Assignment 3

anonymous

1 General information

2 Inference for normal mean and deviation (3 points)

Loading the library and the data.

```
library(aaltobda)
data("windshieldy1")

# The data are now stored in the variable `windshieldy1`.
# The below displays the data:
windshieldy1
```

```
[1] 13.357 14.928 14.896 15.297 14.820 12.067 14.824 13.865 17.447
```

The below data is **only for the tests**, you need to change to the full data `windshieldy1` when reporting your results.

```
windshieldy_test <- c(13.357, 14.928, 14.896, 14.820)
```

2.1 (a)

In this exercise the prior will be uninformative, which is defined in Bayesian Data analysis book as.

$$p(\mu, \sigma^2) \propto \frac{1}{\sigma}$$

The likelihood is defined as

$$p(\mu, \sigma|\theta)$$

The likelihood is the normal distribution

$$p(y|\mu, \sigma^2) = \mathcal{N}(\mu, \sigma^2)$$

We get the posterior by the product

$$p(\mu, \sigma^2)p(y|\mu, \sigma^2) = \frac{\mathcal{N}(\mu, \sigma^2)}{\sigma}$$

The marginalized posterior density for \mu follows Student's t distribution

$$t_{n-1}(\bar{y}, \frac{s^2}{n})$$

2.2 (b)

Write your answers and code here!

Keep the below name and format for the functions to work with `markmyassignment`:

```
# Useful functions: mean(), length(), sqrt(), sum()
# and qtnew(), dtnew() (from aaltobda)

mu_point_est <- function(data){
    return(mean(data))
}


mu_interval <- function(data, prob = 0.95) {
    n = length(data)
    lower = (1 - prob) / 2
    upper = 1 - lower

    probabilities = c(lower,upper)
    degree_of_freedom = n-1
    mu = mu_point_est(data)
    scale= sqrt(var(data)/n)

    qt <- qtnew(probabilities, degree_of_freedom, mean = mu, scale = scale)

    return(qt)
}

mu = mu_point_est(windshieldy1) # testdata, right answer 14.5
interval = mu_interval(windshieldy1) # testdata right answer  c(13.3, 15.7)

mu
```

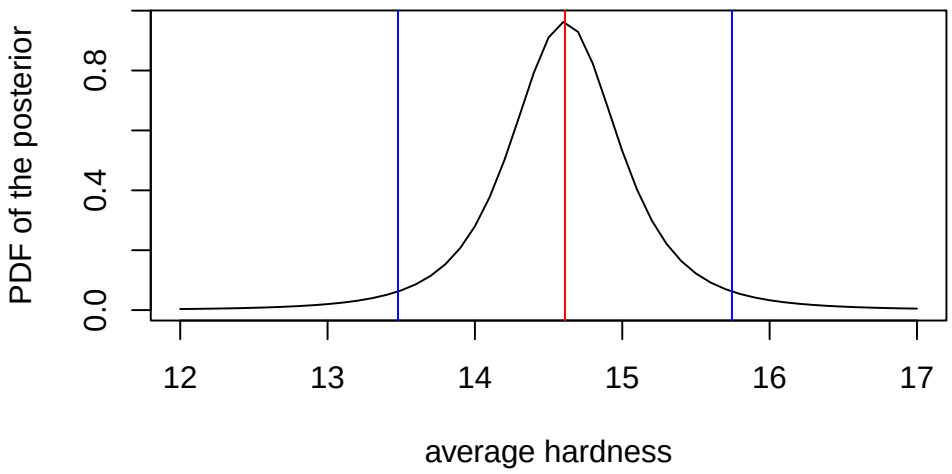
```
[1] 14.61122
```

```
interval
```

```
[1] 13.47808 15.74436
```

```
n = length(windshieldy_test)
x = seq(from=12,to=17,by=0.1)
```

```r
degree_of_freedom = n-1
scale = sqrt(var(windshieldy_test)/n)


density = dtnew(x, degree_of_freedom, mean = mu, scale = scale)

plot(x,density, type = "l",
  xlab='average hardness',
  ylab='PDF of the posterior')
abline(v=mu, col='red')
abline(v=interval[1], col='blue')
abline(v=interval[2], col='blue')
```



