem na našem primeru, kjer si sami izberite neko smiselno začetno vrednost $\theta^{(0)}$ in varianco *predlagalne gostote* (v zgornjem primeru smo jo označili σ_q^2). Opomba: zaradi numerične stabilnosti ob vsaki iteraciji izračunajte logaritem verjetnosti $\rho(\theta^{(n-1)}, y)$ in na podlagi tega logaritma se odločite, kakšen bo $\theta^{(n)}$. Rezultate predstavite na naslednji način:
   - Narišite celotno dobljeno zaporedje $\{\theta^{(0)}, \theta^{(1)}, ...\theta^{(S)}\}$ (naj bo S vsaj 30000 - lahko tudi vzamete več iteracij). Lahko uporabite funkcijo `plot(..., type='l')`.
   - Narišite le prvih 500 ali pa 5000 členov.
   - Narišite celotno zaporedje, kjer uporabite ustrezen *burn-in* parameter B.

- Za tako izbrano zaporedje grafično predstavite aposteriorno porazdelitev in jo grafično primerjajte s pravo (teoretično) aposteriorno porazdelitvijo.
- Ocenite parameter in 95% interval zaupanja za parameter iz izbranega zaporedja ter primerjajte z ocenami iz prave aposterirone porazdelitve.

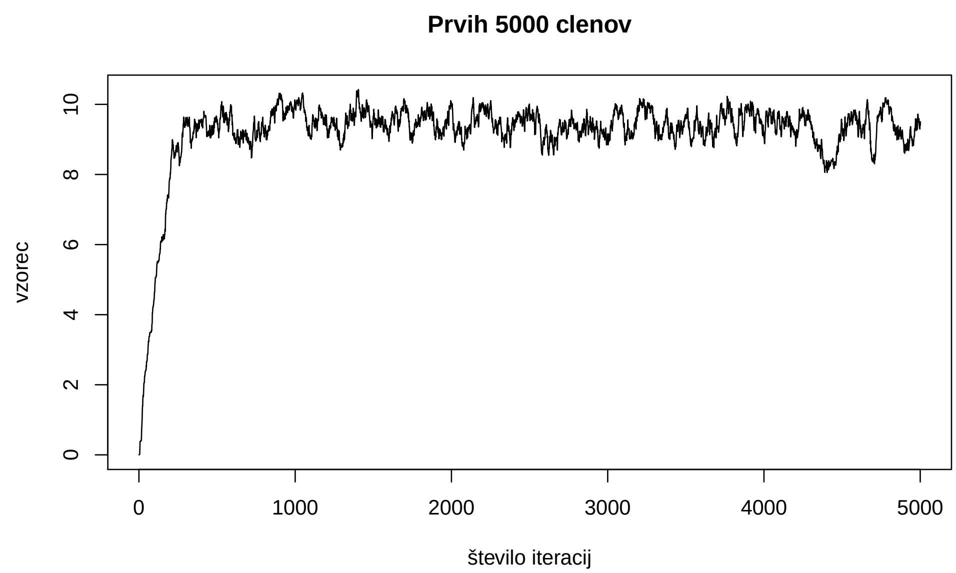
```
zac_vr <- pv

poskus <- MetropolisHastings(zac_vr, 30000, x, sigma.q)
poskus <- poskus[-1]

plot(1:length(poskus), poskus, type = 'l', ylab = 'vzorec',
     xlab = 'število iteracij', main = 'Celotno dobljeno zaporedje')
```



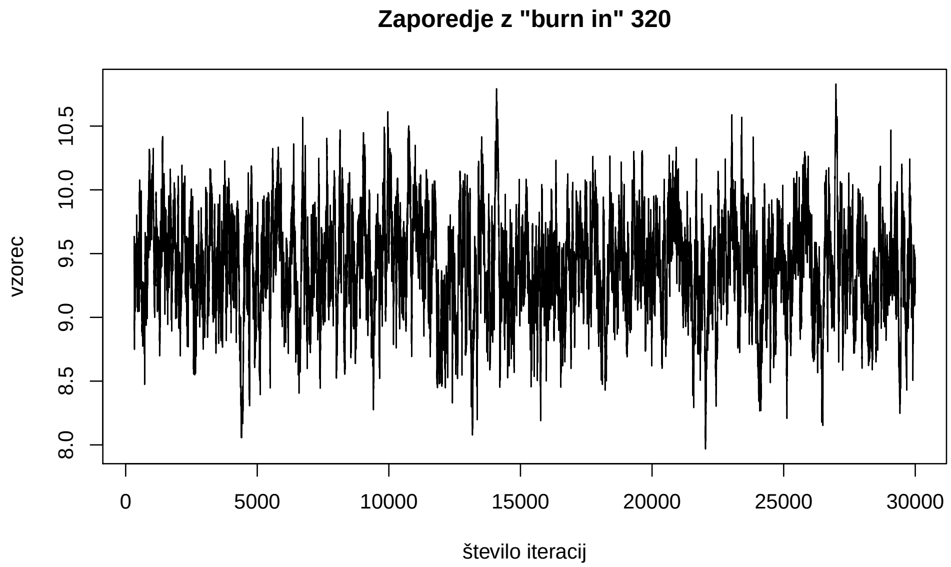

```
plot(1:500, poskus[1:500], type = 'l', ylab = 'vzorec',
     xlab = 'število iteracij', main = 'Prvih 500 členov')
```



