ps8

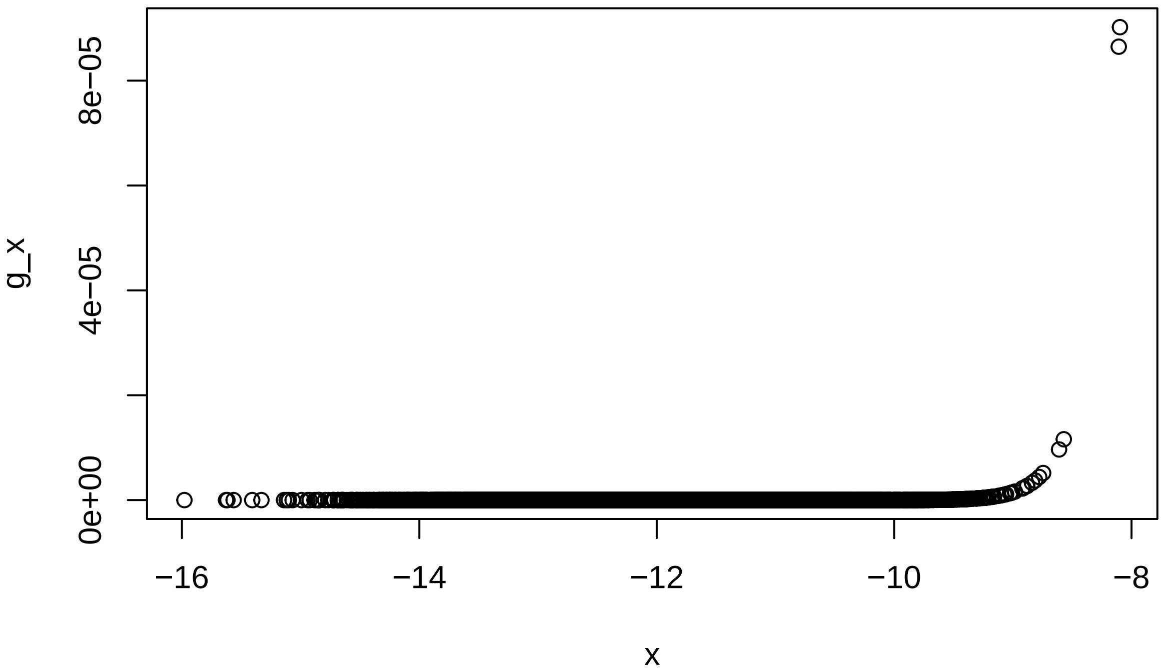
Ramon Crespo

11/20/2018

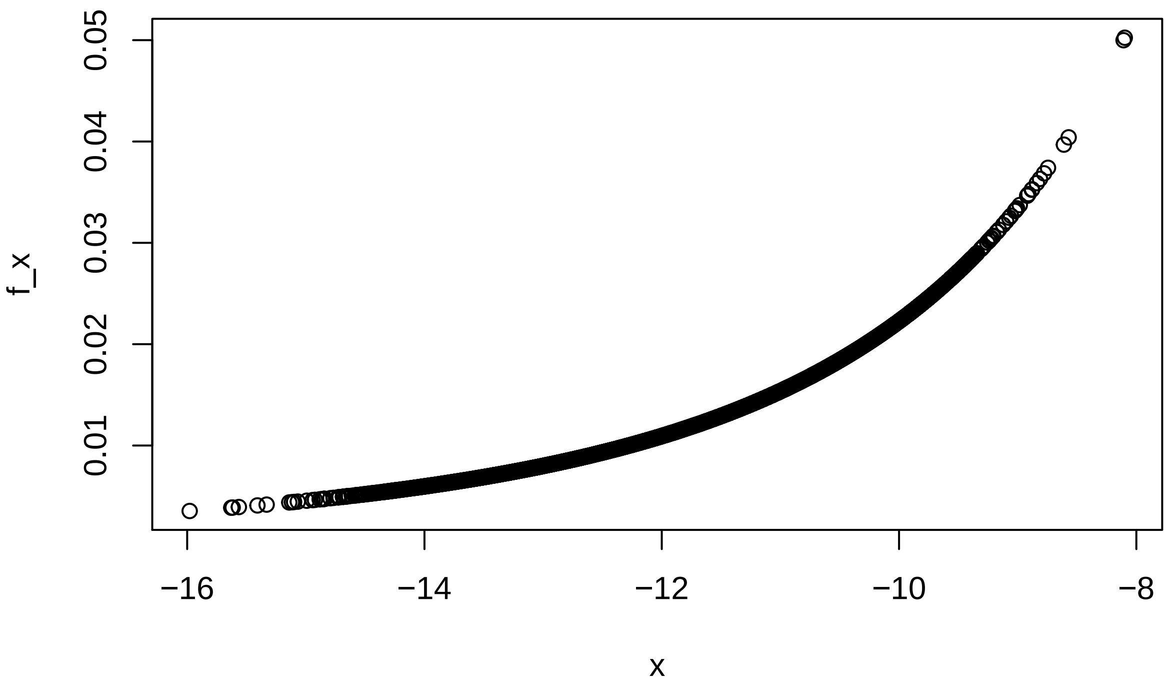
Note: collaborated with Ugur Yildirim. Initially solved the problem set by myself but due to the complexity of the problem set I collaborated with Ugur, who has a stronger statistical background.

Problem1: a)

