Problem 1 Answers
Problem 1 Answers

Problem 1a answers:

Mirror	Statistically different mirror(s)	Statistically not different mirror(s)
Glass-8	Glass-12, Glass-16, Steel-12, Steel-8	None
Glass-12	Glass-8	Glass-16, Steel-12, Steel-8
Glass-16	Glass-8	Glass-12, Steel-12, Steel-8
Steel-12	Glass-8	Glass-12, Glass-16, Steel-8
Steel-8	Glass-8	Glass-12, Glass-16, Steel-12

All conclusions drawn based on ca values. ca[1,2], ca[1,3], ca[1,4], ca[1,5] did not have zero in their 95% credible set. All ca[i,j] for i not equal to 1 and j = {2,3,4,5} had zeros. Based on this the above conclusions were drawn.

Problem 1b answers:

Glass-8 is different from Glass-12, Glass-16, Steel-12 and Steel-8. Glass-12, Glass-16, Steel-12, Steel-8 are all similar to each other. Refer ca values in report.

μ_1-μ_2 has a 95% credible set [3.776,28.61] and μ_3-μ_4 has a 95% credible set [-3.306,17.63]

Problem 1c answers:

Based on the ANOVA following conclusions could be reported to Dr. Michelson

1. There is **insignificant interaction between material of the mirror and number of sides** (alpha.beta(2,2) has a credible set containing zero, please note we do not consider any other alpha.beta as they are just determined by the sum to zero constraints)
2. Material of the mirror is a significant factor (ca(2) has a credible set not containing zero)
3. Number of Sides is a significant factor (cb(2) has a credible set not containing zero)

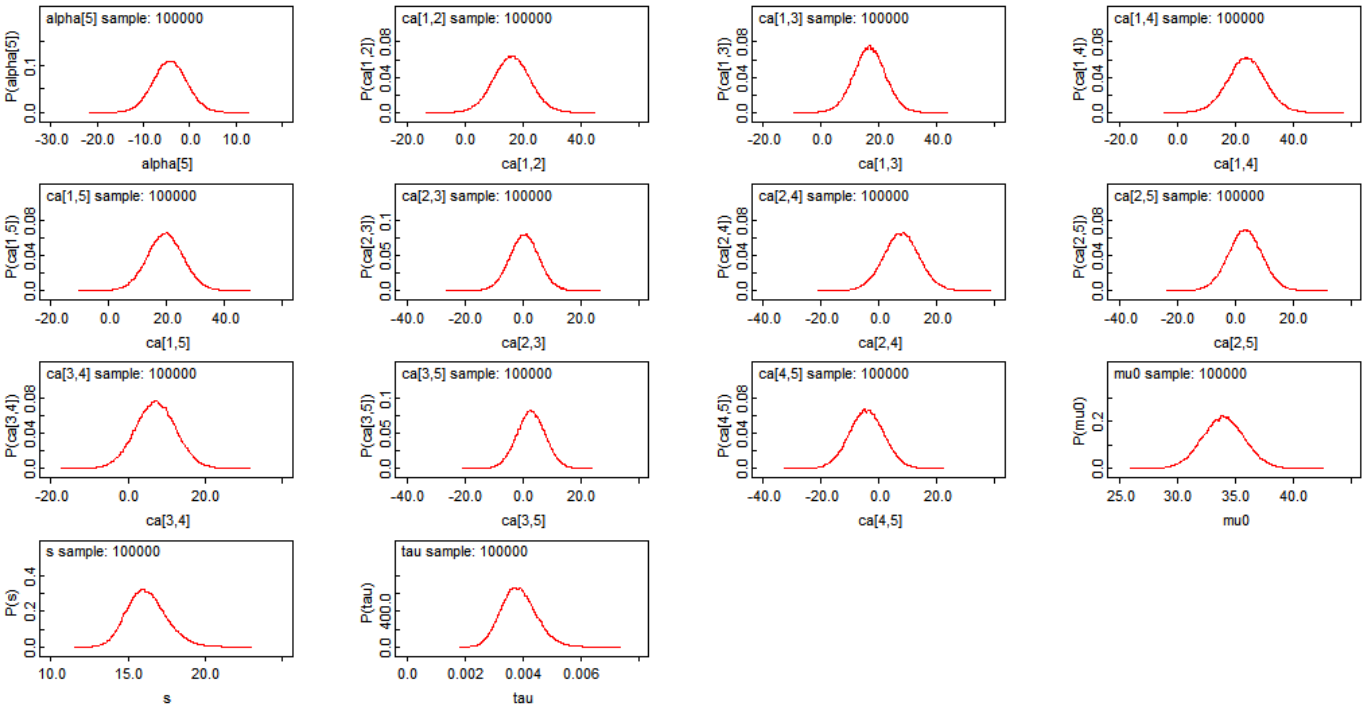
Problem 1 Statistics and graphs

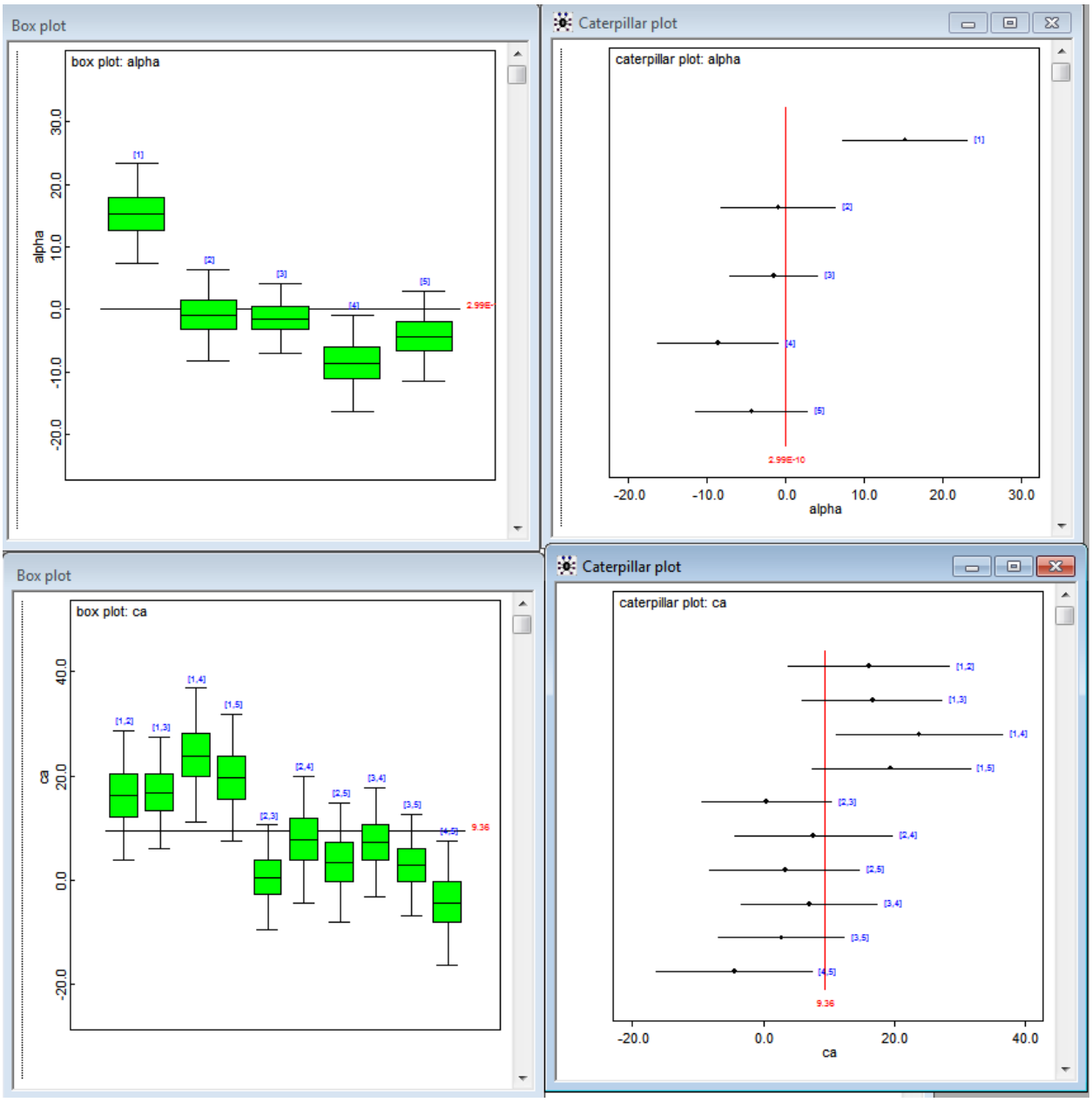
Problem 1a – Statistics and graphs

KEY in statistics:

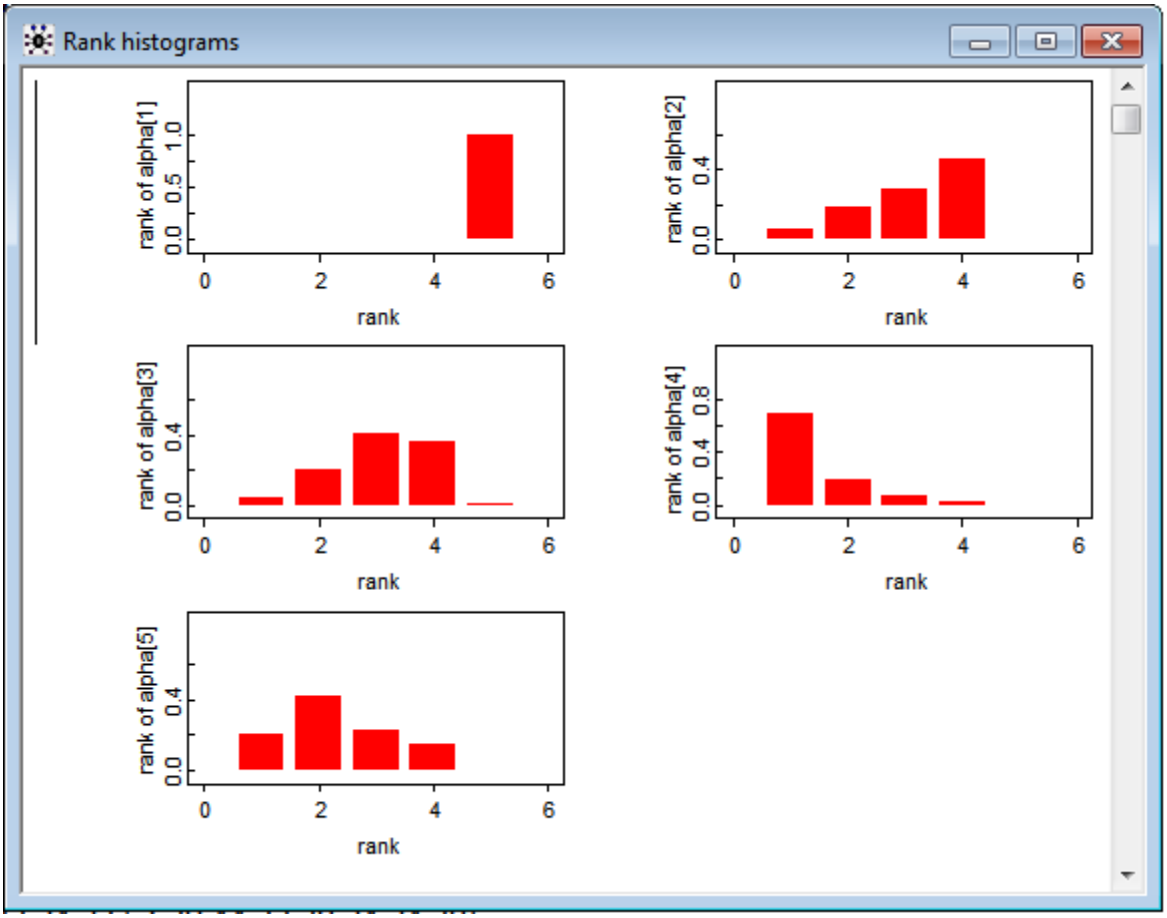
Mirror	Numbering
Glass-8	1
Glass-12	2
Glass-16	3
Steel-12	4
Steel-8	5

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha[1]	15.27	4.075	0.009191	7.322	15.25	23.29	1001	100000
alpha[2]	-0.9187	3.748	0.01475	-8.252	-0.9088	6.453	1001	100000
alpha[3]	-1.453	2.857	0.01217	-7.08	-1.465	4.18	1001	100000
alpha[4]	-8.605	3.945	0.01673	-16.38	-8.602	-0.8503	1001	100000
alpha[5]	-4.289	3.655	0.01366	-11.46	-4.289	2.891	1001	100000
ca[1,2]	16.18	6.31	0.01879	3.776	16.18	28.61	1001	100000
ca[1,3]	16.72	5.47	0.01593	5.971	16.7	27.5	1001	100000
ca[1,4]	23.87	6.502	0.02064	11.15	23.85	36.79	1001	100000
ca[1,5]	19.56	6.225	0.0175	7.388	19.55	31.84	1001	100000
ca[2,3]	0.534	5.075	0.02114	-9.412	0.5333	10.54	1001	100000
ca[2,4]	7.687	6.17	0.02608	-4.395	7.685	19.87	1001	100000
ca[2,5]	3.371	5.858	0.02289	-8.161	3.371	14.85	1001	100000
ca[3,4]	7.153	5.322	0.02395	-3.306	7.144	17.63	1001	100000
ca[3,5]	2.837	4.956	0.01958	-6.882	2.819	12.59	1001	100000
ca[4,5]	-4.316	6.074	0.02521	-16.3	-4.324	7.609	1001	100000
mu0	33.96	1.844	0.006915	30.34	33.96	37.59	1001	100000
S	16.23	1.27	0.004229	13.98	16.15	18.94	1001	100000
tau	0.003864	5.96E-04	1.97E-06	0.002787	0.003835	0.005117	1001	100000

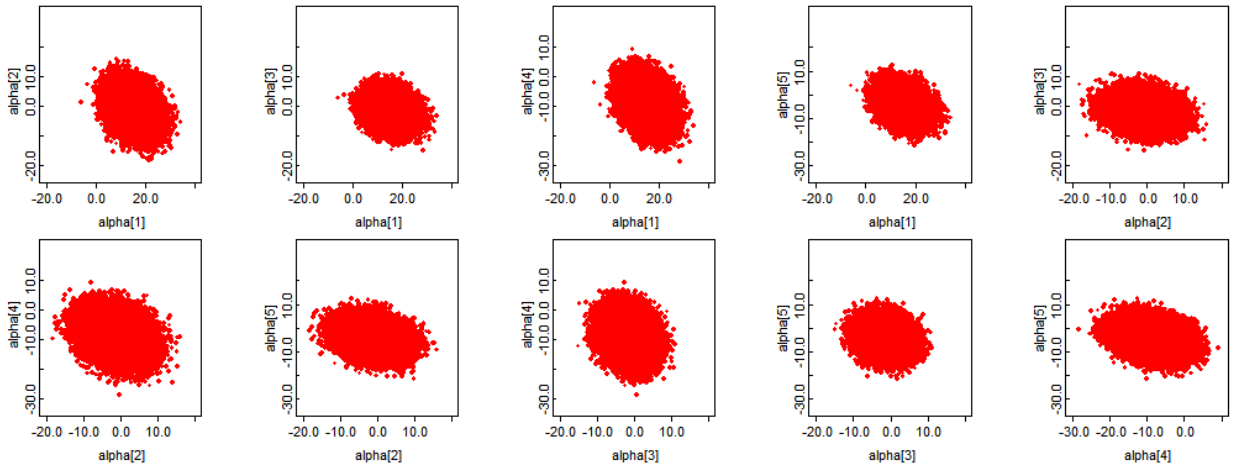

alpha[5] sample: 100000
P(alpha[5])
alpha[5]
ca[1,2] sample: 100000
P(ca[1,2])
ca[1,2]
ca[1,3] sample: 100000
P(ca[1,3])
ca[1,3]
ca[1,4] sample: 100000
P(ca[1,4])
ca[1,4]
ca[1,5] sample: 100000
P(ca[1,5])
ca[1,5]
ca[2,3] sample: 100000
P(ca[2,3])
ca[2,3]
ca[2,4] sample: 100000
P(ca[2,4])
ca[2,4]
ca[2,5] sample: 100000
P(ca[2,5])
ca[2,5]
ca[3,4] sample: 100000
P(ca[3,4])
ca[3,4]
ca[3,5] sample: 100000
P(ca[3,5])
ca[3,5]
ca[4,5] sample: 100000
P(ca[4,5])
ca[4,5]
mu0 sample: 100000
P(mu0)
mu0
s sample: 100000
P(s)
s
tau sample: 100000
P(tau)
tau



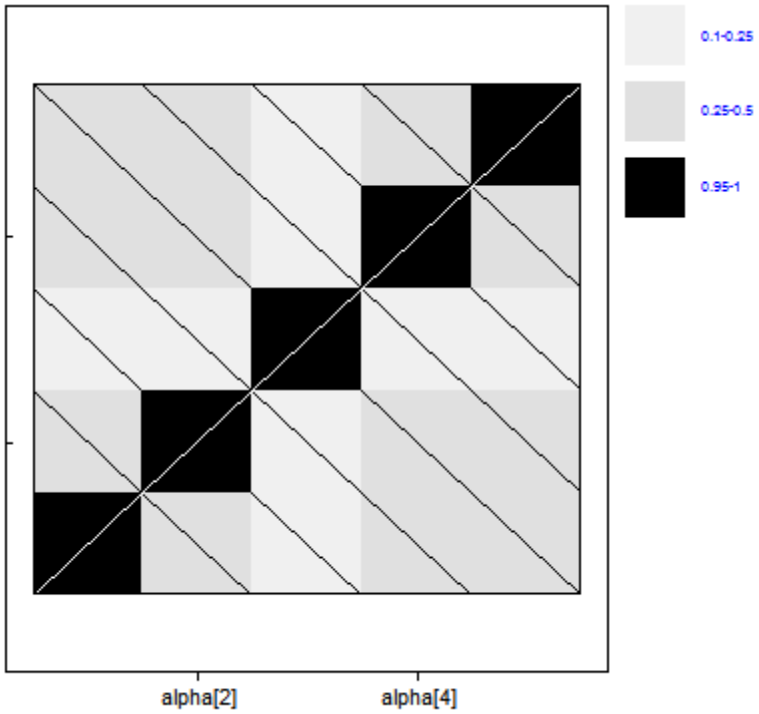
The significantly different types do not have **0** in their credible set.