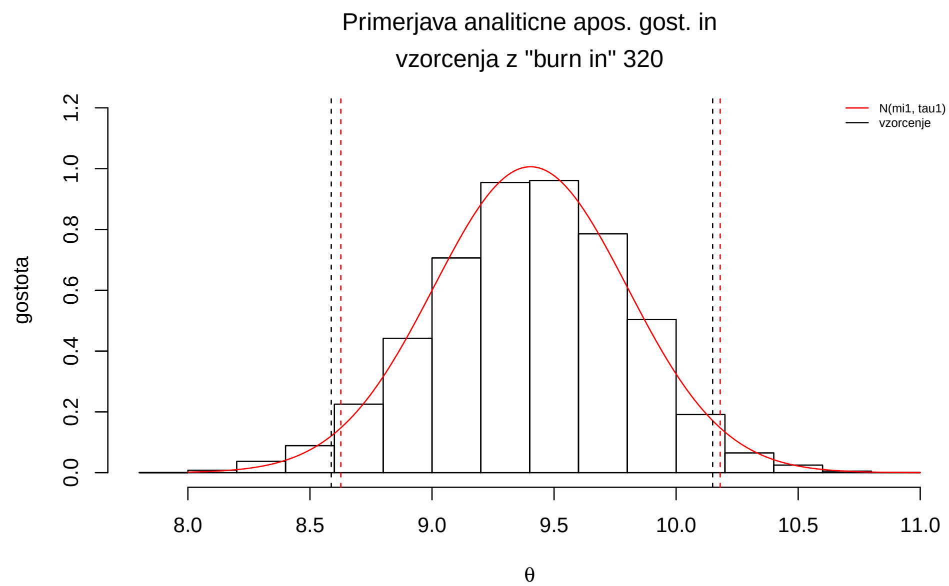

```r
plot(1:5000, poskus[1:5000], type = 'l', ylab = 'vzorec',
     xlab = 'število iteracij', main = 'Prvih 5000 členov')
```




```r
plot(320:length(poskus), poskus[320:length(poskus)], type = 'l',
     ylab = 'vzorec', xlab = 'število iteracij',
     main = 'Zaporedje z "burn in" 320')
```




```
uporabne <- poskus[320:length(poskus)]
hist(uporabne, probability = TRUE, ylab = 'gostota',
     xlab = expression(theta),
     main = expression(
        atop('Primerjava analitične apos. gost. in',
             'vzorčenja z "burn in" 320')),
     ylim = c(0, 1.2)
)
lines(seq(8, 11, 0.001), dnorm(seq(8, 11, 0.001), mi1, tau1),
        col = 'red')
legend("topright", legend = c("N(mi1, tau1)","vzorčenje"),
       col = c("red","black"), lty = 1, bty = "n", cex = 0.59)

kvantila.vz <- quantile(uporabne, c(0.025, 0.975))
kvantila.teor <- qnorm(c(0.025, 0.975), mi1, tau1)

abline(v = kvantila.vz[1], col = "black", lty = 2)
abline(v = kvantila.vz[2], col = "black", lty = 2)
abline(v = kvantila.teor[1], col = "red", lty = 2)
abline(v = kvantila.teor[2], col = "red", lty = 2)
```




```
kvantila.vz # kvantila našega vzorčenja
```

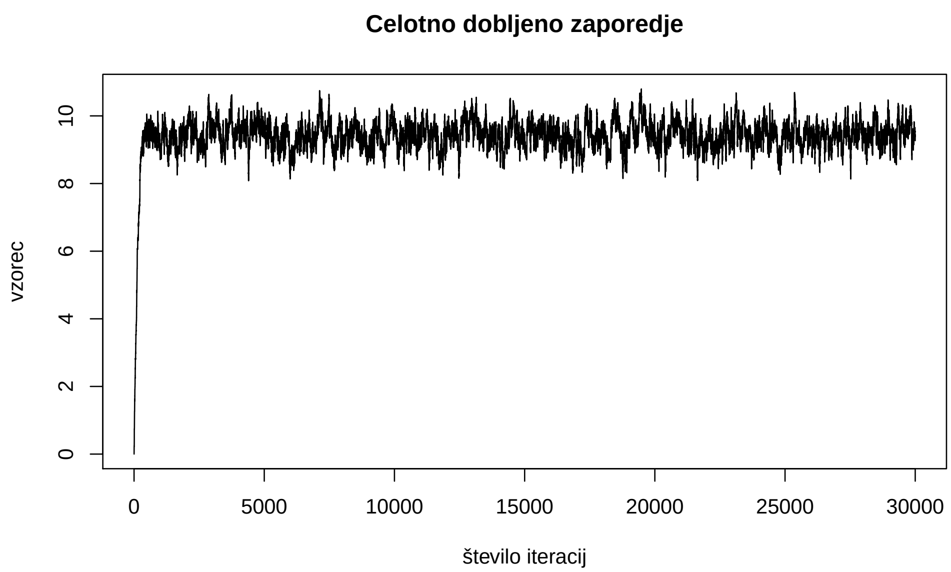
```
##      2.5%     97.5%
##  8.587233 10.149672
```

```
kvantila.teor # kvantila iz teoretične/analitične gostote
```

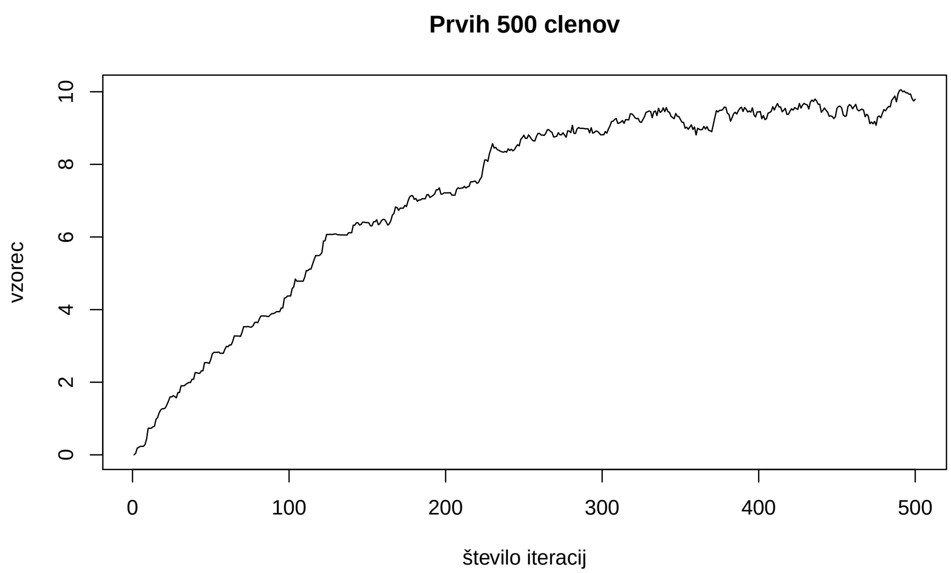
```
## [1]  8.626385 10.180602
```

3. Poženite vas algoritem pri neki nesmiselni zacetni vrednosti. Rezultate predstavite:
   - Narišite celotno dobljeno zaporedje $\{\theta^{(0)}, \theta^{(1)}, ...\theta^{(S)}\}$.
   - Narišite le prvih 500 ali pa 5000 členov.
   - Določite vrednost B, ki bi bila smiselna za vaš primer. Narišite celotno zaporedje, kjer uporabite ustrezen B.

