Bayesian Modelling - Homework 1

Sebastian Kimm-Friedenberg

2022-11-02

Load packages:

```
library(bayesm)
library(data.table)
library(knitr)
library(flextable)
library(tidyverse)
set.seed(60323)

library(compiler)
runireg_c=cmpfun(runireg) # a bit faster
```

Task 1

```
nobs <- c(5,10,100,1000,10000,20000, 30000) # different ns

# matrices in which results are stored
b_b1 <- matrix(0, length(nobs), 3) # m rows, 3 columns
sd_b1 <- matrix(0, length(nobs), 3)
b_ols <- b_b1
sd_ols <- b_b1

iteration <- 0

for (n in nobs){
  print(n)
  iteration <- iteration + 1
  # set up DGP
  #n <- 5
  X <- cbind(rep(1,n),runif(n),runif(n))
  beta <- c(-1,3,6)
  y <- (X %*% beta + rnorm(n, mean = 0, sd = 1))# set up y

  # set up sampling
  R <- 80000 # as in the lecture
  Mcmc1 <- list(R=R,keep=1, nprint = 0)
  Prior <- list(nu=0, ssq=0) # beta = 0 as default

  # run regressions
  ## bayes
  out_b1 <- runireg_c(Data=list(y=y,X=X), Prior = Prior,Mcmc=Mcmc1)
  b_draw1 <-out_b1$betadraw
```

```
  b_b1[iteration,] <- apply(b_draw1, 2, mean) # get mean for each coeff.
  sd_b1[iteration,] <- apply(b_draw1, 2, sd) # standard deviation

  ## ols
  out_ols <- lm(y ~ X[,-1]) # without intercept in x, is added automatically
  summary(out_ols)
  b_ols[iteration,] <- matrix(out_ols$coefficients, 1, 3)
  sd_ols[iteration,] <- matrix(sqrt(diag(vcov(out_ols))), 1, 3)
}
```

The previous code computed the outcomes for the Bayes and OLS estimator along with their standard deviations. The reuslts are presented below:

```
## Warning: fonts used in `flextable` are ignored because the `pdflatex` engine
## is used and not `xelatex` or `lualatex`. You can avoid this warning by using
## the `set_flextable_defaults(fonts_ignore=TRUE)` command or use a compatible
## engine by defining `latex_engine: xelatex` in the YAML header of the R Markdown
## document.
```

n	b1	sd_b1	b2	sd_b2	b3	sd_b3	ols1	sd_ols1	ols2	sd_ols2	ols3	sd_ols3
5	0.1272	1.2480	1.2040	1.4839	6.5530	1.2107	0.0478	1.4324	1.2464	1.7047	6.6661	1.3786
10	-0.0683	1.0388	3.2107	1.0363	4.6992	1.0713	-0.2166	1.0908	3.3151	1.0768	4.8368	1.1239
100	-1.4613	0.2309	3.3252	0.3332	6.6284	0.3038	-1.4683	0.2325	3.3301	0.3341	6.6370	0.3059
1,000	-1.0997	0.0838	3.0957	0.1071	6.0799	0.1113	-1.1002	0.0838	3.0955	0.1074	6.0812	0.1104
10,000	-1.0147	0.0263	3.0308	0.0344	5.9962	0.0345	-1.0149	0.0263	3.0310	0.0345	5.9963	0.0344
20,000	-1.0159	0.0186	3.0272	0.0245	5.9974	0.0244	-1.0159	0.0186	3.0271	0.0245	5.9974	0.0244
30,000	-1.0188	0.0153	3.0090	0.0199	6.0176	0.0200	-1.0188	0.0153	3.0090	0.0199	6.0175	0.0199

We can see that with increasing sample size both estimators converge to the true values of $\beta =$,(-1, 3, 6). The Bayesian estimator constantly has a smaller standard deviation than the OLS estimator, implying the Bayesian being more efficient.