You can see that always alpha[1] is the highest i.e. that treatment gives the highest mean precision value always. The rest are spread around.



Unlike Prob 1c these correlations are not solely determined by the sum to zero constraints. Below is the same information encoded in matrix form.

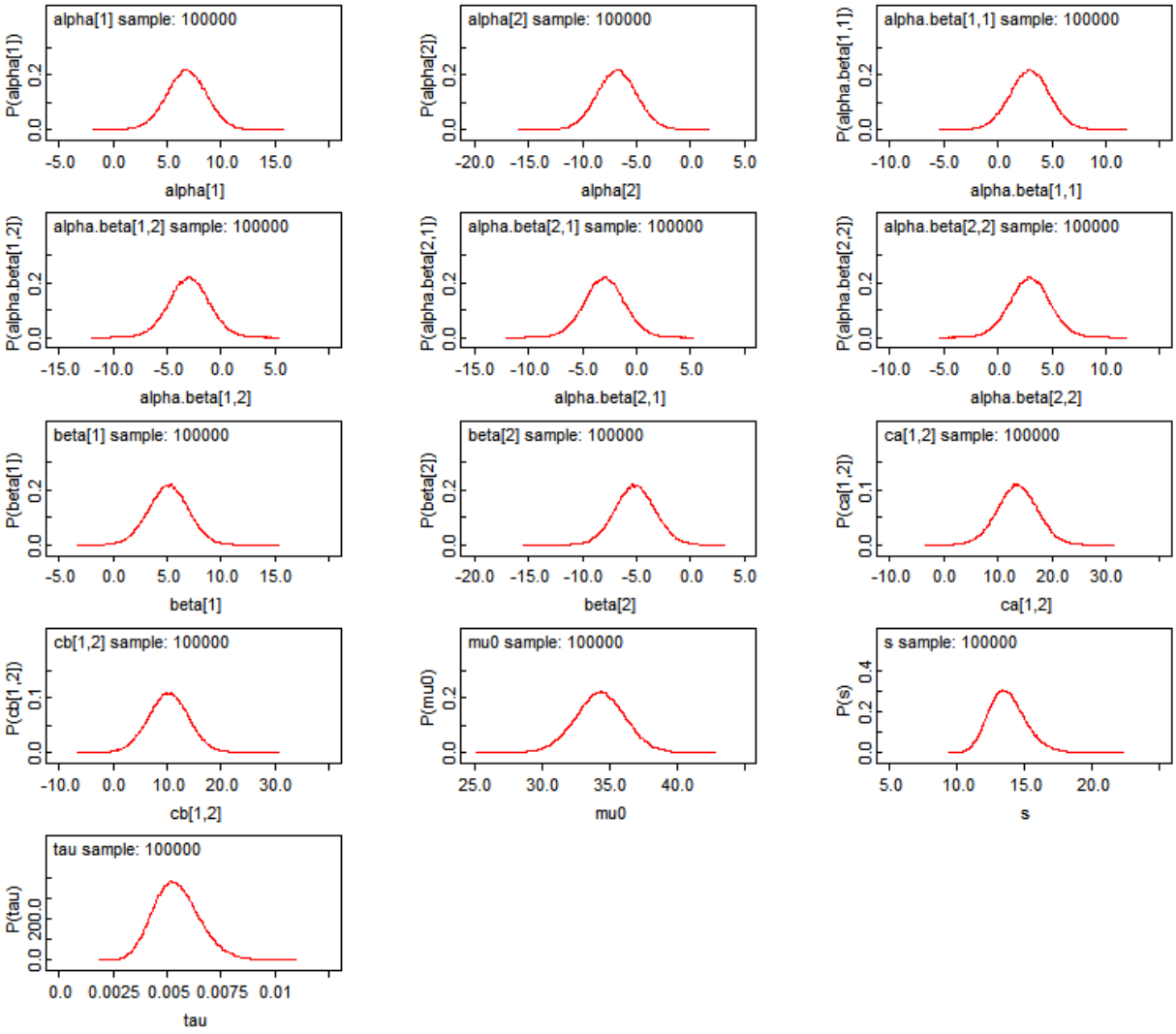


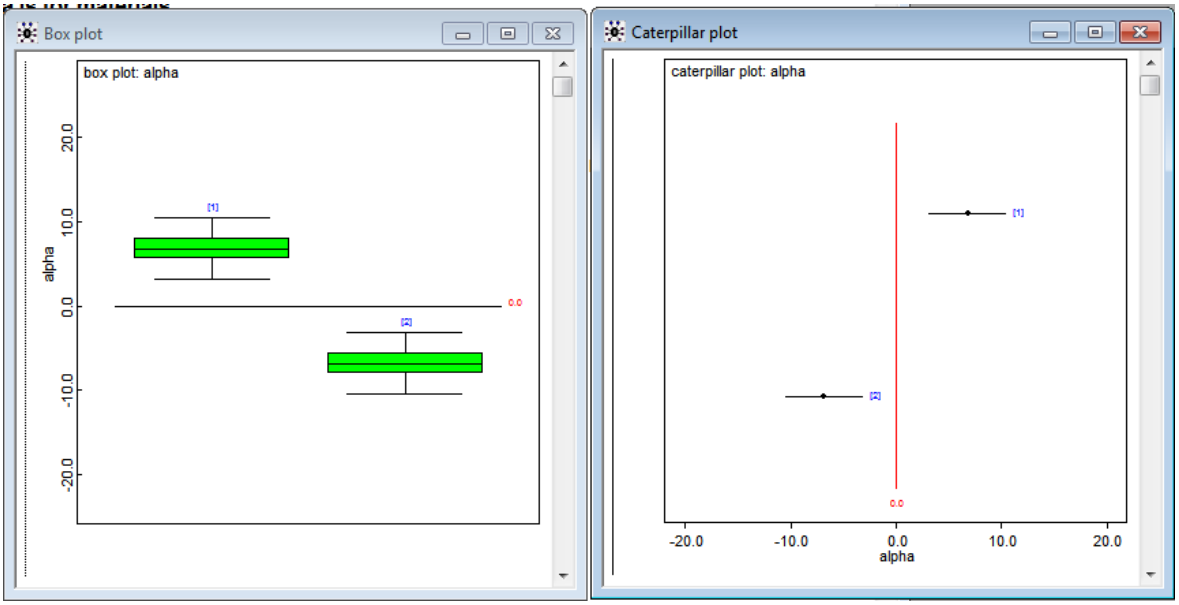
Problem 1c Statistics and graphs

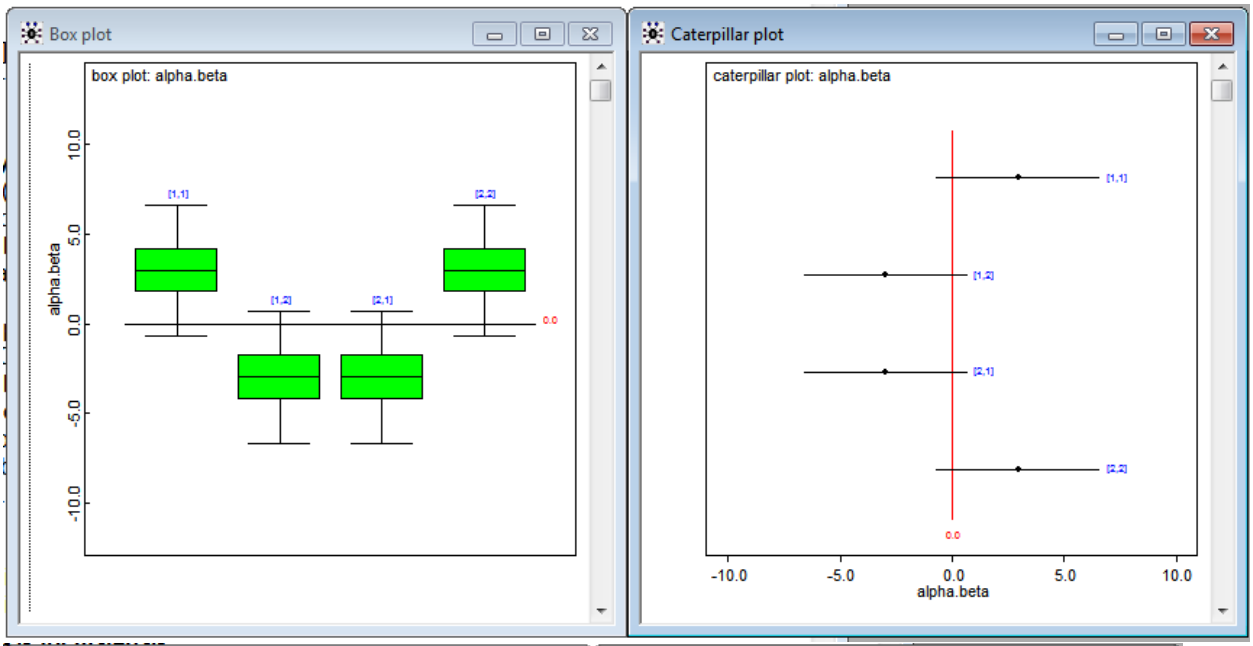
Rows are materials and glass is 1 and steel is 2. Columns are sides 8 is 1 and 12 is 2. Alpha is for materials and beta for sides

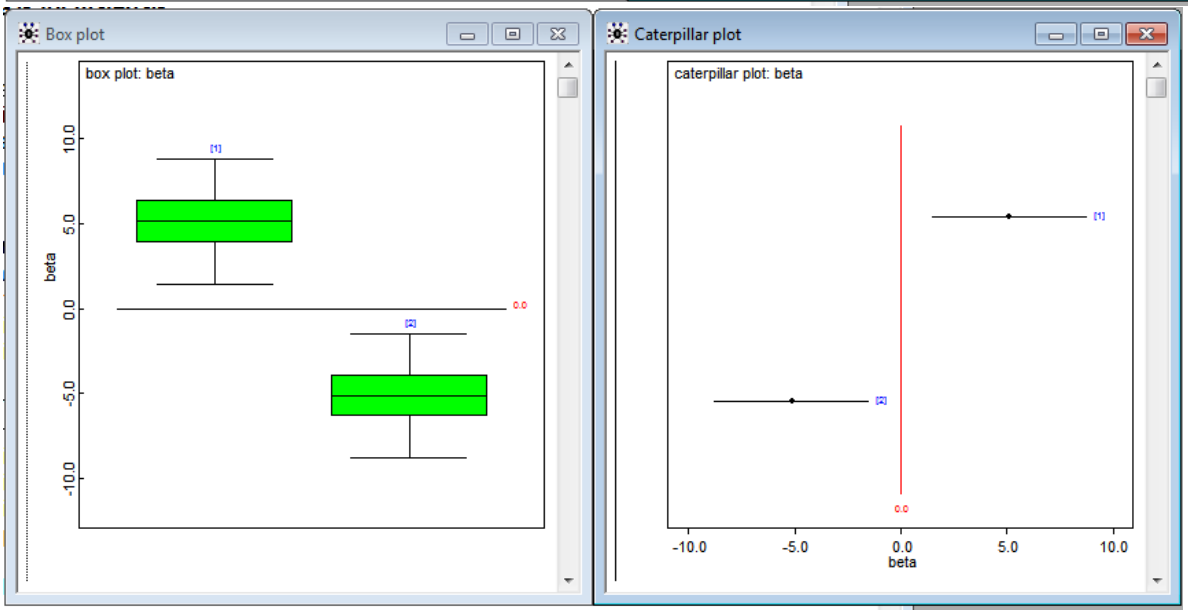
	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha[1]	6.811	1.862	0.005927	3.143	6.808	10.46	1001	100000
alpha[2]	-6.811	1.862	0.005927	-10.46	-6.808	-3.143	1001	100000
alpha.beta[1,1]	2.965	1.86	0.005951	-0.6904	2.966	6.629	1001	100000
alpha.beta[1,2]	-2.965	1.86	0.005951	-6.628	-2.966	0.6907	1001	100000
alpha.beta[2,1]	-2.965	1.86	0.005951	-6.628	-2.966	0.6907	1001	100000
alpha.beta[2,2]	2.965	1.86	0.005951	-0.6904	2.966	6.629	1001	100000
beta[1]	5.129	1.863	0.006065	1.471	5.134	8.794	1001	100000
beta[2]	-5.129	1.863	0.006065	-8.793	-5.133	-1.471	1001	100000
ca[1,2]	13.62	3.724	0.01185	6.287	13.62	20.92	1001	100000
cb[1,2]	10.26	3.726	0.01213	2.942	10.27	17.59	1001	100000
mu0	34.34	1.855	0.005661	30.7	34.34	38.02	1001	100000
tau	0.005458	0.001066	3.74E-06	0.003567	0.005388	0.007742	1001	100000

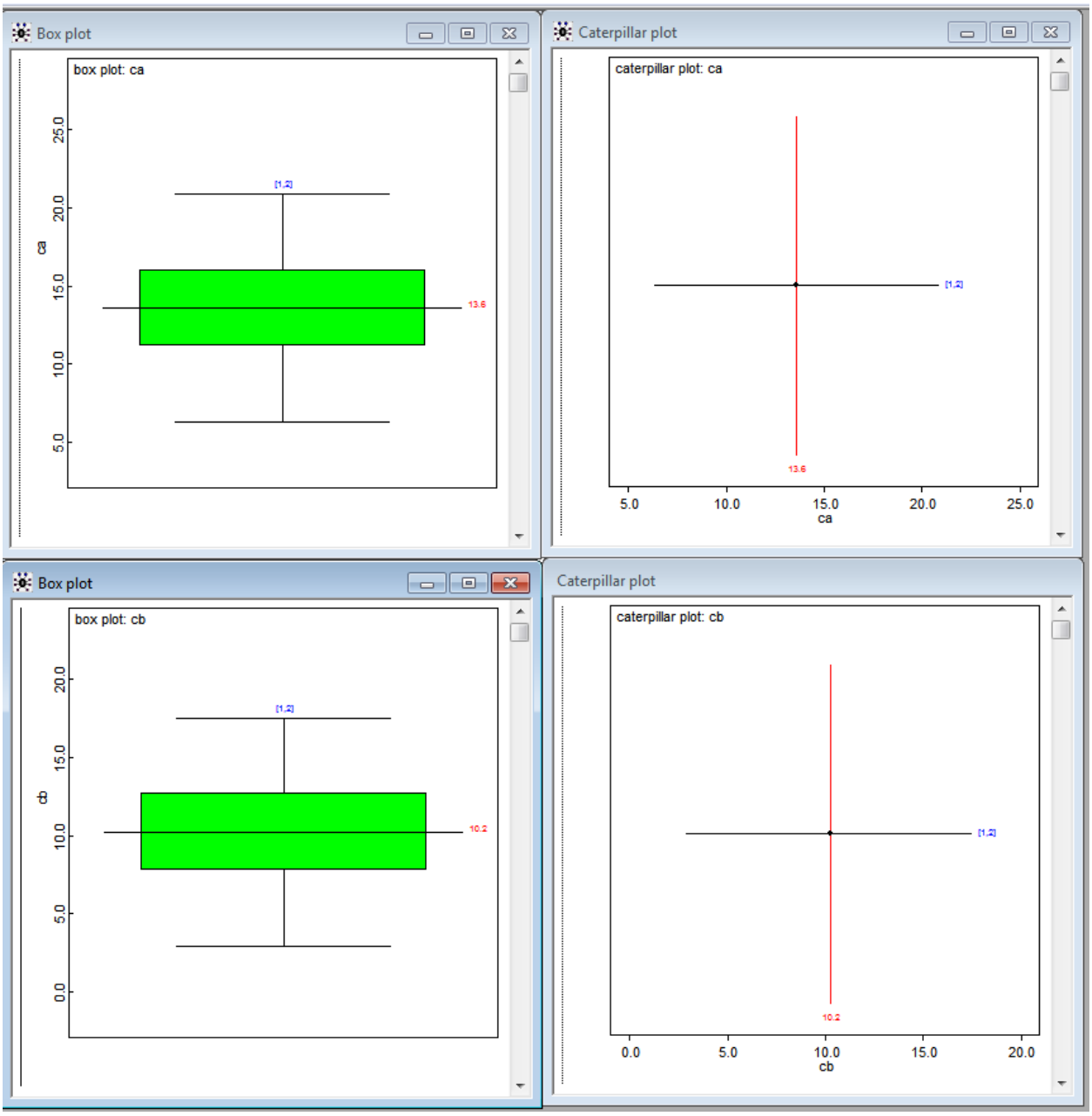
I have included some density graphs,


alpha[1] sample: 100000
P(alpha[1])
alpha[1]
alpha[2] sample: 100000
P(alpha[2])
alpha[2]
alpha.beta[1,1] sample: 100000
P(alpha.beta[1,1])
alpha.beta[1,1]
alpha.beta[1,2] sample: 100000
P(alpha.beta[1,2])
alpha.beta[1,2]
alpha.beta[2,1] sample: 100000
P(alpha.beta[2,1])
alpha.beta[2,1]
alpha.beta[2,2] sample: 100000
P(alpha.beta[2,2])
alpha.beta[2,2]
beta[1] sample: 100000
P(beta[1])
beta[1]
beta[2] sample: 100000
P(beta[2])
beta[2]
ca[1,2] sample: 100000
P(ca[1,2])
ca[1,2]
cb[1,2] sample: 100000
P(cb[1,2])
cb[1,2]
mu0 sample: 100000
P(mu0)
mu0
s sample: 100000
P(s)
s
tau sample: 100000
P(tau)
tau

