Assignment 5

Songze Yang u7192786

Question 1

Consider two p-dimensional populations with covariance matrices I_p and $(Ip + \Delta)$ where

$$\Delta := diag(\delta_1, \delta_2, 0, .., 0)$$

with $\delta_1, \delta_2 \in .R$. Suppose we had p-dimensional random samples $x_1, ..., x_{m+1} \sim Np(0, I_p)$ from the first population and p-dimensional random samples $z_1, ..., z_n + 1 \sim N(0, I_p + \Delta)$ from the second. We stack these random samples to obtain the data matrices X and Z and sample covariance matrices

$$S_1 := \frac{1}{m} XX^T,\ S_2 := \frac{1}{n} ZZ^T,\ S := S_2^{-1} S_1.$$

a. Assume $n, m, p \to \infty$ such that $y_p := p/n \to y \in (0, 1)$ and $c_p := p/m \to c > 0$. Take $\delta_1 = \delta_2 = 0$, $y = 1/4$, and $c = 3/4$, what is the lower bound a and the upper bound b of the limiting spectral distribution of S? For each, give a formula in terms of c and y. Also give a numerical value.

Answer to question 1 (a): When the $\delta_1 = \delta_2 = 0$, the $I_p + \Delta = I_p$. We can see that the X and Z satisfy the null case where $\Sigma_1 = \Sigma_2 = I_p$. Thus, this falls in the case that the matrix entry x_{ij} are $i.i.d.$ random variables with mean zero and variance 1, $y_p := p/n \to y \in (0, 1)$ and $c_p := p/m \to c > 0$. The Fisher LSD $f_{c,y}$ is

$$f_{c,y}(x) = \begin{cases} \frac{(1-y)\sqrt{(b-x)(x-a)}}{2\pi x(c+xy)} & \text{when } a \le x \le b, \\ 0 & \text{otherwise,} \end{cases}$$

where the upper bound a and lower bound b is:

$$a = \left(\frac{1-h}{1-y}\right)^2 \quad b = \left(\frac{1+h}{1-y}\right)^2 \quad h = \sqrt{c + y - cy}$$

Plugging in $c = \frac{3}{4}$ and $y = \frac{1}{4}$, we have:

```
c = 3/4
y = 1/4
h = sqrt(c + y - c*y)
a = (1 - h)^2/(1 - y)^2
b = (1 + h)^2/(1 - y)^2
a
```

```
## [1] 0.01728776
```

```
b
```

```
## [1] 6.427157
```

We have $a = 0.01729$ and $b = 6.42716$.

b. Suppose that $\delta_1 = -\epsilon$ and $\delta_2 = +\epsilon$ for $\epsilon = 1/10$. Would you expect S to have eigenvalues smaller than a and larger than b in that case?

Answer to question 1 (b): By theorem 3.1 of **[B]**, we have the phase transition of the extreme eigenvalues of Fisher matrix F follows:

$$\lambda_i = \begin{cases} \phi(a_i), & \text{if } |a_i - \gamma| > \gamma\sqrt{c + y - cy} \\ b, & \text{if } 1 < a_i \leq \gamma\{1 + \sqrt{c + y - cy}\} \\ b_1, & \text{if } \gamma\{1 - \sqrt{c + y - cy}\} \leq a_i < 1 \end{cases}$$

Where the ϕ function is in the form of:

$$\phi(x) = \frac{\gamma x(x - 1 + c)}{x - \gamma}, \quad x \neq \gamma$$

The $\gamma = 1/(1 - y) \in (1, \infty)$. The paper assume the spike eigenvalues a_i is for Ω_M in Σ_p:

$$\Sigma_p = \begin{pmatrix} \Omega_M & 0 \\ 0 & I_{p-M} \end{pmatrix}$$

which can be written in form of $\Omega_M = U \operatorname{diag}(a_1, \dots, a_1, \dots, a_k, \dots, a_k) U^*$, where U is a $M \times M$ orthogonal matrix.

There is a mismatch between our notation and the notation in the paper. In the paper **[B]**, it is assume that $\Sigma_1 = \Sigma_p$ and $\Sigma_2 = I_p$. However, we assume the opposite in our case. Therefore, we will need to scale the value in our case properly. The Fisher matrix $F = S_2^{-1} S_1$ are invariant under the transformation $S_1 \mapsto \Sigma_2^{-1/2} S_1 \Sigma_2^{-1/2}$, $S_2 \mapsto \Sigma_2^{-1/2} S_2 \Sigma_2^{-1/2}$.