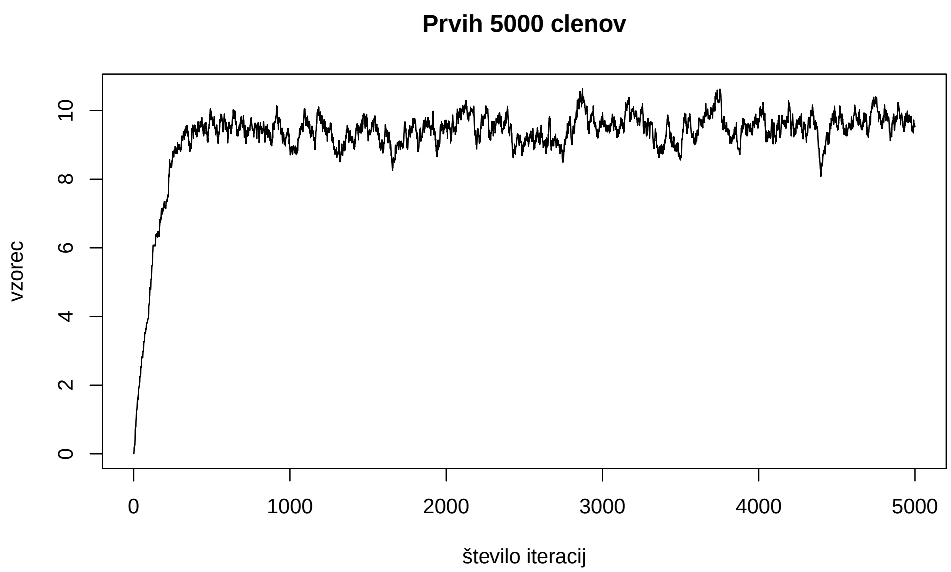
```
zac_vr2 <- 712500 # pričakovana življenjska doba (v urah)

poskus2 <- MetropolisHastings(zac_vr2, 30000, x, sigma.q)

plot(1:length(poskus2), poskus2, type = 'l', ylab = 'vzorec',
     xlab = 'število iteracij', main = 'Celotno dobljeno zaporedje')
```



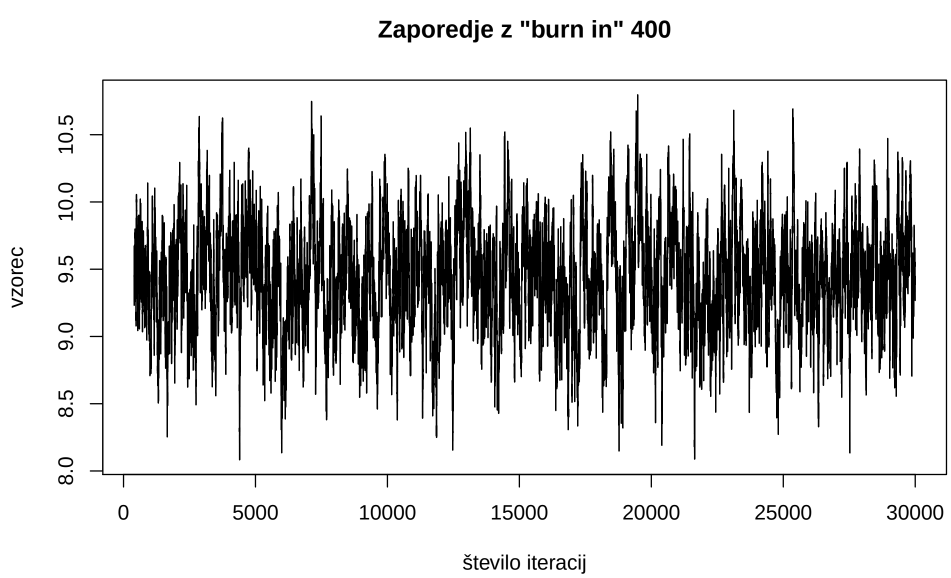

```
plot(1:500, poskus2[1:500], type = 'l', ylab = 'vzorec',
     xlab = 'število iteracij', main = 'Prvih 500 členov')
```