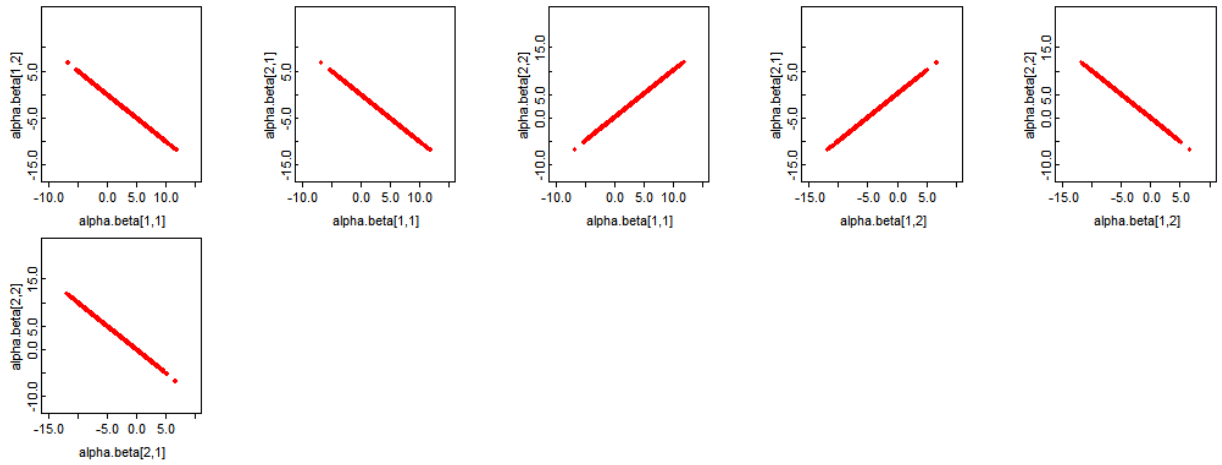

Box plot
box plot: alpha
alpha
Caterpillar plot
caterpillar plot: alpha
alpha

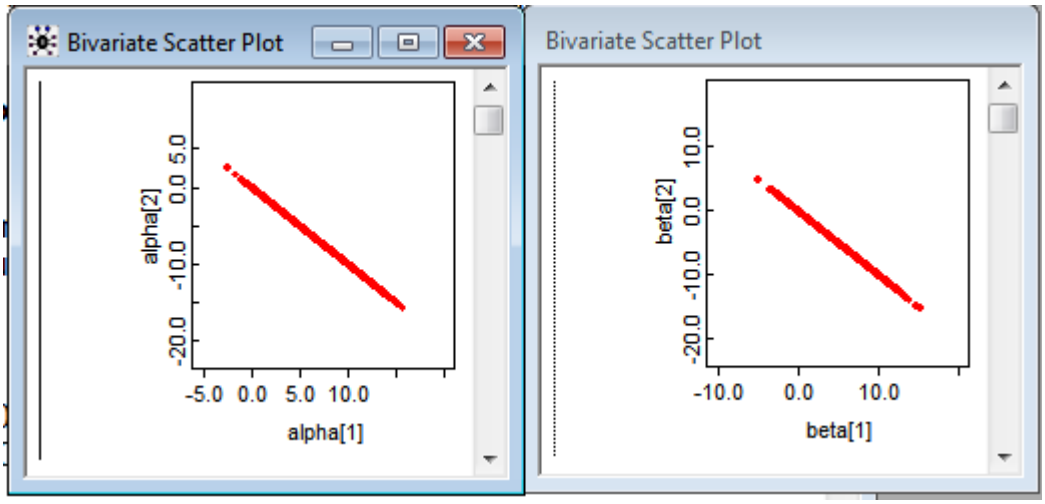

Box plot
box plot: alpha.beta
alpha.beta
10.0
5.0
0.0
-5.0
-10.0
[1,1]
[1,2]
[2,1]
[2,2]
0.0
Caterpillar plot
caterpillar plot: alpha.beta
[1,1]
[1,2]
[2,1]
[2,2]
0.0
-10.0
-5.0
0.0
5.0
10.0
alpha.beta


Box plot
box plot: beta
beta
10.0
5.0
0.0
-5.0
-10.0
[1]
[2]
0.0
Caterpillar plot
caterpillar plot: beta
[1]
[2]
0.0
-10.0
-5.0
0.0
5.0
10.0
beta

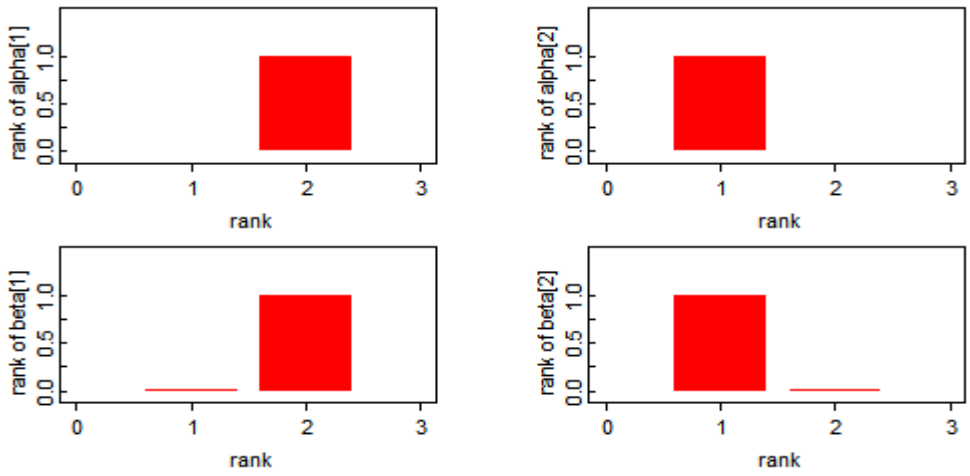


As you can see in ca and cb their credible sets do not include zero and hence the conclusions.





Their relation is along a straight line of slope +1 or -1 passing through zero because of the sum to zero constraints.



Here the rank monitor does not serve any purpose. In problem 3 however it serves a purpose and it will be used there.

Codes

1-way ANOVA

```
# alpha is for type

model{
   for(i in 1:n){
    measurement[i]  ~ dnorm( mu[i], tau )
    mu[i] <-  mu0 + alpha[ type[i] ]
    }

# Apply sum to zero constraints
alpha[1] <- - sum(alpha[2:leva])

# Apply priors
mu0 ~ dnorm(0, 0.0001)
for(a in 2:leva) {
       alpha[a] ~ dnorm(0, 0.0001)
}
tau ~ dgamma(0.0001, 0.0001)
s <- 1/sqrt(tau)

for(i in 1:leva-1)  {
       for(j in i+1:leva) {
              ca[i,j] <- alpha[i]-alpha[j]
       }
}
}

list(mu0=1, alpha=c(NA, 1,1,1,1),tau=1)

list(n=89,leva=5,measurement=c(47,47,38,62,29,59,92,44,41,47,44,41,42,18,36,45,33,30,10,27,18,
27,57,66,48,24,15,3,39,66,21,27,66,27,33,48,15,9,24,3,7,6,39,27,27,42,18,42,37,12,63,69,24,30,
27,63,15,42,42,60,18,9,12,30,30,27,30,39,18,27,48,24,18,30,21,33,18,12,33,24,23,57,39,44,33,30
,24,24,30),
type=c(1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5))
```

2-way ANOVA

```
alpha is for materials
beta is for sides

model{
   for(i in 1:n){
   measurement[i]  ~ dnorm( mu[i], tau )
    mu[i] <-  mu0 + alpha[ material[i] ]  + beta[ sides[i]  ] + alpha.beta[ material[i],
sides[i] ]
    }

# Here we apply the sum to zero contraints on the ANOVA
alpha[1] <- - sum(alpha[2:leva])
beta[1] <- - sum(beta[2:levb])
for(a in 1:leva) {alpha.beta[a,1] <- - sum(alpha.beta[a, 2:levb])}
for(b in  2:levb) {alpha.beta[1,b] <-  - sum(alpha.beta[2:leva, b])}

# Here we apply priors
mu0 ~ dnorm(0, 0.0001)
for(a in 2:leva) {alpha[a] ~ dnorm(0, 0.0001)}
for(b in 2:levb) {beta[b] ~ dnorm(0, 0.0001)}
for(a in 2:leva) {for(b in 2:levb){
      alpha.beta[a,b] ~ dnorm(0, 0.0001) }}
tau ~ dgamma(0.001, 0.001)
s <- 1/sqrt(tau)

for(i in 1:1)  {for(j in i+1:2) {ca[i,j] <- alpha[i]-alpha[j]}}
for(i in 1:1)  {for(j in i+1:2) {cb[i,j] <- beta[i]-beta[j]}}

}


# Data part 1
list(n = 56, leva= 2, levb= 2)

# Initialization
list(mu0=0, alpha=c(NA, 0), beta=c(NA, 0),
alpha.beta = structure(.Data=c(NA, NA, NA, 0),
.Dim=c(2,2)), tau = 1)

# Data part 2 fold below
material []    sides []    measurement []
1       1       47
1       1       47
1       1       38
1       1       62
1       1       29
1       1       59
1       1       92
1       1       44
1       1       41
1       1       47
1       1       44
1       1       41
1       2       42
1       2       18
1       2       36
1       2       45
1       2       33
1       2       30
1       2       10
1       2       27
1       2       18
1       2       27
1       2       57
1       2       66
1       2       48
1       2       24
1       2       15
2       2       18
```

```
2       2       9
2       2       12
2       2       30
2       2       30
2       2       27
2       2       30
2       2       39
2       2       18
2       2       27
2       2       48
2       2       24
2       2       18
2       1       30
2       1       21
2       1       33
2       1       18
2       1       12
2       1       33
2       1       24
2       1       23
2       1       57
2       1       39
2       1       44
2       1       33
2       1       30
2       1       24
2       1       24
2       1       30
END
```

Problem 2 Answers

Since every time I run the code the random values generated are different it may happen that these values below may be different from what is reported when I published the file in MATLAB

Problem 2a Answers

	Mean	Median	Std. dev.	Confidence Interval Lower bound	Confidence Interval Upper bound
μ	10.1187	10.1186	0.2081	9.7094	10.5297
τ	0.4640	0.4581	0.0893	0.3060	0.6550

Problem 2b Answers

Yes, the solution is **different** from Problem 2a.

I verified it using ANOVA on the values of mu and tau from Prob 2a and Prob 2b. Both of them are significantly different. Results are published later.

Using an inverse Gamma prior on $\sigma^2 \sim IGa(2,4)$ is same as $\tau \sim Ga(2,4)$ not $IGa(4,2)$.

	Mean	Median	Std. dev.	Confidence Interval Lower bound	Confidence Interval Upper bound
M	10.1161	10.116	0.1976	9.7294	10.5069
σ^2	2.0087	1.9612	0.3872	1.3883	2.8992
$\tau = 1/\sigma^2$	0.5157	0.5099	0.0958	0.3449	0.7203

Problem 2c Answers

Cauchy prior for μ and Rayleigh prior for τ.

	Mean	Median	Std. dev.	Confidence Interval Lower bound	Confidence Interval Upper bound
μ	10.1218	10.1227	0.2072	10.5310	9.7199
τ	0.4629	0.4578	0.0882	0.6532	0.3063

Prob 2(a)

$\sigma^2 = 1/\tau$ or $\frac{1}{\sigma^2} = \tau$

$$f(y,\mu,\tau) = \frac{(\tau)^{n/2}}{(2\pi)^{n/2}} e^{-\frac{\tau}{2}\sum_{i=1}^{n}(y_i-\mu)^2} \cdot \frac{1}{\sqrt{2\pi}} \cdot \frac{1}{\sqrt{4}} e^{-\frac{1}{8}(\mu-12)^2} \times \frac{4^2}{\Gamma(2)} e^{-4\tau} \cdot \tau$$

$$\propto \tau^{n/2} \cdot e^{-\frac{\tau}{2}\sum_{i=1}^{n}(y_i-\mu)^2} \cdot e^{-\frac{1}{8}(\mu-12)^2} \cdot e^{-4\tau} \cdot \tau$$

$$\pi(\mu|y,\tau) \propto e^{-\frac{\tau}{2}\sum_{i=1}^{n}(y_i-\mu)^2} \cdot e^{-\frac{1}{8}(\mu-12)^2}$$

$$\propto \exp\left(-\frac{\tau}{2}\sum_{i=1}^{n}(y_i-\mu)^2 - \frac{1}{8}(\mu-12)^2\right)$$

$$\propto \exp\left(-\frac{1}{8}\left(4\tau\sum_{i=1}^{n}(y_i-\mu)^2 + (\mu-12)^2\right)\right)$$

$$\propto \exp\left(-\frac{1}{8}\left(4n\tau\mu^2 - 8\tau\mu\sum_{i=1}^{n}y_i + \mu^2 - 24\mu \text{ ---- }\right)\right)$$

$$\propto \exp\left(-\frac{1}{8}\left(\mu^2(4n\tau+1) - 2\mu\left(4\tau\sum_{i=1}^{n}y_i + 12\right) + \text{-------}\right)\right)$$

constants.

$$\pi(\mu|y,\tau) \sim N\left(\frac{4\tau\sum_{i=1}^{n}y_i + 12}{4n\tau+1}, \frac{4}{4n\tau+1}\right) \checkmark$$

$$\pi(\tau|\mu,y) \propto \tau^{(\frac{n}{2}+1)} \cdot e^{-4\tau} \cdot e^{-\frac{\tau}{2}\left(\sum_{i=1}^{n}(y_i-\mu)^2\right)}$$

$$\tau^{(\frac{n}{2}+1)} \cdot e^{-\tau\left(4+\frac{1}{2}\sum_{i=1}^{n}(y_i-\mu)^2\right)}$$

$$\pi(\tau|\mu,y) = Ga\left(\frac{n}{2}+2,\ 4+\frac{1}{2}\sum_{i=1}^{n}(y_i-\mu)^2\right)$$

Prob 2(b)

$$IG(\alpha,\beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{-\alpha-1} e^{-\beta/x}. \qquad \tau^2 = \xi_q.$$

$$\therefore f(\mu,\xi_q,\tau) = \frac{1}{(2\pi)^{n/2}\,\xi_q^{n/2}} \cdot e^{-\frac{1}{2\xi_q}\left(\sum_{i=1}^{n}(y_i-\mu)^2\right)} \cdot \frac{1}{\sqrt{2\pi}\sqrt{4}} e^{-\frac{1}{8}(\mu-12)^2} \cdot \frac{2^4}{\Gamma(4)} \xi_q^{-5} \cdot e^{-2/\xi_q}.$$

$$\pi(\mu|\xi_q,\tau) \propto \exp\left(-\frac{1}{2\xi_q}\left(\sum_{i=1}^{n}(y_i-\mu)^2\right) - \frac{1}{8}(\mu-12)^2\right)$$

$$\propto \exp\left(-\frac{1}{8\xi_q}\left(4\left[\sum_{i=1}^{n}(y_i-\mu)^2\right] + \xi_q(\mu-12)^2\right)\right)$$

$$\propto \exp\left(-\frac{1}{8\xi_q}\left(4n\mu^2 - 8\mu\sum_{i=1}^{n} y_i + \xi_q\mu^2 - 24\mu\xi_q + \cdots\right)\right)$$

$$\propto \exp\left(-\frac{1}{8\xi_q}\left(\mu^2(4n+\xi_q) - 2\mu\left(4\sum_{i=1}^{n} y_i + 12\xi_q\right) + \cdots\right)\right)$$

$$= \mathcal{N}\left(\frac{4\sum_{i=1}^{n} y_i + 12\xi_q}{4n+\xi_q}, \frac{4\xi_q}{4n+\xi_q}\right)$$

$$\pi(\xi_q|\mu,\tau) \propto \xi_q^{-5} \cdot \xi_q^{-n/2} \cdot e^{-2/\xi_q} \cdot e^{-\frac{1}{2\xi_q}\left(\sum_{i=1}^{n}(y_i-\mu)^2\right)}$$

$$\propto \xi_q^{-(4+\frac{n}{2})-1} \cdot e^{-\frac{1}{\xi_q}\left(2+\frac{1}{2}\left(\sum_{i=1}^{n}(y_i-\mu)^2\right)\right)}$$

$$\sim IG\left(4+\frac{n}{2},\ 2+\frac{1}{2}\left(\sum_{i=1}^{n}(y_i-\mu)^2\right)\right)$$