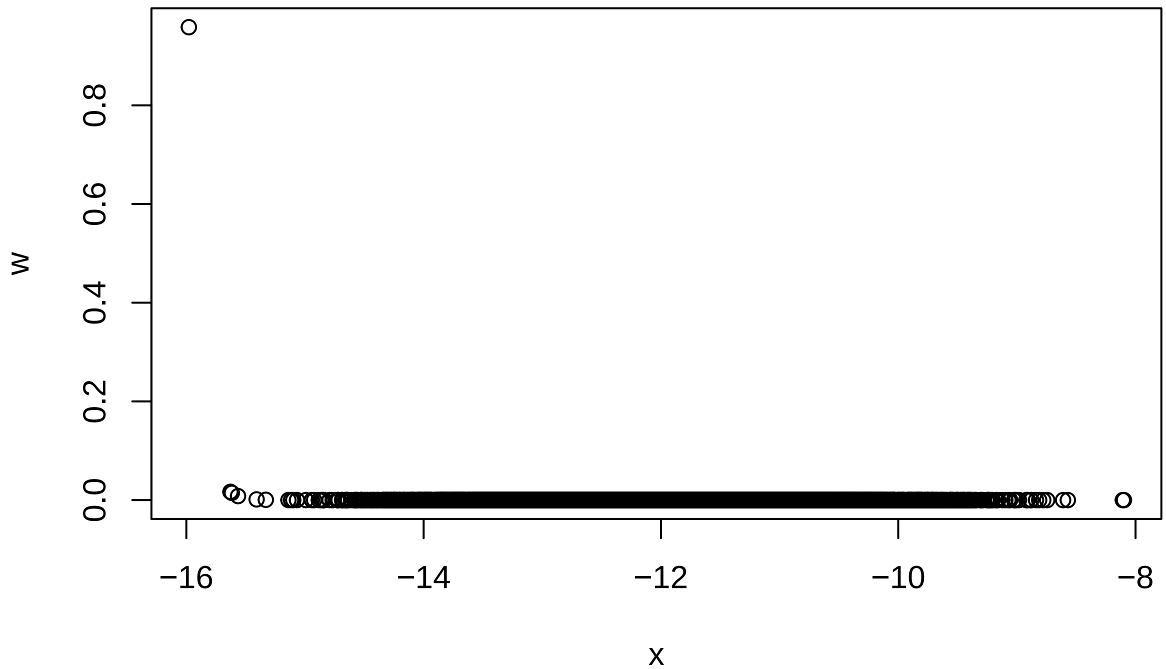
```
#define parameters
#variance
sigma <- 1
#median
mu <- -4
#number of samples
m <- 10000

#Generate samples
#sample from half normal distribution and correct
x <- rnorm(m,mean = -mu,sigma)
x[x>-4] = -4+(-4-x[x>-4])

#degrees of freedom
v <- 3

#pdf of t distribution
f_x <- dt(x,v)/pt(mu,v)

#pdf of half normal distribution
g_x <- dnorm(x, mu, sigma)/pt(m,v)

w_star <- f_x/g_x
w <- w_star/sum(w_star)
plot(x,g_x)
```



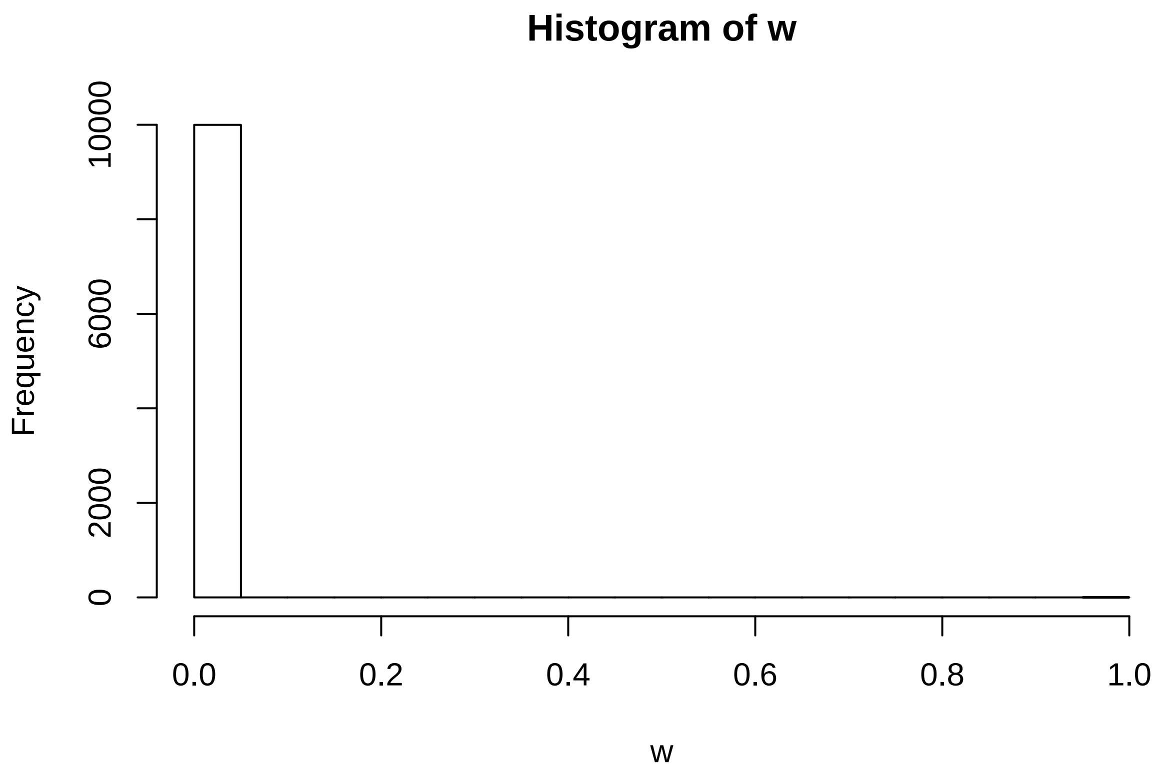

```
plot(x,f_x)
```




```
plot(x,w)
```




```
hist(w,seq(0,1,.05))
```




```
exp_f_x= (1/m)*sum(w*x)
#compute the variance of phi hat

var_phi_hat <- (1/m)*sqrt(sum((exp_f_x-w*x)**2))
print(exp_f_x)
```

```
## [1] -0.0015963
```