Task 2

```
nobs <- 10000 # medium sized observation size
noby <- n # faster then changing all "n" to "nobs"
# matrices in which results are stored
b_b <- matrix(0, length(nobs), 3) # m rows, 3 columns
sd_b <- matrix(0, length(nobs), 3)
b_ols <- b_b
sd_ols <- b_b

# set up DGP
#n <- 5
hilf= runif(nobs)
X <- cbind(rep(1,n),hilf, hilf)
beta <- c(-1,3,6)
y <- (X %*% beta + rnorm(n, mean = 0, sd = 1))# set up y
```

```r
# set up sampling
R <- 80000 # as in the lecture
Mcmc1 <- list(R=R,keep=1, nprint = 0)
Prior <- list(nu=0, ssq=0) # beta = 0 as default

# run regressions
## bayes
out_b <- runireg_c(Data=list(y=y,X=X), Prior = Prior,Mcmc=Mcmc1)
b_draw <-out_b$betadraw
b_b[1,] <- apply(b_draw, 2, mean) # get mean for each coeff.
sd_b[1,] <- apply(b_draw, 2, sd) # standard deviation

## ols
out_ols <- lm(y ~ X[,-1]) # without intercept in x, is added automatically
summary(out_ols)
b_ols[1,] <- matrix(out_ols$coefficients, 1, 3)
sd_ols[1,] <- matrix(sqrt(diag(vcov(out_ols))), 1, 3)
```

```r
res <- data.frame(n = nobs, b1 = b_b[,1], sd_b1 = sd_b[,1],
                            b2 = b_b[,2], sd_b2 = sd_b[,2],
                            b3 = b_b[,3], sd_b3 = sd_b[,3],
                            ols1 = b_ols[,1], sd_ols1 = sd_ols[,1],
                            ols2 = b_ols[,2], sd_ols2 = sd_ols[,2],
                            ols3 = b_ols[,3], sd_ols3 = sd_ols[,3])

round(res,4) %>% regulartable() %>% autofit() %>% fit_to_width(max_width = 6.5)
```

```
## Warning: fonts used in `flextable` are ignored because the `pdflatex` engine
## is used and not `xelatex` or `lualatex`. You can avoid this warning by using
## the `set_flextable_defaults(fonts_ignore=TRUE)` command or use a compatible
## engine by defining `latex_engine: xelatex` in the YAML header of the R Markdown
## document.
```

n	b1	sd_b1	b2	sd_b2	b3	sd_b3	ols1	sd_ols1	ols2	sd_ols2	ols3	sd_ols3
10,000	-1.0066	0.0116	4.5245	7.071	4.4935	7.071	-1.0067	0.0116	9.0181	0.0202		

```r
kable(summary(b_draw), digits = 4)
```

Summary of Posterior Marginal Distributions Moments mean std dev num se rel eff sam size 1 -1.0 0.012 0.00004 0.87 72000 2 4.5 7.079 0.02579 0.96 72000 3 4.5 7.079 0.02580 0.96 72000

Quantiles 2.5% 5% 50% 95% 97.5% 1 -1.0 -1.0 -1.0 -0.99 -0.98 2 -9.3 -7.2 4.5 16.17 18.42 3 -9.4 -7.1 4.5 16.19 18.32 based on 72000 valid draws (burn-in=8000)

