Statistical inference: MCMC methods

Frank van der Meulen (TU Delft)

```r
setwd("~/.julia/dev/WI4455/scr/mcmc")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

SIMPLE EXAMPLE OF METROPOLIS-HASTINGS ALGORITHM

Suppose we wish to simulate from the beta distribution. We can do this with the Metropolis–Hastings algorithm, where the target density is the $\beta(a, b)$-distribution. Of course this is just an example for illustration, as there exist direct ways for simulating independent realisations of the beta distribution.

MH with independent $Unif(0, 1)$-proposals.

```r
  a=2.7; b=6.3; # choose parameters of the beta-distribution

  Nsim = 5000
  X = c(runif(1), rep(0,Nsim-1))  # initialize the chain
  acc = rep(0,Nsim)
  for (i in 2:Nsim)
  {
    Y=runif(1)  # proposal
    A=dbeta(Y,a,b)/dbeta(X[i-1],a,b)
    acc[i-1] <- (runif(1)<A)
    X[i]=X[i-1] + (Y-X[i-1])*acc[i-1]
```

```r
  }
  df = data.frame(iterate=1:Nsim, vals=X)
  p1 = df %>% ggplot(aes(x=iterate, y=vals)) + geom_line()  + ylab("")
  p2 = ggplot()  + geom_histogram(data=df, mapping=aes(x=vals,y=..density..), colour='white', bins=50) +
      xlab("") + ylab("") +
      stat_function(data=data.frame(x=c(0,1)), mapping=aes(x), fun=function(x) dbeta(x,a,b),colour="ora


pdf('Beta-ind.pdf',width=7,height=4)
grid.arrange(p1,p2)
dev.off()
```

```
## pdf
##   2
```

```r
cat('average acceptance probability equals: ',mean(acc))
```

```
## average acceptance probability equals:  0.4644
```

MH with smmetric random-walk MH proposals.

If x is the current iterate, then propose $x + U(-\eta, \eta)$. Experiment yourself with different values for η.

```r
  eta=10
  X=rep(runif(1),Nsim)  # initialize the chain
  acc=rep(0,Nsim)
  for (i in 2:Nsim)
  {
    Y=X[i-1] + runif(1,-eta,eta)
    A=dbeta(Y,a,b)/dbeta(X[i-1],a,b)
    acc[i-1] <- (runif(1)<A)
    X[i]=X[i-1] + (Y-X[i-1])*acc[i-1]
  }


df = data.frame(iterate=1:Nsim, vals=X)
p1 = df %>% ggplot(aes(x=iterate, y=vals)) + geom_line()  + ylab("")
p2 = ggplot()  + geom_histogram(data=df, mapping=aes(x=vals,y=..density..), colour='white', bins=50) +
      xlab("") + ylab("") +
      stat_function(data=data.frame(x=c(0,1)), mapping=aes(x), fun=function(x) dbeta(x,a,b),colour="ora


pdf('Beta-rw10.pdf',width=7,height=4)
grid.arrange(p1,p2)
dev.off()
```

```
## pdf
##   2
```

```r
cat('average acceptance probability equals: ',mean(acc))
```

```
## average acceptance probability equals:  0.021
```

BASEBALL EXAMPLE, MODEL 2

The data are

```
players <- c('McGwire', 'Sosa', 'Griffey', 'Castilla', 'Gonzalez', 'Galaragga',  'Palmeiro',
'Vaughn',' Bonds', 'Bagwell', 'Piazza', 'Thome', 'Thomas','T. Martinez', 'Walker', 'Burks', 'Buhner')
Y <- c(7,9,4,7,3,6,2,10,2,2,4,3,2,5,3,2,6)
n <- c(58,59,74,84,69,63,60,54,53,60,66,66,72,64,42,38,58)
AB <- c(509,643,633,645,606,555,619,609,552,540,561,440,585,531,454,504,244)
HR <- c(70,66,56,46,45,44,43,40,37,34,32,30,29,28,23,21,15)
print(baseball<-data.frame(players=players,PS_AB=n,PS_HR=Y,S_AB=AB,S_HR=HR))
```