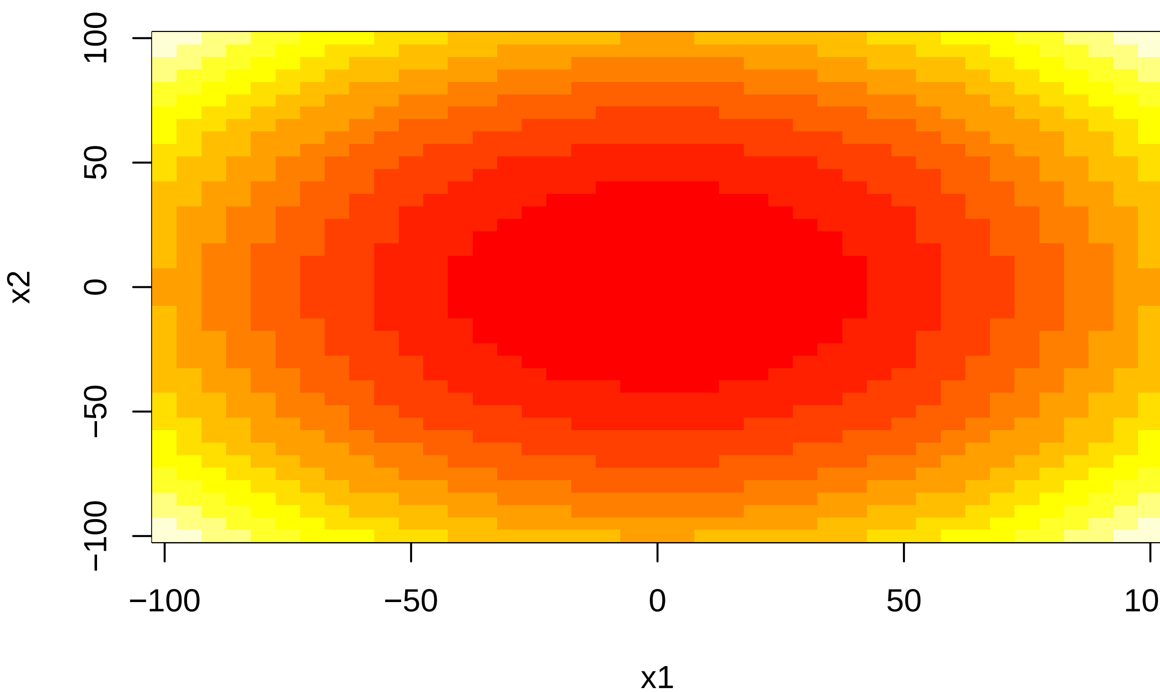
```r
print(var_phi_hat)
```

```
## [1] 0.001531713
```

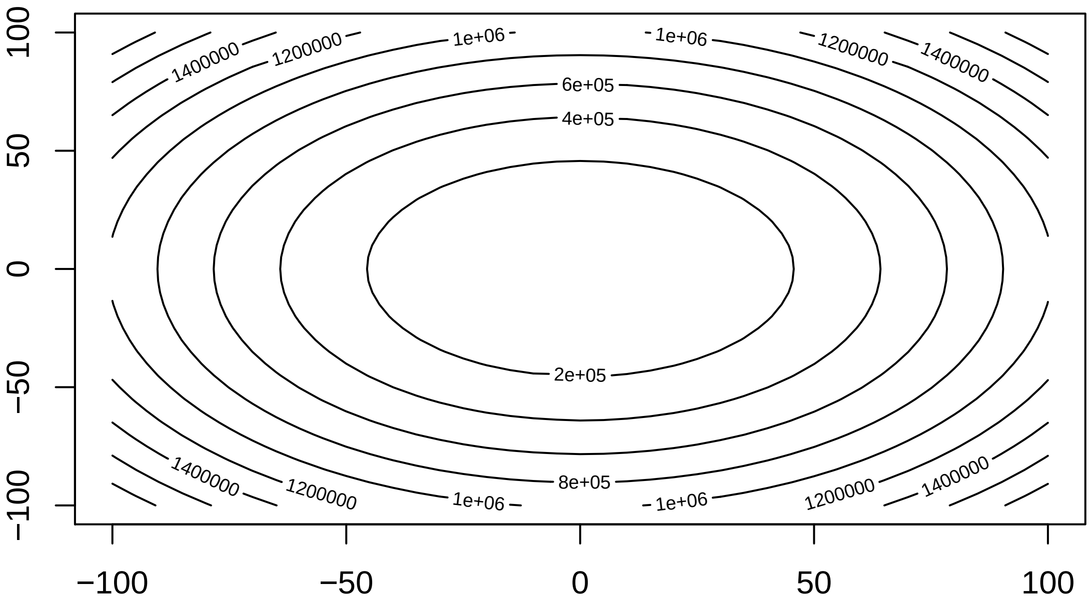
```r
help("hist")
```

Problem 2 Consider the "helical valley" function. Plot slices of the function to get a sense for how it behaves.

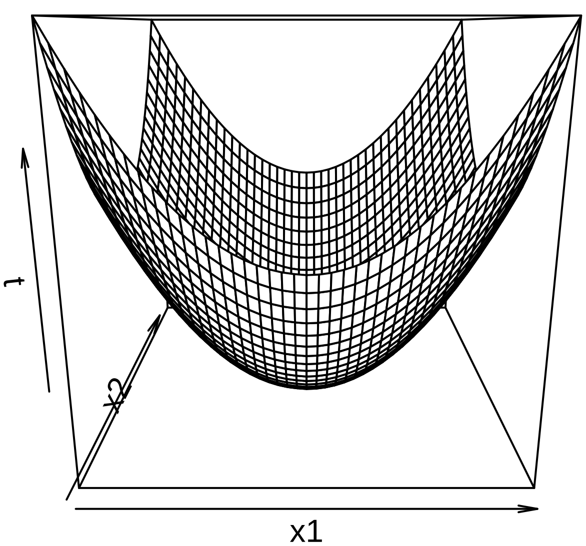
```r
#Plot x1 and x2 to obtain a sense of how the function varies with respect to the variables x1 and x2
x1 <- seq(-100, 100, by=5)
x2 <- seq(-100, 100, by=5)
x3 <- 1

theta <- function(x1,x2) atan2(x2, x1)/(2*pi)
x <- cbind(x1, x2, x3)
f <- function(x1,x2) {
  f1 <- 10*(x3 - 10*theta(x1,x2))
  f2 <- 10*(sqrt(x1^2 + x2^2) - 1)
  f3 <- x3
  return(f1^2 + f2^2 + f3^2)
}

t <- sapply(x1,f,x2)
image(x1,x2,t)
```