4You can plot the density as below if you implement `mu_pdf` to compute the PDF of the posterior $p(\mu|y)$ of the average hardness μ.

```r
mu_pdf <- function(data, x){
    # Compute necessary parameters here.
    # These are the correct parameters for `windshieldy_test`
    # with the provided uninformative prior.
    df = length(data) - 1
    location = mean(data)
    scale = sqrt(var(data)/n)

    # Use the computed parameters as below to compute the PDF:

    return(dtnew(x, degree_of_freedom, mean = mu, scale = scale))

}
```

```r
x_interval = mu_interval(windshieldy1, 0.95)
lower_x = x_interval[1]
upper_x = x_interval[2]
x = seq(lower_x, upper_x, length.out=1000)
plot(x, mu_pdf(windshieldy1, x), type="l",
     xlab=TeX(r'(average hardness $\mu$)'),
     ylab=TeX(r'(PDF of the posterior $p(\mu|y)$)'))
```

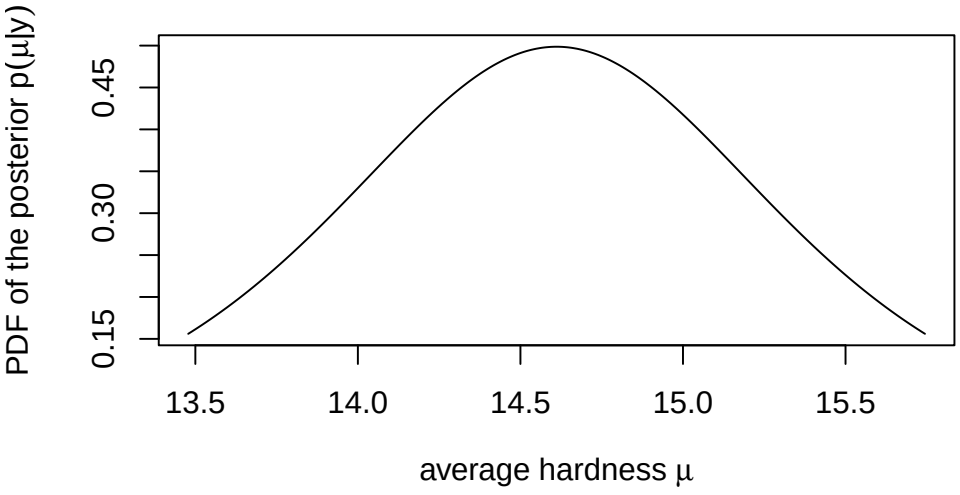


Figure 1: PDF of the posterior $p(\mu|y)$ of the average hardness μ

2.3 (c)

Write your answers and code here!

Keep the below name and format for the functions to work with `markmyassignment`:

```r
# Useful functions: mean(), length(), sqrt(), sum()
# and qtnew(), dtnew() (from aaltobda)

mu_pred_point_est <- function(data) {
    # Do computation here, and return as below.
    # This is the correct return value for the test data provided above.
    return(mean(data))
    14.5
}

mu_pred_point_est(windshieldy1)
```

```
[1] 14.61122
```

```
mu_pred_interval <- function(data, prob = 0.95) {
    # Do computation here, and return as below.
    # This is the correct return value for the test data provided above.
    mu = mu_point_est(data)
    n = length(data)
    freedom = n-1

    scale =  1/(n-1)*sum((data-mu)^2)
      #1/(n-2)*sum((data-mu)^2)
    low = (1-prob)/2
    high= prob+low
    int = c(low,high)
    res <- qtnew(int, freedom, mean=mu, scale = scale)
    return(res)
    c(11.8, 17.2)

}

mu_pred_interval(windshieldy1)
```

```
[1]  9.599923 19.622522
```

You can plot the density as below if you implement `mu_pred_pdf` to compute the PDF of the posterior predictive $p(\tilde{y}|y)$ of a new hardness observation $\tilde{y}$.

```
mu_pred_pdf <- function(data, x){
    # Compute necessary parameters here.
    # These are the correct parameters for `windshieldy_test`
    # with the provided uninformative prior.
    df = length(data) - 1
    location = mean(data)
    mu = location
    scale = 1/(n-2)*sum((data-mu)^2)
      #0.8536316
    # Use the computed parameters as below to compute the PDF:

    return(dtnew(x, df, location, scale))
}

x_interval = mu_pred_interval(windshieldy1, .95)
lower_x = x_interval[1]
upper_x = x_interval[2]
x = seq(lower_x, upper_x, length.out=1000)
pdf = mu_pred_pdf(windshieldy1, x)
plot(
    x, pdf, type="l",
    xlab=TeX(r'(new hardness observation $\tilde{y}$)'),
    ylab=TeX(r'(PDF of the posterior predictive $p(\tilde{y}|y)$)')
)
```