```
##        players PS_AB PS_HR S_AB S_HR
## 1      McGwire    58     7  509   70
## 2         Sosa    59     9  643   66
## 3      Griffey    74     4  633   56
## 4     Castilla    84     7  645   46
## 5     Gonzalez    69     3  606   45
## 6    Galaragga    63     6  555   44
## 7     Palmeiro    60     2  619   43
## 8       Vaughn    54    10  609   40
## 9        Bonds    53     2  552   37
## 10     Bagwell    60     2  540   34
## 11      Piazza    66     4  561   32
## 12       Thome    66     3  440   30
## 13      Thomas    72     2  585   29
## 14 T. Martinez    64     5  531   28
## 15      Walker    42     3  454   23
## 16       Burks    38     2  504   21
## 17      Buhner    58     6  244   15
```

We need the following 2 functions for updating the coefficients μ_i, $i = 1, \ldots, 17$.

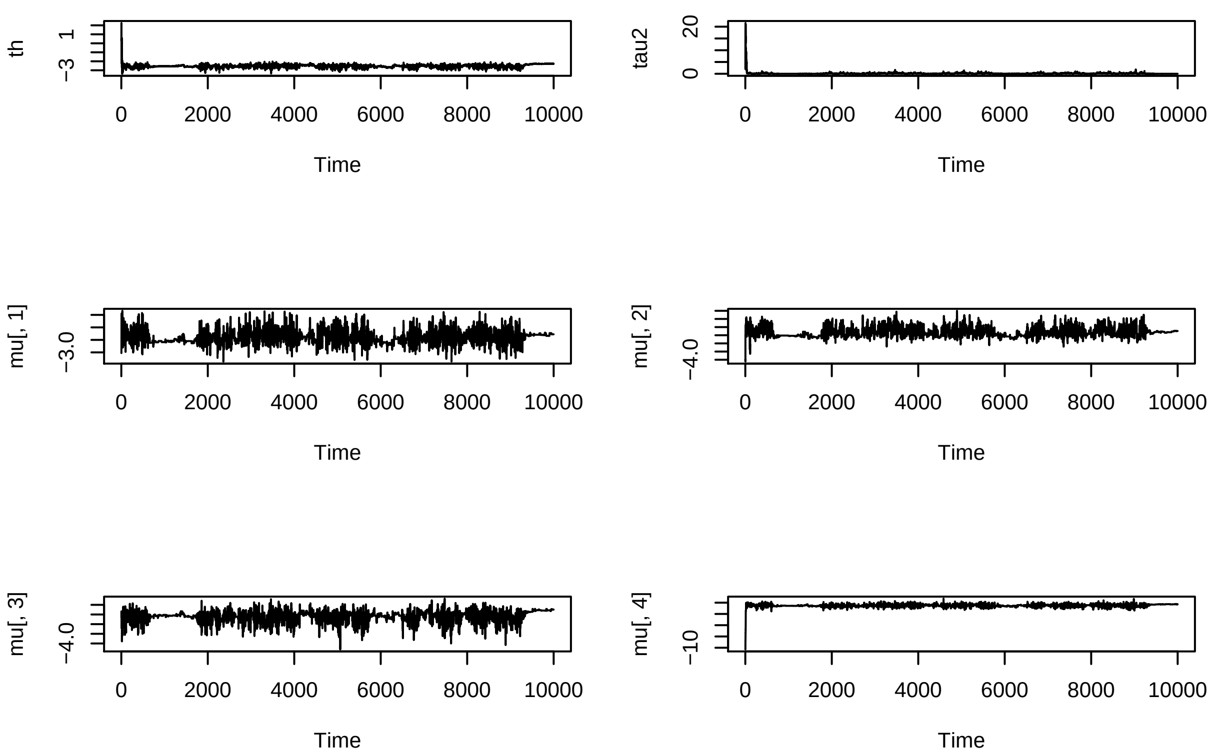
```
logtargetMui <- function(Yi, ni, mui, th, tau2,tunePar)
    Yi*mui-((mui-th)^2)/(2*tau2)-ni*log(1+exp(mui))

updateMui <- function(Yi, ni, mui, th, tau2, tunePar)
{
  muiNew <- mui + tunePar * rnorm(1)
  A <- exp(logtargetMui(Yi,ni,muiNew,th, tau2,tunePar)-
             logtargetMui(Yi,ni,mui,th, tau2,tunePar))
  ifelse (runif(1)<A, muiNew, mui)
}
```

```
N <- length(Y)
IT <- 10000  # number of iterations
tunePar <- 1 # tuning par for MH step updating mu (sd of normal distr)

# prior hyperpars
alpha <- 0.001
beta <- 0.001

# save iterates in matrix and vectors
mu <- matrix(0,IT,N)
th <- rep(0,IT)
tau2 <- rep(0,IT)
```

```
# initialise
mu[1,] <- rnorm(N,sd=5)  # arbitrary
th[1] <- rnorm(1)
tau2[1] <-2

