HW5_TFang

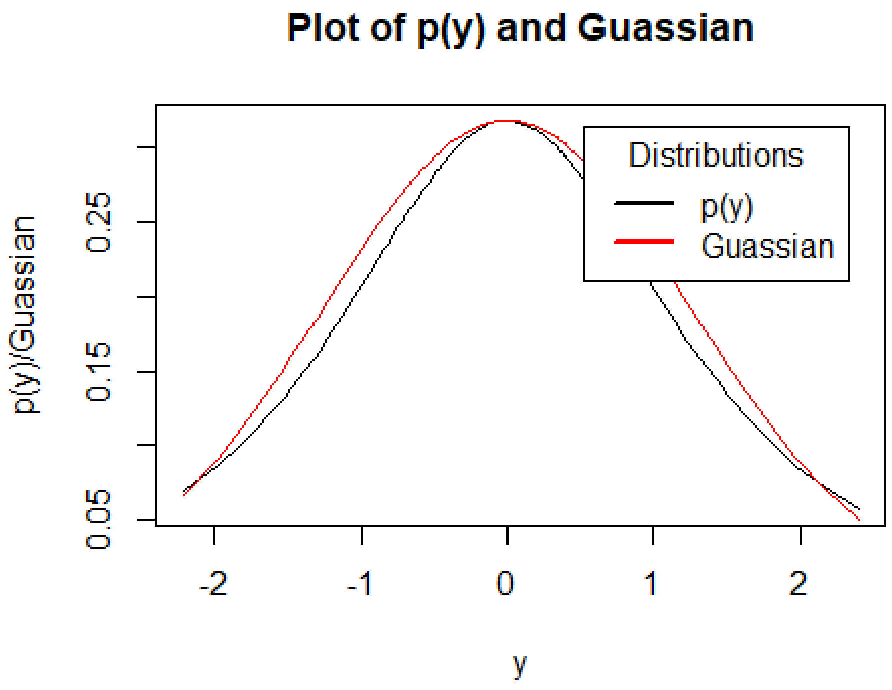
Tianyi Fang

October 29, 2017

Gradient descent for blind source separation

1.Plot p(y)

```
set.seed(1)
y <- sort(rnorm(100, 0,1))
p <- function(y) {1/(pi*cosh(y))}
set.seed(2)
g <- dnorm(y, 0, sqrt(pi/2))
plot(y, p(y), type = "l", ylab = "p(y)/Guassian", main = "Plot of p(y) and Gu
assian")
points(y, g, type = "l", col = "red")
legend("topright", inset = 0.05, title = "Distributions",c("p(y)","Guassian")
, lwd =2, lty = c(1,1),col = c("black", "red"))
```



2.

$logp(X|W) = Tlog(|W|) + \sum_{t=1}^{T}\sum_{s=1}^{3} l\, ogp(w^s x_t).\, logP(X|W) = log(\prod_t^T P\,(x_t|A^{-1}))$

$$= log(\prod_{t}^{T}\frac{1}{|A^{-1}|}P_y(\frac{x}{A^{-1}})) =$$

$$log(\prod_{t}^{T}|W|\,P_y(Wx_t)) =$$

$$log(\prod_{t}^{T}\prod_{s}^{3}|\,W|P_y(w^sx_t)) = \sum_{t}^{T}l\,og|W| + \sum_{t}^{T}\sum_{s}^{3}l\,ogP_y(w^sx_t)$$

$$= Tlog|W| + \sum_{t}^{T}\sum_{s}^{3}l\,ogP_y(w^sx_t)$$

3.Show that

$\frac{\partial logp(X|W)}{\partial w_{i,j}} = Ta_{j,i} + \frac{\partial logp(y_{i,t}}{\partial y_{i,t}}x_{j,t}$. Proof:

$$Wadj(W) = det(W)I, \frac{adj(w)}{det(W)} = \frac{I}{W} = W^{-1}$$

$$\frac{\partial log(|W|)}{\partial W_{ij}} = \frac{1}{det(W)}adj^T(W_{ij}) = W_{ij}^{-1} = a_{ji}$$

$\frac{\partial \sum_{t=1}^{T}\sum_{s=1}^{3}P(w^sx_t)}{\partial w_{ij}} = \frac{\partial \sum_{t=1}^{T}P(w_{ij}x_{jt})}{\partial y_{it}}x_{jt} = \frac{\partial \sum_{t=1}^{T}P(Y_{it})}{\partial y_{it}}x_{jt}$ For $w_{ij}x_{jt} = y_{it}, w_{ij} = \frac{y_{it}}{x_{jt}}$

4.Plug in the expression for the gradient to write the update rule.

$$W_{new} = W_old + \eta A_{old}^T + \frac{1}{T}(-tanh(Y)X^T))$$

5. Function of gradient descent

```
gradient_descent <- function (matrix, dl){
  #initialize w
  t <- dim(x)[2]#4900000
  set.seed(3)
  aa <- rnorm(9,0,1)
  A_old <- t(matrix(aa, ncol=3, nrow = 3))
  W_new <- solve(A_old) #W=A^{-1}
  #learning rate
  eta <- 1
  #stop criteria:Flag
  #max likelihood estimate
  mle <-c()
  #iteration
  iteration = 0
  stop_value = 10
  while(stop_value > 0.000001){
    iteration = iteration + 1
    W_old <- W_new
    Y_old <- W_old %*% x
    W_new <- W_old + eta*(t(A_old) + (1/t) *(-tanh(Y_old)%*%t(x)))
    eta = eta*dl
    Y_new <- W_new %*% x
    A_old <- solve(W_new)
    mle[iteration] = t*log(abs(det(W_new))) + sum(log(1/(pi*cosh(Y_new))))
    stop_value = abs(sum(W_new - W_old))
  }
  W_hat <- W_new %*% solve(sqCov)
  return(list(W_new, mle, iteration, W_hat))
}
```