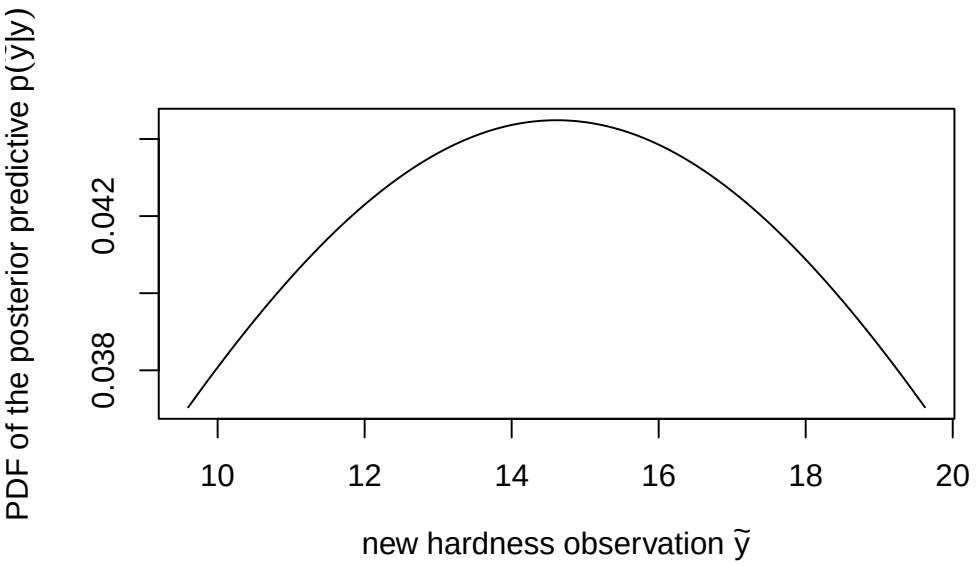


Figure 2: PDF of the posterior predictive $p(\tilde{y}|y)$ of a new hardness observation $\tilde{y}$

3 Inference for the difference between proportions (3 points)

3.1 (a)

The likelihood

$p(y|\pi) = \binom{n}{y}\pi^y(1-\pi)^{n-y}$

the prior $

$p(\pi) = Beta(\alpha, \beta)$

and thus the likelihood becomes

$p(y|\pi) = \binom{n}{y}\pi^y(1-\pi)^{n-y}$.

Using a uniform Beta, $Beta(1,1)$ as our weakly informative prior we get that the posterior is

$p(\pi|y) = Beta(\pi|1+y, 1+n-y)$

3.2 (b)

Write your answers and code here!

The below data is **only for the tests**:

Keep the below name and format for the functions to work with `markmyassignment`:

```
# Useful function: mean(), quantile()

posterior_odds_ratio_point_est <- function(p0, p1) {
```

```r
    # Do computation here, and return as below.
    # This is the correct return value for the test data provided above.
    #2.650172
    ratio = (p1/(1 - p1))/(p0/(1 - p0))
    return(mean(ratio))
}


posterior_odds_ratio_interval <- function(p0, p1, prob = 0.95) {
    # Do computation here, and return as below.
    # This is the correct return value for the test data provided above.
    lower = (1-prob)/2
    upper = prob+lower
    data = (p1/(1 - p1))/(p0/(1 - p0))
    interval = c(lower,upper)
    return(quantile(data,interval))
    #c(0.6796942,7.3015964)
}

n0 = 674
y0 = 39

n1 = 680
y1 = 22

prior_alpha = 1
prior_beta = 1

post_alpha0 = prior_alpha + n0 - y0
post_beta0 = prior_beta + y0

post_alpha1 = prior_alpha + n1 - y1
post_beta1 = prior_beta + y1

set.seed(4711)
ndraws = 10000

p0 <- rbeta(ndraws, post_alpha0, post_beta0)
p1 <- rbeta(ndraws, post_alpha1, post_beta1)

point_est = posterior_odds_ratio_point_est(p0 = p0, p1 = p1)
interval = posterior_odds_ratio_interval(p0 = p0, p1 = p1, prob = 0.95)
paste0("The point estimate is: ", point_est)
```