# Gibbs sampler:
for (it in 2:IT)
{
  # update mu
  for (i in 1:N)
  {  mu[it,i] <- updateMui(Y[i],n[i],mu[it-1,i],
                           th[it-1],tau2[it-1],tunePar)  }
  # update th
  th[it] <- rnorm(1,mean(mu[it,]),sqrt(tau2[it-1]/N))
  # update tau2
  tau2[it] <- 1/(rgamma(1,shape=N/2+alpha,
                       rate=beta+.5*sum((mu[it,]-th[it])^2)))
}
```

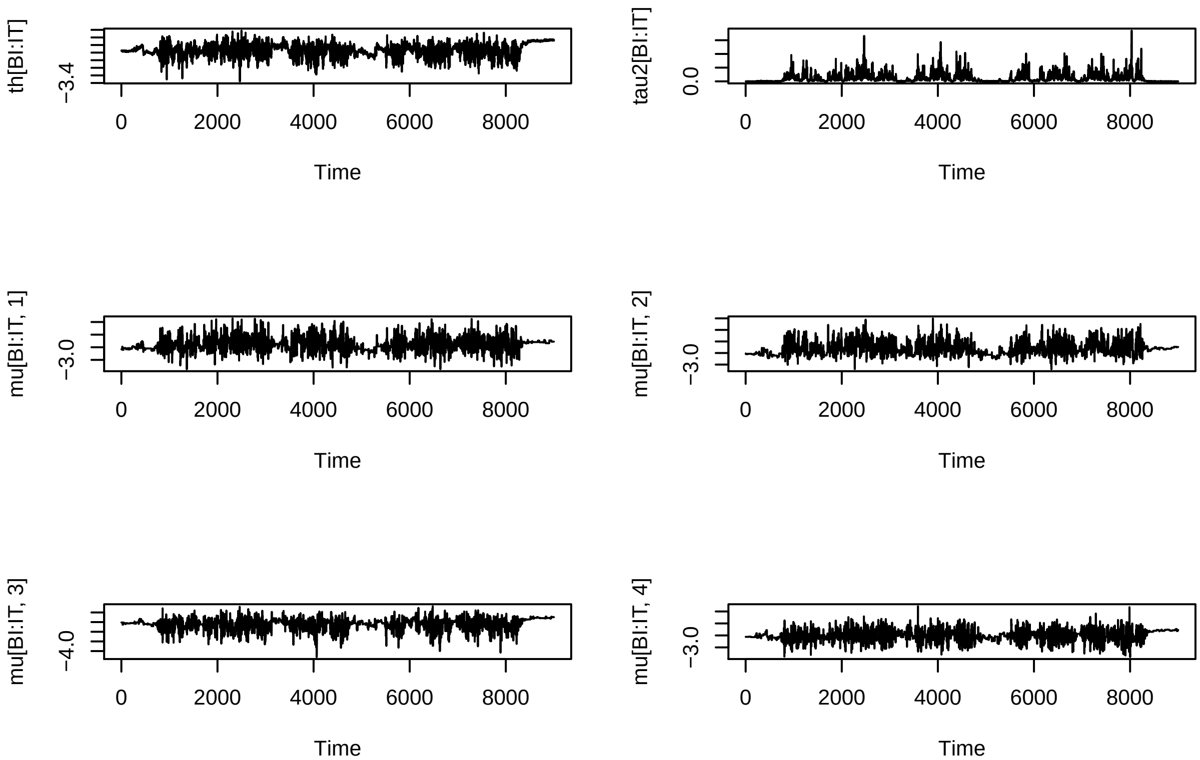
Visualisation

```
par(mfrow=c(3,2)) # make trace plots for th, tau2, mu1, mu2, mu3, mu4
plot.ts(th);plot.ts(tau2)
plot.ts(mu[,1]);plot.ts(mu[,2])
plot.ts(mu[,3]);plot.ts(mu[,4])
```




```
BI <- 1000 # discard first BI samples as BurnIn

plot.ts(th[BI:IT]);plot.ts(tau2[BI:IT])
plot.ts(mu[BI:IT,1]);plot.ts(mu[BI:IT,2])
```

```
plot.ts(mu[BI:IT,3]);plot.ts(mu[BI:IT,4])
```




```
# computate posterior means
th.pm <- mean(th[BI:IT])
tau2.pm <- mean(tau2[BI:IT])
mu.pm <- colMeans(mu[BI:IT,])
p.pm <- colMeans(1/(1+exp(-mu[BI:IT,])))

print(th.pm)
```

```
## [1] -2.543294
```

```
print(tau2.pm)
```

```
## [1] 0.1065423
```

```
print(mu.pm)
```

```
##  [1] -2.381159 -2.300796 -2.616010 -2.497637 -2.673856 -2.453892 -2.697256
##  [8] -2.213297 -2.663254 -2.689669 -2.584318 -2.645289 -2.725899 -2.517498
## [15] -2.540705 -2.591036 -2.451152
```

Add results to baseball dataframe