loadin X

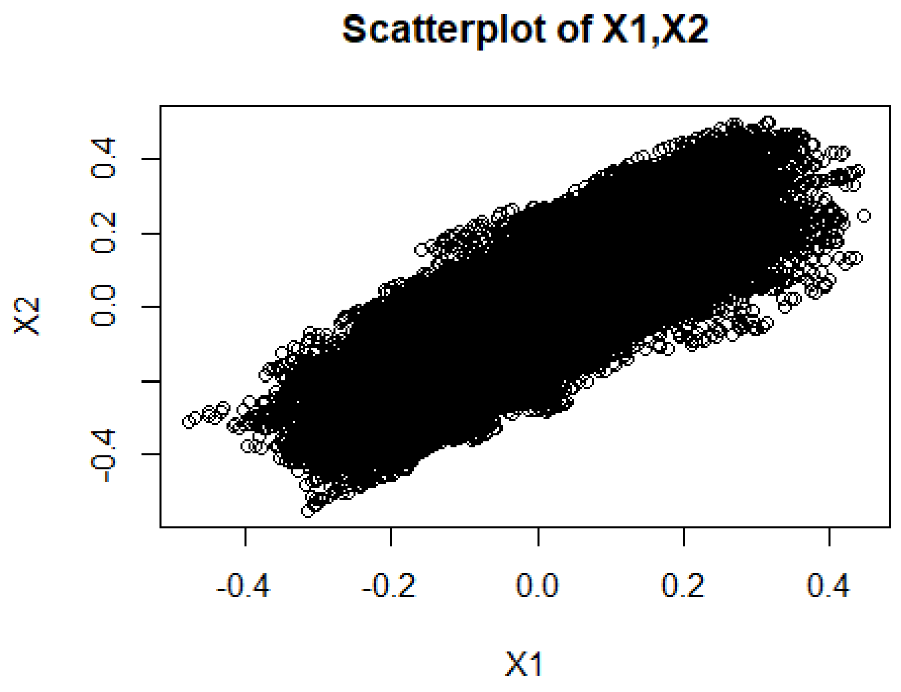
```
library(audio)

## Warning: package 'audio' was built under R version 3.4.1

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.4.1

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.4.1

X <- matrix(0, ncol=490000, nrow = 3)
w1 <- load.wave("mike1.wav")
w2 <- load.wave("mike2.wav")
w3 <- load.wave("mike3.wav")
X <- rbind(w1,w2,w3)
```

6. Write a 3x3 covariance matrix of X.

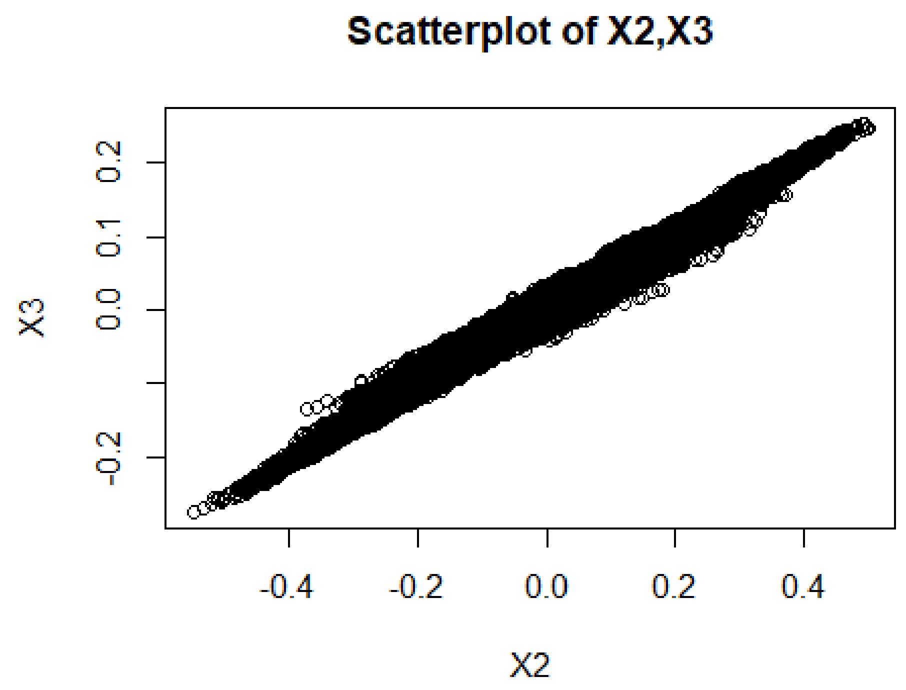
```
cov <- matrix(0, ncol=3, nrow=3)
t <- dim(X)[2]
for(i in (1:3)){
  for(j in (1:3)){
    cov[i,j] <- 1/t*(t(X[i,])%*%X[j,])-(1/t^2)*(sum(X[i,]*sum(X[j,])))
  }
}
print(cov)
```

```
##              [,1]        [,2]        [,3]
## [1,] 0.007731817 0.009049549 0.004389545
## [2,] 0.009049549 0.014568355 0.007381064
## [3,] 0.004389545 0.007381064 0.003785347
```

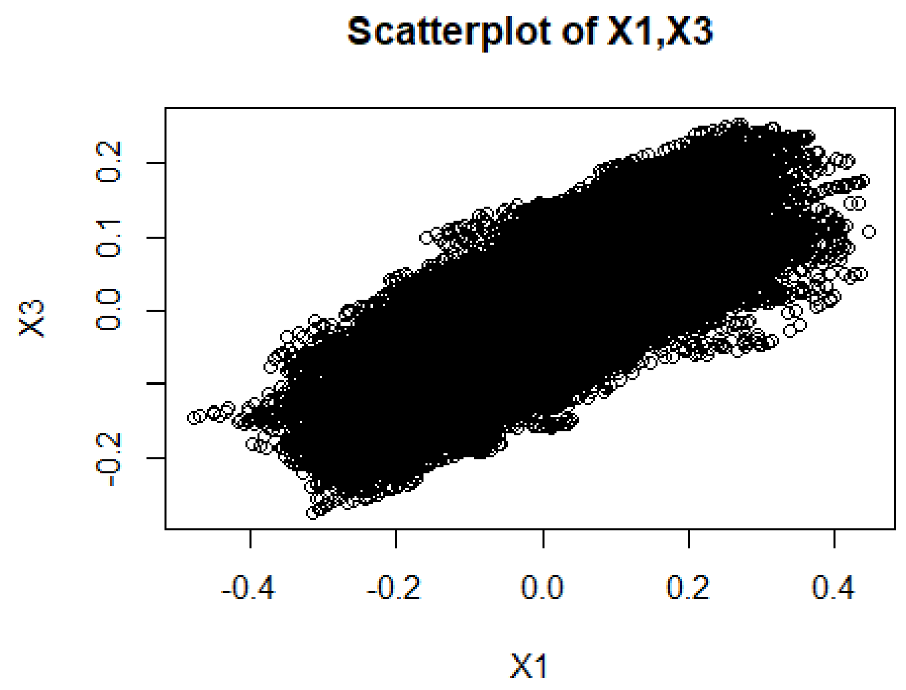
```
#plot

plot(X[1,],X[2,], xlab = "X1", ylab = "X2", main = "Scatterplot of X1,X2")
```



```
plot(X[2,],X[3,], xlab = "X2", ylab = "X3", main = "Scatterplot of X2,X3")
```



```
plot(X[1,],X[3,], xlab = "X1", ylab = "X3", main = "Scatterplot of X1,X3")
```



####7. get sqCov

```
library(expm)
```

```
## Warning: package 'expm' was built under R version 3.4.2

## Loading required package: Matrix

##
## Attaching package: 'expm'

## The following object is masked from 'package:Matrix':
##
##     expm

sqCov <- sqrtm(cov)

## Note: method with signature 'symmetricMatrix#missing' chosen for function
'Schur',
##  target signature 'dsyMatrix#missing'.
##  "dsyMatrix#ANY" would also be valid

X_white <- solve(sqCov)%*%X
cov_white <- matrix(0, ncol=3, nrow=3)
t_white <- dim(X_white)[2]
for(i in (1:3)){
  for(j in (1:3)){
    cov_white[i,j] <- 1/t*(t(X_white[i,])%*%X_white[j,])-(1/t^2)*(sum(X_white
[i,]*sum(X_white[j,])))
  }
}
cov_white

##               [,1]          [,2]          [,3]
## [1,]  1.000000e+00 -9.370382e-13  1.363192e-12
## [2,] -9.370382e-13  1.000000e+00 -9.817656e-12
## [3,]  1.363192e-12 -9.817656e-12  1.000000e+00

#almost =1
```

If we calculate the covariance matrix of X_white, we can see the covariance matrix of X_white is approximately identity matrix.