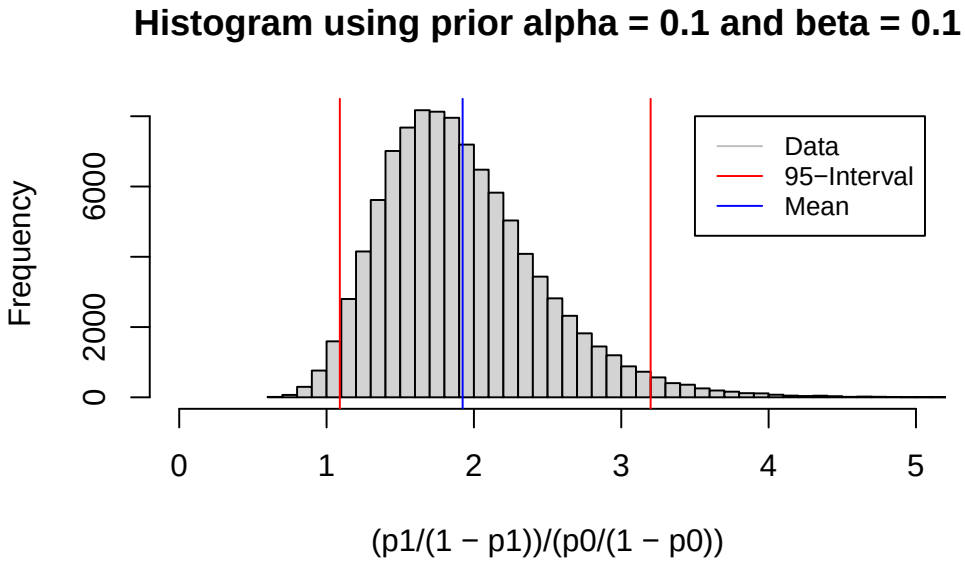
```
[1] "The point estimate is: 1.88460915313603"
```

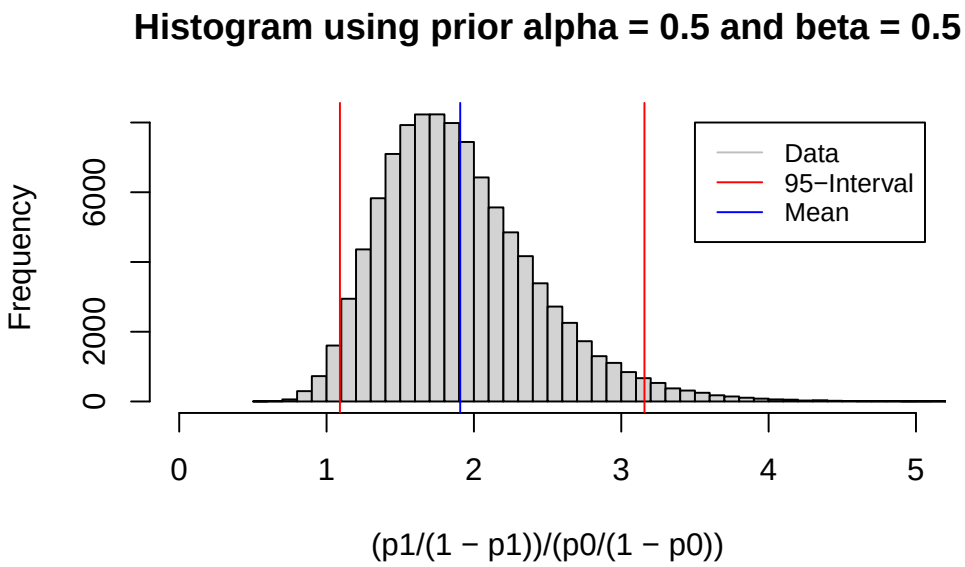
```r
paste0("The interval  is: [", interval[1], ",", interval[2],"]")
```