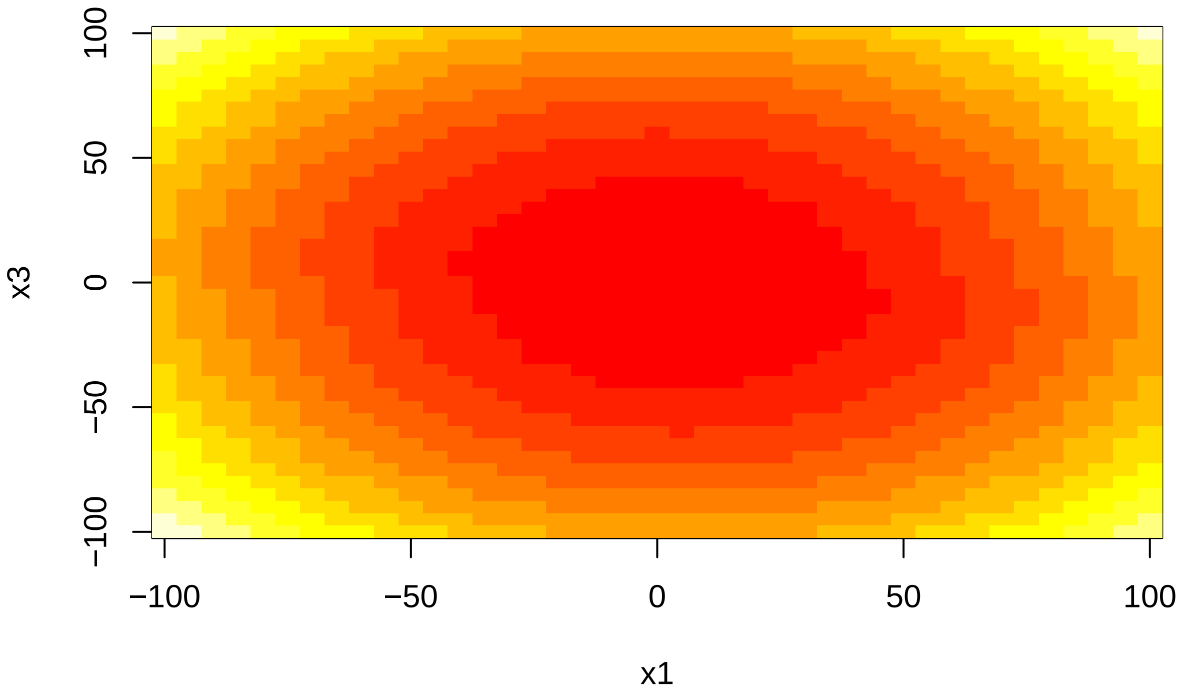

```r
contour(x1,x2,t)
```



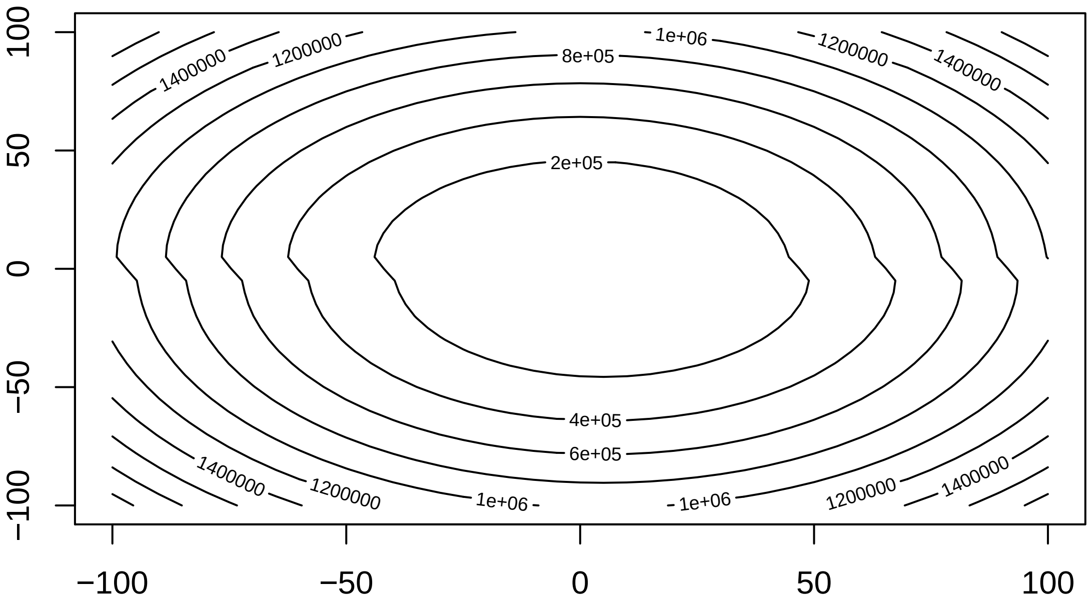

```
persp(x1,x2,t)
```



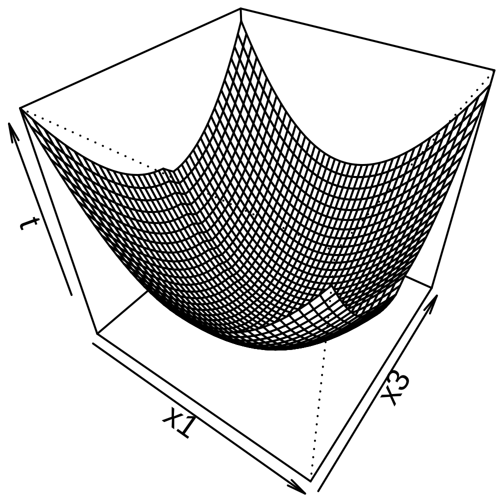

```
#Plot x1 and x3 to obtain a sense of how the function varies with respect to the variables x1 and x3 by
x1 <- seq(-100, 100, by=5)
x2 <- 1
x3 <- seq(-100, 100, by=5)

theta <- function(x1,x2) atan2(x2, x1)/(2*pi)
f <- function(x1,x3) {
  f1 <- 10*(x3 - 10*theta(x1,x2))
  f2 <- 10*(sqrt(x1^2 + x2^2) - 1)
  f3 <- x3
  return(f1^2 + f2^2 + f3^2)
}

t <- sapply(x1,f,x3)
image(x1,x3,t)
```



