Homework 9

Shao-Ting Chiu (UIN:433002162)

11/17/22

Table of contents

Computational Environment

```julia
using Pkg
Pkg.activate("hw9")
using Distributions
using DataFrames
using Plots
using DelimitedFiles
using LinearAlgebra
using Statistics
using ProtoStructs
using CSV
import Random
Random.seed!(2022)
```

```
  Activating project at `~/Documents/GitHub/STAT638_Applied-Bayes-Methods/hw/hw9`
```

```
Random.TaskLocalRNG()
```

Description

- Course: STAT638, 2022 Fall
- Deadline: 2022/11/17, 12:00 pm

  Read Chapter 9 in the Hoff book. Then do Problems 9.1 and 9.2 in Hoff.

  For both regression models, please include an intercept term (β_0).

  In 9.1(b), please replace "max" by "min". (This is not listed in the official book errata, but appears to be a typo.)

  For 9.2, the azdiabetes.dat data are described in *Exercise* 6 of Chapter 7 (see errata).

- Note: This PDF file is generated by Quarto and LualaTeX. There is unsolved problem to display the special character in the source code. Thus, I leave the html version here for reference that displays the complete source code:

  https://stchiu.quarto.pub/stat_638_hw_9/

Problem 9.1

Extrapolation: The file `swim.dat` contains data on the amount of time in seconds, it takes each of four high school swimmers to swim 50 yards. Each swimmer has 6 times, taken on a biweekly basis.

(a)

Perform the following data analysis for each swimmer separately:

1. Fit a linear regression model of swimming time as the response and week as the explanatory variable. To formulate your prior, use the information that competitive times for this age group generally range from 22 to 24 seconds.
2. For each swimmer j, obtain a posterior predictive distribution for Y_j^*, their time if they were to swim 2 weeks from the last recorded time.

- Suppose a linear model

$$Y = X\beta + \epsilon$$

$$Y_i = x_{i,1}\beta_1 + x_{i,2}\beta_2 + \epsilon_i$$

- $Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_6 \end{bmatrix}$. A swimmer's record of 6. Series in time
- $X = \begin{bmatrix} x_{1,1} & x_{1,2} \\ \vdots & \vdots \\ x_{6,1} & x_{6,2} \end{bmatrix}$
  - $x_{j,1}$: jth record with swim score in the range of 22 to 24 second
  - $x_{j,2}$: Weeks of training
- $\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}$.
  - $\mu_0 = \begin{bmatrix} 23 \\ 0 \end{bmatrix}$
    - The prior expectation of intercept of y is 23.
  - $\beta_0 \sim N_p(\mu_0, \Sigma_0)$.
    1. FCD: $\beta | y, \sigma^2 \sim N_p(\beta_n, \Sigma_n)$
    2. $\Sigma_n^{-1} = \Sigma_0^{-1} + \frac{X^T X}{\sigma^2}$
    3. $\beta_n = \Sigma_n(\Sigma_0^{-1}\beta_0 + \frac{X^T y}{\sigma^2})$