8.Run gradient_descent on white data. How to initialize W? set eta, stop_sign, get the # of iteration, Plot the evoluation of log-likelihood. What?

```
gradient_descent <- function (x, dl){
  #initialize w
  t <- dim(x)[2]#4900000
  set.seed(3)
  aa <- rnorm(9,0,1)
  A_old <- t(matrix(aa, ncol=3, nrow = 3))
  W_new <- solve(A_old) #W=A^{-1}
  #learning rate
  eta <- 1
  #stop criteria:Flag
  #max likelihood estimate
  mle <-c()
  #iteration
  iteration = 0
  stop_value = 10
  while(stop_value > 0.000001){
    iteration = iteration + 1
    W_old <- W_new
    Y_old <- W_old %*% x
    W_new <- W_old + eta*(t(A_old) + (1/t) *(-tanh(Y_old)%*%t(x)))
    eta = eta*dl
    Y_new <- W_new %*% x
    A_old = solve(W_old)
    mle[iteration] = t*log(abs(det(W_new))) + sum(log(1/(pi*cosh(Y_new))))
    stop_value = abs(sum(W_new - W_old))
    A_old = solve(W_old)
    #print(W_new)
  }
  W_hat <- W_new %*% solve(sqCov)
  return(list(W_new, mle, iteration, W_hat))
}

#
white_0.9 <- gradient_descent(X_white, 0.9)
white_0.7 <- gradient_descent(X_white, 0.7)
white_0.5 <- gradient_descent(X_white, 0.5)
#W_new_0.9 = white_0.9[[1]]
mle_white_0.9 <-white_0.9[[2]]
iteration_0.9 <- white_0.9[[3]]
W_hat_white_0.9 <- white_0.9[[4]]
#W_new_0.7 = white_0.9[[1]]
mle_white_0.7 <-white_0.7[[2]]
iteration_0.7 <- white_0.7[[3]]
W_hat_white_0.7 <- white_0.7[[4]]
#W_new_0.5 = white_0.5[[1]]
mle_white_0.5 <-white_0.5[[2]]
iteration_0.5 <- white_0.5[[3]]
W_hat_white_0.5 <- white_0.5[[4]]
```

How initialize W I initialized my W as the inverse of A, where A is a 3x3 matrix contain random normally distributed values from 0:1 How set eta/stop_cretiera I set my learning rate as 1 and every iteration I decrease my learning rate by 10%, that is, every iteration it will descent in a smaller steps. I also tried 0.7, 0.5 to see whether how the steps of learning rate will affect the converge trend of mle.\ My stop cretiera is: when W_new has little different from W_old, which means converge and get the min point, so we stop. How many iteration For eta=0.9*eta, eta=0.7*eta, eta=0.5*eta

```
ite <- c(iteration_0.9, iteration_0.7, iteration_0.5)
ite

## [1] 108  39  21
```

So we can see the larger step learning rate change, the less iteration time it will need to converge. Plot evolution of mle

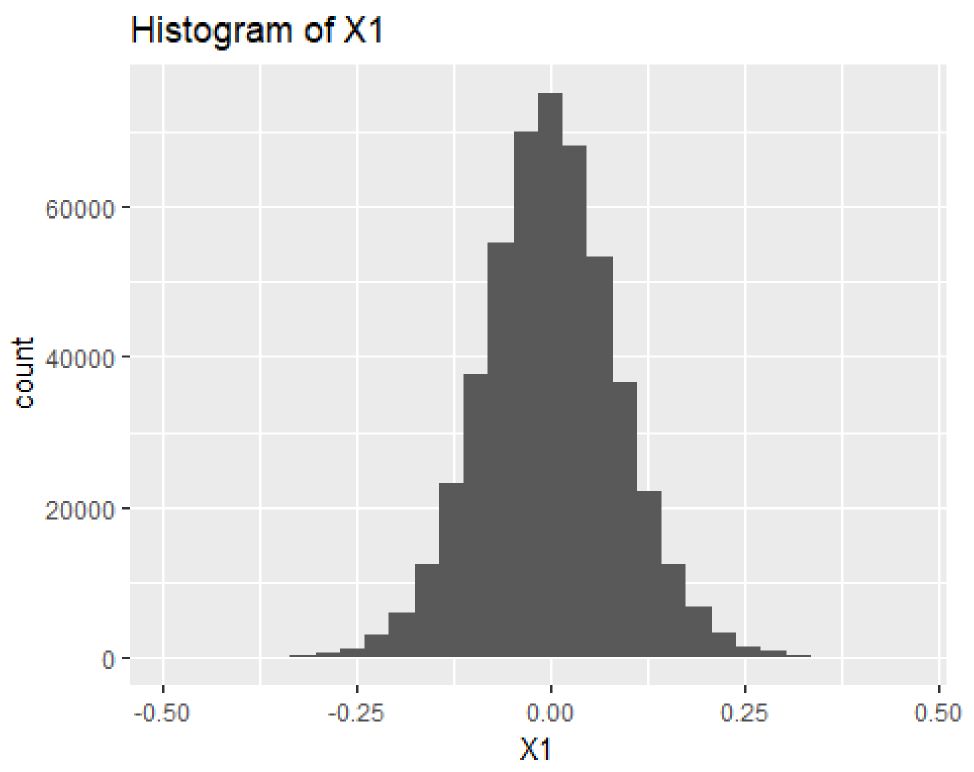
```
mle <-as.data.frame(mle_white_0.9)
mle$iteration <- c(1:108)
mle$eta0.7 <- c(mle_white_0.7, rep_len(max(mle_white_0.7), 108-39))
mle$eta0.5 <- c(mle_white_0.5, rep_len(max(mle_white_0.5), 108-21))
colnames(mle)<- c("eta0.9","eta0.7","eta0.5")
library(reshape2)

## Warning: package 'reshape2' was built under R version 3.4.1

#mle.melt <- melt(mle, id.var = "iteration")
#mle.melt%>%
 # ggplot(aes(x=iteration, y = value, color = variable)) + geom_line(size = 2
) + labs(y="Max Likelihood Estimate",title = "Different Change rate of Learni
ng rate vs MLE")
```