```
baseball$bayes <- p.pm
baseball$mle <- baseball$PS_HR/baseball$PS_AB # equals empirical fraction
baseball
```

```
##       players PS_AB PS_HR S_AB S_HR      bayes        mle
## 1     McGwire    58     7  509   70 0.08688297 0.12068966
## 2        Sosa    59     9  643   66 0.09392363 0.15254237
## 3     Griffey    74     4  633   56 0.07031249 0.05405405
## 4    Castilla    84     7  645   46 0.07757542 0.08333333
```

```
## 5     Gonzalez    69     3  606   45 0.06681850 0.04347826
## 6    Galaragga    63     6  555   44 0.08101138 0.09523810
## 7     Palmeiro    60     2  619   43 0.06581345 0.03333333
## 8       Vaughn    54    10  609   40 0.10251493 0.18518519
## 9        Bonds    53     2  552   37 0.06779007 0.03773585
## 10     Bagwell    60     2  540   34 0.06608699 0.03333333
## 11      Piazza    66     4  561   32 0.07195323 0.06060606
## 12       Thome    66     3  440   30 0.06846106 0.04545455
## 13      Thomas    72     2  585   29 0.06410743 0.02777778
## 14 T. Martinez    64     5  531   28 0.07665730 0.07812500
## 15      Walker    42     3  454   23 0.07536535 0.07142857
## 16       Burks    38     2  504   21 0.07210634 0.05263158
## 17      Buhner    58     6  244   15 0.08131376 0.10344828
```

```
library(ggplot2)
p1 <- ggplot(baseball, aes(x=players)) +
  geom_point(aes(y = bayes, shape="Bayes"),size=2.5) +
  geom_point(aes(y = mle, shape="mle"),size=2.5)+
  theme_minimal()+
    theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5),legend.position='none') +ylab('probabili
```

Add predictions for season

```
baseball$S_HR_Bayes <- baseball$S_AB * baseball$bayes
baseball$S_HR_mle <- baseball$S_AB * baseball$mle
baseball
```

```
##         players PS_AB PS_HR S_AB S_HR      bayes        mle S_HR_Bayes  S_HR_mle
## 1       McGwire    58     7  509   70 0.08688297 0.12068966   44.22343  61.43103
## 2          Sosa    59     9  643   66 0.09392363 0.15254237   60.39289  98.08475
## 3       Griffey    74     4  633   56 0.07031249 0.05405405   44.50780  34.21622
## 4      Castilla    84     7  645   46 0.07757542 0.08333333   50.03615  53.75000
## 5      Gonzalez    69     3  606   45 0.06681850 0.04347826   40.49201  26.34783
## 6     Galaragga    63     6  555   44 0.08101138 0.09523810   44.96132  52.85714
## 7      Palmeiro    60     2  619   43 0.06581345 0.03333333   40.73853  20.63333
## 8        Vaughn    54    10  609   40 0.10251493 0.18518519   62.43159 112.77778
## 9         Bonds    53     2  552   37 0.06779007 0.03773585   37.42012  20.83019
## 10      Bagwell    60     2  540   34 0.06608699 0.03333333   35.68697  18.00000
## 11       Piazza    66     4  561   32 0.07195323 0.06060606   40.36576  34.00000
## 12        Thome    66     3  440   30 0.06846106 0.04545455   30.12287  20.00000
## 13       Thomas    72     2  585   29 0.06410743 0.02777778   37.50285  16.25000
## 14 T. Martinez    64     5  531   28 0.07665730 0.07812500   40.70503  41.48438
## 15       Walker    42     3  454   23 0.07536535 0.07142857   34.21587  32.42857
## 16        Burks    38     2  504   21 0.07210634 0.05263158   36.34159  26.52632
## 17       Buhner    58     6  244   15 0.08131376 0.10344828   19.84056  25.24138
```

Compare performance