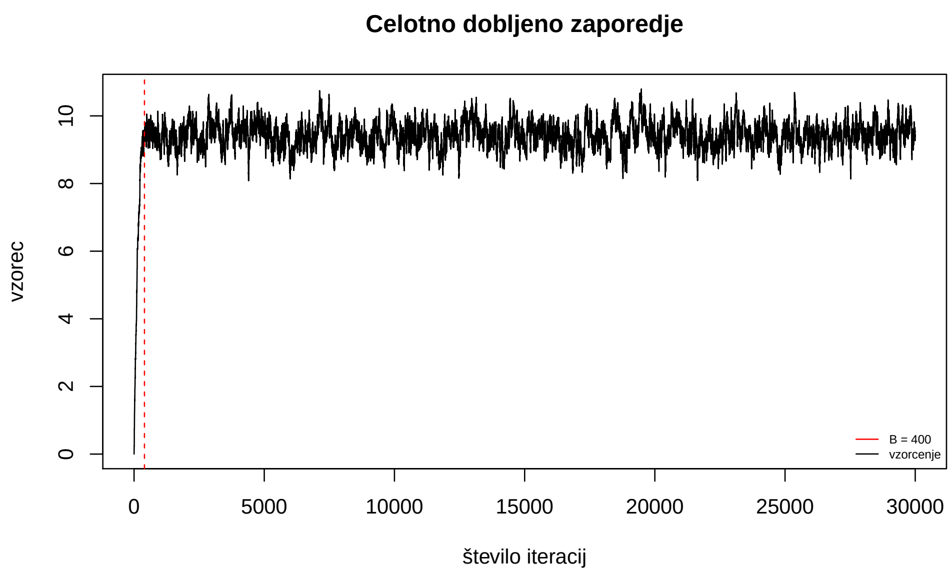

```
plot(1:5000, poskus2[1:5000], type = 'l', ylab = 'vzorec',
     xlab = 'število iteracij', main = 'Prvih 5000 členov')
```




```r
plot(400:length(poskus2), poskus2[400:length(poskus2)], type = 'l',
     ylab = 'vzorec', xlab = 'število iteracij',
     main = 'Zaporedje z "burn in" 400')
```




```r
plot(1:length(poskus2), poskus2, type = 'l', ylab = 'vzorec',
     xlab = 'število iteracij', main = 'Celotno dobljeno zaporedje')
abline(v = 400, col = "red", lty = 2)
legend("bottomright", legend = c("B = 400","vzorčenje"),
        col = c("red","black"), lty = 1, bty = "n", cex = 0.59)
```