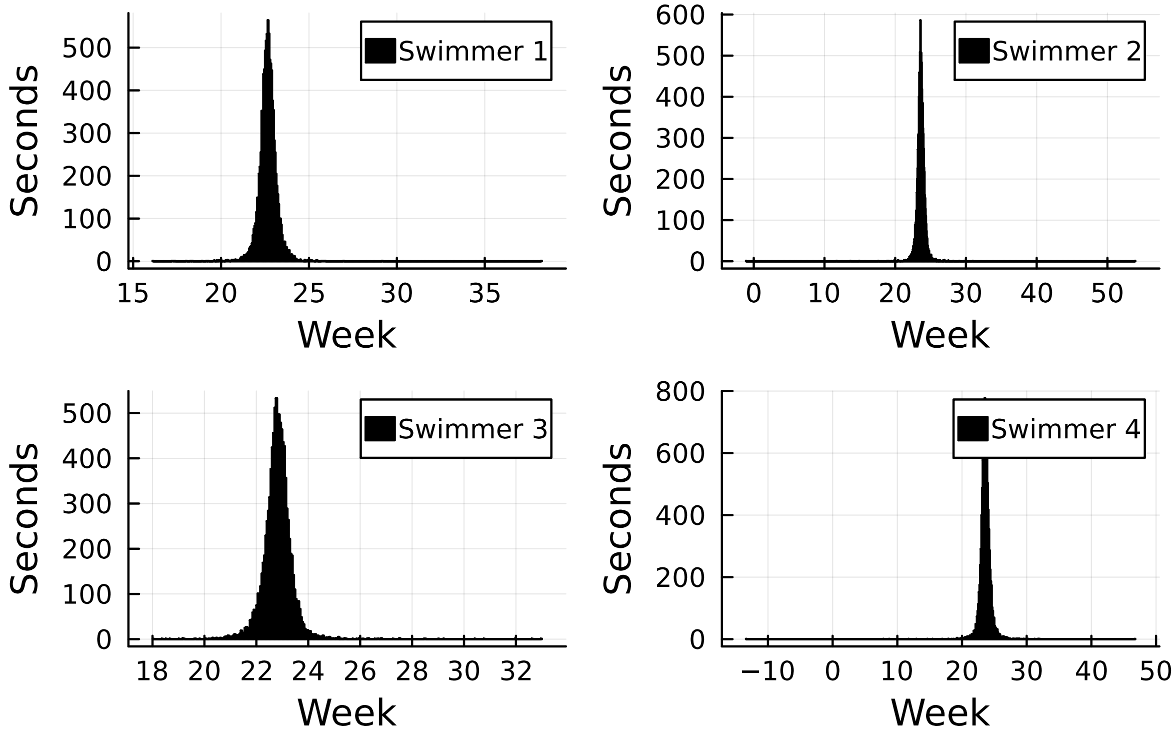
Prior setting

- $\Sigma_0 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$
  - There is uncertainty about β estimation.
  - Covariance of time and intersept is believe as 0
- $\sigma^2 \sim IG(\nu_0/2, \nu_0\sigma_0^2/2)$
  - FCD: $\sigma^2 | y, \beta \sim IG((\nu_0 + n)/2, (\nu_0\sigma_0^2) + SSR(\beta)/2)$
- $SSR(\beta) = (y - X\beta)^T(y - X\beta)$

```
ys = readdlm("data/swim.dat")
```

```
4×6 Matrix{Float64}:
 23.1  23.2  22.9  22.9  22.8  22.7
 23.2  23.1  23.4  23.5  23.5  23.4
 22.7  22.6  22.8  22.8  22.9  22.8
 23.7  23.6  23.7  23.5  23.5  23.4
```

```
"""
Problem 9.1 (a)
"""

@proto struct SwimmingModel
    S = 1000 # Number of sampling
    # Data
    y
    n = length(y) # number of records
    # Model
    X = hcat( ones(n), collect(0:2:10) )
    p = size(X)[2]
    # Prior
      = MvNormal([23., 0.], [0.1 0; 0 0.1])
      = 1.
     ² = 0.2
end


function SSR( , y, X)
    ssrV = (y - X* )' * (y - X* )
    return sum(ssrV)
end

function  _FCD( ², m::SwimmingModel)
    Σ  =( m.  .Σ^-1 + m.X' * m.X /  ²)^-1
      = Σ *(m.  .Σ^-1 * m.  .  + m.X' * m.y /  ²)
    return MvNormal(vec( ), Hermitian(Σ ))
end

function  ²_FCD( , m::SwimmingModel)
       = (m.   + m.n)/2
       = (m.  *m.  ²) + SSR( , m.y, m.X)
    return InverseGamma( ,  )
end

function pred(X, m::SwimmingModel)
    # Sampling vector
     smp = zeros(m.S, length(m.  .))
     ²smp = zeros(m.S)
    y = zeros(m.S)
    # Init
    smp[1,:] = rand(m. )
    ²smp[1] = m. ²
    y[1] = m.y[1]
    for i in 2:m.S
        smp[i,:] = rand( _FCD( ²smp[i-1], m))
        ²smp[i] = rand( ²_FCD( smp[i-1,:], m))

        # Predict
        y[i] =  smp[i,:]' * X + rand(Normal(0.,  ²smp[i]))
    end

    return (y=y,  = smp,  ²= ²smp)
end

j_swim = 1
ms = [ SwimmingModel(y = hcat(ys[i,:]), S=10000 ) for i in 1:size(ys)[1] ]
ys_pred = zeros(size(ys)[1], ms[1].S)
X_pred = [1,12]

for i in eachindex(ms)
    ys_pred[i,:] = pred([1,12], ms[i]).y
end

## Plotting
p = [histogram(ys_pred[i,:], label="Swimmer $i", color="black",
    xlabel="Week", ylabel="Seconds"
    ) for i in 1:size(ys)[1]]
plot(p...)
```