Prob2 codes – sub functions used

I have attached the codes and their results published but they do not publish some of the smaller functions that I wrote. I am posting only those here.

calculate_CI

Calculate the 95% confidence interval of a set of values.

```
function [CI_upper,CI_lower] = calculate_CI(data)
    no_of_data_points = numel(data);
    data = sort(data);
    no_lower = round(no_of_data_points*0.025);
    no_higher = round(no_of_data_points*0.975);
    CI_lower = data(no_lower);
    CI_upper = data(no_higher);
End
```

inv_gamma_pdf

Generate the inverse gamma pdf at the points specified in *pts* for parameters *alpha* and *beta*

```
function [pdf_values] = inv_gam_pdf(pts,r_shape,lambda_rate)
    pdf_values = (lambda_rate^r_shape)/(gamma(r_shape)) * pts.^(-r_shape-1) .* exp(-
lambda_rate./pts);
end
```

inv_gamma_rnd

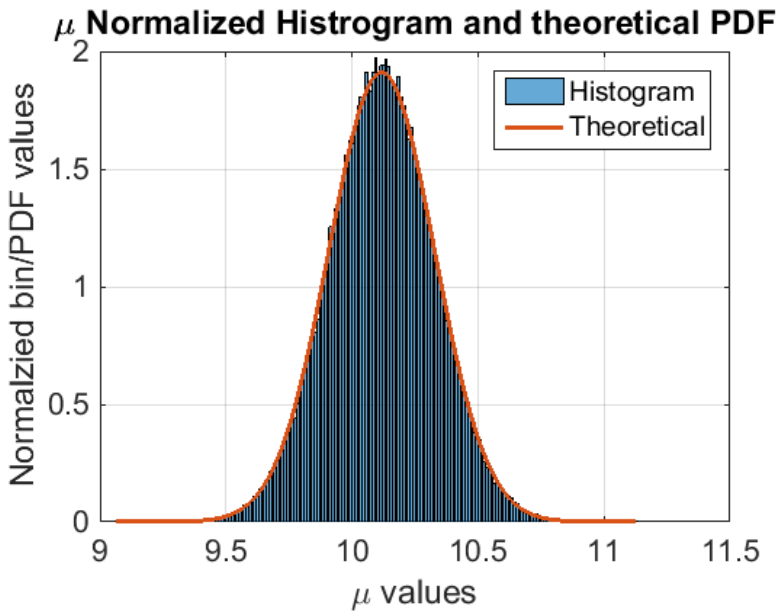
Generate a an inverse gamma random variable by inverse CDF sampling method with parameters *alpha* and *beta*.

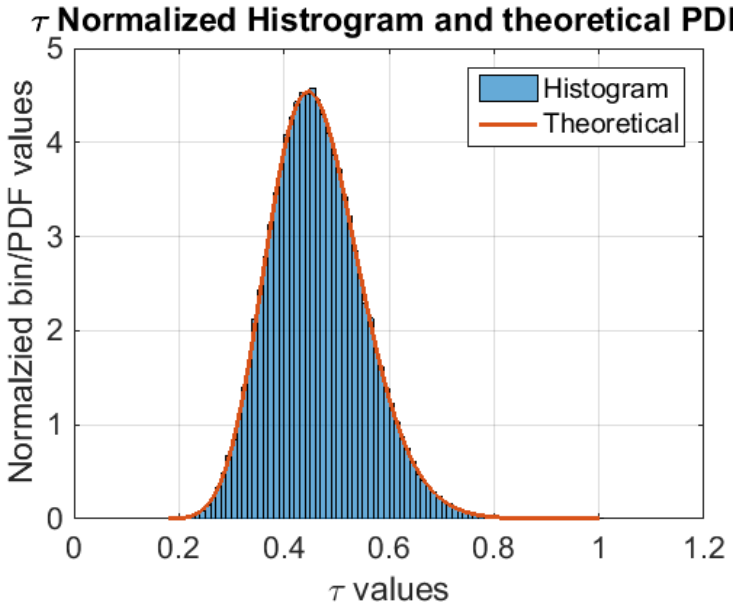
```
function [op] = inv_gam_rnd(alpha,beta)
    op = beta/gammaincinv(rand(),alpha,'upper');
end
```

```
function [mu_rel,tau_rel] = Prob2a()
    clc
    clear
    close all

    load('data_amanita.mat');
    data = data_amanita;
    n = length(data);
    iters = 201000;
    burn = 1000;
    mu = 100;
    tau = 0.01;
    mu_vals = mu;
    tau_vals = tau;

    for indx=1:1:iters
        mu_prev = mu(end);
        tau_prev = tau(end);

        % The conditional of mu given data and Sigma square is a normal
        % distribution with the parameters as computed below. For a
        % simplifcation as to how this is obtained I have attached a scanned
        % copy of the sheet where I simplify this
        norm_mean = (4*tau*sum(data)+12)/(1+4*tau_prev*n);
        norm_var = 4/(1+4*tau_prev*n);

        % The conditional of tau given data and mu is a is a
        % Gamma distribution. For a simplifcation as to how this is obtained I
        % have attached a scanned copy of the sheet where I simplify this.

        % There is a gamma random number generator in matlab, so I use that.
        % Also the gamrnd and gampdf in MATLAB employ the shape and scale
        % parameters as inputs so I invert the rate parameter to get the scale
        % parameter.
        gamma_rate = (n/2)+2;
        data_cent = data-mu_prev;
        data_cent_sum_sq = sum(data_cent.^2);
        gamma_shape = 4 + 0.5*data_cent_sum_sq;

        mu = sqrt(norm_var)*randn(1,1)+norm_mean;
        tau = gamrnd(gamma_rate,1/gamma_shape);
        mu_vals = [mu_vals,mu];
        tau_vals = [tau_vals,tau];

    end

    % Here I am bruning the first 1000 values and calculating the staistics
    % based on the next 100000

    % First, I do this for mu
    mu_rel = mu_vals(burn+1:end);
    mean_mu = mean(mu_rel);
    std_dev_mu = std(mu_rel);
    median_mu = median(mu_rel);

    % Second, I do this for tau
    tau_rel = tau_vals(burn+1:end);
    mean_tau = mean(tau_rel);
    std_dev_tau = std(tau_rel);
    median_tau = median(tau_rel);

    % Calculating the 95% equal tail confidence intervals
    [CI_upper_mu,CI_lower_mu] = calculate_CI(mu_rel);
    [CI_upper_tau,CI_lower_tau] = calculate_CI(tau_rel);

    % I plot the histogram for mu values
    % The mu should be a normal distribution and I plot the theoretical
    % distribution based on the mean and standard deviation calculated
    % empirically.
    histogram(mu_rel,'BinWidth',0.01,'Normalization','pdf');
    hold on
    plot(min(mu_rel):0.01:max(mu_rel),normpdf(min(mu_rel):0.01:max(mu_rel),mean_mu,std_dev_mu),'LineWidth',2);
    title('\mu Normalized Histrogram and theoretical PDF');
    legend('Histogram','Theoretical');
    xlabel('\mu values');
    ylabel('Normalzied bin/PDF values');
    grid on
    hold off

    % Here I plot the histogram for tau values
    % Tau is a gamma distribution and I plot the theoretical
    % distribution based on the parameters calculated empirically.
```

```
    figure
    rate_tau = mean_tau/(std_dev_tau)^2;
    shape_tau = mean_tau*rate_tau;
    histogram(tau_rel,'BinWidth',0.01,'Normalization','pdf');
    hold on
    plot(min(tau_rel):0.01:max(tau_rel),gampdf(min(tau_rel):0.01:max(tau_rel),shape_tau,1/rate_tau),'LineWidth',2);
    title('\tau Normalized Histrogram and theoretical PDF');
    legend('Histogram','Theoretical');
    xlabel('\tau values');
    ylabel('Normalzied bin/PDF values');
    grid on
    hold off

    fprintf('Mean of mu is %0.4f, median of mu is %0.4f and std dev is %0.4f\n',mean_mu,median_mu,std_dev_mu);
    fprintf('For the equal tail 95percent confidence interval of mu, the upper bound is %0.4f and lower bound is %0.4f\n\n',CI_upper_mu,CI_lower_mu);

    fprintf('Mean of tau is %0.4f, meadian of tau is %0.4f and std dev is %0.4f\n',mean_tau,median_tau,std_dev_tau);
    fprintf('For the equal tail 95percent confidence interval of tau, the upper bound is %0.4f and lower bound is %0.4f\n',CI_upper_tau,CI_lower_tau);
end
```

```
Mean of mu is 10.1179, median of mu is 10.1176 and std dev is 0.2088
For the equal tail 95percent confidence interval of mu, the upper bound is 10.5289 and lower bound is 9.7073

Mean of tau is 0.4634, meadian of tau is 0.4576 and std dev is 0.0892
For the equal tail 95percent confidence interval of tau, the upper bound is 0.6533 and lower bound is 0.3052
```



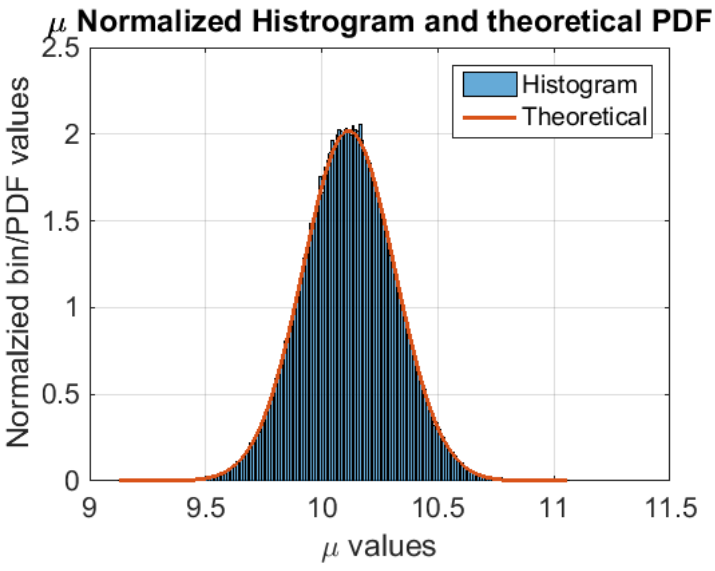


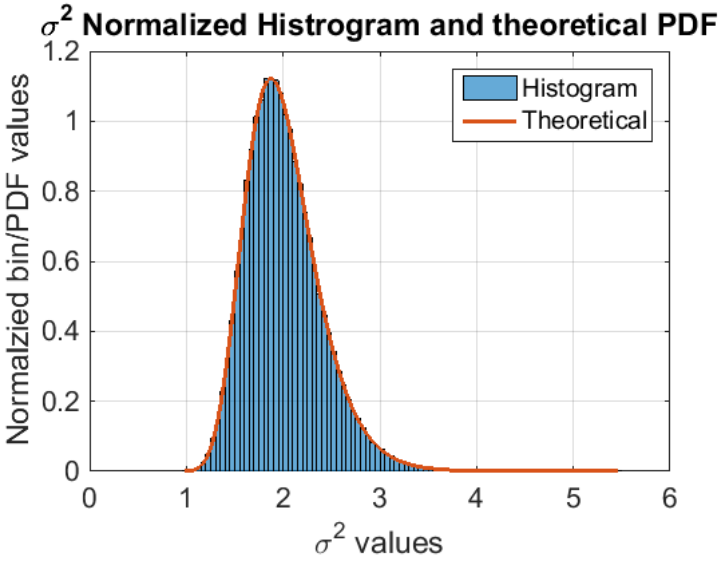
Published with MATLAB® R2014b

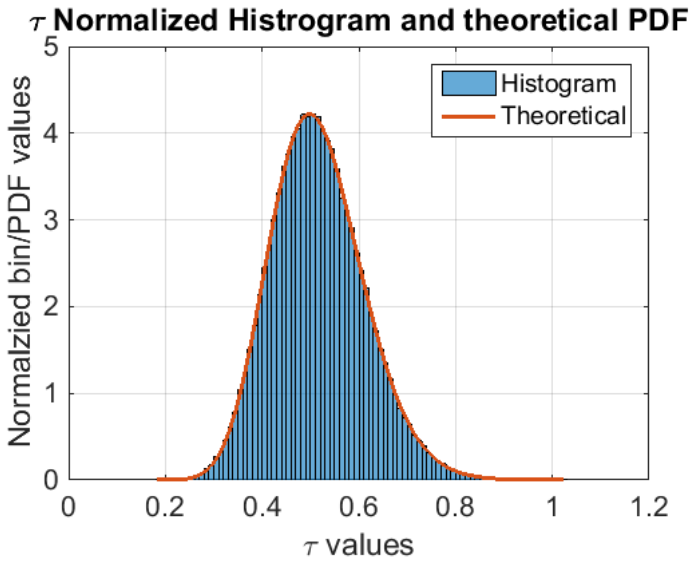
```
function [mu_rel,tau_rel] = Prob2b()
    clc
    clear
    close all

    load('data_amanita.mat');
    data = data_amanita;
    n = length(data);
    iters = 201000;
    burn = 1000;
    mu = 100;
    sigma_sq = 0.01;
    mu_vals = mu;
    sigma_sq_vals = sigma_sq;



    for indx=1:1:iters
        mu_prev = mu(end);
        sigma_sq_prev = sigma_sq(end);

        % The conditional mu given data and Sigma square is a normal
        % distribution with the parameters as computed below. For a
        % simplifcation as to how this is obtained I have attached a scanned
        % copy of the sheet where I simplify this
        norm_mean = (4*sum(data)+12*sigma_sq_prev)/(sigma_sq_prev+4*n);
        norm_var = 4*sigma_sq_prev/(sigma_sq_prev+4*n);

        % The conditional Sigma square given data and mu is a is an inverse
        % Gamma distribution. For a simplifcation as to how this is obtained I
        % have attached a scanned copy of the sheet where I simplify this.

        % There is no inverse gamma random number generator in matlab, so I
        % employ the gamma random number generator and simply invert the value.
        % Also the gamrnd and gampdf in MATLAB employ the shape and scale
        % parameters as inputs so I invert the rate parameter to get the scale
        % parameter.
        inv_gamma_shape = (n/2)+4;
        data_cent = data-mu_prev;
        data_cent_sum_sq = sum(data_cent.^2);
        inv_gamma_rate = 2 + 0.5*data_cent_sum_sq;

        mu = sqrt(norm_var)*randn(1,1)+norm_mean;
%         sigma_sq = 1/gamrnd(inv_gamma_shape,1/inv_gamma_rate);
        sigma_sq = inv_gam_rnd(inv_gamma_shape,inv_gamma_rate);
        mu_vals = [mu_vals,mu];
        sigma_sq_vals = [sigma_sq_vals,sigma_sq];

    end

    % Here I am bruning the first 1000 values and calculating the staistics
    % based on the next 100000

    % First, I do this for mu
    mu_rel = mu_vals(burn+1:end);
    mean_mu = mean(mu_rel);
    std_dev_mu = std(mu_rel);
    median_mu = median(mu_rel);

    % Second, I do this for sigma square
    sigma_sq_rel = sigma_sq_vals(burn+1:end);
    mean_sigma_sq = mean(sigma_sq_rel);
    std_dev_sigma_sq = std(sigma_sq_rel);
    median_sigma_sq = median(sigma_sq_rel);

    % Calculating the 95% equal tail confidence intervals
    [CI_upper_mu,CI_lower_mu] = calculate_CI(mu_rel);
    [CI_upper_sigma_sq,CI_lower_sigma_sq] = calculate_CI(sigma_sq_rel);


    % I plot the histogram for mu values
    % The mu should be a normal distribution and I plot the theoretical
    % distribution based on the mean and standard deviation calculated
    % empirically.
    histogram(mu_rel,'BinWidth',0.01,'Normalization','pdf');
    hold on
    plot(min(mu_rel):0.01:max(mu_rel),normpdf(min(mu_rel):0.01:max(mu_rel),mean_mu,std_dev_mu),'LineWidth',2);
    title('\mu Normalized Histrogram and theoretical PDF');
    legend('Histogram','Theoretical');
    xlabel('\mu values');
    ylabel('Normalzied bin/PDF values');
    grid on
    hold off

    % Here I plot the histogram for sigma square values
    % Sigma square should be a inverse-gamma distribution and I plot the theoretical
    % distribution based on the parameters calculated empirically.
    figure
    % I use the equations provided in Chapter 5 of Statbook to get the
```

```
    % following equations for parameter from the mean and standard deviation.
    % As there is no inverse gamma pdf inbuit in MATLAB i wrote a small snippet
    % of code to do the same called as inv_gam_pdf function. I have submitted
    % the same.
    inv_gamma_shape_tau = (mean_sigma_sq/std_dev_sigma_sq)^2 + 2;
    inv_gamma_rate_tau = std_dev_sigma_sq * (inv_gamma_shape_tau-1)* sqrt(inv_gamma_shape_tau - 2);
    histogram(sigma_sq_rel,'BinWidth',0.05,'Normalization','pdf');
    hold on
    plot(min(sigma_sq_rel):0.01:max(sigma_sq_rel),inv_gam_pdf(min(sigma_sq_rel):0.01:max(sigma_sq_rel),inv_gamma_shape_tau,inv_gamma_rate_tau),'LineWidth',2);
    title('\sigma^2 Normalized Histrogram and theoretical PDF');
    xlabel('\sigma^2 values');
    ylabel('Normalzied bin/PDF values');
    legend('Histogram','Theoretical');
    grid on
    hold off

    % I just convert the sigma square values back to tau values by inverting them
    % and see whether they fit a gamma distribution
    tau_rel = 1./sigma_sq_rel;

    mean_tau = mean(tau_rel);
    std_dev_tau = std(tau_rel);
    median_tau = median(tau_rel);
    [CI_upper_tau,CI_lower_tau] = calculate_CI(tau_rel);

    rate_tau = mean_tau/(std_dev_tau)^2;
    shape_tau = mean_tau*rate_tau;
    figure
    histogram(tau_rel,'BinWidth',0.01,'Normalization','pdf');
    hold on
    plot(min(tau_rel):0.01:max(tau_rel),gampdf(min(tau_rel):0.01:max(tau_rel),shape_tau,1/rate_tau),'LineWidth',2);
    title('\tau Normalized Histrogram and theoretical PDF');
    legend('Histogram','Theoretical');
    xlabel('\tau values');
    ylabel('Normalzied bin/PDF values');
    grid on
    hold off


    % Printing out the values of relevant parameters
    fprintf('Mean of mu is %0.4f, median of mu is %0.4f and std dev is %0.4f\n',mean_mu,median_mu,std_dev_mu);
    fprintf('For the equal tail 95percent confidence interval of mu, the upper bound is %0.4f and lower bound is %0.4f\n\n',CI_upper_mu,CI_lower_mu);

    fprintf('Mean of sigma square is %0.4f, meadian of sigma square is %0.4f and std dev is %0.4f\n',mean_sigma_sq,median_sigma_sq,std_dev_sigma_sq);
    fprintf('For the equal tail 95percent confidence interval of sigma square, the upper bound is %0.4f and lower bound is %0.4f\n\n',CI_upper_sigma_sq,CI_lower_sig

    fprintf('(1/Sigma_square) = Tau, hence I just invert the numbers\n');
    fprintf('Mean of tau is %0.4f, median of tau is %0.4f and std dev is %0.4f\n',mean_tau,median_tau,std_dev_tau);
    fprintf('For the equal tail 95percent confidence interval of sigma square, the upper bound is %0.4f and lower bound is %0.4f\n',CI_upper_tau,CI_lower_tau);
end
```

```
Mean of mu is 10.1165, median of mu is 10.1160 and std dev is 0.1976
For the equal tail 95percent confidence interval of mu, the upper bound is 10.5069 and lower bound is 9.7294

Mean of sigma square is 2.0087, meadian of sigma square is 1.9612 and std dev is 0.3872
For the equal tail 95percent confidence interval of sigma square, the upper bound is 2.8992 and lower bound is 1.3883

(1/Sigma_square) = Tau, hence I just invert the numbers
Mean of tau is 0.5157, median of tau is 0.5099 and std dev is 0.0958
For the equal tail 95percent confidence interval of sigma square, the upper bound is 0.7203 and lower bound is 0.3449
```