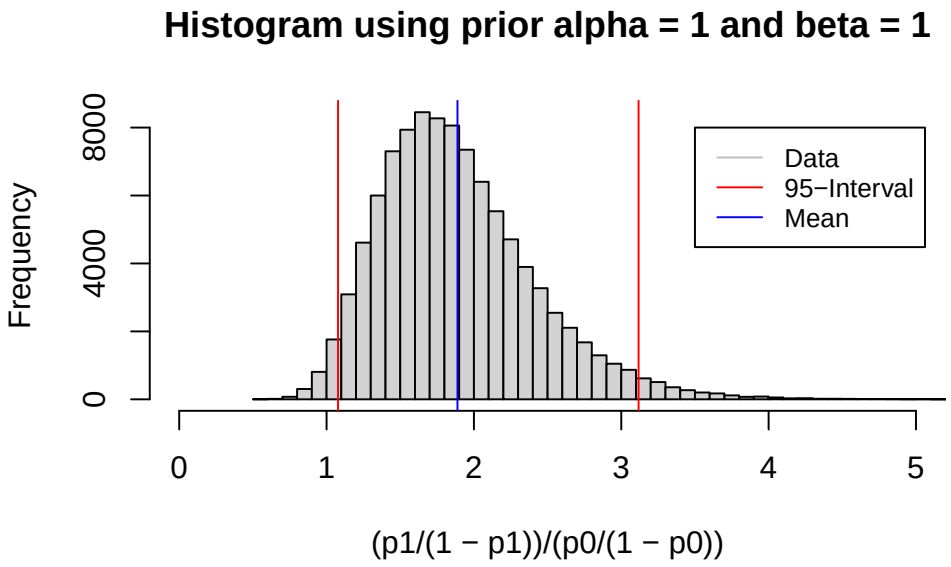
```
[1] "The interval  is: [1.07711580756948,3.11545144407049]"
```

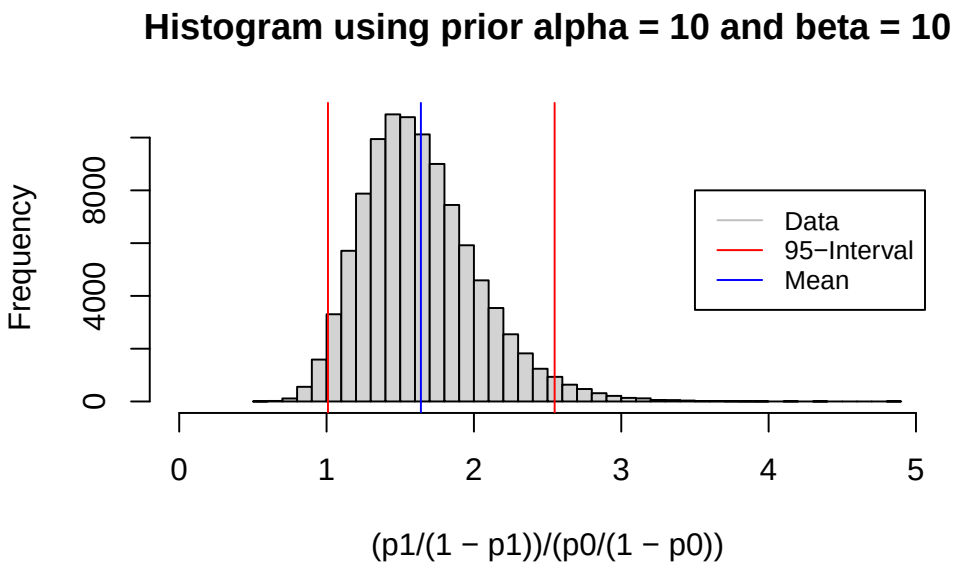
3.3 (c)