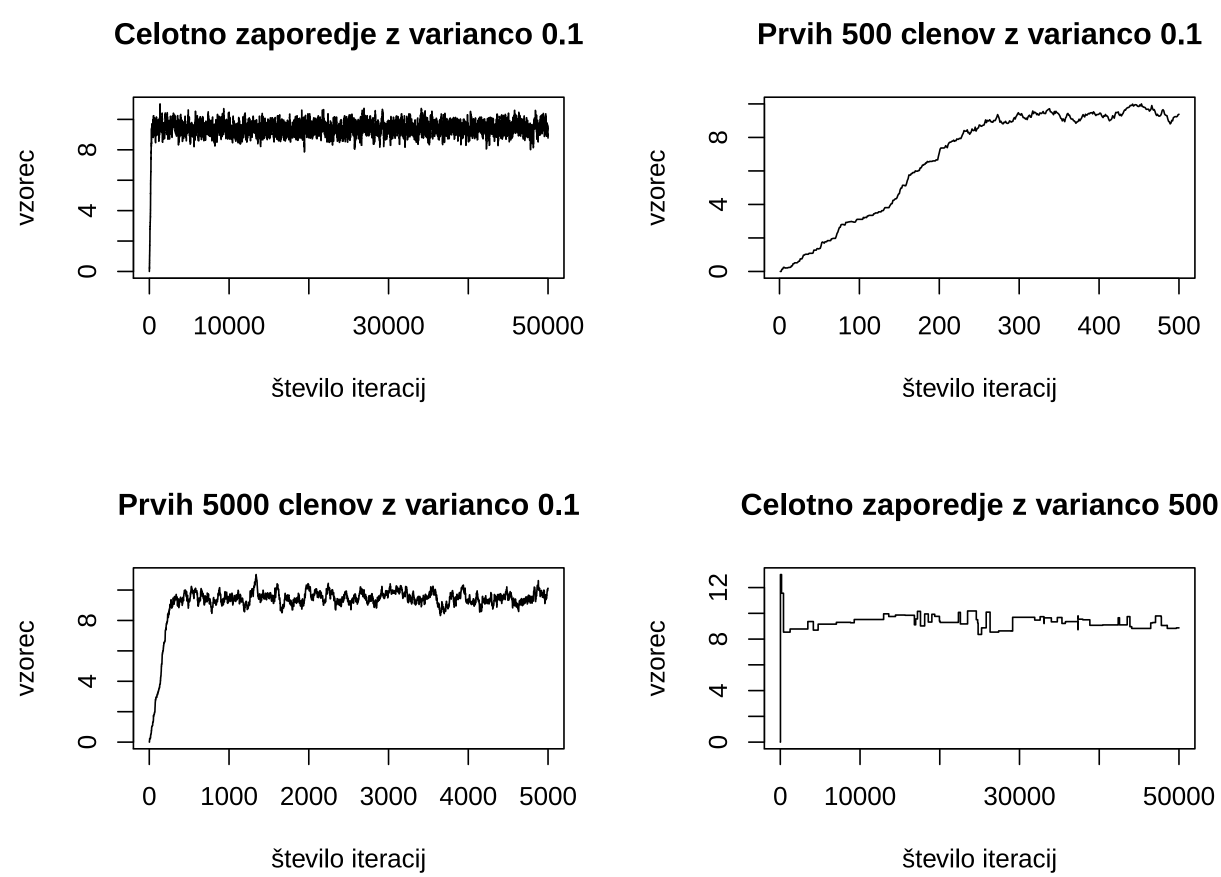


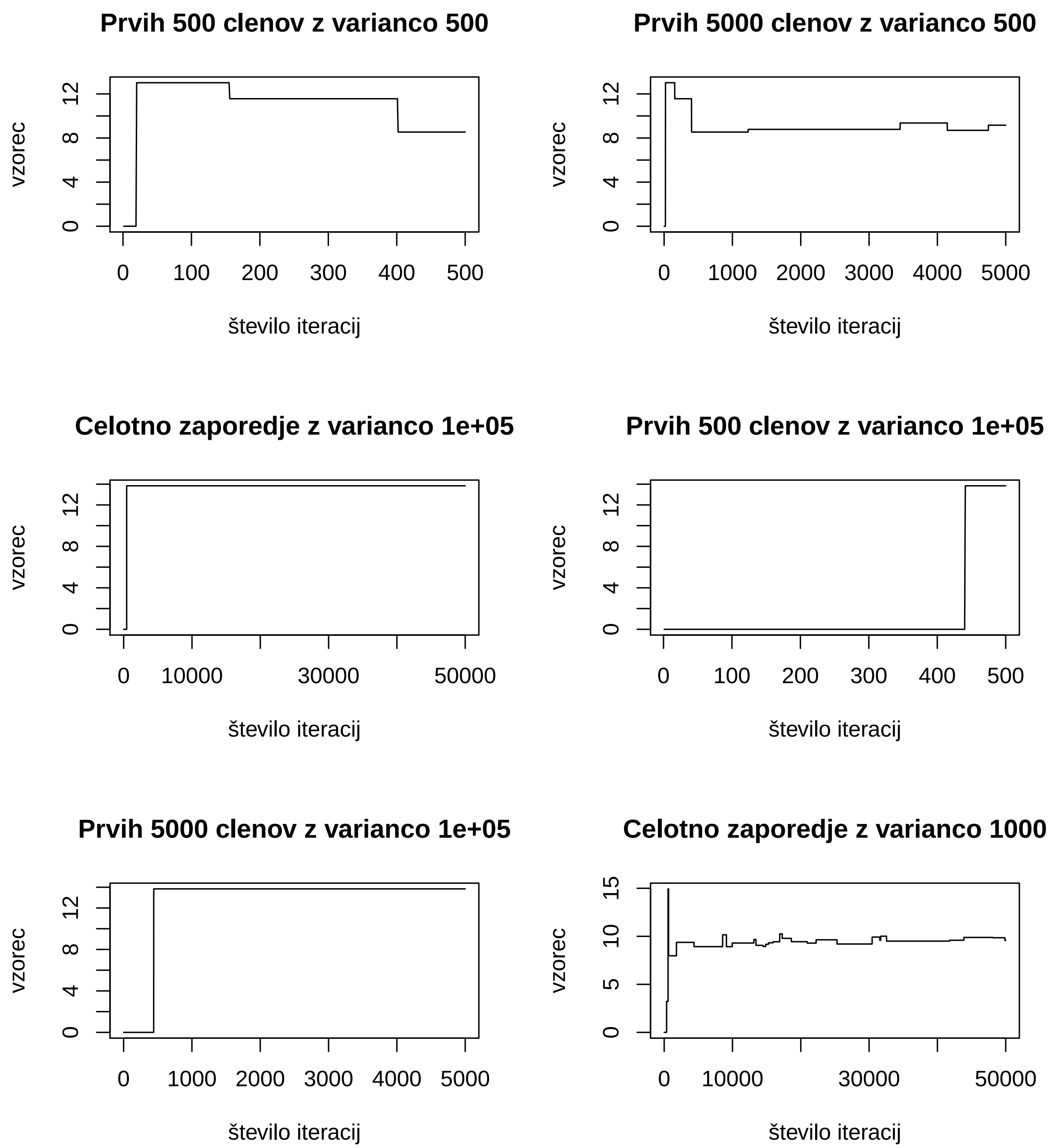
4. Pri neki smiselni začetni vrednosti poženite algoritem pri nekaj razlicnih variancah za *predlagalno jedro*. Pri izboru pretiravajte v obe smeri (spomnite se, kakšni so po velikosti naši podatki), tako da boste grafično opazili razlike na prvih npr. 500 iteracijah. Rezultate predstavite:

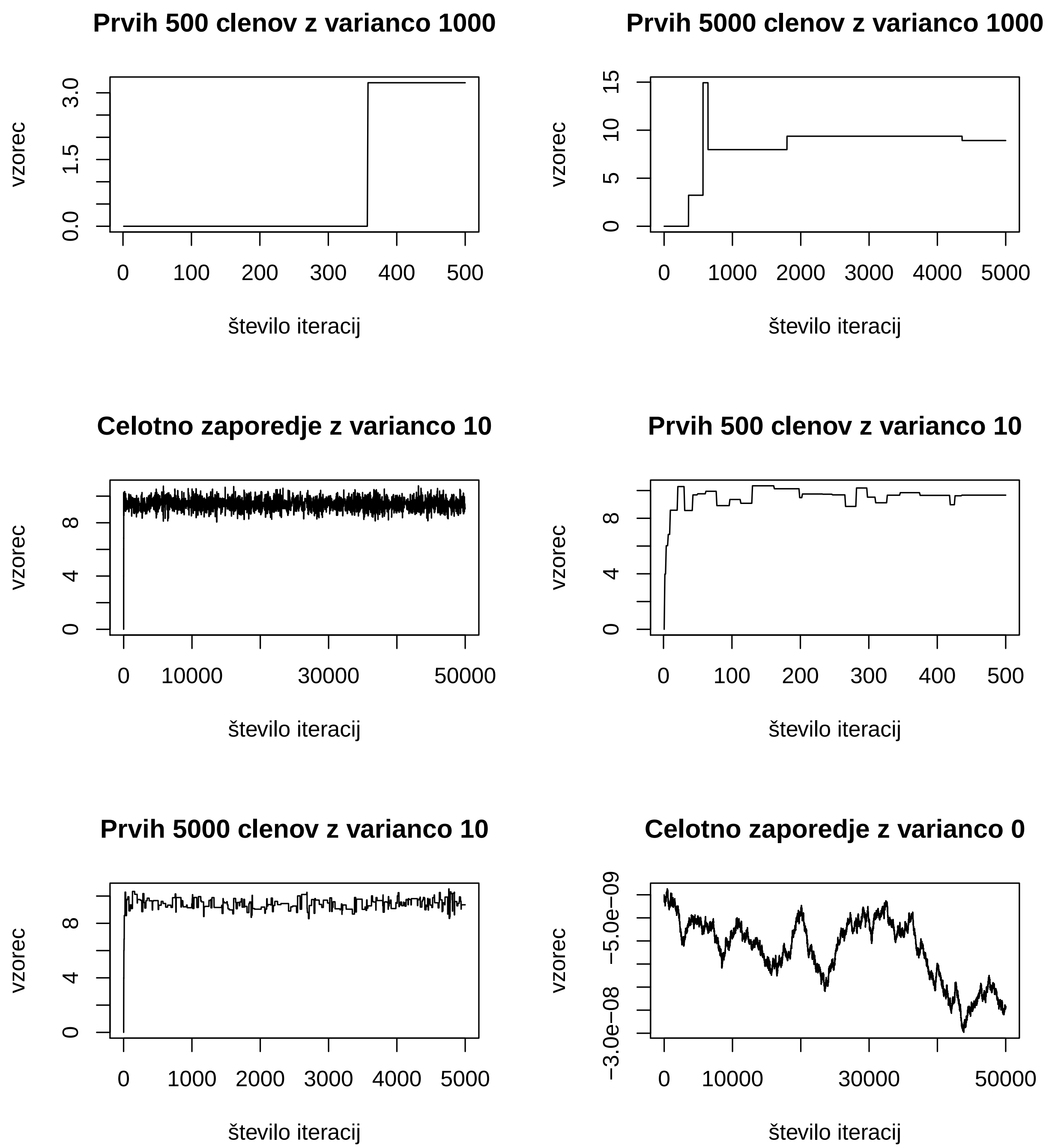
   - Za vsak primer narisite prvih nekaj (nekje med 500 in 5000) členov in še celotno zaporedje.
   - Komentirajte razlike in zakaj do njih pride. Kaj in zakaj vas moti pri izbranih primerih?
   - Kakšen bi bil v splošnem (ne vezano na naš vzorec) vaš predlog glede izbora variance *predlagalnega jedra* oz. kakšen bi bil predlog za izbor končnega zaporedja?