Published with MATLAB® R2014b

```
clc
clear
close all

% Here I use ANOVA to compare if the two values of mu and tau gotten by fitting a
% Gamma prior of Ga(2,4) to tau directly to it in Prob(a) and got indirectly by
% inverting the sigma sqaure values which have an Inverse gamma prior  of
% IGa (4,2) as in Prob2(b) are statistically same or not. For this I
% generate both and run an ANOVA
close all
fprintf('-----------------------------------------------------------------\n');
fprintf('These are Prob2a prints ignore\n');
[mu_rel_a,tau_rel_a] = Prob2a;
fprintf('These are Prob2a prints ignore\n');
fprintf('-----------------------------------------------------------------\n\n');
clc
close all
fprintf('-----------------------------------------------------------------\n');
fprintf('These are Prob2b prints ignore\n');
[mu_rel_b,tau_rel_b] = Prob2b;
fprintf('These are Prob2b prints ignore\n');
fprintf('-----------------------------------------------------------------\n\n');
clc
close all
```

```
-----------------------------------------------------------------
These are Prob2a prints ignore
Mean of mu is 10.1185, median of mu is 10.1185 and std dev is 0.2077
For the equal tail 95percent confidence interval of mu, the upper bound is 10.5299 and lower bound is 9.7123

Mean of tau is 0.4634, meadian of tau is 0.4577 and std dev is 0.0891
For the equal tail 95percent confidence interval of tau, the upper bound is 0.6541 and lower bound is 0.3051
These are Prob2a prints ignore
-----------------------------------------------------------------

-----------------------------------------------------------------
These are Prob2b prints ignore
Mean of mu is 10.1157, median of mu is 10.1150 and std dev is 0.1978
For the equal tail 95percent confidence interval of mu, the upper bound is 10.5056 and lower bound is 9.7293

Mean of sigma square is 2.0083, meadian of sigma square is 1.9633 and std dev is 0.3855
For the equal tail 95percent confidence interval of sigma square, the upper bound is 2.8948 and lower bound is 1.3905

(1/Sigma_square) = Tau, hence I just invert the numbers
Mean of tau is 0.5157, median of tau is 0.5093 and std dev is 0.0958
For the equal tail 95percent confidence interval of sigma square, the upper bound is 0.7192 and lower bound is 0.3454
These are Prob2b prints ignore
-----------------------------------------------------------------
```

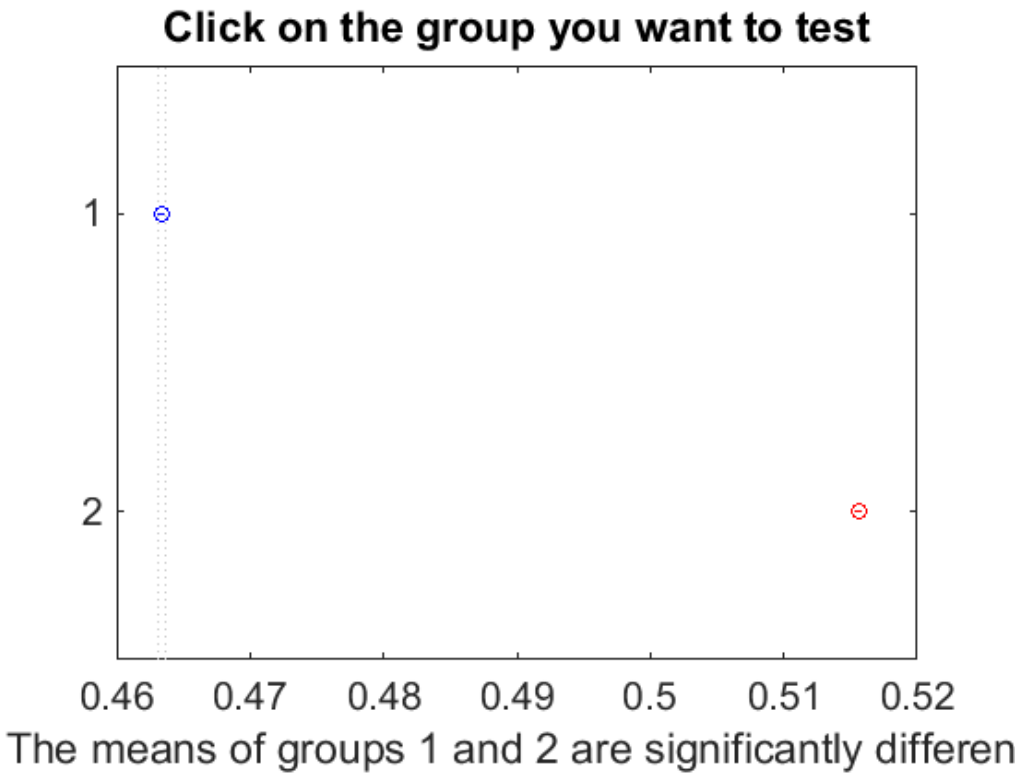
```
if isrow(tau_rel_a)
    tau_rel_a = tau_rel_a';
end
type_a = ones(size(tau_rel_a));
if isrow(tau_rel_b)
    tau_rel_b = tau_rel_b';
end
type_b = 2*ones(size(tau_rel_b));
tau_vals = [tau_rel_a;tau_rel_b];
types = [type_a;type_b];
% Tau from problem a is type 1 and tau from prob b is type 2
```

```
% I then run an ANOVA on them
[p,atab,stats] = anova1(tau_vals,types);
signifcance_of_mean = multcompare(stats);
fprintf('Below is the results of multcompare\n');
disp(signifcance_of_mean);
fprintf('\n');
% I store the lower bounds of the confidence intervals
lower_bound = signifcance_of_mean(3);
upper_bound = signifcance_of_mean(5);
if ((lower_bound < 0 && upper_bound < 0) || (lower_bound > 0 && upper_bound > 0))
    fprintf('The tau values are significantly different\n');
else
    fprintf('The tau values are not significantly different\n');
end
```

```
Below is the results of multcompare
    1.0000    2.0000   -0.0528   -0.0523   -0.0517    0.0000


The tau values are significantly different
```

ANOVA Table

Source	SS	df	MS	F	Prob>F
Groups	273.11	1	273.106	31924.26	0
Error	3421.93	400000	0.009		
Total	3695.03	400001			

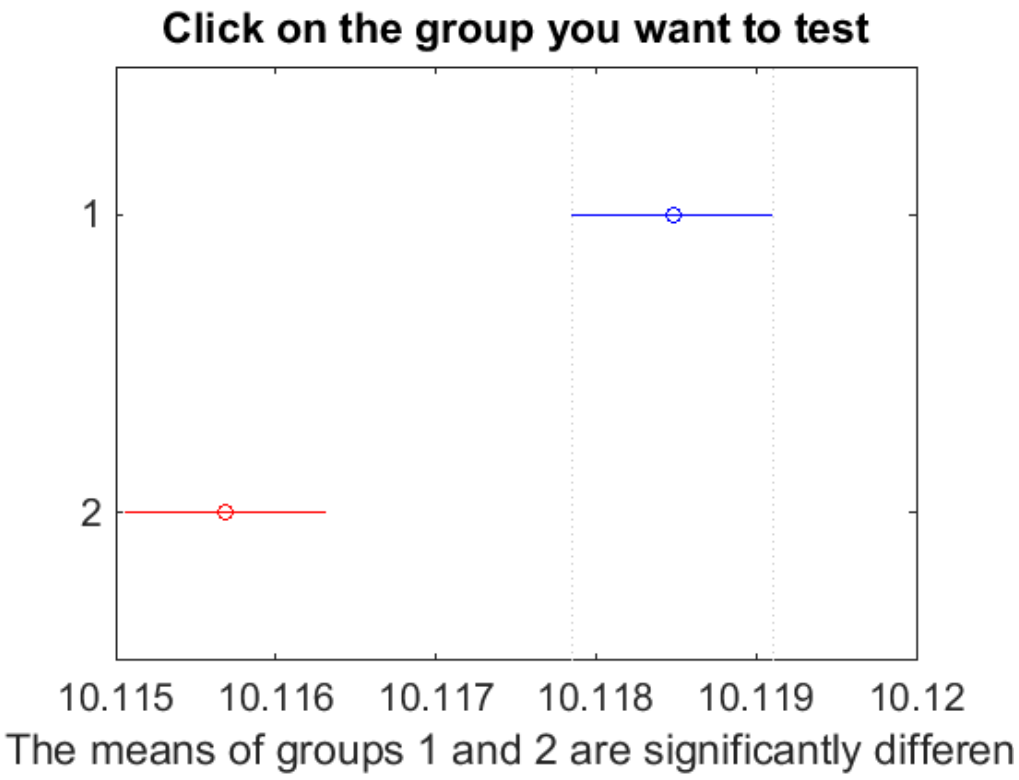



```
if isrow(mu_rel_a)
    mu_rel_a = mu_rel_a';
end
type_a = ones(size(mu_rel_a));
if isrow(mu_rel_b)
    mu_rel_b = mu_rel_b';
end
type_b = 2*ones(size(mu_rel_b));
mu_vals = [mu_rel_a;mu_rel_b];
types_mu = [type_a;type_b];
% Mu from problem a is type 1 and mu from prob b is type 2
% I then run an ANOVA on them
[p,atab,stats] = anova1(mu_vals,types_mu);
signifcance_of_mean = multcompare(stats);
fprintf('Below is the results of multcompare\n');
disp(signifcance_of_mean);
fprintf('\n');
% I store the lower bounds of the confidence intervals
lower_bound = signifcance_of_mean(3);
upper_bound = signifcance_of_mean(5);
if ((lower_bound < 0 && upper_bound < 0) || (lower_bound > 0 && upper_bound > 0))
    fprintf('The mu values are significantly different\n');
else
    fprintf('The mu values are not significantly different\n');
end
```

```
Below is the results of multcompare
    1.0000    2.0000    0.0015    0.0028    0.0041    0.0000


The mu values are significantly different
```

ANOVA Table

Source	SS	df	MS	F	Prob>F
Groups	0.8	1	0.7805	18.98	1.32063e-05
Error	16448.1	400000	0.04112		
Total	16448.9	400001			