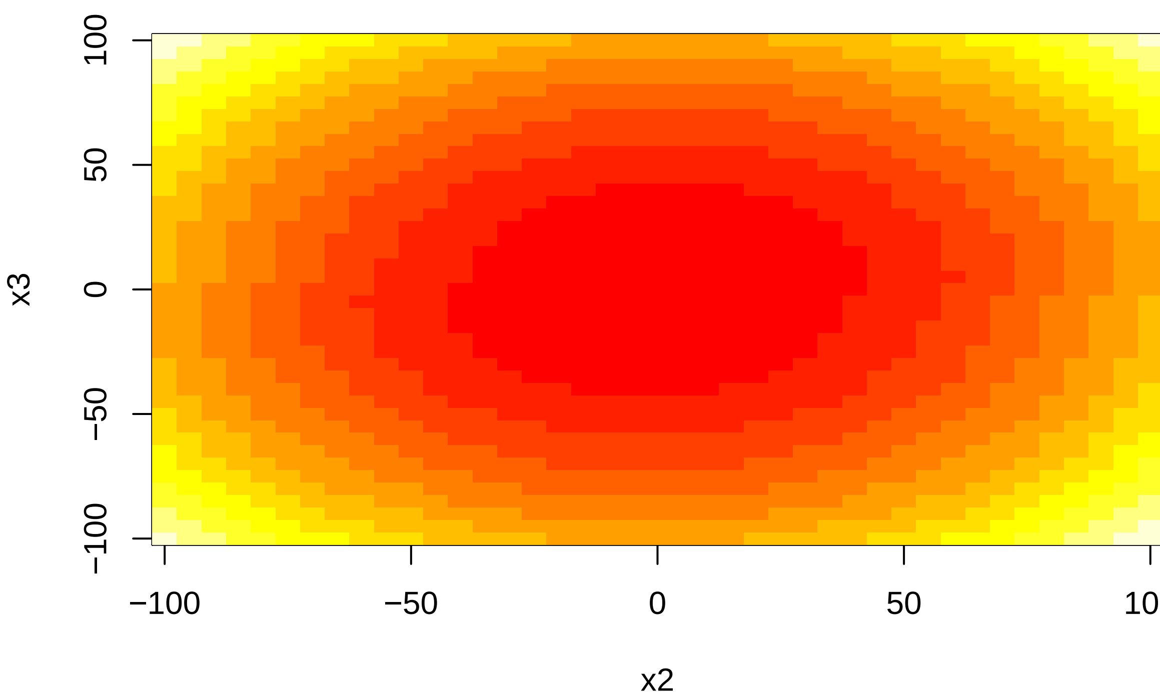

```r
contour(x1,x3,t)
```



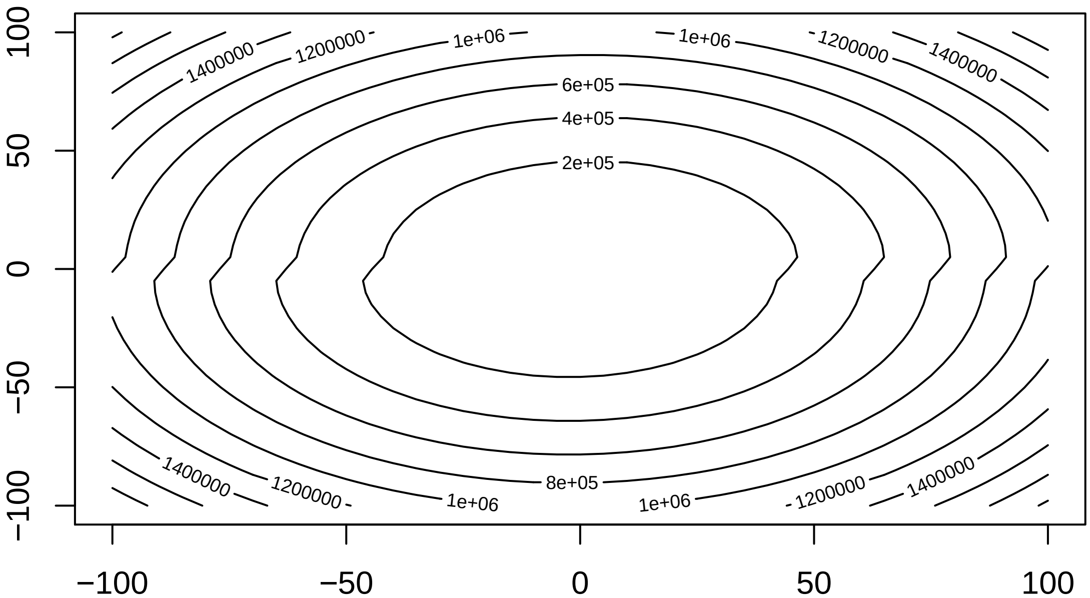

```r
persp(x1,x3,t,theta = 35, phi = 40)
```



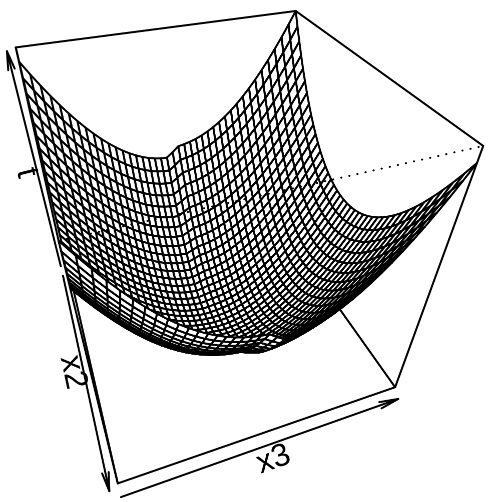

```
#Plot x2 and x3 to obtain a sense of how the function varies with respect to the variables x2 and x3 by
x1 <- 1
x2 <- seq(-100, 100, by=5)
x3 <- seq(-100, 100, by=5)

theta <- function(x1,x2) atan2(x2, x1)/(2*pi)
f <- function(x2,x3) {
  f1 <- 10*(x3 - 10*theta(x1,x2))
  f2 <- 10*(sqrt(x1^2 + x2^2) - 1)
  f3 <- x3
  return(f1^2 + f2^2 + f3^2)
}

t <- sapply(x2,f,x3)
image(x2,x3,t)
```



```r
contour(x2,x3,t)
```




```r
persp(x2,x3,t,theta = 70, phi = 40)
```