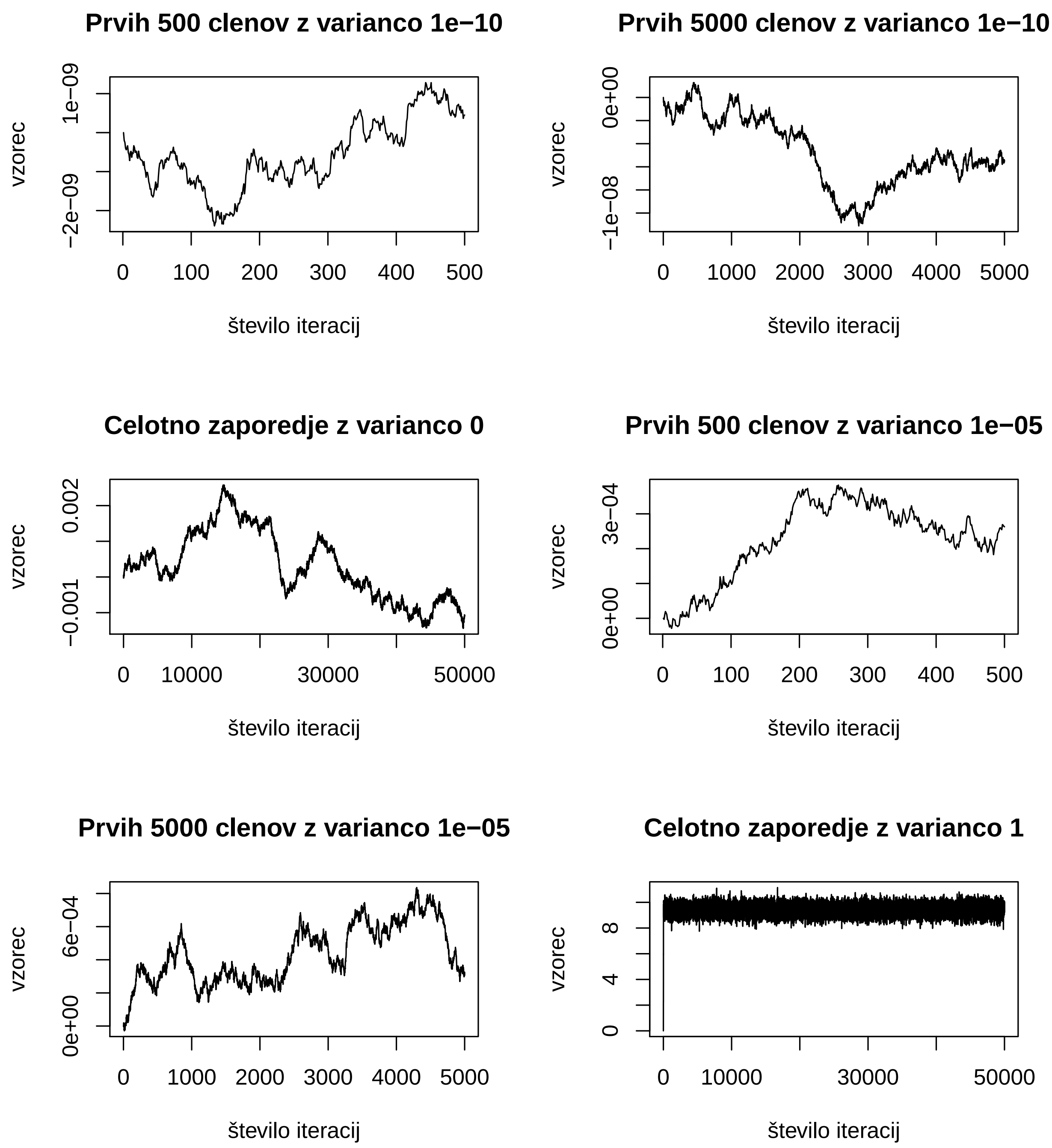
```
variance <- c(sigma.q, 500, 10^5, 1000, 10, 10^(-10), 10^(-5), 1, pv, var(x))

upanja <- rep(0, length(variance)) # upanja zadnjih 25000 vzorčenj
st.odkloni <- rep(0, length(variance)) # standardni odkloni zadnjih 25000 vzorčenj

for (i in 1:length(variance)) {
   poskusek <- MetropolisHastings(zac_vr, 50000, x, variance[i])

   upanja[i] <- mean(poskusek[25000:length(poskusek)])
   st.odkloni[i] <- sd(poskusek[25000:length(poskusek)])

   plot(1:length(poskusek), poskusek, type = 'l', ylab = 'vzorec',
     xlab = 'število iteracij',
     main = glue("Celotno zaporedje z varianco {round(variance[i], 4)}"))
```

```
  plot(1:500, poskusek[1:500], type = 'l', ylab = 'vzorec',
    xlab = 'število iteracij',
    main = glue('Prvih 500 členov z varianco {variance[i]}'))

  plot(1:5000, poskusek[1:5000], type = 'l', ylab = 'vzorec',
    xlab = 'število iteracij',
    main = glue('Prvih 5000 členov z varianco {variance[i]}'))
}
```