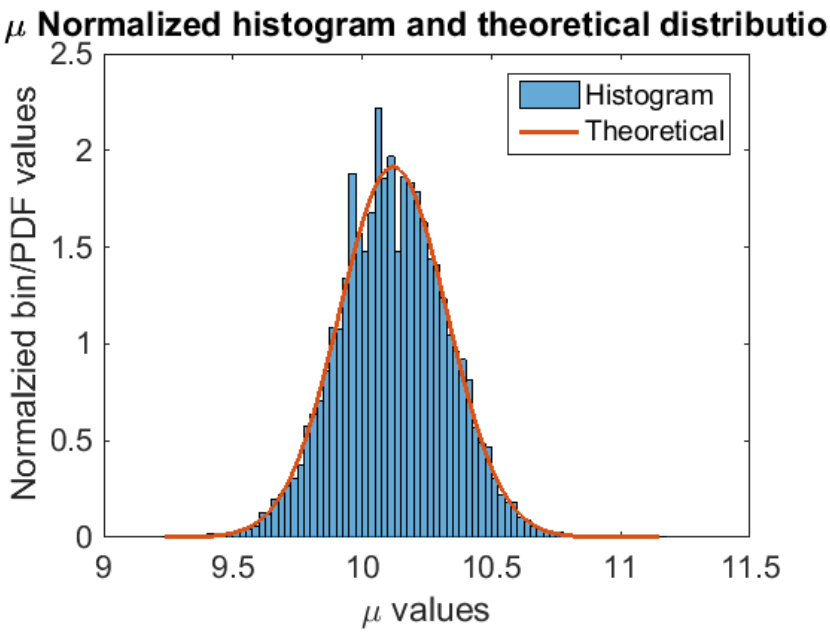


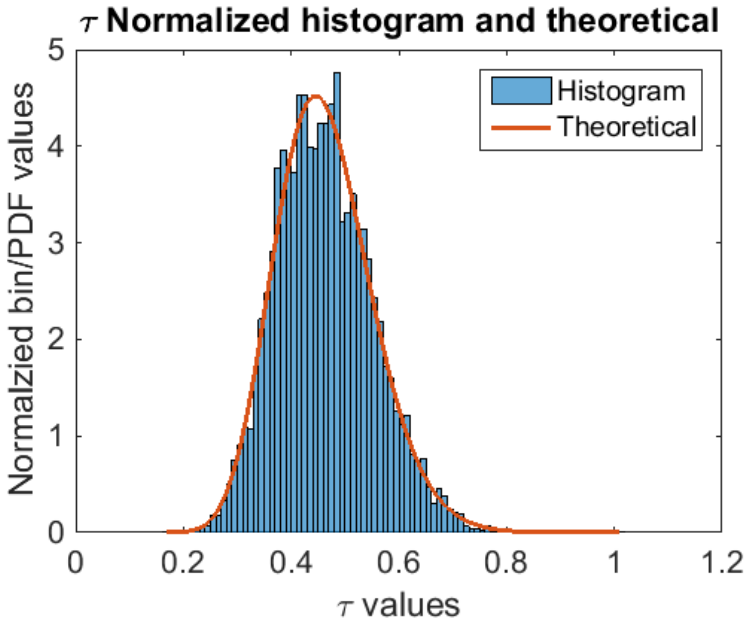
Published with MATLAB® R2014b

```
clc
clear
close all

load('data_amanita.mat');
data = data_amanita;
n = length(data);

mu = 8;
tau = 4;
burn = 1000;
no_simuls = 501000;
mu_vals = -100*ones(1,no_simuls); %generate a matrix because matrix concatenation slows down loop
tau_vals = -100*ones(1,no_simuls); %generate a matrix because matrix concatenation slows down loop
s = 4;

no_of_accepts = 0;
pi_muprop_tauprop = 0;
pi_mu_tau = 0;


for indx=1:1:no_simuls
    mu_prop = mu + 5*tan(pi*(rand(1,1)-0.5)); % Used the Cauchy as proposal for mu
    tau_prop = raylrnd(tau); % Used rayleigh as proposal for tau

    pi_muprop_tauprop = prod(normpdf(data,mu_prop,sqrt(1/tau_prop)))*normpdf(mu_prop,12,sqrt(4))*gampdf(tau_prop,2,0.25);
    pi_mu_tau = prod(normpdf(data,mu,sqrt(1/tau)))*normpdf(mu,12,sqrt(4))*gampdf(tau,2,0.25);

    pdf_tau_given_tau_prop = raylpdf(tau,tau_prop);
    pdf_tau_prop_given_tau = raylpdf(tau_prop,tau);

    % Since proposal for mu depends on (mu-mu_prop)^2 exchanging it does not
    % matter and hence during evaluation we do not factor that in
    num = pdf_tau_given_tau_prop*pi_muprop_tauprop;
    den = pdf_tau_prop_given_tau*pi_mu_tau;

    rho = min([num/den,1]); %Probability of acceptance for this iteration
    if rand(1,1) <= rho
        mu = mu_prop;
        tau = tau_prop;
        no_of_accepts = no_of_accepts + 1;
    end
    mu_vals(indx) = mu;
    tau_vals(indx) = tau;
end

% Here I am bruning the first 1000 values and calculating the staistics
% based on the next 100000

% First, I do this for mu
mu_rel = mu_vals(burn+1:end);
post_mu_mean = mean(mu_rel);
post_mu_std_dev = std(mu_rel);
post_mu_median = median(mu_rel);

% Second I do this for tau
tau_rel = tau_vals(burn+1:end);
post_tau_mean = mean(tau_rel);
post_tau_std_dev = std(tau_rel);
post_tau_median = median(tau_rel);

% Calculating the 95% equal tail confidence intervals
[CI_mu_upper,CI_mu_lower] = calculate_CI(mu_rel);
[CI_tau_upper,CI_tau_lower] = calculate_CI(tau_rel);

% Printing out the values
fprintf('Mean of mu is %0.4f, median of mu is %0.4f and std dev is %0.4f\n',post_mu_mean,post_mu_median,post_mu_std_dev);
fprintf('For the equal tail 95percent confidence interval of mu, the upper bound is %0.4f and lower bound is %0.4f\n\n',CI_mu_upper,CI_mu_lower);

fprintf('Mean of tau is %0.4f, meadian of tau is %0.4f and std dev is %0.4f\n',post_tau_mean,post_tau_median,post_tau_std_dev);
fprintf('For the equal tail 95percent confidence interval of tau, the upper bound is %0.4f and lower bound is %0.4f\n',CI_tau_upper,CI_tau_lower);

% I plot the histogram for mu values
% The mu should be a normal distribution and I plot the theoretical
% distribution based on the mean and standard deviation calculated
% empirically.
histogram(mu_rel,'BinWidth',0.025,'Normalization','pdf');
hold on
plot(min(mu_rel):0.01:max(mu_rel),normpdf(min(mu_rel):0.01:max(mu_rel),post_mu_mean,post_mu_std_dev),'LineWidth',2);
legend('Histogram','Theoretical');
```

```
title('\mu Normalized histogram and theoretical distribution')
xlabel('\mu values');
ylabel('Normalzied bin/PDF values');
hold off

% Here I plot the histogram for tau values
% Tau should be a gamma distribution and I plot the theoretical
% distribution based on the parameters calculated empirically.
rate_tau = post_tau_mean/(post_tau_std_dev)^2;
shape_tau = post_tau_mean*rate_tau;
figure
histogram(tau_rel,'BinWidth',0.01,'Normalization','pdf');
hold on
plot(min(tau_rel):0.01:max(tau_rel),gampdf(min(tau_rel):0.01:max(tau_rel),shape_tau,1/rate_tau),'LineWidth',2);
legend('Histogram','Theoretical');
title('\tau Normalized histogram and theoretical')
xlabel('\tau values');
ylabel('Normalzied bin/PDF values');
hold off
```

```
Mean of mu is 10.1198, median of mu is 10.1154 and std dev is 0.2082
For the equal tail 95percent confidence interval of mu, the upper bound is 10.5246 and lower bound is 9.7093

Mean of tau is 0.4628, meadian of tau is 0.4575 and std dev is 0.0895
For the equal tail 95percent confidence interval of tau, the upper bound is 0.6522 and lower bound is 0.3043
```





Published with MATLAB® R2014b

Problem3 Answers

Problem 3a – answers

I did this problem with both normal and Zellner's g-prior

Normal Prior

The model is

$mu[i] = b0 + b1 * x_1[i] + b2 * x_2[i] + b3 * x_3[i] + b4 * x_4[i] + b5 * x_5[i]$ for the ith observation

a. The node info gave a value of 0.9669. The Bayesian R^2 is a random variable because it depends on τ through SSE and it was found to have a mean of 0.9561, median of 0.9301 and a 95% credible set of [0.9091,0.9792].
b. Estimate for x1[11]:
   The mean is 1.397, median is 1.368 and the 95% credible set is [0.629,2.271]
c. Estimate for y[21]:
   The mean is 9.329, median is 9.327 and the 95% credible set is [7.514,11.15]

Zellners Prior

The model is

$mu[i] = b[1] + b[2] * x_1[i] + b[3] * x_2[i] + b[4] * x_3[i] + b[5] * x_4[i] + b[6] * x_5[i]$ for the ith observation

a. The node info gave a value of 0.8964. The Bayesian R^2 is a random variable because it depends on τ through SSE and it was found to have a mean of 0.8653, median of 0.8744 and a 95% credible set of [0.7481,0.9285].
b. Estimate for x1[11]:
   The mean is 1.5, median is 1.501 and the 95% credible set is [0.5158,2.48]
c. Estimate for y[21]:
   The mean is 8.739, median is 8.738 and the 95% credible set is [5.683,11.78]

Problem 3b – answers

1. The 95% credible set assuming normal prior is [-30.68,38.29]
2. The 95% credible set assuming Zellner's g-prior is [-53.06,59.99]

Problem 3c answers

Note: I have estimated x1[11] and y[21] for each model separately. This can be seen in the statistics. All models gave estimated values close to each other.

Assuming normal prior I would recommend the model in which x1 is eliminated

Assuming Zellner's g-prior with g=22, I would recommend using the model in which x3 is eliminated

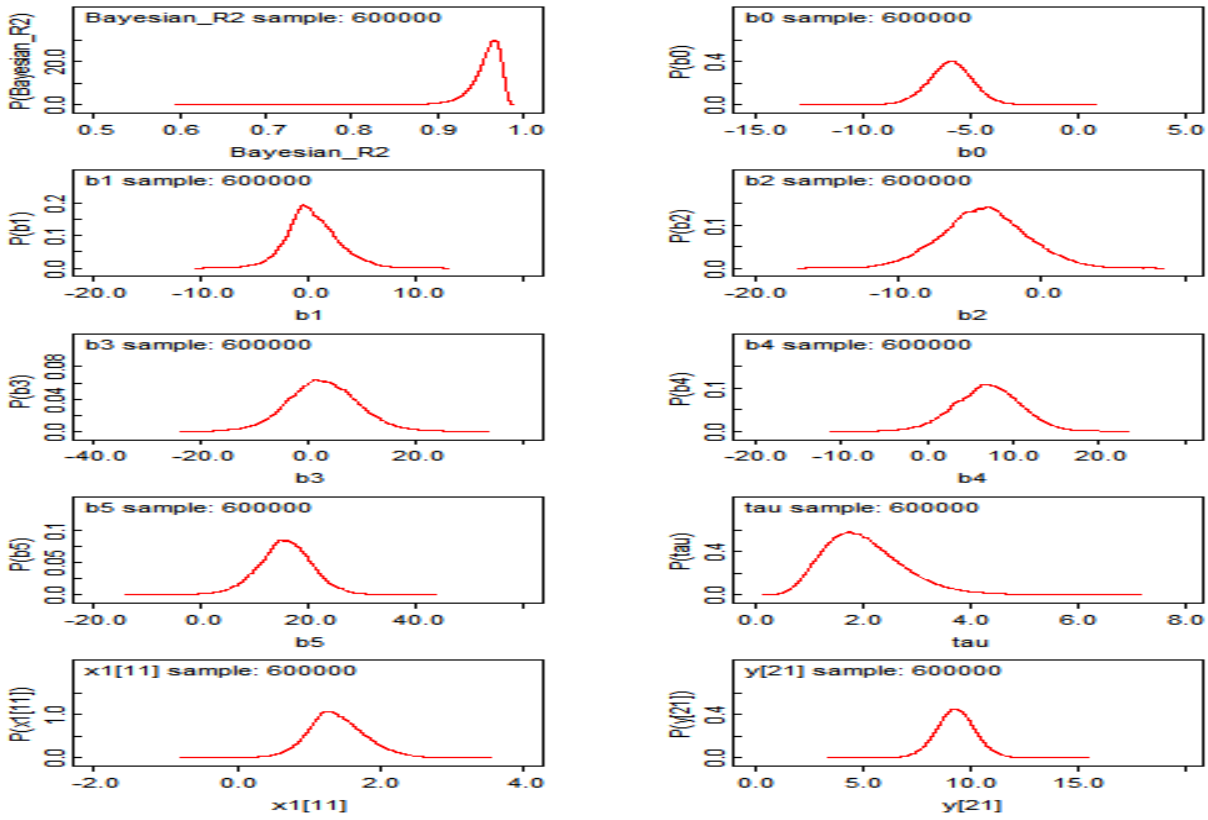
Prob3 Statistics and densities(for 3a) or rank graphs (for 3c)

Problem 3a

Normal prior

Model is $mu[i] = b0 + b1 * x_1[i] + b2 * x_2[i] + b3 * x_3[i] + b4 * x_4[i] + b5 * x_5[i]$ for the ith observation

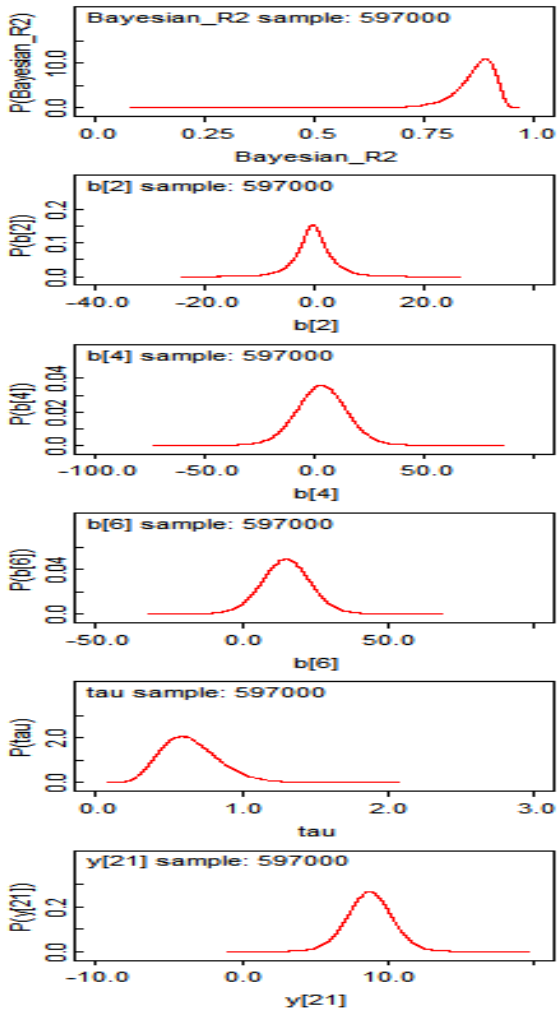
	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Bayesian_R2	0.9561	0.01855	2.04E-04	0.9091	0.9601	0.9792	1001	600000
b0	-5.917	1.055	0.02393	-8.044	-5.903	-3.871	1001	600000
b1	0.3612	2.433	0.06797	-4.203	0.1645	5.611	1001	600000
b2	-4.037	3.011	0.08656	-9.995	-4.038	1.933	1001	600000
b3	2.697	6.541	0.185	-10	2.547	15.92	1001	600000
b4	7.025	3.878	0.1157	-0.7051	7.061	14.54	1001	600000
b5	15.6	5.055	0.1343	5.364	15.67	25.47	1001	600000
tau	2.024	0.7349	0.007468	0.8522	1.934	3.7	1001	600000
x1[11]	1.397	0.4097	0.004344	0.629	1.368	2.271	1001	600000
y[21]	9.329	0.9181	0.01102	7.514	9.327	11.15	1001	600000



Zellner's prior

Model is $mu[i] = b[1] + b[2] * x_1[i] + b[3] * x_2[i] + b[4] * x_3[i] + b[5] * x_4[i] + b[6] * x_5[i]$ for the i^{th} observation

	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Bayesian_R2	0.8653	0.04713	5.73E-05	0.7481	0.8744	0.9285	4001	597000
b[1]	-5.408	1.697	0.002246	-8.772	-5.409	-2.038	4001	597000
b[2]	-0.2438	3.467	0.004778	-7.39	-0.3114	7.188	4001	597000
b[3]	-4.147	5.086	0.006518	-14.22	-4.153	5.936	4001	597000
b[4]	3.056	11.55	0.0152	-19.72	3.046	25.94	4001	597000
b[5]	7.449	6.211	0.008183	-4.89	7.455	19.74	4001	597000
b[6]	15.09	8.408	0.01101	-1.52	15.1	31.69	4001	597000
sigma_sq	1.705	0.5863	7.09E-04	0.913	1.592	3.156	4001	597000
tau	0.6486	0.2003	1.95E-04	0.3168	0.6282	1.095	4001	597000
x[11,2]	1.5	0.5012	7.10E-04	0.5158	1.501	2.48	4001	597000
y[21]	8.739	1.543	0.002053	5.683	8.738	11.78	4001	597000



Problem 3b

Nothing here

Problem 3c

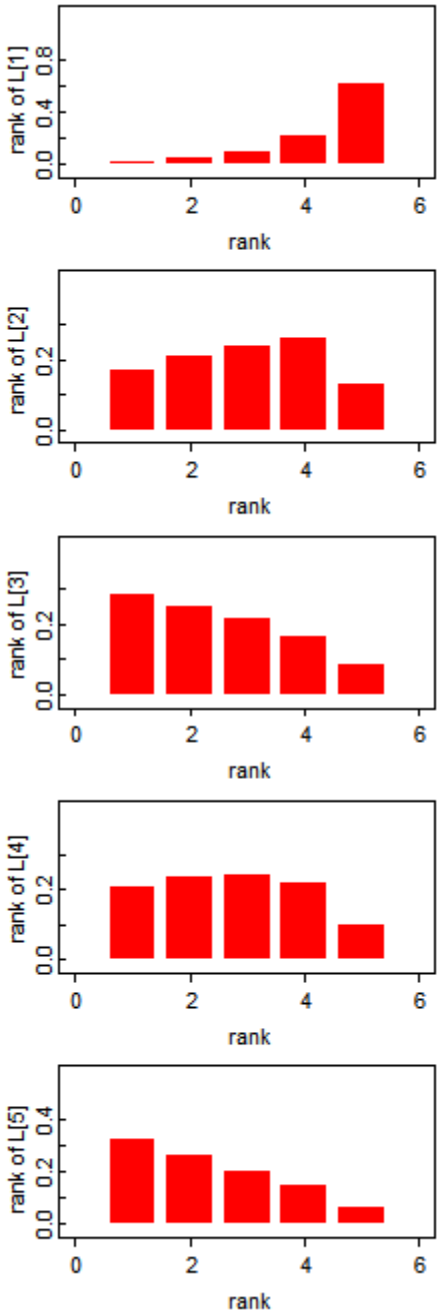
Normal prior

Model No.	Model equation	Missing
Model 1	$mu[i] = a[1] + a[2] * x_1[i] + a[3] * x_2[i] + a[4] * x_3[i] + a[5] * x_4[i]$	x_5
Model 2	$mu[i] = b[1] + b[2] * x_1[i] + b[3] * x_2[i] + b[4] * x_3[i] + b[5] * x_5[i]$	x_4
Model 3	$mu[i] = c[1] + c[2] * x_1[i] + c[3] * x_2[i] + c[4] * x_4[i] + c[5] * x_5[i]$	x_3
Model 4	$mu[i] = d[1] + d[2] * x_1[i] + d[3] * x_3[i] + d[4] * x_4[i] + d[5] * x_5[i]$	x_2
Model 5	$mu[i] = e[0] + e[2] * x_2[i] + e[3] * x_3[i] + e[4] * x_4[i] + e[5] * x_5[i]$	x_1

Statistics:

Node	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Comp[1,2]	0.1995	0.3996	0.00249	0	0	1	1001	600000
Comp[1,3]	0.1524	0.3594	0.002966	0	0	1	1001	600000
Comp[1,4]	0.1662	0.3723	0.002586	0	0	1	1001	600000
Comp[1,5]	0.1196	0.3245	0.002318	0	0	1	1001	600000
Comp[2,3]	0.4012	0.4901	0.003588	0	0	1	1001	600000
Comp[2,4]	0.4473	0.4972	0.00324	0	0	1	1001	600000
Comp[2,5]	0.3631	0.4809	0.003268	0	0	1	1001	600000
Comp[3,4]	0.5483	0.4977	0.003604	0	1	1	1001	600000
Comp[3,5]	0.4629	0.4986	0.003661	0	0	1	1001	600000
Comp[4,5]	0.4143	0.4926	0.003288	0	0	1	1001	600000
L[1]	7.244	1.188	0.01219	5.403	7.089	9.955	1001	600000
L[2]	6.079	0.8741	0.007524	4.732	5.958	8.137	1001	600000
L[3]	5.807	0.8836	0.01081	4.531	5.675	7.83	1001	600000
L[4]	5.971	1.455	0.03118	4.636	5.804	7.967	1001	600000
L[5]	5.7	0.8638	0.01131	4.487	5.576	7.59	1001	600000
Y[1,21]	9.063	1.196	0.008988	6.668	9.073	11.4	1001	600000
Y[2,21]	9.845	0.9396	0.006242	7.972	9.849	11.7	1001	600000
Y[3,21]	9.555	0.93	0.009175	7.728	9.549	11.42	1001	600000
Y[4,21]	9.543	0.9842	0.009253	7.672	9.543	11.4	1001	600000
Y[5,21]	9.281	0.8399	0.004095	7.625	9.281	10.94	1001	600000
a[1]	-7.499	1.039	0.01065	-9.523	-7.512	-5.407	1001	600000
a[2]	0.4384	2.634	0.05646	-4.716	0.3304	5.865	1001	600000
a[3]	-7.217	3.777	0.08804	-14.45	-7.174	-0.09135	1001	600000
a[4]	18.3	7.019	0.1502	4.559	18.33	32.24	1001	600000
a[5]	12.34	3.438	0.07609	5.552	12.38	18.7	1001	600000
b[1]	-6.615	1.037	0.01709	-8.655	-6.62	-4.554	1001	600000
b[2]	3.121	2.334	0.05149	-1.202	3.089	7.701	1001	600000
b[3]	-1.085	2.798	0.06311	-6.542	-1.127	4.575	1001	600000
b[4]	2.346	7.139	0.1637	-11.85	2.478	16.46	1001	600000
b[5]	13.73	5.293	0.1108	3.349	13.66	24.29	1001	600000
c[1]	-6.387	0.9419	0.01587	-8.306	-6.359	-4.616	1001	600000
c[2]	1.645	2.58	0.06117	-2.965	1.463	7.2	1001	600000
c[3]	-2.021	2.748	0.06342	-7.086	-2.115	3.55	1001	600000
c[4]	5.819	3.997	0.09338	-2.249	5.873	13.52	1001	600000
c[5]	0.746	315.8	0.4246	-618.8	1.043	619.2	1001	600000
d[1]	-6.113	1.396	0.02997	-8.275	-6.075	-3.942	1001	600000
d[2]	0.2375	2.753	0.06354	-4.693	0.08591	5.354	1001	600000
d[3]	-1.257	8.756	0.2168	-13.87	-1.608	11.92	1001	600000
d[4]	4.117	3.584	0.08187	-2.53	4.173	10.81	1001	600000
d[5]	15.24	7.386	0.1773	4.526	15.57	25.63	1001	600000
e[1]	-5.937	0.876	0.01469	-7.605	-5.931	-4.259	1001	600000

e[2]	-4.065	2.985	0.07179	-9.71	-4.109	1.819	1001	600000
e[3]	3.427	6.993	0.1689	-10.14	3.371	16.6	1001	600000
e[4]	7.302	3.074	0.06862	1.414	7.288	13.36	1001	600000
e[5]	15.49	4.843	0.1061	6.369	15.55	24.66	1001	600000
tau[1]	1.118	0.4109	0.003714	0.4742	1.065	2.067	1001	600000
tau[2]	1.768	0.6351	0.004927	0.7559	1.691	3.218	1001	600000
tau[3]	2.001	0.7401	0.006987	0.821	1.911	3.688	1001	600000
tau[4]	1.884	0.6772	0.005571	0.7894	1.808	3.415	1001	600000
tau[5]	2.101	0.7499	0.006347	0.8828	2.017	3.798	1001	600000
x11[11]	1.395	0.4231	0.003392	0.6043	1.367	2.288	1001	600000
x12[11]	1.26	0.3087	0.002005	0.6984	1.232	2.016	1001	600000
x13[11]	1.312	0.3882	0.0038	0.6152	1.263	2.201	1001	600000
x14[11]	1.403	0.4047	0.003498	0.6404	1.376	2.267	1001	600000



L[5] has the highest probability of being the lowest which is also the same conclusion we come at by looking at the Comp values.

Zellner's g-prior

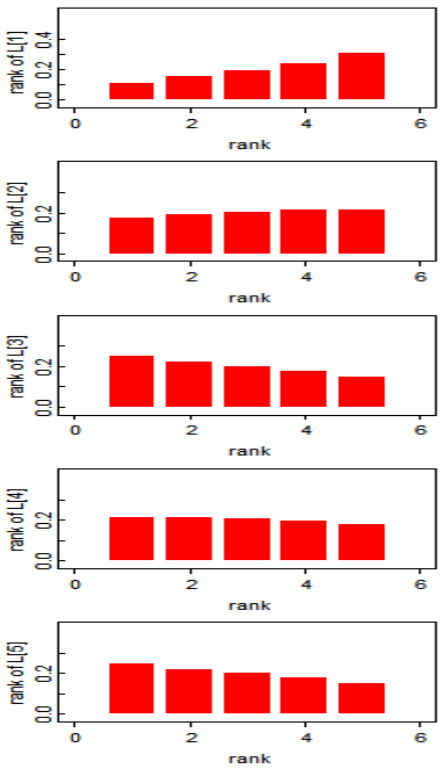
Model No.	Model equation	Missing
Model 1	$mu[i] = \beta_1[1] + \beta_1[2] * x_1[i] + \beta_1[3] * x_2[i] + \beta_1[4] * x_3[i] + \beta_1[5] * x_4[i]$	x_5
Model 2	$mu[i] = \beta_2[1] + \beta_2[2] * x_1[i] + \beta_2[3] * x_2[i] + \beta_2[4] * x_3[i] + \beta_2[5] * x_5[i]$	x_4
Model 3	$mu[i] = \beta_3[1] + \beta_3[2] * x_1[i] + \beta_3[3] * x_2[i] + \beta_3[4] * x_4[i] + \beta_3[5] * x_5[i]$	x_3
Model 4	$mu[i] = \beta_4[1] + \beta_4[2] * x_1[i] + \beta_4[3] * x_3[i] + \beta_4[4] * x_4[i] + \beta_4[5] * x_5[i]$	x_2

Model 5	$mu[i] = \beta_5[0] + \beta_5[2] * x_2[i] + \beta_5[3] * x_3[i] + \beta_5[4] * x_4[i] + \beta_5[5] * x_5[i]$	x_1

Statistics:

Node	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
Comp[1,2]	0.4195	0.4935	0.001075	0	0	1	4001	197000
Comp[1,3]	0.3517	0.4775	0.001051	0	0	1	4001	197000
Comp[1,4]	0.3819	0.4858	0.001087	0	0	1	4001	197000
Comp[1,5]	0.3536	0.4781	0.001465	0	0	1	4001	197000
Comp[2,3]	0.4297	0.495	0.001047	0	0	1	4001	197000
Comp[2,4]	0.462	0.4986	0.001049	0	0	1	4001	197000
Comp[2,5]	0.4312	0.4952	0.001445	0	0	1	4001	197000
Comp[3,4]	0.5312	0.499	0.001149	0	1	1	4001	197000
Comp[3,5]	0.5023	0.5	0.001457	0	1	1	4001	197000
Comp[4,5]	0.4709	0.4992	0.001522	0	0	1	4001	197000
L[1]	8.819	1.567	0.003752	6.288	8.635	12.4	4001	197000
L[2]	8.392	1.498	0.003336	5.974	8.221	11.83	4001	197000
L[3]	8.04	1.437	0.003211	5.727	7.869	11.32	4001	197000
L[4]	8.196	1.46	0.003396	5.84	8.022	11.52	4001	197000
L[5]	8.052	1.437	0.006108	5.748	7.879	11.34	4001	197000
Y[1,21]	8.903	1.681	0.003808	5.572	8.904	12.22	4001	197000
Y[2,21]	9.191	1.56	0.003557	6.111	9.191	12.27	4001	197000
Y[3,21]	8.768	1.543	0.003437	5.718	8.767	11.81	4001	197000
Y[4,21]	8.953	1.547	0.003431	5.914	8.949	12.01	4001	197000
Y[5,21]	8.774	1.478	0.003422	5.831	8.777	11.69	4001	197000
beta1[1]	-7.322	1.441	0.003258	-10.18	-7.317	-4.483	4001	197000
beta1[2]	1.805	3.874	0.01077	-4.976	1.301	10.61	4001	197000
beta1[3]	-4.017	5.484	0.01209	-14.85	-4.029	6.841	4001	197000
beta1[4]	16.54	9.516	0.02182	-2.445	16.58	35.22	4001	197000
beta1[5]	5.916	6.784	0.0168	-7.724	5.995	19.06	4001	197000
beta2[1]	-5.975	1.693	0.003878	-9.342	-5.972	-2.64	4001	197000
beta2[2]	1.887	3.241	0.008724	-3.763	1.505	9.115	4001	197000
beta2[3]	-0.5236	4.328	0.009903	-9.206	-0.4683	7.915	4001	197000
beta2[4]	2.705	11.98	0.02703	-21.02	2.703	26.53	4001	197000
beta2[5]	13.85	8.667	0.01956	-3.394	13.89	30.87	4001	197000
beta3[1]	-5.229	1.55	0.003614	-8.308	-5.23	-2.155	4001	197000
beta3[2]	-0.2465	3.484	0.009045	-7.434	-0.3107	7.197	4001	197000
beta3[3]	-3.519	4.496	0.009903	-12.39	-3.521	5.408	4001	197000
beta3[4]	7.399	6.225	0.01455	-4.963	7.397	19.74	4001	197000
beta3[5]	16.54	6.355	0.01491	3.903	16.56	29.05	4001	197000
beta4[1]	-5.564	1.708	0.003633	-8.939	-5.567	-2.158	4001	197000

beta4[2]	-0.365	3.519	0.007951	-7.753	-0.3825	7.085	4001	197000
beta4[3]	-1.318	10.37	0.02389	-21.86	-1.298	19.15	4001	197000
beta4[4]	4.446	5.103	0.01126	-5.704	4.432	14.61	4001	197000
beta4[5]	14.99	8.518	0.01968	-1.781	14.97	31.9	4001	197000
beta5[1]	-5.515	1.484	0.003132	-8.449	-5.517	-2.58	4001	197000
beta5[2]	-4.109	5.082	0.01166	-14.17	-4.11	5.932	4001	197000
beta5[3]	3.119	11.58	0.02581	-19.88	3.121	25.95	4001	197000
beta5[4]	7.12	5.276	0.01153	-3.28	7.116	17.61	4001	197000
beta5[5]	14.95	8.012	0.01708	-0.8476	14.94	30.77	4001	197000
tau[1]	0.5555	0.1723	3.03E-04	0.2716	0.5372	0.9408	4001	197000
tau[2]	0.6024	0.1871	3.13E-04	0.294	0.5819	1.023	4001	197000
tau[3]	0.647	0.1998	3.06E-04	0.3176	0.6266	1.094	4001	197000
tau[4]	0.6267	0.1934	3.46E-04	0.3075	0.6068	1.058	4001	197000
tau[5]	0.6461	0.1993	0.001089	0.3167	0.6251	1.091	4001	197000
x1[11,2]	1.428	0.4904	0.001315	0.5138	1.387	2.44	4001	197000
x2[11,2]	1.426	0.4819	0.001343	0.5229	1.389	2.428	4001	197000
x3[11,2]	1.5	0.5009	0.001381	0.5181	1.5	2.485	4001	197000
x4[11,2]	1.498	0.4983	0.001177	0.5218	1.498	2.48	4001	197000



Here we cannot conclusively draw the conclusion that Model 3 or Model 5 is better just based on rank graphs but based on the Comp values in the table we can see that Model 3 is better that Model 5.

Problem 3 Codes

Problem 3a

Normal prior

```
model {
        for (i in 1:n) {
                y[ obs[i] ] ~ dnorm(mu[obs[i]],tau)
                mu[obs[i]] <- b0 + b1*x1[obs[i]] + b2*x2[obs[i]] + b3*x3[obs[i]] +
        b4*x4[obs[i]] + b5*x5[obs[i]]

                # Prior on x1
                x1[obs[i]] ~ dnorm(1.5,4)
        }
        tau ~ dgamma(0.0001,0.0001)
        b0 ~ dnorm(0,0.0001)
        b1 ~ dnorm(0,0.0001)
        b2 ~ dnorm(0,0.0001)
        b3 ~ dnorm(0,0.0001)
        b4 ~ dnorm(0,0.0001)
        b5 ~ dnorm(0,0.0001)

        # Bayesian R2 calculation
        sum_sq_err <- (n-preds)/tau
        for (i in 1:n) {
                centered_y[i] <- y[i] - mean(y[])
        }
        sum_sq_total <- inprod(centered_y[],centered_y[])
        Bayesian_R2 <- 1-sum_sq_err/sum_sq_total
}

#INITS
list(tau=1,b0=1,b1=0,b2=0,b3=0,b4=0,b5=0)

#DATA1
list(n=22,preds=6)

#Data
obs[] x1[] x2[] x3[] x4[] x5[] y[]
1       1.31    1.07    0.44    0.75    0.35    1.95
2       1.55    1.49    0.53    0.9     0.47    2.9
3       0.99    0.84    0.34    0.57    0.32    0.72
4       0.99    0.83    0.34    0.54    0.27    0.81
5       1.05    0.9     0.36    0.64    0.3     1.09
6       1.09    0.93    0.42    0.61    0.31    1.22
7       1.08    0.9     0.4     0.51    0.31    1.02
8       1.27    1.08    0.44    0.77    0.34    1.93
9       0.99    0.85    0.36    0.56    0.29    0.64
10      1.34    1.13    0.45    0.77    0.37    2.08
11      NA      1.1     0.45    0.76    0.38    1.98
12      1.33    1.1     0.48    0.77    0.38    1.9
13      1.86    1.47    0.6     1.01    0.65    8.56
14      1.58    1.34    0.52    0.95    0.5     4.49
15      1.97    1.59    0.67    1.2     0.59    8.49
16      1.8     1.56    0.66    1.02    0.59    6.17
17      1.75    1.58    0.63    1.09    0.59    7.54
18      1.72    1.43    0.64    1.02    0.63    6.36
19      1.68    1.57    0.72    0.96    0.68    7.63
20      1.75    1.59    0.68    1.08    0.62    7.78
21      2.19    1.86    0.75    1.24    0.72    NA
22      1.73    1.67    0.64    1.14    0.55    6.88
END
```

Zellner's g-prior

```
model {
        for (i in 1:n) {
                x[obs[i],1]<- 1
                y[ obs[i] ] ~ dnorm(mu[obs[i]],tau)
                mu[obs[i]] <- inprod(b[],x[obs[i],])
                x[obs[i],2] ~ dnorm(1.5,4)
        }
        log.sigma_sq ~ dflat()
        log(sigma_sq) <- log.sigma_sq
        tau <- 1/sigma_sq

        b[1:preds] ~ dmnorm(mu_b[],tau_b[,])
        for (j in 1:preds) {
                mu_b[j] <- 0
        }
        zellner_g <- 22
        for (i in 1:preds) {
                for (j in 1:preds) {
                        tau_b[i,j] <- tau/zellner_g * inprod(x[,i],x[,j])
                }
        }

        # Bayesian R2 calculation
        sum_sq_err <- (n-preds)/tau
        for (i in 1:n) {
                centered_y[i] <- y[i] - mean(y[])
        }
        sum_sq_total <- inprod(centered_y[],centered_y[])
        Bayesian_R2 <- 1-sum_sq_err/sum_sq_total
}

#INITS
list(log.sigma_sq=1,b=c(1,0,0,0,0,0))

#DATA1
list(n=22,preds=6)

#Data
obs[]   x[,2]   x[,3]   x[,4]   x[,5]   x[,6]   y[]
1       1.31    1.07    0.44    0.75    0.35    1.95
2       1.55    1.49    0.53    0.9     0.47    2.9
3       0.99    0.84    0.34    0.57    0.32    0.72
4       0.99    0.83    0.34    0.54    0.27    0.81
5       1.05    0.9     0.36    0.64    0.3     1.09
6       1.09    0.93    0.42    0.61    0.31    1.22
7       1.08    0.9     0.4     0.51    0.31    1.02
8       1.27    1.08    0.44    0.77    0.34    1.93
9       0.99    0.85    0.36    0.56    0.29    0.64
10      1.34    1.13    0.45    0.77    0.37    2.08
11      NA      1.1     0.45    0.76    0.38    1.98
12      1.33    1.1     0.48    0.77    0.38    1.9
13      1.86    1.47    0.6     1.01    0.65    8.56
14      1.58    1.34    0.52    0.95    0.5     4.49
15      1.97    1.59    0.67    1.2     0.59    8.49
16      1.8     1.56    0.66    1.02    0.59    6.17
17      1.75    1.58    0.63    1.09    0.59    7.54
18      1.72    1.43    0.64    1.02    0.63    6.36
19      1.68    1.57    0.72    0.96    0.68    7.63
20      1.75    1.59    0.68    1.08    0.62    7.78
21      2.19    1.86    0.75    1.24    0.72    NA
22      1.73    1.67    0.64    1.14    0.55    6.88
END
```