Explore the posibility o local minima by using different initial values. In the function below I wxplor the posibility of passing different initial values for x1, while leaving the other values x2 and x3 with a constant initial value. From the results we conclude that the algorithm has different results depending on the initial value of x1. The solution does not vary too much (it is in the orther of 1e-5) but the algorithm does find different values for the optimal point depending on the initialization

```r
?optim()
theta <- function(x1,x2) atan2(x2, x1)/(2*pi)

f <- function(x) {
  f1 <- 10*(x[3] - 10*theta(x[1],x[2]))
  f2 <- 10*(sqrt(x[1]^2 + x[2]^2) - 1)
  f3 <- x[3]
  return(f1^2 + f2^2 + f3^2)
}
```

```r
optimization_values <- function(x1){
  return(optim(c(x1,1,1),f))
}

t <- lapply(seq(-10, 10, by=.5),optimization_values)
t[c(1,20,30)]
```

```
## [[1]]
## [[1]]$par
## [1]  1.0001558192 -0.0008084661 -0.0008045370
##
## [[1]]$value
## [1] 2.631552e-05
##
## [[1]]$counts
## function gradient
##      288       NA
##
## [[1]]$convergence
## [1] 0
##
## [[1]]$message
## NULL
##
##
## [[2]]
## [[2]]$par
## [1] 1.000075e+00 1.310945e-07 3.589943e-05
##
## [[2]]$value
## [1] 6.874854e-07
##
## [[2]]$counts
## function gradient
##      182       NA
##
## [[2]]$convergence
## [1] 0
##
## [[2]]$message
## NULL
##
##
## [[3]]
## [[3]]$par
## [1] 1.0001481371 0.0009570948 0.0010930247
##
## [[3]]$value
## [1] 2.189387e-05
##
## [[3]]$counts
## function gradient
##      208       NA
```

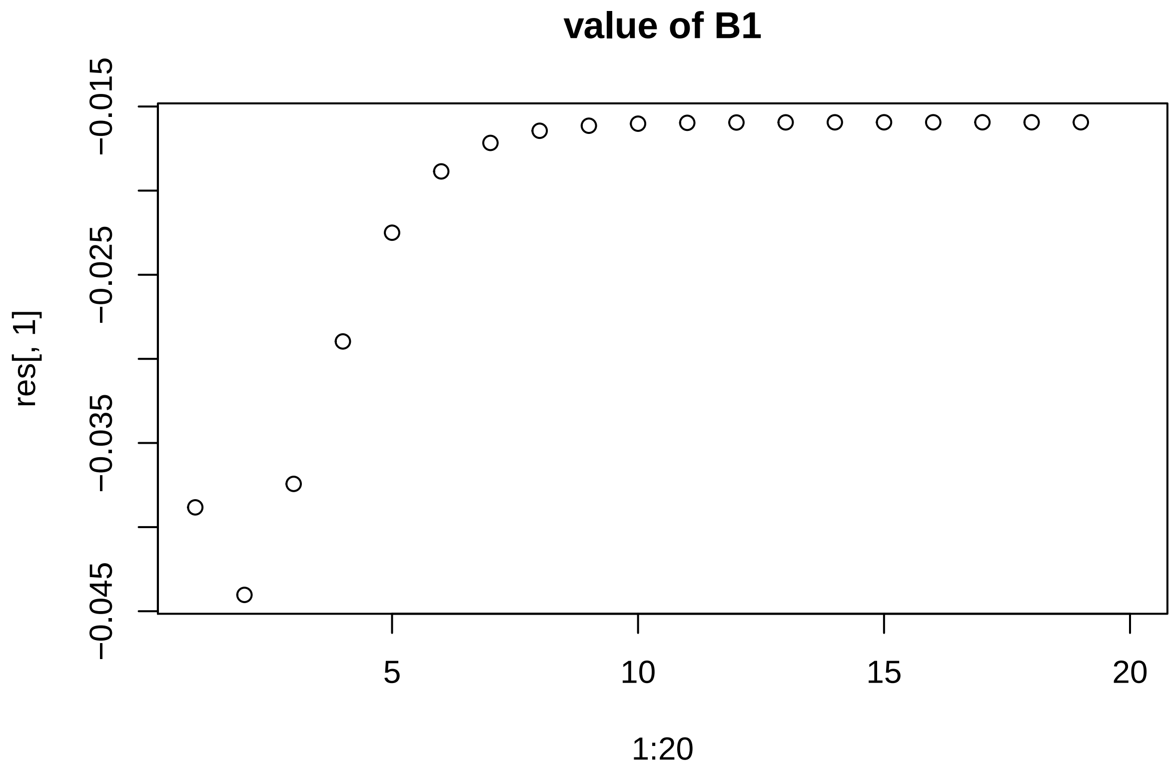
```
##
## [[3]]$convergence
## [1] 0
##
## [[3]]$message
## NULL
```

Including Plots

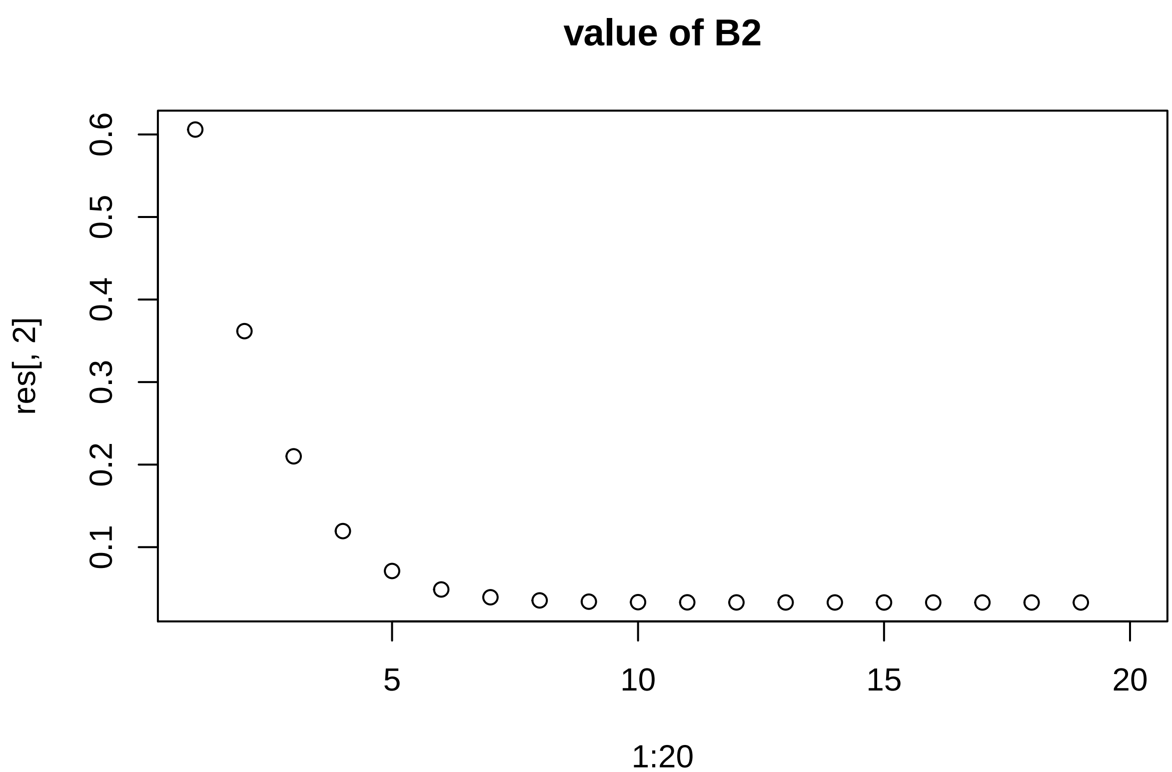
Problem 3b Given that at this point in the problem we dont really have any further information for the problem I would do some search for convergence by starting B1, B2, B3, B4 with a convinatio of values ranging from -1 to 1. An example could be 1,1,-1,-1