(b)

The coach of the team has to decide which of the four swimmers will compete in a swimming meet in 2 weeks. Using your predictive distributions, compute $Pr(Y_j^* = \max\{Y_1^*, \dots, Y_4^*\}|Y)$ for each swimmer j, and based on this make a recommendation to the coach.

```
am = argmax(ys_pred, dims=1)

y_count = zeros(1, size(ys)[1])

for a in am
    y_count[a[1]] += 1
end

pmax = vec(y_count ./ length(am))

## Recommendation
ds = DataFrame( Dict("Swimmer"=> collect(1:size(ys)[1]),"Pr(Y_i is max)" => pmax ))
```

	Pr(Y_i is max)	Swimmer
	Float64	Int64
1	0.023	1
2	0.4773	2
3	0.0424	3
4	0.4573	4

```
Swimmer 2 is the most probable winner.
```

Problem 9.2

Model selection: As described in Example 6 of Chapter 7, the file `azdiabetes.dat` contains data on health-related variables of a population of 532 women. In this exercise we will be modeling the conditional distribution of glucose level (`glu`) as a linear combination of the other variables, excluding the variable `diabetes`.

(a)

Fit a regression model using the g-prior with $g = n$, $\nu_0 = 2$ and $\sigma_0^2 = 1$. Obtain posterior confidence intervals for all of the parameters.