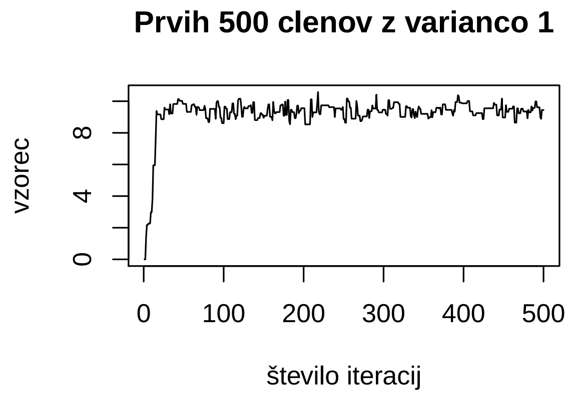


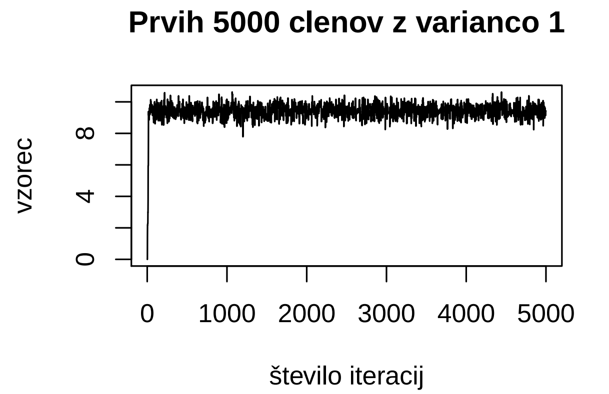

Prvih 500 clenov z varianco 500
vzorec
število iteracij
Prvih 5000 clenov z varianco 500
vzorec
število iteracij
Celotno zaporedje z varianco 1e+05
vzorec
število iteracij
Prvih 500 clenov z varianco 1e+05
vzorec
število iteracij
Prvih 5000 clenov z varianco 1e+05
vzorec
število iteracij
Celotno zaporedje z varianco 1000
vzorec
število iteracij

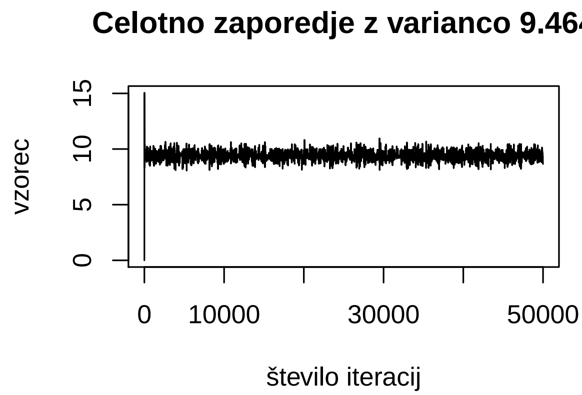

Prvih 500 clenov z varianco 1000
vzorec
0.0
1.5
3.0
0
100
200
300
400
500
število iteracij
Prvih 5000 clenov z varianco 1000
vzorec
0
5
10
15
0
1000
2000
3000
4000
5000
število iteracij
Celotno zaporedje z varianco 10
vzorec
0
4
8
0
10000
30000
50000
število iteracij
Prvih 500 clenov z varianco 10
vzorec
0
4
8
0
100
200
300
400
500
število iteracij
Prvih 5000 clenov z varianco 10
vzorec
0
4
8
0
1000
2000
3000
4000
5000
število iteracij
Celotno zaporedje z varianco 0
vzorec
−3.0e−08
−5.0e−09
0
10000
30000
50000
število iteracij

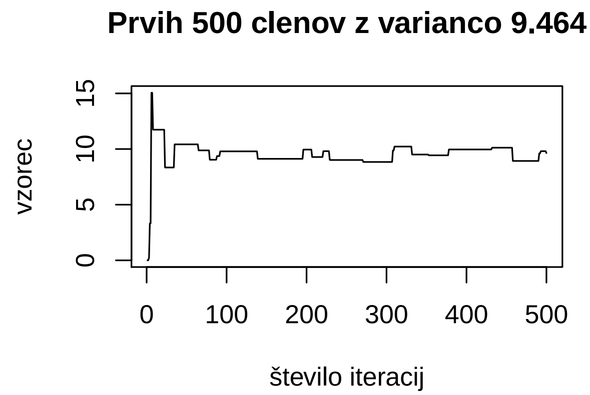


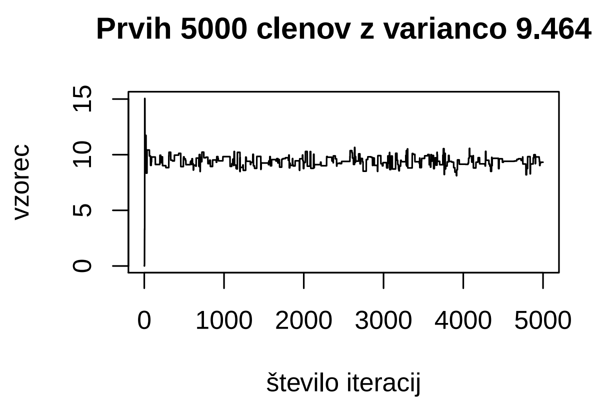
Prvih 500 clenov z varianco 1e−10
vzorec
−2e−09
1e−09
0
100
200
300
400
500
število iteracij
Prvih 5000 clenov z varianco 1e−10
vzorec
−1e−08
0e+00
0
1000
2000
3000
4000
5000
število iteracij
Celotno zaporedje z varianco 0
vzorec
−0.001
0.002
0
10000
30000
50000
število iteracij
Prvih 500 clenov z varianco 1e−05
vzorec
0e+00
3e−04
0
100
200
300
400
500
število iteracij
Prvih 5000 clenov z varianco 1e−05
vzorec
0e+00
6e−04
0
1000
2000
3000
4000
5000
število iteracij
Celotno zaporedje z varianco 1
vzorec
0
4
8
0
10000
30000
50000
število iteracij