To sum up, with larger step of learning rate, The converge will be faster but the value of MLE is less than smaller step pf learning rate. For example, if learning rate reduce 0.5 every time, it is easily to go over the min MLE, and end up with local optima. Return the W_hat

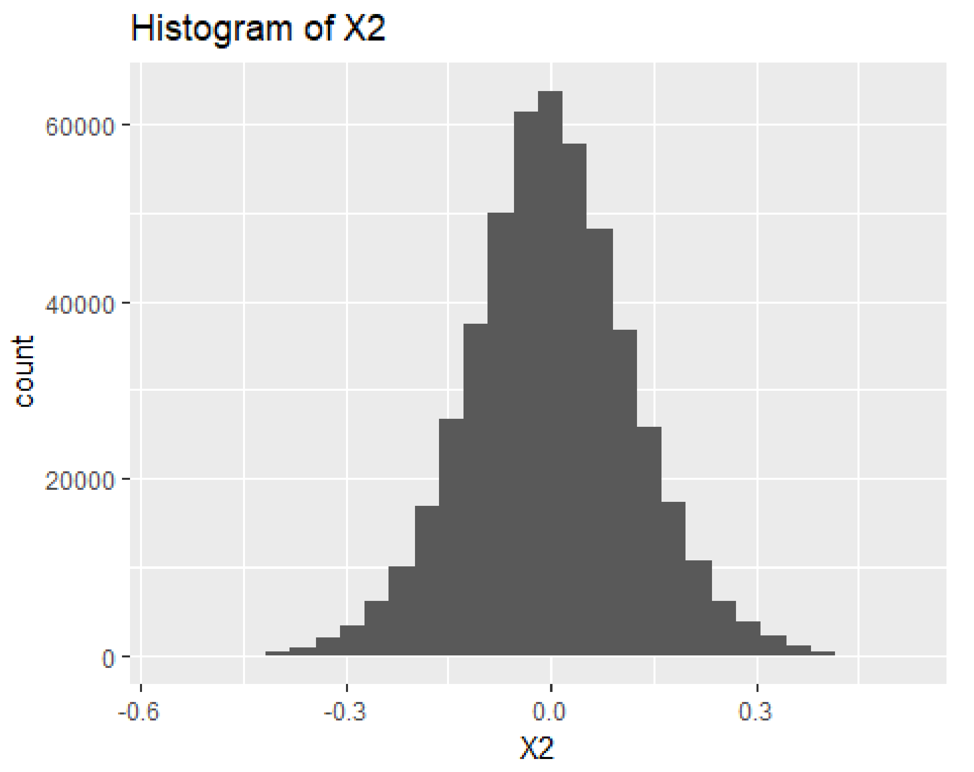
```
print(W_hat_white_0.5)

##             [,1]      [,2]      [,3]
## [1,]   -6.892822 -73.41449  124.7718
## [2,] -108.988308 144.66397 -131.1854
## [3,] -120.497252 435.68729 -716.9090

print(W_hat_white_0.7)

##             [,1]      [,2]       [,3]
## [1,]   -5.206619 -71.89902  120.94370
## [2,]  -90.283603  98.02767  -63.66504
## [3,] -102.233137 390.46704 -651.59875

print(W_hat_white_0.9)

##            [,1]      [,2]       [,3]
## [1,]  -2.150677 -52.70061   84.63622
## [2,] -34.026429  10.09271   31.02256
## [3,] -40.800239 185.87519 -322.15781
```

plot a histogram of raw, white data, recover data, Here I only plot the result of eta =eta*0.9

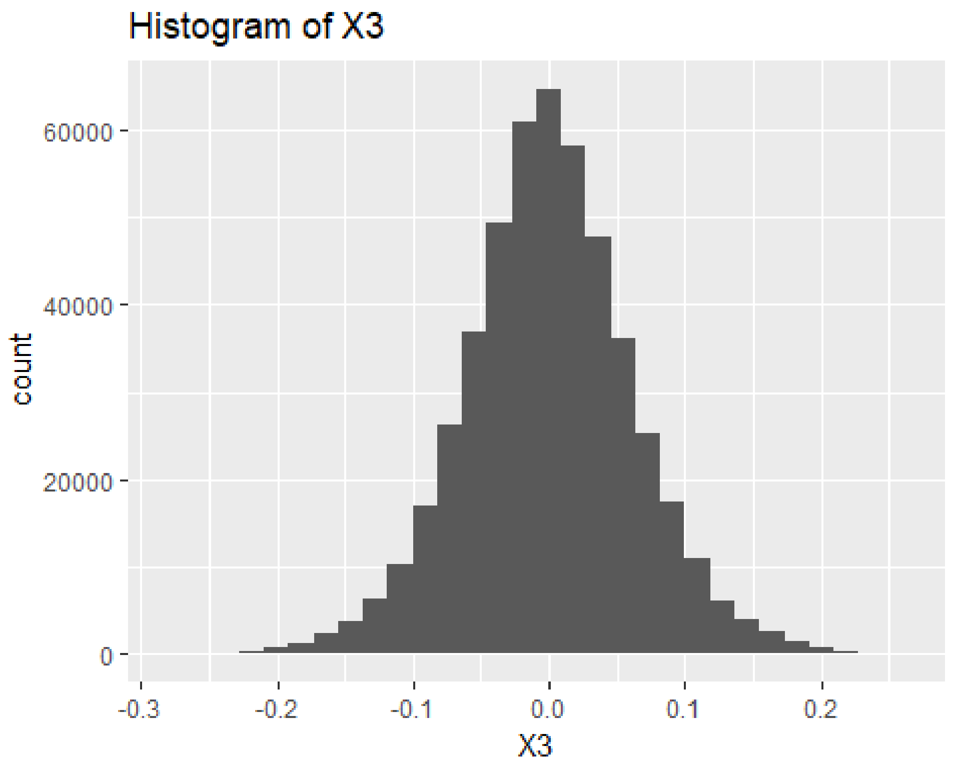
```
#raw data
X.df <- as.data.frame(t(X))
colnames(X.df) <- c("X1","X2","X3")
X.df %>% ggplot(aes(X1))+geom_histogram()+ labs(title = "Histogram of X1")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



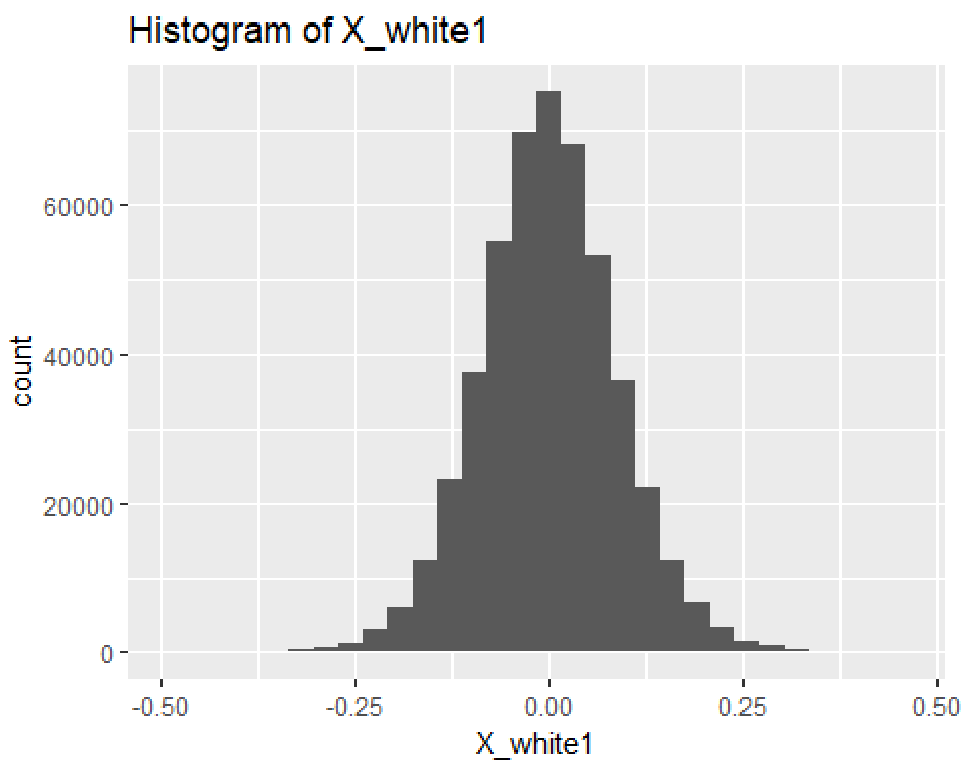
```
X.df %>% ggplot(aes(X2))+geom_histogram()+ labs(title = "Histogram of X2")
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