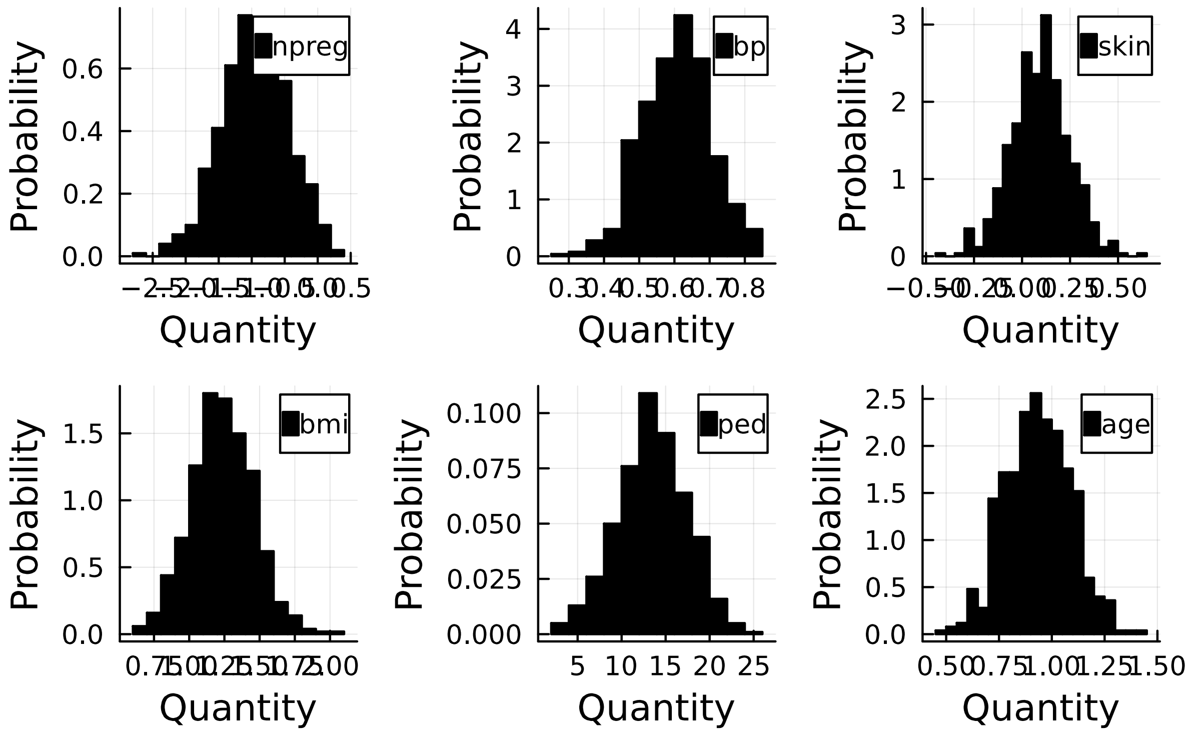
```
data = readdlm("data/azdiabetes.dat")
```

```
533×8 Matrix{Any}:
   "npreg"     "glu"    "bp"    "skin"     "bmi"    "ped"     "age"   "diabetes"
  5           86      68      28          30.2     0.364    24        "No"
  7          195      70      33          25.1     0.163    55        "Yes"
  5           77      82      41          35.8     0.156    35        "No"
  0          165      76      43          47.9     0.259    26        "No"
  0          107      60      25          26.4     0.133    23        "No"
  5           97      76      27          35.6     0.378    52        "Yes"
  3           83      58      31          34.3     0.336    25        "No"
  1          193      50      16          25.9     0.655    24        "No"
  3          142      80      15          32.4     0.2      63        "No"
  2          128      78      37          43.3     1.224    31        "Yes"
  0          137      40      35          43.1     2.288    33        "Yes"
  9          154      78      30          30.9     0.164    45        "No"

 12          100      84      33          30.0     0.488    46        "No"
  1          147      94      41          49.3     0.358    27        "Yes"
```

3	187	70	22	36.4	0.408	36	"Yes"
1	121	78	39	39.0	0.261	28	"No"
3	108	62	24	26.0	0.223	25	"No"
0	181	88	44	43.3	0.222	26	"Yes"
1	128	88	39	36.5	1.057	37	"Yes"
2	88	58	26	28.4	0.766	22	"No"