Thus, let's scale the S_2 by $\Sigma_2^{-1/2} = (I_p + \Delta)^{-1/2} = diag(\frac{1}{\sqrt{1+\delta_1}}, \frac{1}{\sqrt{1+\delta_2}}, 1, ..., 1)$. Thus, $\Sigma_2^{-1/2} S_1 \Sigma_2^{-1/2}$ will be equal to $diag(\frac{1}{(1+\delta_1)}, \frac{1}{(1+\delta_2)}, 1, ..., 1)$.

In our formulation, the Ω_M is equal to $diag(\frac{1}{1+\delta_1}, \frac{1}{1+\delta_2})$, where the eigenvalues of this matrix are $a_1 = \frac{1}{1+\delta_1}, a_2 = \frac{1}{1+\delta_2}$. Thus, Let's check their phase transition.

```
gamma = 1/(1 - y)
a1 = 1/(1 - 1/10)
a2 = 1/(1 + 1/10)
abs(a1 - gamma) > gamma*h
```

```
## [1] FALSE
```

```
abs(a2 - gamma) > gamma*h
```

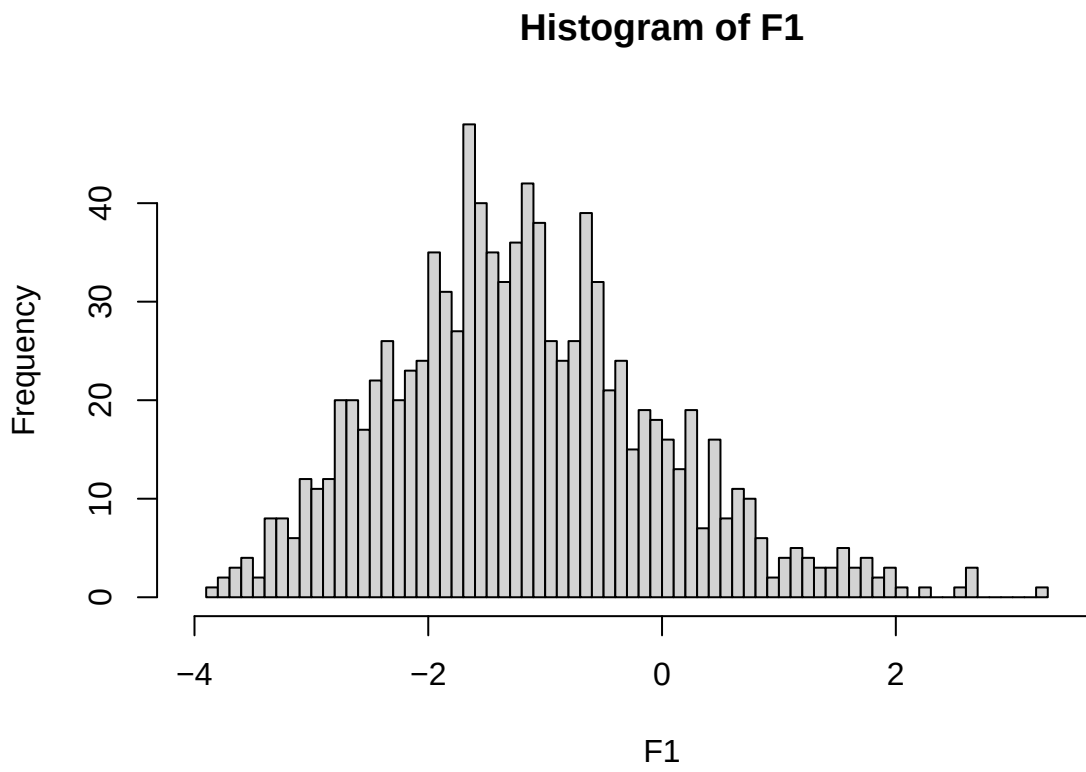
```
## [1] FALSE
```

Therefore, we do not expect S to have eigenvalues smaller than a and larger than b.

(c) In the paper [A] (see also [B]) it is suggested that the largest eigenvalue λ_1 of $\mathbf{S}$, scaled as $\frac{\lambda_1 - b}{sp}$ where b is from question (a) and $s_p := \left(\frac{1}{m}(\sqrt{m}+\sqrt{p})(\sqrt{\frac{1}{m}}+\frac{1}{\sqrt{p}})\right)^{1/3}$, behaves like a Tracy-Widom distribution of order 1. Show this using a simulation in the case $n = 400$, $y_n = \frac{1}{4}$ and $c_n = \frac{3}{4}$. Plot the histogram and compare it against the Tracy-Widom distribution of order 1.