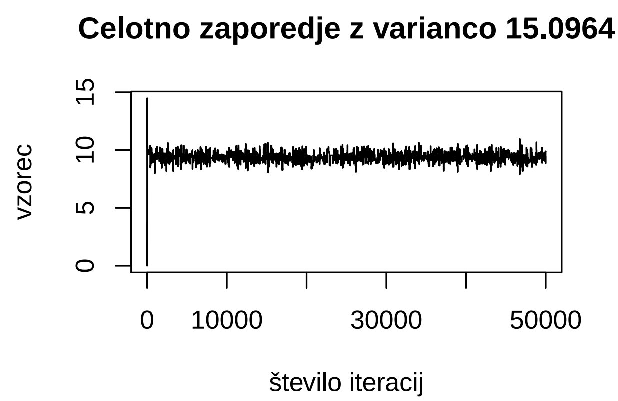

Prvih 500 clenov z varianco 1
vzorec
0
4
8
0
100
200
300
400
500
število iteracij

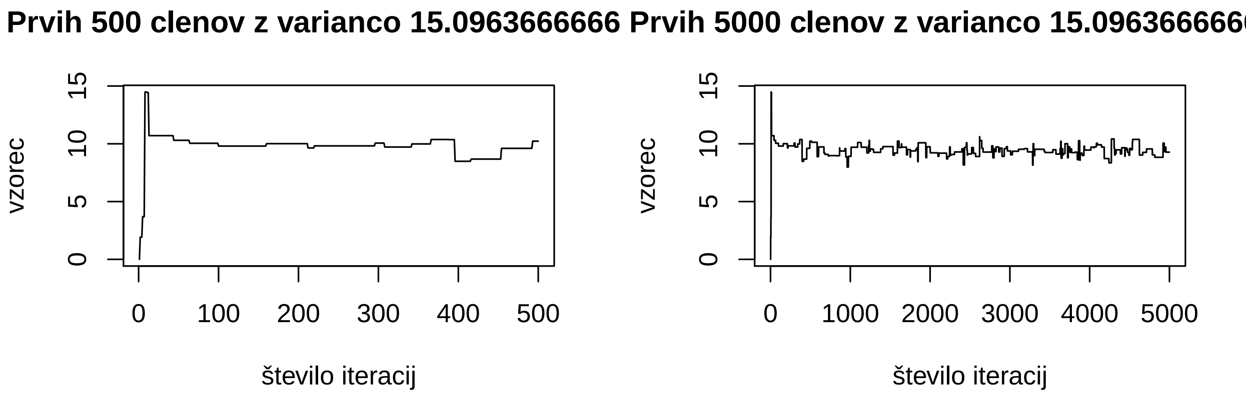

Prvih 5000 clenov z varianco 1
vzorec
0
4
8
0
1000
2000
3000
4000
5000
število iteracij


Celotno zaporedje z varianco 9.464
vzorec
0
5
10
15
0
10000
30000
50000
število iteracij


Prvih 500 clenov z varianco 9.464
vzorec
0
5
10
15
0
100
200
300
400
500
število iteracij


Prvih 5000 clenov z varianco 9.464
vzorec
0
5
10
15
0
1000
2000
3000
4000
5000
število iteracij


Celotno zaporedje z varianco 15.0964
vzorec
0
5
10
15
0
10000
30000
50000
število iteracij




```r
tabela <- data.frame("Varianca jedra" = variance,
                     "Upanje realiziranega vzorca" = round(upanja, 4),
                     "St. odklon realiziranega vzorca" = round(st.odkloni, 4))

knitr::kable(tabela, caption =
               "Primerjava vzorca v odvisnosti od variance predlagalnega jedra")
```

Table 1: Primerjava vzorca v odvisnosti od variance predlagalnega jedra

Varianca.jedra	Upanje.realiziranega.vzorca	St..odklon.realiziranega.vzorca
0.10000	9.4259	0.3993
500.00000	9.2134	0.4013
100000.00000	13.8427	0.0000
1000.00000	9.5743	0.2554
10.00000	9.4018	0.3959
0.00000	0.0000	0.0000
0.00001	-0.0003	0.0007
1.00000	9.4110	0.3977
9.46400	9.3818	0.4025
15.09637	9.3927	0.3964

```r
pv # povprečna vrednost osvežitvenega vzorca
```

```
## [1] 9.464
```

```r
sd(x) # standardni odklon osvežitvenega vzorca
```

```
## [1] 3.885404
```

KOMENTAR:

Ob veliki varianci predlagalnega jedra je, da dosežemo ustalitev vzorečenih vrednosti okoli naše pričakovane vrednosti potrebnih več “burn in” iteracij, prav tako le te večkrat zapored vrnejo enake vrednosti, kar glede na to, da vemo da vzorčimo iz zvezne normalne porazdelitve ni najbolj pravilno.Sicer se pri velikih vrednostih povprecje nasega vzorca dokaj dobro ujema z povprecjem nasega osvežitvenega vzorca.

Enako ne velja za izredno nizke variance. Pri njih so vzorčena povprečja okoli 0 medtem, ko bi pričakovali vrednosti okoli 9. Pri nizkih variancah se prav tako kot pri zredno viskokih zaporedje ne ustali, torej markovska veriga stacionarnost, ki bi odražala porazdelitev parametra doseže kasneje.

Po ogledu primerjave 25000 vzorčenih količin (za 25000, se se odločil po ogledu vseh grafov) me najbolj prepričajo rezultat v katerem za varianco predlagalnega jedra vzamemo kar varianco osvežitvenega vzorca x.