9	170	74	31	44.0	0.403	43	"Yes"
10	101	76	48	32.9	0.171	63	"No"
5	121	72	23	26.2	0.245	30	"No"
1	93	70	31	30.4	0.315	23	"No"

```
dt = data[1:end , 1:end-1]
y = float.(dt[2:end, 2])
X = float.(dt[2:end, 1:end .!= 2])
ns = data[1,1:end-1]
ns = ns[1:end .!=2]

@proto struct DiabetesModel
    S = 500 # Number of sampling
    # Data
    y
    X
    n = length(y) # number of records
    p = size(X)[2]
    # Model
    # Prior
    g = n # g prior
      = 2.
     ² = 1.
end

function  _FCD( ², m::DiabetesModel)
    return  _FCD( ², m.g, m.X, m.y)
end

function  _FCD( ², g, X, y)
    Σ  = g/(g+1) *  ² * (X'X)^-1
      = g/(g+1) * ^( ², y, X)
      =  MvNormal( , Hermitian(Σ ))
    return
end
```

```
function ^( ², y, X)
    return  ² * (X'X)^-1 * (X'y /  ²)
end



function  ²_FCD(m::DiabetesModel)
    return  ²_FCD(m. , m. ², m.n, m.X, m.y, m.g)
end

function  ²_FCD( ,  ², n, X, y, g)
      =   + n / 2.
      = (  *  ² + SSR(X, y, g))/2.
     ² = InverseGamma( ,  )
    return  ²
end

function SSR(m::DiabetesModel)
    return SSR(m.X, m.y, m.g)
end

function SSR(X, y, g)
    return y'*(I - g/(g+1)*X*(X'X)^-1*X')*y
end


m = DiabetesModel(y=y, X=X )
 ²smp = zeros(m.S, 1)
 smp = zeros(m.S, size(m.X)[2])

for i in 1: m.S
     ²smp[i] = rand( ²_FCD(m))
     smp[i,:] = rand( _FCD( ²smp[i], m))
end

ps = [histogram( smp[:,i], xlabel="Quantity",
      ylabel="Probability",normalize=true, label="$(ns[i])", color="black")
      for i in 1:m.p]
plot(ps...)
```