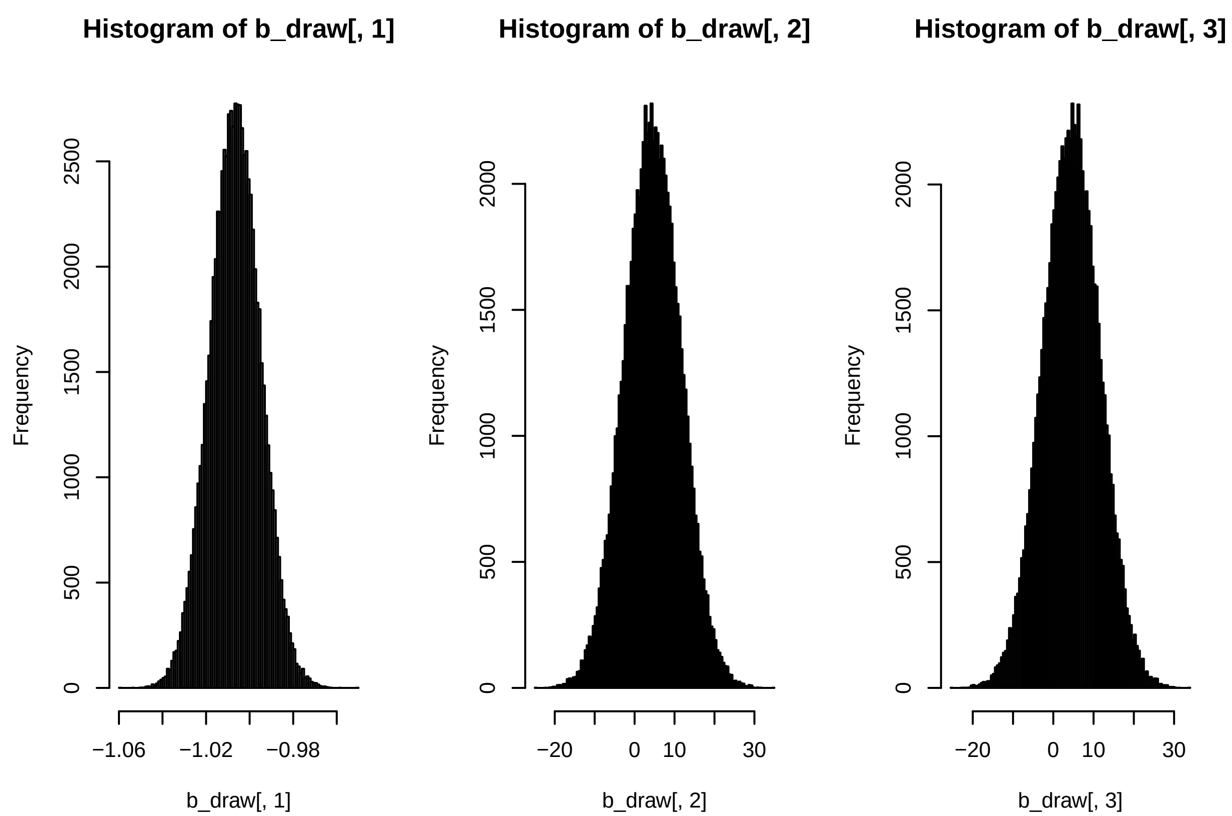
meanstd devnum se rel effsam size			
-1.0066	0.0116	0.00000.8744	72000
4.5339	7.0793	0.02580.9556	72000
4.4841	7.0793	0.02580.9560	72000

```r
kable(t(apply(b_draw,2,quantile)), digits = 4)
```

0%	25%	50%	75%	100%
-1.0594	-1.0144	-1.0066	-0.9989	-0.9506
-24.7829	-0.2485	4.4883	9.2924	34.5134
-25.4765	-0.2735	4.5263	9.2687	33.7629

The `lm` command, i.e. the OLS estimation, automatically omits one of the identical variables. Due to multicolinearity, the matrix $X'X$ is not invertible and the OLS estimator would not be feasible. In contrast, the Bayesian posterior distribution is computed for both variables and it is almost identical. The third variable has a slightly smaller mean and seems to be a bit more skewed to the right, according to the quantiles.

Plotting the three variables confirms that the poserior distribtuion of 2 and 3 is almost identical.



Task 3

```r
nobs <- c(5,10,100,1000,10000,20000, 30000) # different ns

# matrices in which results are stored
b_b3 <- matrix(0, length(nobs), 3) # m rows, 3 columns
sd_b3 <- matrix(0, length(nobs), 3)
b_pro <- b_b3
```

```r
sd_pro <- b_b3

iteration <- 0

for (n in nobs){
  print(n)
  iteration <- iteration + 1
  # set up DGP
  #n <- 5
  X <- cbind(rep(1,n),runif(n),runif(n))
  beta <- c(-1,3,6)
  y <- (X %*% beta + rnorm(n, mean = 0, sd = 1))# set up y
  y <- ifelse(y<0,0,1)

  # set up sampling
  R <- 80000 # as in the lecture
  Mcmc1 <- list(R=R,keep=1, nprint = 0)
  #Prior <- list(nu=0, ssq=0) # beta = 0 as default

  # run regressions
  ## bayes
  out_b3 <- runireg_c(Data=list(y=y,X=X),Mcmc=Mcmc1)
  b_draw3 <-out_b3$betadraw
  b_b3[iteration,] <- apply(b_draw3, 2, mean) # get mean for each coeff.
  sd_b3[iteration,] <- apply(b_draw3, 2, sd) # standard deviation

  ## ols
  out_pro <- glm(y ~ X[,-1], family=binomial(link="probit")) # without intercept in x, is added automat
  summary(out_pro)
  b_pro[iteration,] <- matrix(out_pro$coefficients, 1, 3)
  sd_pro[iteration,] <- matrix(sqrt(diag(vcov(out_pro))), 1, 3)
}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Below, the first table shows the results of the Bayes estimator and the probit estimator for the probit data. The second shows the comparison of the Bayes estimator performance for linear and probit data. Clearly, the probit model converges (slowly) to the true parameters whereas the Bayesian model is consistently wrong, i.e. unable to recover the true parameters -1, 3, 6. Regarding the trade-off between sample size and having access to the linear observations or only to the probit observations, the access to linear observations should always be preferred. In the linear case, the estimator converges relatively quickly whereas in the probit case it quite far away from the true result regardless of the sample size.