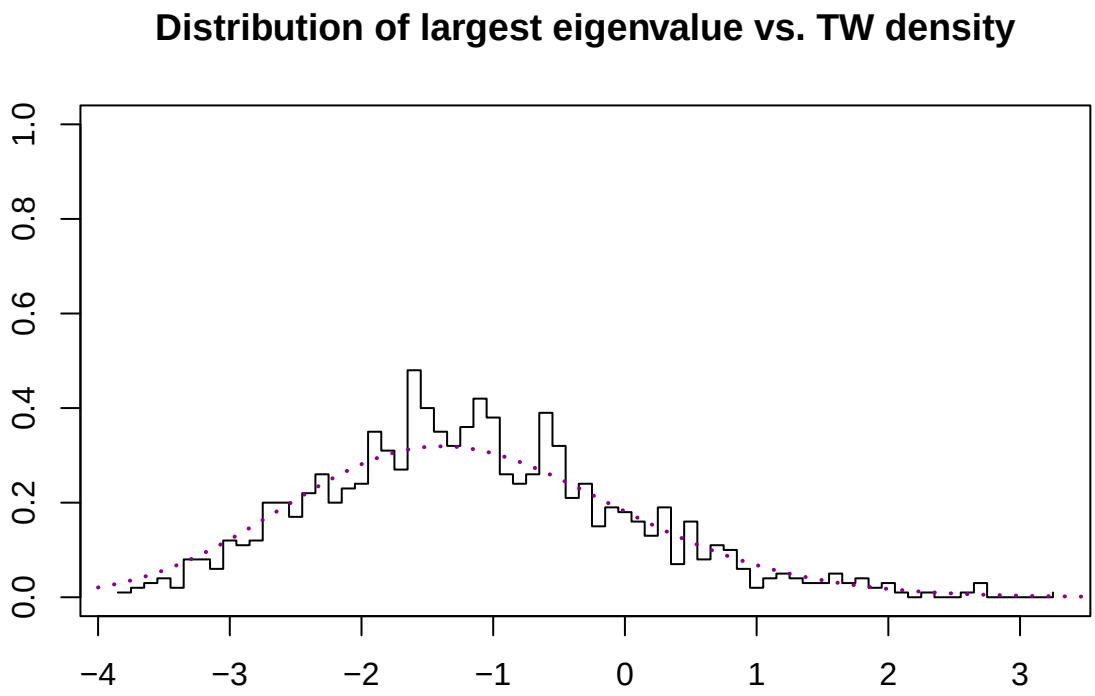
```
n = 400
sim.F.max.eigenvalues = function(delta_1, delta_2, n = 400, c, y, n_sims = 1) {
  p = n * y
  m = p / c
  evalues = c()
  for (sim in 1:n_sims) {
    Sigma1 = diag(p)
    Sigma2 = diag(p) + diag(x = c(delta_1, delta_2, rep(0, p - 2)), p)
    mu = rep(0, p)
    X = rmvn(m+1, mu, Sigma1)
    Z = rmvn(n+1, mu, Sigma2)
    S1 = cov(X)
    S2 = cov(Z)
    FF = solve(S2) %*% S1
    evalues = c(evalues, max(eigen(FF, only.values = TRUE)$values))
    }
  evalues
}
p = n * y
m = p / c
s_p = (1/m * (sqrt(m) + sqrt(p)) * (1/sqrt(m) + 1/sqrt(p)))^{1/3}
lambdas = sim.F.max.eigenvalues(delta_1 = -1/10, delta_2 = 1/10, c = c, y = y, n_sims = 1000)
```

```
F1 = (lambdas - b) / s_p
hhh <- hist(F1, breaks=80, xlim=c(-4,4))
```



Answer to question 1 (c):

```
plot(hhh$mids, hhh$density, type="s", xlab="", ylab="", ylim=c(0,1))
curve(dtw, -4, 4, lwd=2, lty=3, col="darkmagenta", add=TRUE)
title(main="Distribution of largest eigenvalue vs. TW density")
```



(d) Considering [B], suppose that $\delta_1 < \ell$ and $\delta_2 > \kappa$ for some choice of ℓ and κ. What would be the critical values of ℓ and κ that would ensure you would have a large fundamental spike and a small fundamental spike? Give a formula for ℓ and κ and also give a numerical value in the case $y = \frac{1}{4}$ and $c = \frac{3}{4}$.