(b)

Perform the model selection and averaging procedure described in Section 9.3. Obtain $Pr(\beta_j \neq 0|y)$, as well as posterior confidence intervals for all of the parameters. Compare to the results in part (a).

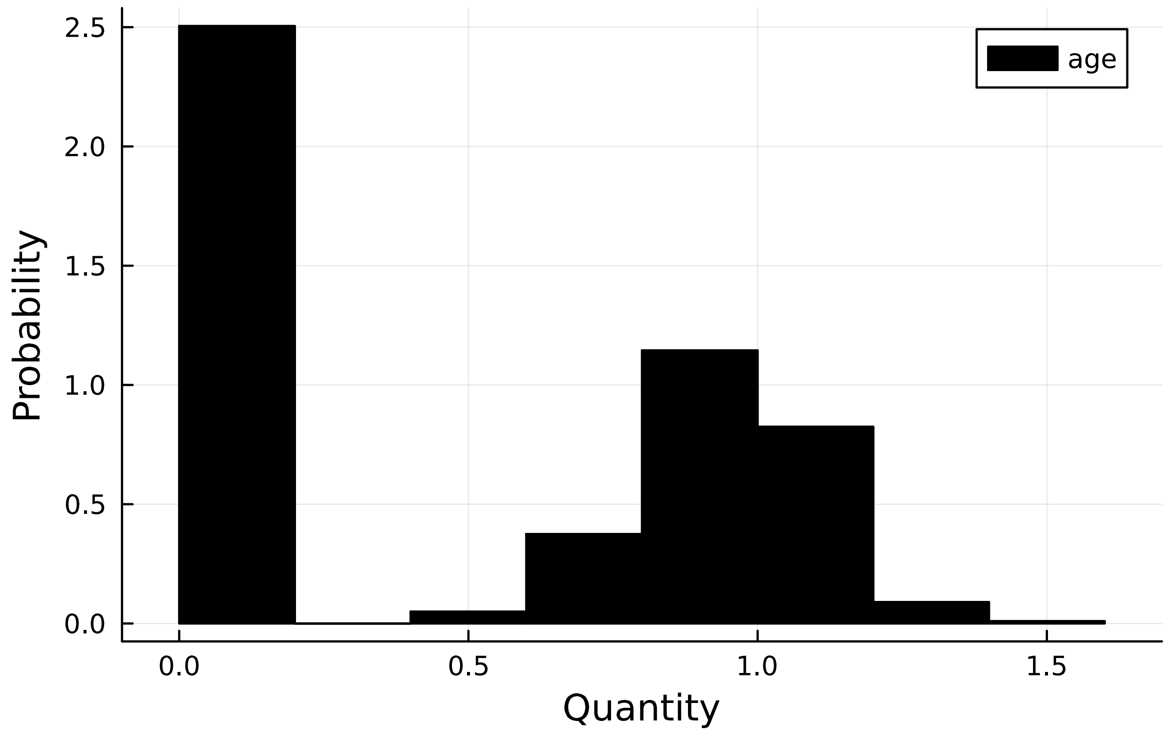
$$\{y|X_{z(k)}, \beta_{z(k)}, \sigma^2\} \sim \text{ multivariate normal } (X_{z(k)}\beta_{z(k)}, \sigma^2 I)$$

```
m = DiabetesModel(y=y, X=X,  S=1000)


function  ²_FCD(m::DiabetesModel, zs)
    Xz = @view m.X[1:end, Bool.(zs)]
    return  ²_FCD(m.  , m.  ², m.n, Xz, m.y, m.g)
end

function  _FCD( ², m::DiabetesModel, zs)
    Xz  = @view m.X[1:end, Bool.(zs)]
    return  _FCD( ², m.g, Xz, m.y)
end

function y_margin( z², m::DiabetesModel, zs)
       = m.
    n = m.n
    y = m.y
    g = m.g
    pz = sum(zs)
    Xz  = @view m.X[1:end, Bool.(zs)]
    ssr = SSR(Xz, y, g)

    pyl = (pz/2.)log(1. +g) + ( /2.)*log( z²) - (( +n)/2)log(( * z² + ssr))
    return pyl
end

function z_FCD(i  ,  z², zsmp, nSmp,m::DiabetesModel)
    zs = zsmp[nSmp,:]
    pj1 = sum(zsmp[1:nSmp, i]) / length(zsmp[1:nSmp, i])
    pj0 = 1. - pj1
    pj1_FCD_l =  pj1 * y_margin( z², m, ones(length(zs)))
    pj0_FCD_l =  pj0 * y_margin( z², m, zs)
    O = exp(pj0_FCD_l - pj1_FCD_l)
    return Bernoulli( 1/(1+O))
end


zsmp = ones(m.S, size(m.X)[2])
 ²smp = zeros(m.S, 1)
 smp = zeros(m.S, size(m.X)[2])

 ²smp[1] = 0.1
# Gibbs sampling
for i in 2:m.S
    for j in Random.shuffle(1:m.p)
        zsmp[i, j] = rand(z_FCD(j,  ²smp[i-1], zsmp, i-1, m))
    end

     ²smp[i] = rand( ²_FCD(m, zsmp[i,:]))
     smp[i, Bool.(zsmp[i,:])] = rand( _FCD( ²smp[i], m, zsmp[i,:]))
end
```

```
sum(zsmp, dims=1)/size(zsmp)[1]
prB = 1. .- vec(sum(zsmp, dims=1))./size(zsmp)[1]
DataFrame(Dict( "Bi"=> ns, "Pr(Bi != 0 |y)"=> prB ))
```

	Bi	Pr(Bi != 0 |y)
	Any	Float64
1	npreg	0.499
2	bp	0.499
3	skin	0.499
4	bmi	0.499
5	ped	0.499
6	age	0.5

```
inds = prB .>= 0.5
b_select = prB[inds]
ps = [histogram( smp[:, i], xlabel="Quantity",
      ylabel="Probability",normalize=true, label="$(ns[i])", color="black")
      for i in  findall(inds .== 1)]
plot(ps...)
```



```
DataFrame(Dict("Parameters"=> ns[inds],
               "Confidence interval"=> [quantile( smp[:,i], [0.25, 0.975]) for i in  findal
```

	Confidence interval	Parameters
	Array...	Any
1	[0.0, 1.1847]	age

Conclusion

There might be some bugs in FCD formulation. The current results show that the `age` is the only factor remained after model selection. The distribution is far different from what it is in part (a).