```
performance_Bayes = sum((baseball$S_HR-baseball$S_HR_Bayes)^2)
performance_mle = sum((baseball$S_HR-baseball$S_HR_mle)^2)
p2<- ggplot(baseball, aes(x=players)) +
     geom_point(aes(y = S_HR_Bayes, shape ="Bayes"),size=2.5)+
  geom_point(aes(y = S_HR_mle, shape = "mle"),size=2.5)+
  geom_point(aes(y = S_HR),shape=8,size=2.5,colour='blue')+theme_light()+
    theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5),legend.position='bottom') +ylab('nr of h
pdf('output-baseball_combined.pdf',width=8,height=8)
```

```r
grid.arrange(p1,p2,ncol=1)
dev.off()
```

```
## pdf
##   2
```

```r
cat('performance Bayes equals: ',performance_Bayes)
```

```
## performance Bayes equals:  2065.1
```

```r
cat('performance mle equals: ',performance_mle)
```

```
## performance mle equals:  9050.857
```

Compute predictive distributions

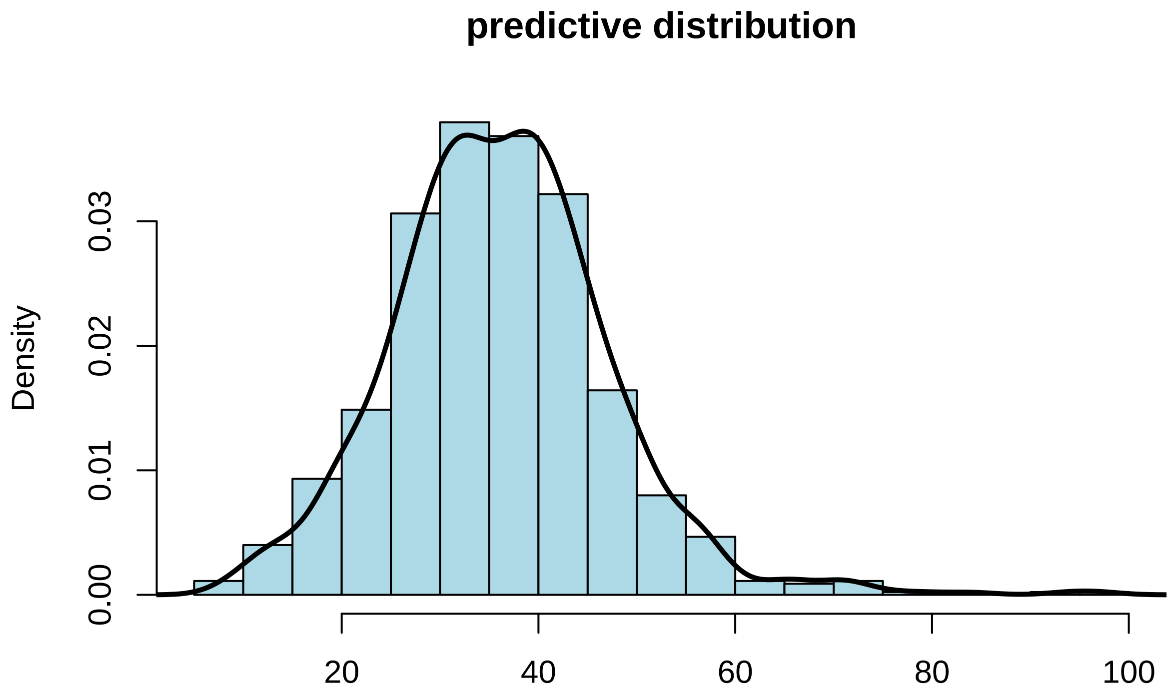
```r
ind <- seq(BI,IT,by=10) # use every 10-th iterate from non-burnin samples
L <- length(ind)
pred <- matrix(0,L,N)

for (i in 1:N)
{
    for (j in 1:L)
        pred[j,i] <- rbinom(1,AB[i], 1/(1+exp(-mu[ind[j],i])))
}

meanPred <- colMeans(pred)
meanPred
```

```
##  [1] 44.16981 60.85461 44.57270 50.20311 40.08657 45.02109 40.80355 62.65705
##  [9] 37.39734 35.63152 40.16648 30.02109 37.28635 40.93452 33.88901 36.24084
## [17] 19.56715
```

```r
cat('Sum of squared prediction error equals: ',sum((meanPred-HR)^2))
```

```
## Sum of squared prediction error equals:  2062.463
```

```r
# plot for the i-th second player the predictive distr
i<-16
hist(pred[,i],breaks='FD',prob=TRUE, main='predictive distribution',xlab='',col='lightblue')
lines(density(pred[,i]),lwd=2.5)
```



Note that the results are sensitive to the choice of hyperpars. $\alpha = \beta = 0.01$ and $\alpha = \beta = 0.001$ give quite different sum of squared prediction errors!