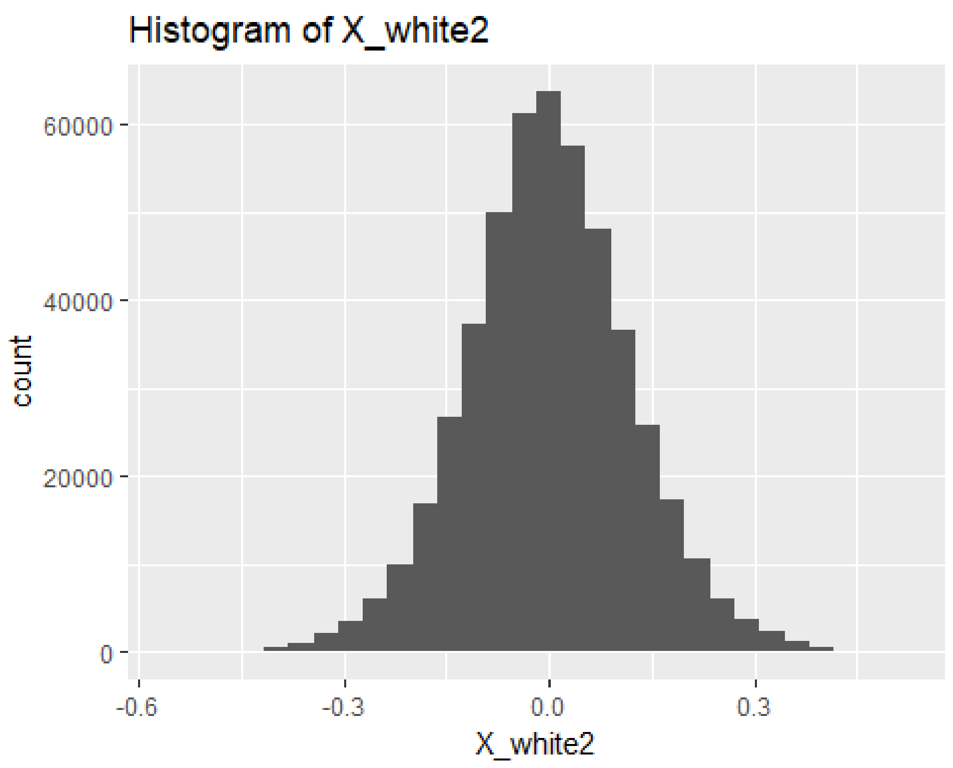
```
X.df %>% ggplot(aes(X3))+geom_histogram()+ labs(title = "Histogram of X3")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



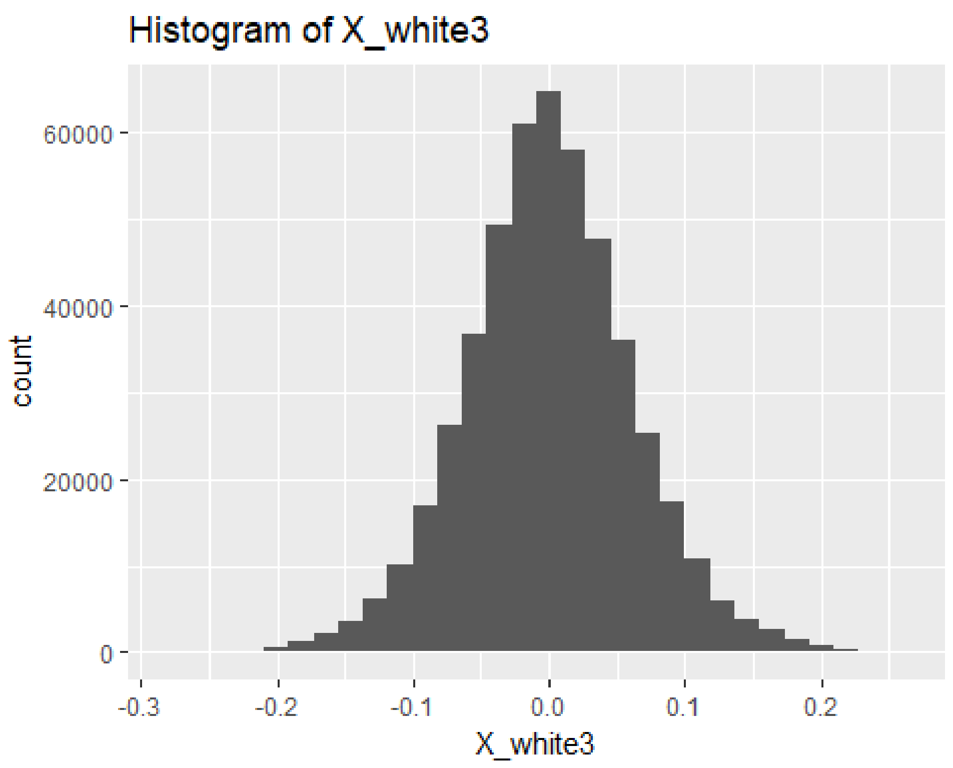
```
#White data
X_white.df <- as.data.frame(t(X))
colnames(X_white.df) <- c("X_white1","X_white2","X_white3")
X_white.df %>% ggplot(aes(X_white1))+geom_histogram()+ labs(title = "Histogra
m of X_white1")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