Problem 3c

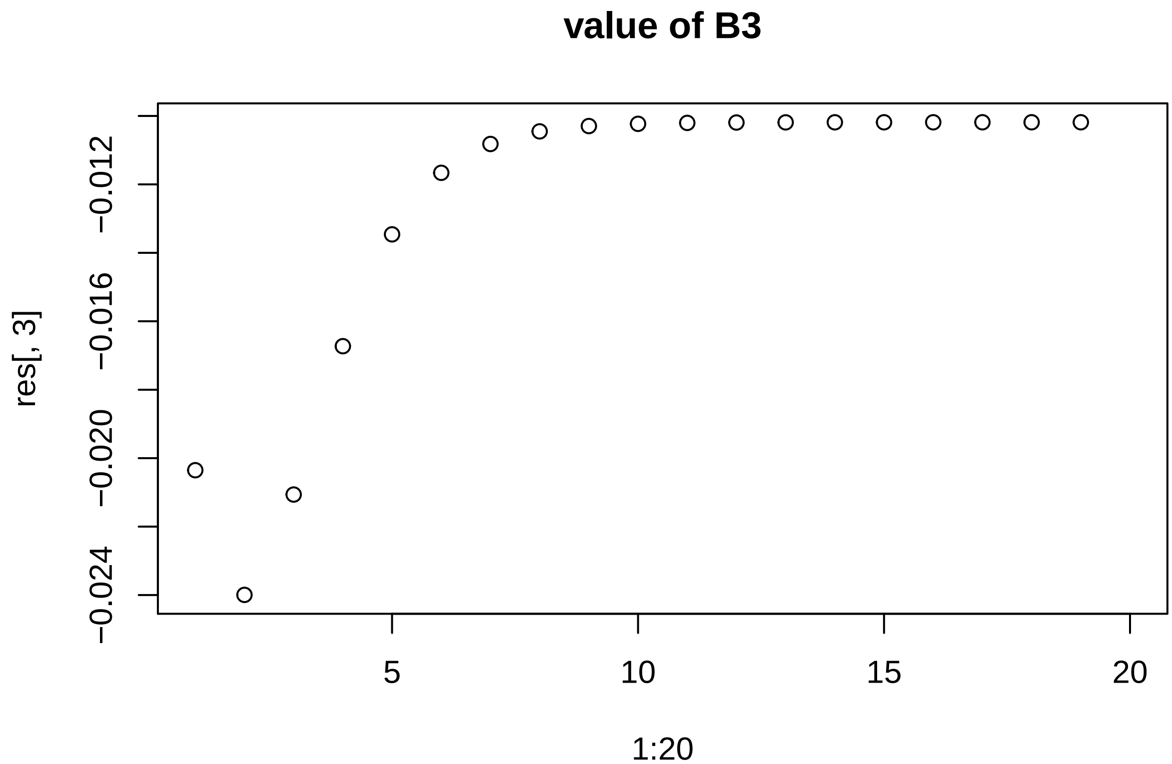
```
calculate_EM <- function(data,observation,B_t){
  classification <- data %*% B_t
  # Using QR decomposition
  QR <- qr(data)
  Q <- qr.Q(QR)
  R <- qr.R(QR)
  #E step
  E_zi <- ifelse(observation == 1,
                 classification + dnorm(classification)/pnorm(classification),
                 classification - dnorm(-classification)/(pnorm(-classification))
                 )
  #M step
  #B_t_1 <- (((t(data)%*%data + lambda*diag(4))^-1)%*%(t(data)%*%E_zi))
  B_t_1 <- backsolve(R, crossprod(Q, E_zi))
  return (B_t_1)
}

#Define some initial value to start the iteration algorithm
count <- 1
max_count <- 20
y <- ifelse(rnorm(1000)<0,0,1)
data <- matrix(rnorm(1000*4, sd = 10), nrow = 1000, ncol = 4)
data[,1] <- 1
data[,2] <- y + rnorm(1000, sd = 5)
B_t <- c(0,1,0,0)

res <- matrix(data=NA,nrow=20,ncol=4)

#Generate a loop and visualy interpret when convergence is occuring
while (count < max_count){
  B_t_1 <- calculate_EM(data,observation = y,B_t)
  count <- count + 1
  B_t <- B_t_1
  res[count-1,] <-B_t
}

#Plot results for the value of B
plot(1:20,res[,1], main = "value of B1")
```