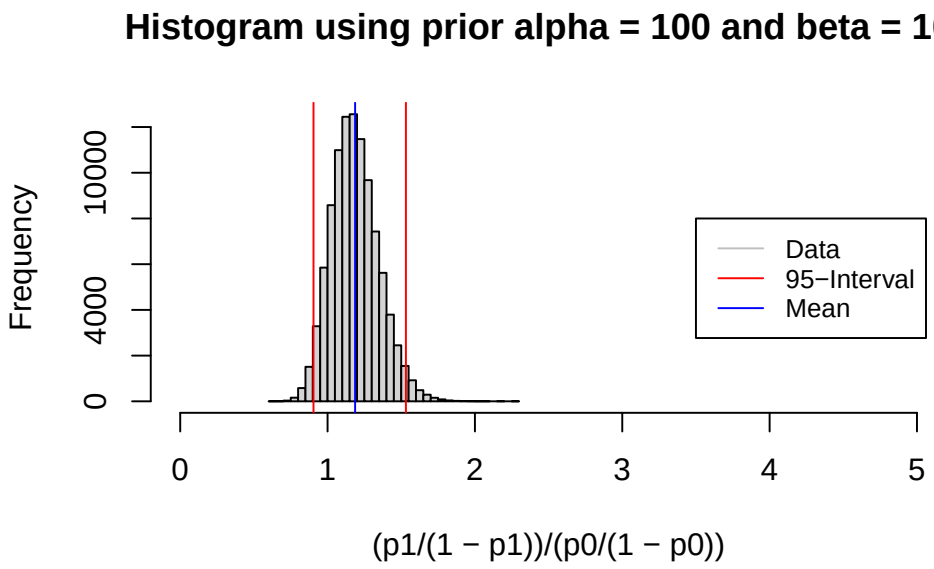
```
# Define prior parameters
prior_alpha = c(0.1, 0.5, 1, 10, 100)
prior_beta = prior_alpha

# Initialize arrays to store results
n_prior = length(prior_alpha)
post_alpha0 = numeric(n_prior)
post_beta0 = numeric(n_prior)
post_alpha1 = numeric(n_prior)
post_beta1 = numeric(n_prior)
point_est = numeric(n_prior)
interval = matrix(nrow=n_prior, ncol=2)

# Perform analysis for each set of priors and create separate plots
for (i in 1:n_prior) {
  post_alpha0[i] = prior_alpha[i] + n0 - y0
  post_beta0[i] = prior_beta[i] + y0
  post_alpha1[i] = prior_alpha[i] + n1 - y1
  post_beta1[i] = prior_beta[i] + y1

  p0 <- rbeta(100000, post_alpha0[i], post_beta0[i])
  p1 <- rbeta(100000, post_alpha1[i], post_beta1[i])

  point_est[i] = posterior_odds_ratio_point_est(p0 = p0, p1 = p1)
  interval[i, ] = posterior_odds_ratio_interval(p0 = p0, p1 = p1, prob = 0.95)

  # Create a separate plot for each prior setting
  par(mfrow=c(1, 1))
  hist((p1 / (1 - p1)) / (p0 / (1 - p0)),
       breaks=50,
       main=paste0("Histogram using prior alpha = ", prior_alpha[i], " and beta = ", prior_beta[i]),
       xlim=c(0, 5))
  abline(v = interval[i, 1], col='red')
  abline(v = interval[i, 2], col='red')
  abline(v = point_est[i], col='blue')
  legend(3.5, 8000, legend=c("Data", "95-Interval", "Mean"), col=c("grey", "red", "blue"), lty=1, ce
}
```


Histogram using prior alpha = 0.1 and beta = 0.1
Frequency
0
2000
6000
0
1
2
3
4
5
(p1/(1 − p1))/(p0/(1 − p0))
Data
95−Interval
Mean


Histogram using prior alpha = 0.5 and beta = 0.5
Frequency
0
2000
6000
0
1
2
3
4
5
(p1/(1 − p1))/(p0/(1 − p0))
Data
95−Interval
Mean


Histogram using prior alpha = 1 and beta = 1
Frequency
0
4000
8000
0
1
2
3
4
5
(p1/(1 − p1))/(p0/(1 − p0))
Data
95−Interval
Mean


Histogram using prior alpha = 10 and beta = 10
Frequency
0
4000
8000
0
1
2
3
4
5
(p1/(1 − p1))/(p0/(1 − p0))
Data
95−Interval
Mean



First conclusion one can see that a bigger priori alpha and beta makes the distribution more narrower in width and shfits it closer to the mean value. On the other hand smaller priori doesnt have a big difference.

4 Inference for the difference between normal means (3 points)

Loading the library and the data.

```
data("windshieldy2")
# The new data are now stored in the variable `windshieldy2`.
# The below displays the first few rows of the new data:
head(windshieldy2)
```

```
[1] 15.980 14.206 16.011 17.250 15.993 15.722
```

4.1 (a)

In the exercise we will be using a uninformative prior $p(\mu, \sigma^2) \propto \frac{1}{\sigma}$.

The likelihood is the normal distribution $p(y|\mu, \sigma^2) = \mathcal{N}(\mu, \sigma^2)$.

We get the posterior by the product $p(\mu, \sigma^2)p(y|\mu, \sigma^2) = \frac{\mathcal{N}(\mu, \sigma^2)}{\sigma}$

The marginalized posterior density for μ follows Student's t distribution

$t_{n-1}(\bar{y}, \frac{s^2}{n})$

is the same as in part 1.