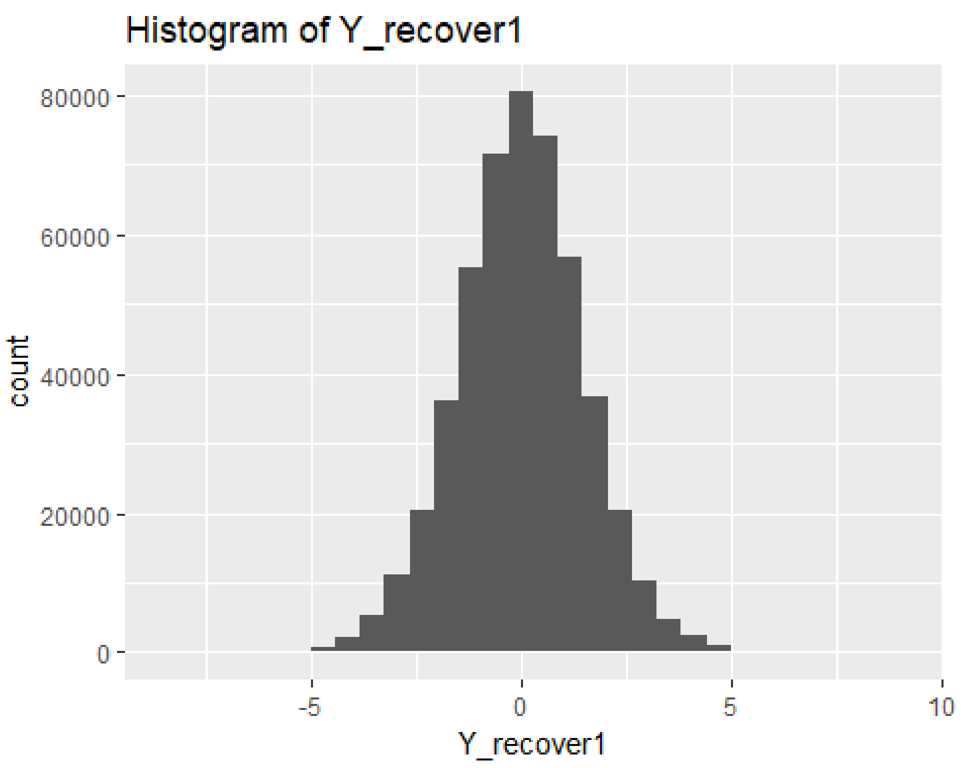
```
X_white.df %>% ggplot(aes(X_white2))+geom_histogram()+ labs(title = "Histogra
m of X_white2")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



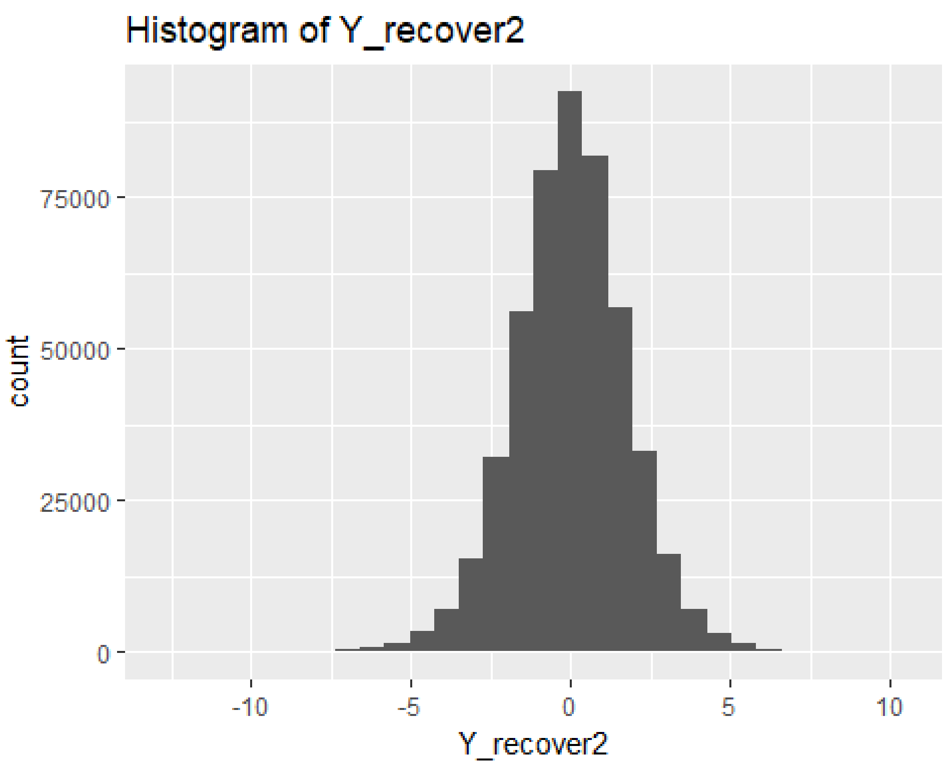
```
X_white.df %>% ggplot(aes(X_white3))+geom_histogram()+ labs(title = "Histogra
m of X_white3")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