```
## Warning: fonts used in `flextable` are ignored because the `pdflatex` engine
## is used and not `xelatex` or `lualatex`. You can avoid this warning by using
## the `set_flextable_defaults(fonts_ignore=TRUE)` command or use a compatible
## engine by defining `latex_engine: xelatex` in the YAML header of the R Markdown
## document.
```

n	b1	sd_b1	b2	sd_b2	b3	sd_b3	pro1	sd_pro1	pro2	sd_pro2	pro3	sd_pro3
5	0.9822	0.0543	0.0249	0.1080	-0.0096	0.2255	6.5528	24,074.9174	0.0000	54,796.1802	0.0000	120,125.7980
10	0.9925	0.0262	0.0073	0.0319	0.0059	0.0296	6.5528	15,218.3276	0.0000	18,556.5465	0.0000	17,132.8289
100	0.6354	0.0617	0.2503	0.0786	0.3367	0.0823	-3.2218	1.3506	7.2138	2.7046	9.7542	4.2229
1,000	0.7555	0.0169	0.1788	0.0224	0.2131	0.0218	-0.7046	0.2176	2.9741	0.4333	5.1495	0.7671
10,000	0.7311	0.0056	0.1658	0.0074	0.2592	0.0074	-1.0011	0.0743	2.7890	0.1313	6.0738	0.2566
20,000	0.7385	0.0039	0.1754	0.0050	0.2414	0.0051	-1.0801	0.0550	3.2650	0.1065	6.0528	0.1877
30,000	0.7370	0.0032	0.1741	0.0041	0.2446	0.0041	-1.0379	0.0436	3.1232	0.0828	6.0727	0.1529

```
## Warning: fonts used in `flextable` are ignored because the `pdflatex` engine
## is used and not `xelatex` or `lualatex`. You can avoid this warning by using
## the `set_flextable_defaults(fonts_ignore=TRUE)` command or use a compatible
## engine by defining `latex_engine: xelatex` in the YAML header of the R Markdown
## document.
```

n	b1.1	sd_b1.1	b2.1	sd_b2.1	b3.1	sd_b3.1	b1.3	sd_b1.3	b2.3	sd_b2.3	b3.3	sd_b3.3
5	0.1272	1.2480	1.2040	1.4839	6.5530	1.2107	0.9822	0.0543	0.0249	0.1080	-0.0096	0.2255
10	-0.0683	1.0388	3.2107	1.0363	4.6992	1.0713	0.9925	0.0262	0.0073	0.0319	0.0059	0.0296
100	-1.4613	0.2309	3.3252	0.3332	6.6284	0.3038	0.6354	0.0617	0.2503	0.0786	0.3367	0.0823
1,000	-1.0997	0.0838	3.0957	0.1071	6.0799	0.1113	0.7555	0.0169	0.1788	0.0224	0.2131	0.0218
10,000	-1.0147	0.0263	3.0308	0.0344	5.9962	0.0345	0.7311	0.0056	0.1658	0.0074	0.2592	0.0074
20,000	-1.0159	0.0186	3.0272	0.0245	5.9974	0.0244	0.7385	0.0039	0.1754	0.0050	0.2414	0.0051
30,000	-1.0188	0.0153	3.0090	0.0199	6.0176	0.0200	0.7370	0.0032	0.1741	0.0041	0.2446	0.0041