4.2 (b)

Write your answers and code here!

```r
# Useful functions: mean(), length(), sqrt(), sum(),
# rtnew() (from aaltobda), quantile() and hist().
rm(list = ls())
data("windshieldy1")
data("windshieldy2")
```

```r
mu_point_est = function(data){
  return(mean(data))
}

n1 = length(windshieldy1)
df1 = n1 - 1
mu1 = mu_point_est(windshieldy1)
scale1 = sqrt(var(windshieldy1)/n1)

n2 = length(windshieldy2)
df2 = n2 - 1
mu2 = mu_point_est(windshieldy2)
scale2 = sqrt(var(windshieldy2)/n2)

n_samples <- 10000
samples1 <- rtnew(n_samples, df=df1, mean = mu1, scale = scale1)
samples2 <- rtnew(n_samples, df=df2, mean = mu2, scale = scale2)

mu_d <- mu1 - mu2
paste0("Point estimat for the difference in the means are: ", mu_d)
```

```
[1] "Point estimat for the difference in the means are: -1.2098547008547"
```

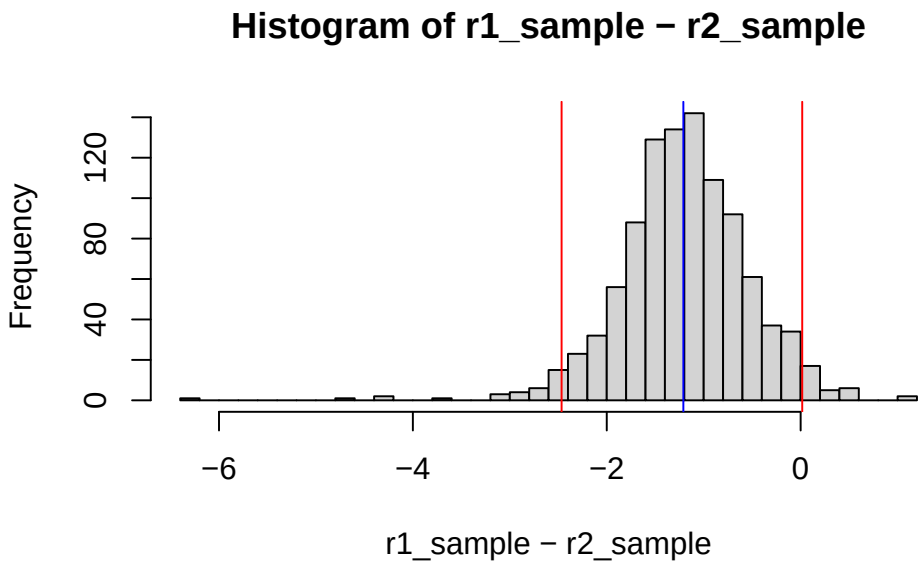
As we can see, the differences in the means are approximately -1.21.

```r
r_sample = function(samples){
  n = length(samples)
  scale = sd(samples)/sqrt(n)
  return(rtnew(1000, n-1, mean(samples), scale))
}

r1_sample = r_sample(windshieldy1)
r2_sample = r_sample(windshieldy2)

interval = quantile(r1_sample-r2_sample, c(0.025,0.975))


paste0("Interval low: ", round(interval[1],digits=2), " and high: ", round(interval[2],digits=2))
```

```
[1] "Interval low: -2.47 and high: 0.02"
```

```r
hist(r1_sample-r2_sample,
       breaks=50)
abline(v = interval[1], col='red')
```

```
abline(v = interval[2], col='red')
abline(v = mu_d, col='blue')
  legend(0.2 ,8000, legend=c("Data", "95-Interval", "mu_d"),col=c("grey", "red", "blue"), lty=1, cex
```



4.3 (c)

For any continuous variable, the probability of it assuming any exact, precise value, referred to as the point density, is infinitesimal, i.e., it approaches zero. Consequently, the probability that the variable exactly equals zero, denoted as P(d = 0), is indeed zero.

Examining the credible intervals, we observe that the value zero falls within the 95% interval, although it lies in close proximity to its boundary. This implies that the likelihood of the variable being exactly zero is exceedingly low.

AI was used to brainstorm, optimize code and better formulate my opinions.