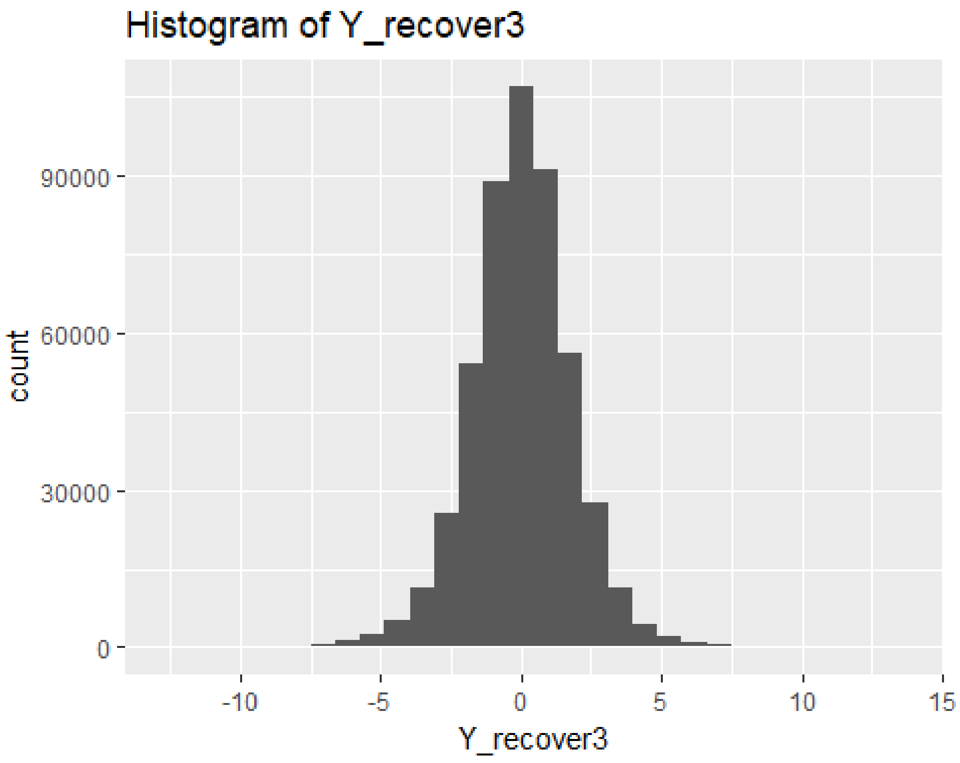
```
#recovered data
recover_Y <- W_hat_white_0.9 %*% X
recover.df <- as.data.frame(t(recover_Y))
colnames(recover.df) <- c("Y_recover1","Y_recover2","Y_recover3")
recover.df %>% ggplot(aes(Y_recover1))+geom_histogram()+ labs(title = "Histog
ram of Y_recover1")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
recover.df %>% ggplot(aes(Y_recover2))+geom_histogram()+ labs(title = "Histog
ram of Y_recover2")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

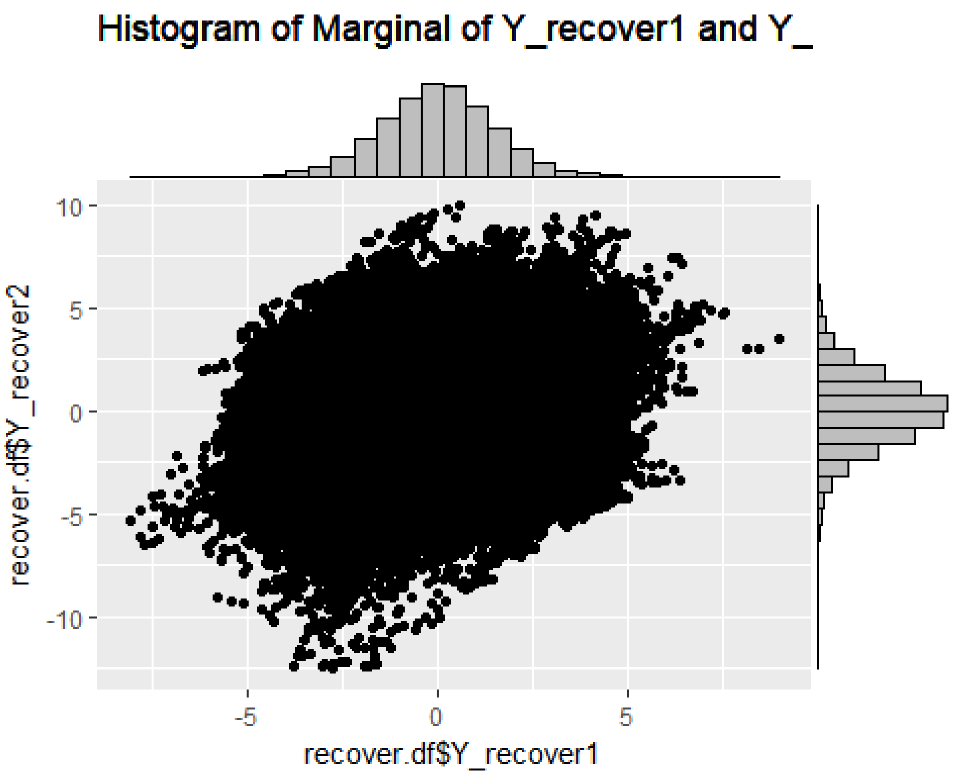


```
recover.df %>% ggplot(aes(Y_recover3))+geom_histogram()+ labs(title = "Histog
ram of Y_recover3")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

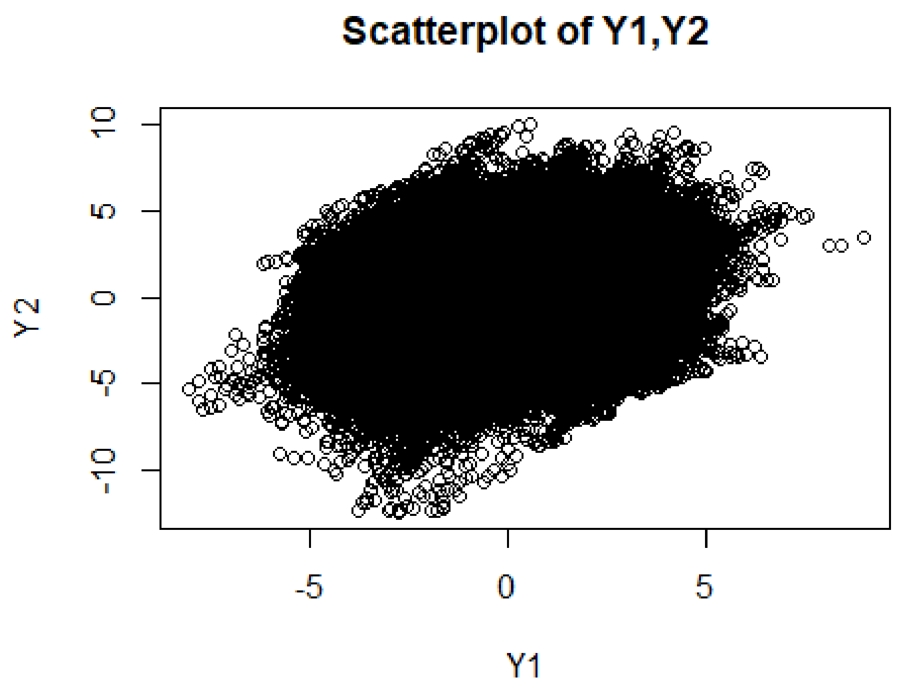


####10. Plot marginals for each source, What is the covarience?Plot three pairwise scatterplots