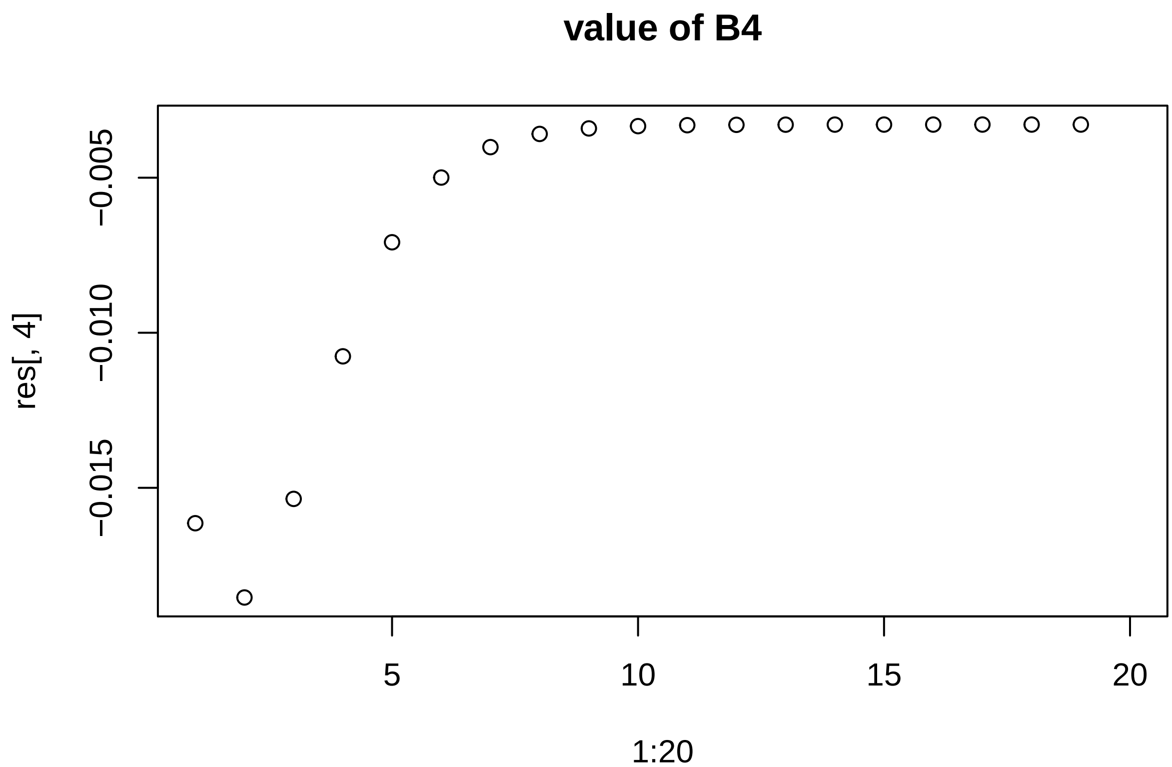

```r
plot(1:20,res[,2], main = "value of B2")
```




```r
plot(1:20,res[,3], main = "value of B3")
```




```
plot(1:20,res[,4], main = "value of B4")
```



Problem 3d Solve the problem by

```
count <- 1
max_count <- 20
```

```r
data <- matrix(rnorm(1000*4, sd = 10), nrow = 1000, ncol = 4)
data[,1] <- 1
data[,2] <- y + rnorm(1000, sd = 5)

observation <- ifelse(rnorm(1000)<0,0,1)
B_t <- c(0,1,0,0)

calculate_BFGS <- function(B_t){
  classification <- data %*% B_t
  #Calculate expectation and variance
  E_zi <- ifelse(observation == 1,
                 classification + dnorm(classification)/pnorm(classification),
                 classification - dnorm(-classification)/(pnorm(-classification))
                 )
  var_zi <- ifelse(observation==1,
                     1 - classification * dnorm(classification) / pnorm(classification) - (dnorm(classifi
                     1 + classification * dnorm(-classification) / pnorm(-classification) - (dnorm(-class

  #calculate log likelihood
  #log_liklihood <- -(1/2)*(E_zi%*%E_zi)+ B_t%*%c(1,1,1,1)
  log_liklihood <- -1/2*sum((E_zi-data%*%B_t)^2)
  return (-log_liklihood)
}

optim(par=c(0,1,0,0),calculate_BFGS,method = "BFGS")
```

```
## $par
## [1]  0.071300346 -0.009937004  0.003811126 -0.005281261
##
## $value
## [1] 315.4587
##
## $counts
## function gradient
##       41        8
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Expectation Maximization Edutt 243

↳ ... write the Expectation of z_i as

$$E(z_i/z_i>0) = x_i^T B + \frac{\phi(x_i^T B)}{\Phi(x_i^T \beta)} \qquad E(z_i/z_i \leq 0) = x_i^T B + \frac{\phi(x_i^T B)}{1-\Phi(x_i^T B)}$$

↳ Write variance of z_i as

$$Var(z_i/z_i>0) = 1 - (x_i^T B)\frac{\phi x_i^T B}{\Phi x_i^T \beta} - \left(\frac{\phi(x_i^T\beta)}{\Phi(x_i^T\beta)}\right)^2$$

$$= 1 + (x_i^T\beta)\frac{\phi(x_i^T\beta)}{1-\Phi(x_i^T B)} - \left(\frac{\phi(x_i^T\beta)}{1-\Phi(x_i^T\beta)}\right)^2$$

↳ likelihood function $f(y, z; \beta)$

$$f(y/z,\beta)\, f(z/\beta) = \prod_{i=1}^n I(z_i>0)\, e^{\left(-\frac{1}{2}(z_i - x_i^T\beta)^2\right)} = \boxed{\underbrace{\left(I(z_i>0)\right)^n}\left(\frac{1}{\sqrt{2\pi}}\right)^n e^{\left(-\frac{1}{2}(z_i - x_i^T\beta)^2\right)}}$$

zero out of terms are ...

Log - likelihood

$$\sum_{i=1}^n \cancel{n \log(1)} \underbrace{- \frac{n}{2}\log 2\pi}_{\text{constant value}} - \frac{1}{2}(z_i - x_i^T B)^2$$

* expected log likelihood becomes $E[\log(f(y, z/B))/y, B^t] =$

$$E\left[\sum_{i=1}^n -\frac{1}{2}(z_i - x_i^T B)^2 / y, B^t\right]$$

* Expand $= -\frac{1}{2}\left[\sum_{i=1}^n E\left[(z_i - x_i^T B)^T(z_i - x_i^T B)/y, B^t\right]\right] = -\frac{1}{2}\left[\sum_{i=1}^n E\left[\left(z_i^T z_i - 2 z_i x_i^T B + B^T x_i x_i^T B\right)/y, B^t\right]\right]$

$$-\frac{1}{2}\left(E[Z^T Z/y, B^t] - E[2ZXB] + E[B^T X X^T B]/y, B^t\right)$$

$$\frac{\partial E[\log(f(y, z/B))/y, B^t]}{\partial B} = -\frac{1}{2}\left(0 - 2E[ZX/y, B^t] + E[2X^T X B/y, B^t]\right) = 0$$

$E[X^T X B/y, B^t] = E[ZX/y, B^t]$ ~ Noticing that only Z is a random variable

$$\boxed{B = (X^T X)^{-1} X^T E[Z/y, B^t]}$$