Problem 3b

```
clc
clear

% The values were got from WINBUGS answers of part(a)
b_zellner_lower = [-8.772,-7.39,-14.22,-19.72,-4.89,-1.52];
b_zellner_upper = [-2.038,7.188,5.936,25.94,19.74,31.69];
b_normal_lower = [-8.044,-4.203,-9.995,-10,-0.7051,5.364];
b_normal_upper = [-3.871,5.611,1.933,15.92,14.54,25.47];

x = [1.52,1.112,0.622,0.917,0.324]; % Given

fprintf('Since all xs are positive the value of y evaluated corresponding to the 2.5percent
interval of betas will be the lower end of the 95percent confidence interval\n');
fprintf('Similarly the value of y evaluated corresponding to the 97.5percent interval of betas
will be the upper end of the 95percent confidence interval\n\n');


y_zellner_lower = b_zellner_lower*[1 x]';
y_zellner_upper = b_zellner_upper*[1 x]';
fprintf('Assuming Zellner prior the 95percent credible set of y is
[%0.2f,%0.2f]\n',y_zellner_lower,y_zellner_upper);

y_normal_lower = b_normal_lower*[1 x]';
y_normal_upper = b_normal_upper*[1 x]';
fprintf('Assuming Normal prior the 95percent credible set of y is
[%0.2f,%0.2f]\n',y_normal_lower,y_normal_upper);

% % ANSWERS FOLLOW
% % Since all xs are positive the value of y evaluated corresponding to the 2.5percent
interval of betas will be the lower end of the 95percent confidence interval
% % Similarly the value of y evaluated corresponding to the 97.5percent interval of betas will
be the upper end of the 95percent confidence interval
% %
% % Assuming Zellner prior the 95percent credible set of y is [-53.06,59.99]
% % Assuming Normal prior the 95percent credible set of y is [-30.68,38.29]
```

Problem 3c

Normal prior

```
model {
        for (j in 1:N){
           x11[j] ~ dnorm(1.5,4) #We have a missing x1 so we have a distribution for it
       mu[1,j] <- a[1] + a[2]*x11[j] + a[3]*x2[j] + a[4]*x3[j]+a[5]*x4[j]

           x12[j] ~ dnorm(1.5,4)
       mu[2,j] <- b[1] + b[2]*x12[j] + b[3]*x2[j] + b[4]*x3[j]+b[5]*x5[j]

           x13[j] ~ dnorm(1.5,4)
       mu[3,j] <- c[1] + c[2]*x13[j] + c[3]*x2[j] + c[4]*x4[j]+a[5]*x5[j]

           x14[j] ~ dnorm(1.5,4)
       mu[4,j] <- d[1] + d[2]*x14[j] + d[3]*x3[j] + d[4]*x4[j]+ d[5]*x5[j]

       mu[5,j] <- e[1] + e[2]*x2[j] + e[3]*x3[j] + e[4]*x4[j] + e[5]*x5[j]

        }

        # Compute the Ibrahim-Laud criterion for each model
        for (i in 1:5) {
                tau[i] ~ dgamma(0.0001,0.0001)
            L[i] <- sqrt(sum(D2[i,]) + pow(sd(Y.new[i,]),2) )
               for (j in 1:N){
        Y[i,j]    ~ dnorm(mu[i,j],tau[i])
               D2[i,j] <- pow(Y[i,j]-Y.new[i,j],2)
```

```
               Y.new[i,j] ~ dnorm(mu[i,j],tau[i])
                }
        }

        # Comparing the Ibrahim Laud Criterion in a pairwise manner
        # Smaller the Ibrahim-Laud criterion is better
        for (i in 1:5) {
        for (j in i+1:5) {
                        Comp[i,j] <- step(L[j]-L[i])
                }
        }

        # Prior distirbutions for the covariates
   for (j in 1:5) {
        a[j] ~ dnorm(0,0.00001)
    b[j] ~ dnorm(0,0.00001)
    c[j] ~ dnorm(0,0.00001)
    d[j] ~ dnorm(0,0.00001)
    e[j] ~ dnorm(0,0.00001)
        }
}

# Here we just initialize tau and the rest we use the 'Generate Inits' in BUGS
list(tau=c(1,1,1,1,1))

# Data Part-1
list(N=22)

# Data Part-2
x11[]   x12[]   x13[]   x14[]   x2[]    x3[]    x4[]    x5[]
1.31    1.31    1.31    1.31    1.07    0.44    0.75    0.35
1.55    1.55    1.55    1.55    1.49    0.53    0.9     0.47
0.99    0.99    0.99    0.99    0.84    0.34    0.57    0.32
0.99    0.99    0.99    0.99    0.83    0.34    0.54    0.27
1.05    1.05    1.05    1.05    0.9     0.36    0.64    0.3
1.09    1.09    1.09    1.09    0.93    0.42    0.61    0.31
1.08    1.08    1.08    1.08    0.9     0.4     0.51    0.31
1.27    1.27    1.27    1.27    1.08    0.44    0.77    0.34
0.99    0.99    0.99    0.99    0.85    0.36    0.56    0.29
1.34    1.34    1.34    1.34    1.13    0.45    0.77    0.37
NA      NA      NA      NA      1.1     0.45    0.76    0.38
1.33    1.33    1.33    1.33    1.1     0.48    0.77    0.38
1.86    1.86    1.86    1.86    1.47    0.6     1.01    0.65
1.58    1.58    1.58    1.58    1.34    0.52    0.95    0.5
1.97    1.97    1.97    1.97    1.59    0.67    1.2     0.59
1.8     1.8     1.8     1.8     1.56    0.66    1.02    0.59
1.75    1.75    1.75    1.75    1.58    0.63    1.09    0.59
1.72    1.72    1.72    1.72    1.43    0.64    1.02    0.63
1.68    1.68    1.68    1.68    1.57    0.72    0.96    0.68
1.75    1.75    1.75    1.75    1.59    0.68    1.08    0.62
2.19    2.19    2.19    2.19    1.86    0.75    1.24    0.72
1.73    1.73    1.73    1.73    1.67    0.64    1.14    0.55
END


# Data Part-3
Y[,1]   Y[,2]   Y[,3]   Y[,4]   Y[,5]   Y[,6]   Y[,7]   Y[,8]   Y[,9]   Y[,10]  Y[,11]  Y[,12]
        Y[,13]  Y[,14]  Y[,15]  Y[,16]  Y[,17]  Y[,18]  Y[,19]  Y[,20]  Y[,21]  Y[,22]
1.95    2.9     0.72    0.81    1.09    1.22    1.02    1.93    0.64    2.08    1.98    1.9     8.56
        4.49    8.49    6.17    7.54    6.36    7.63    7.78    NA      6.88
1.95    2.9     0.72    0.81    1.09    1.22    1.02    1.93    0.64    2.08    1.98    1.9     8.56
        4.49    8.49    6.17    7.54    6.36    7.63    7.78    NA      6.88
1.95    2.9     0.72    0.81    1.09    1.22    1.02    1.93    0.64    2.08    1.98    1.9     8.56
        4.49    8.49    6.17    7.54    6.36    7.63    7.78    NA      6.88
1.95    2.9     0.72    0.81    1.09    1.22    1.02    1.93    0.64    2.08    1.98    1.9     8.56
        4.49    8.49    6.17    7.54    6.36    7.63    7.78    NA      6.88
1.95    2.9     0.72    0.81    1.09    1.22    1.02    1.93    0.64    2.08    1.98    1.9     8.56
        4.49    8.49    6.17    7.54    6.36    7.63    7.78    NA      6.88
END
```

Zellner's g-prior

```
model{
for(j in 1:n ){
       x1[j,1]<-1.0 # The first term has to be one for all models
       x1[j,2] ~ dnorm(1.5,4) #We have a missing x1 so we have a distribution for it
       mu[1,j] <- inprod( beta1[], x1[j,])

       x2[j,1]<-1.0
       x2[j,2] ~ dnorm(1.5,4)
       mu[2,j] <- inprod( beta2[], x2[j,])

       x3[j,1]<-1.0
       x3[j,2] ~ dnorm(1.5,4)
       mu[3,j] <- inprod( beta3[], x3[j,])

       x4[j,1]<-1.0
       x4[j,2] ~ dnorm(1.5,4)
       mu[4,j] <- inprod( beta4[], x4[j,])

       x5[j,1]<-1.0
       mu[5,j] <- inprod( beta5[], x5[j,])

}

# In Zellner's g-prior we have that covariates in a model can be correlated and hence
# they are generated as Multivariate Normal Variables (MVN)
beta1[1:p] ~ dmnorm(mu.beta1[], T.beta1[,])
beta2[1:p] ~ dmnorm(mu.beta2[], T.beta2[,])
beta3[1:p] ~ dmnorm(mu.beta3[], T.beta3[,])
beta4[1:p] ~ dmnorm(mu.beta4[], T.beta4[,])
beta5[1:p] ~ dmnorm(mu.beta5[], T.beta5[,])

# The priors to the covariates are zero mean MVN priors
for( i in 1:p ){
 mu.beta1[i]<-0
 mu.beta2[i]<-0
 mu.beta3[i]<-0
 mu.beta4[i]<-0
 mu.beta5[i]<-0
}

g <- 22 # By putting g=n I am making the prior as non informative
for(i in 1:p){
    for(j in 1:p){
    T.beta1[i,j] <- tau[1]/g * inprod(x1[,i],x1[,j])
    T.beta2[i,j] <- tau[2]/g * inprod(x2[,i],x2[,j])
    T.beta3[i,j] <- tau[3]/g * inprod(x3[,i],x3[,j])
    T.beta4[i,j] <- tau[4]/g * inprod(x4[,i],x4[,j])
    T.beta5[i,j] <- tau[5]/g * inprod(x5[,i],x5[,j])
     }
 }

       # Compute the Ibrahim-Laud criterion for each model
       for (i in 1:5) {
              log.sigma2[i] ~ dflat()
              log(sigma2[i]) <-  log.sigma2[i]
              tau[i] <- 1/sigma2[i]
          L[i] <- sqrt(sum(D2[i,]) + pow(sd(Y.new[i,]),2) )
              for (j in 1:n){
       Y[i,j]    ~ dnorm(mu[i,j],tau[i])
             D2[i,j] <- pow(Y[i,j]-Y.new[i,j],2)
             Y.new[i,j] ~ dnorm(mu[i,j],tau[i])
              }
       }

       # Comparing the Ibrahim Laud Criterion in a pairwise manner
       # Smaller the Ibrahim-Laud criterion is better
       for (i in 1:5) {
       for (j in i+1:5) {
                     Comp[i,j] <- step(L[j]-L[i])
```

```
                }
        }
}


# Data Part-1
list(n = 22,p=5)

# Data Part-2 Fold below
x1[,2]  x1[,3]  x1[,4]  x1[,5]  x2[,2]  x2[,3]  x2[,4]  x2[,5]  x3[,2]  x3[,3]  x3[,4]  x3[,5]
        x4[,2]  x4[,3]  x4[,4]  x4[,5]  x5[,2]  x5[,3]  x5[,4]  x5[,5]
1.31    1.07    0.44    0.75    1.31    1.07    0.44    0.35    1.31    1.07    0.75    0.35    1.31
        0.44    0.75    0.35    1.07    0.44    0.75    0.35
1.55    1.49    0.53    0.9     1.55    1.49    0.53    0.47    1.55    1.49    0.9     0.47    1.55
        0.53    0.9     0.47    1.49    0.53    0.9     0.47
0.99    0.84    0.34    0.57    0.99    0.84    0.34    0.32    0.99    0.84    0.57    0.32    0.99
        0.34    0.57    0.32    0.84    0.34    0.57    0.32
0.99    0.83    0.34    0.54    0.99    0.83    0.34    0.27    0.99    0.83    0.54    0.27    0.99
        0.34    0.54    0.27    0.83    0.34    0.54    0.27
1.05    0.9     0.36    0.64    1.05    0.9     0.36    0.3     1.05    0.9     0.64    0.3     1.05
        0.36    0.64    0.3     0.9     0.36    0.64    0.3
1.09    0.93    0.42    0.61    1.09    0.93    0.42    0.31    1.09    0.93    0.61    0.31    1.09
        0.42    0.61    0.31    0.93    0.42    0.61    0.31
1.08    0.9     0.4     0.51    1.08    0.9     0.4     0.31    1.08    0.9     0.51    0.31    1.08
        0.4     0.51    0.31    0.9     0.4     0.51    0.31
1.27    1.08    0.44    0.77    1.27    1.08    0.44    0.34    1.27    1.08    0.77    0.34    1.27
        0.44    0.77    0.34    1.08    0.44    0.77    0.34
0.99    0.85    0.36    0.56    0.99    0.85    0.36    0.29    0.99    0.85    0.56    0.29    0.99
        0.36    0.56    0.29    0.85    0.36    0.56    0.29
1.34    1.13    0.45    0.77    1.34    1.13    0.45    0.37    1.34    1.13    0.77    0.37    1.34
        0.45    0.77    0.37    1.13    0.45    0.77    0.37
NA      1.1     0.45    0.76    NA      1.1     0.45    0.38    NA      1.1     0.76    0.38    NA
        0.45    0.76    0.38    1.1     0.45    0.76    0.38
1.33    1.1     0.48    0.77    1.33    1.1     0.48    0.38    1.33    1.1     0.77    0.38    1.33
        0.48    0.77    0.38    1.1     0.48    0.77    0.38
1.86    1.47    0.6     1.01    1.86    1.47    0.6     0.65    1.86    1.47    1.01    0.65    1.86
        0.6     1.01    0.65    1.47    0.6     1.01    0.65
1.58    1.34    0.52    0.95    1.58    1.34    0.52    0.5     1.58    1.34    0.95    0.5     1.58
        0.52    0.95    0.5     1.34    0.52    0.95    0.5
1.97    1.59    0.67    1.2     1.97    1.59    0.67    0.59    1.97    1.59    1.2     0.59    1.97
        0.67    1.2     0.59    1.59    0.67    1.2     0.59
1.8     1.56    0.66    1.02    1.8     1.56    0.66    0.59    1.8     1.56    1.02    0.59    1.8
        0.66    1.02    0.59    1.56    0.66    1.02    0.59
1.75    1.58    0.63    1.09    1.75    1.58    0.63    0.59    1.75    1.58    1.09    0.59    1.75
        0.63    1.09    0.59    1.58    0.63    1.09    0.59
1.72    1.43    0.64    1.02    1.72    1.43    0.64    0.63    1.72    1.43    1.02    0.63    1.72
        0.64    1.02    0.63    1.43    0.64    1.02    0.63
1.68    1.57    0.72    0.96    1.68    1.57    0.72    0.68    1.68    1.57    0.96    0.68    1.68
        0.72    0.96    0.68    1.57    0.72    0.96    0.68
1.75    1.59    0.68    1.08    1.75    1.59    0.68    0.62    1.75    1.59    1.08    0.62    1.75
        0.68    1.08    0.62    1.59    0.68    1.08    0.62
2.19    1.86    0.75    1.24    2.19    1.86    0.75    0.72    2.19    1.86    1.24    0.72    2.19
        0.75    1.24    0.72    1.86    0.75    1.24    0.72
1.73    1.67    0.64    1.14    1.73    1.67    0.64    0.55    1.73    1.67    1.14    0.55    1.73
        0.64    1.14    0.55    1.67    0.64    1.14    0.55
END



# Data Part-3 fold below
Y[,1]   Y[,2]   Y[,3]   Y[,4]   Y[,5]   Y[,6]   Y[,7]   Y[,8]   Y[,9]   Y[,10]  Y[,11]  Y[,12]
        Y[,13]  Y[,14]  Y[,15]  Y[,16]  Y[,17]  Y[,18]  Y[,19]  Y[,20]  Y[,21]  Y[,22]
1.95    2.9     0.72    0.81    1.09    1.22    1.02    1.93    0.64    2.08    1.98    1.9     8.56
        4.49    8.49    6.17    7.54    6.36    7.63    7.78    NA      6.88
1.95    2.9     0.72    0.81    1.09    1.22    1.02    1.93    0.64    2.08    1.98    1.9     8.56
        4.49    8.49    6.17    7.54    6.36    7.63    7.78    NA      6.88
1.95    2.9     0.72    0.81    1.09    1.22    1.02    1.93    0.64    2.08    1.98    1.9     8.56
        4.49    8.49    6.17    7.54    6.36    7.63    7.78    NA      6.88
1.95    2.9     0.72    0.81    1.09    1.22    1.02    1.93    0.64    2.08    1.98    1.9     8.56
        4.49    8.49    6.17    7.54    6.36    7.63    7.78    NA      6.88
1.95    2.9     0.72    0.81    1.09    1.22    1.02    1.93    0.64    2.08    1.98    1.9     8.56
        4.49    8.49    6.17    7.54    6.36    7.63    7.78    NA      6.88
END
```

```
# Inits
list(beta1=c(1,0,0,0,0),beta2=c(1,0,0,0,0),beta3=c(1,0,0,0,0),beta4=c(1,0,0,0,0),beta5=c(1,0,0
,0,0),log.sigma2=c(0,0,0,0,0))
```