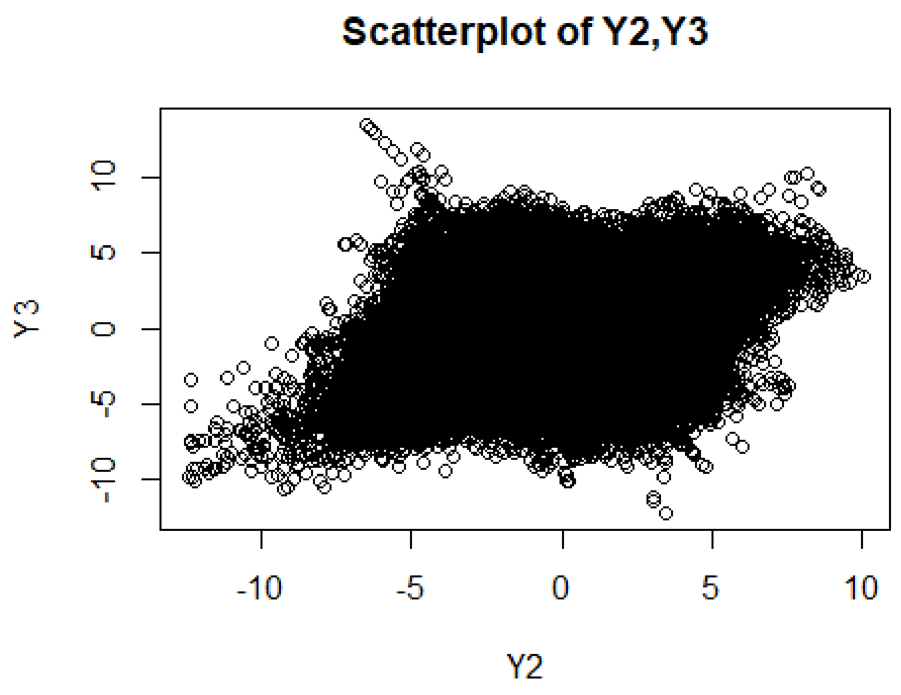
```
library(ggExtra)

## Warning: package 'ggExtra' was built under R version 3.4.2

#marginal of Y1,Y2
p1 <- ggplot(recover.df, aes(recover.df$Y_recover1, recover.df$Y_recover2))+g
eom_point()+ labs(title = "Histogram of Marginal of Y_recover1 and Y_recover2
")
ggMarginal(p1, type = "histogram")

#marginal of Y2,3
p2 <- ggplot(recover.df, aes(recover.df$Y_recover2, recover.df$Y_recover3))+g
eom_point()+ labs(title = "Histogram of Marginal of Y_recover2 and Y_recover3
")
ggMarginal(p1, type = "histogram")

#marginal of Y1,3
p3 <- ggplot(recover.df, aes(recover.df$Y_recover1, recover.df$Y_recover3))+g
eom_point()+ labs(title = "Histogram of Marginal of Y_recover1 and Y_recover3
")
ggMarginal(p1, type = "histogram")
```



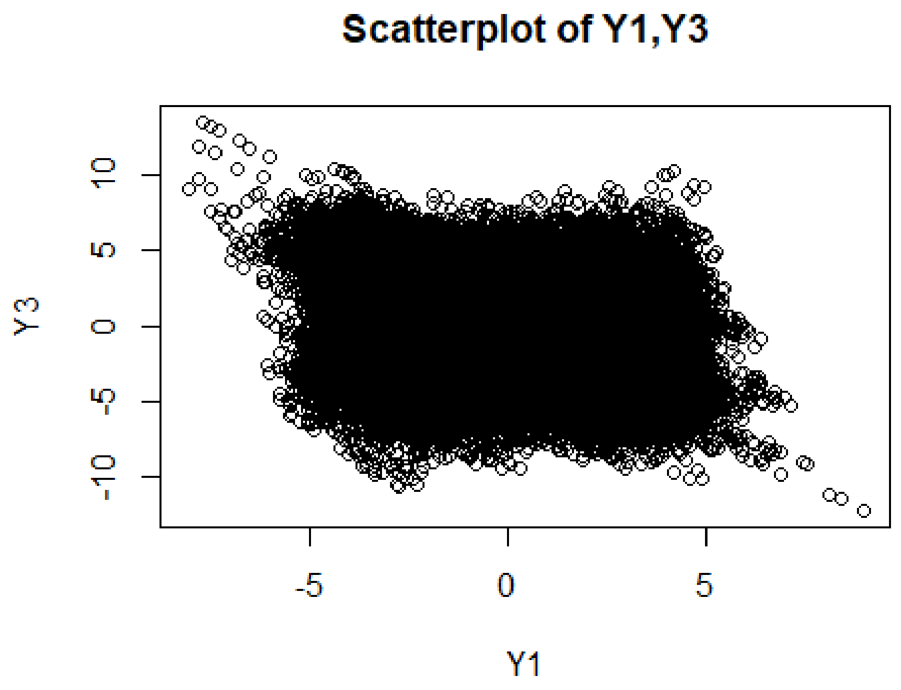
```r
#cov
cov_y <- matrix(0, ncol=3, nrow=3)
t_y <- dim(recover_Y)[2]
for(i in (1:3)){
  for(j in (1:3)){
    cov_y[i,j] <- 1/t*(t(recover_Y[i,])%*%recover_Y[j,])-(1/t_y^2)*(sum(recov
er_Y[i,]*sum(recover_Y[j,])))
  }
}
cov_y

##             [,1]       [,2]        [,3]
## [1,]  2.22145535 0.09072695 -0.08498905
## [2,]  0.09072695 3.21825389  0.39491844
## [3,] -0.08498905 0.39491844  3.22742529

#
plot(recover_Y[1,],recover_Y[2,], xlab = "Y1", ylab = "Y2", main = "Scatterpl
ot of Y1,Y2")
```



```
plot(recover_Y[2,],recover_Y[3,], xlab = "Y2", ylab = "Y3", main = "Scatterpl
ot of Y2,Y3")
```



```
plot(recover_Y[1,],recover_Y[3,], xlab = "Y1", ylab = "Y3", main = "Scatterpl
ot of Y1,Y3")
```



####11.Is the MLE unique? Explain

```
norm_y <- matrix(0, nrow= 3, ncol = dim(X)[2])
for(i in 1:3){
  norm_y[i,] <- recover_Y[i,]/(2*max(recover_Y[i,]))
}
save.wave(norm_y[1,], where = "C:\\Users\\Tianyi Fang\\Desktop\\stat545\\hw5\
\recover1.wav")
save.wave(norm_y[2,], where = "C:\\Users\\Tianyi Fang\\Desktop\\stat545\\hw5\
\recover2.wav")
save.wave(norm_y[3,], where = "C:\\Users\\Tianyi Fang\\Desktop\\stat545\\hw5\
\recover3.wav")
```

MLE of W is not always unique. For a convex-optimization problem, it will always find the global optimum, which is unique. But in general, It also depends on the choices of your step size(learning rate), the initial set of W. If you want to find the global optimum, that is a NP - hard problem. Unless you tried as many as possible, you cannot decide whether it is local or global optima.