Answer to question 1 (d): From

$$|a_i - \gamma| > \gamma\sqrt{c + y - cy}$$

The critical values are $a_i > \gamma * (1 + h)$ and $a_i < \gamma * (1 - h)$ for $a_1 = \frac{1}{1+\delta_1}, a_2 = \frac{1}{1+\delta_2}$. Let's first explore the ranges: $\sqrt{c + y - cy} > 0$, $\gamma > 1$, thus $\gamma * (1 + h) > 0$. Assume $0 \leq 1 + \delta_1 \leq 1 + \delta_2 \rightarrow \frac{1}{1+\delta_1} \geq \frac{1}{1+\delta_2} > 0$ and $h < 1 \rightarrow 1 - h > 0$, we have the following:

$$\begin{cases} \frac{1}{1+\delta_1} & > \gamma(1+h) \\ \frac{1}{1+\delta_2} & < \gamma(1-h) \end{cases} \Rightarrow \begin{cases} \delta_1 & < \frac{1}{\gamma(1+h)} - 1 \\ \delta_2 & > \frac{1}{\gamma(1-h)} - 1 \end{cases}$$

Assume $1 + \delta_1 \leq 0 \leq 1 + \delta_2 \rightarrow \frac{1}{1+\delta_1} \leq \frac{1}{1+\delta_2}$ and $h < 1 \rightarrow 1 - h > 0$, we have the following:

$$\begin{cases} \frac{1}{1+\delta_2} & > \gamma(1+h) \\ \frac{1}{1+\delta_1} & < \gamma(1-h) \end{cases} \Rightarrow \begin{cases} \delta_2 & < \frac{1}{\gamma(1+h)} - 1 \\ \delta_1 & > \frac{1}{\gamma(1-h)} - 1 \end{cases}$$

Otherwise, we do not have ℓ and κ simultaneously. We can see that upper critical value is always $\frac{1}{\gamma(1-h)} - 1$ and lower critical value is always $\frac{1}{\gamma(1+h)} - 1$.

```
ell <- 1/(gamma*(1 + h)) - 1
kappa <- 1/(gamma*(1 - h)) - 1
ell
```

```
## [1] -0.6055513
```

```
kappa
```

```
## [1] 6.605551
```

If $y = \frac{1}{4}$ and $c = \frac{3}{4}$, we have $\ell = -0.60555$ and $\kappa = 6.60555$.

(e) Suppose that $\delta_1 = \ell - \frac{1}{100}$ and $\delta_2 = \kappa + \frac{1}{100}$ for your critical values of κ and ℓ you found in (d), then give a formula for each of the two locations where you think the spike eigenvalues will cluster around and also a numerical value for each.

Also, perform a simulation experiment to illustrate this phenomena. That is, sample data and plot a histogram of eigenvalues of $\mathbf{S}$, compare it to the theoretical density expected if $\delta_1 = \delta_2 = 0$, and plot the location where you expect spike eigenvalues to cluster around. Take $n = 400$, $y_n = \frac{1}{4}$, and $c_n = \frac{3}{4}$.

Answer to question 1 (e): Let's plug $a_1 = \frac{1}{1+\delta_1}, a_2 = \frac{1}{1+\delta_2}$ into

$$\phi(x) = \frac{\gamma x(x - 1 + c)}{x - \gamma}, \quad x \neq \gamma$$

Thus, we get the

$$\phi(\delta) = \frac{\gamma \frac{1}{1+\delta}(\frac{1}{1+\delta} - 1 + c)}{\frac{1}{1+\delta} - \gamma}, \quad \frac{1}{1+\delta} \neq \gamma$$

Let's code up and check the phase transition condition. The case correspond to case where $0 \leq 1+\delta_1 \leq 1+\delta_2$.

```
delta1 = ell - 1/100
delta2 = kappa + 1/100

a1 = 1/(1 + delta1)
a2 = 1/(1 + delta2)
a1
```

```
## [1] 2.601127
```

```
a2
```

```
## [1] 0.1313103
```

```
a1 > gamma*(1 + h)
```

```
## [1] TRUE
```

```
a2 < gamma*(1 - h)
```

```
## [1] TRUE
```

The phase transition condition is satisfied and thus let's compute the expected location of the spikes.

```
phi <- function(x ,gamma, c){
  if (x == gamma) {
    print("The x is equal to gamma so the value is not defined")
    return(NA)  # Return NA if x is equal to gamma
  }
  numerator = gamma*x*(x - 1 + c)
  denominator = x - gamma
  result = numerator / denominator
  return(result)
}
lambda_1 <- phi(a1, gamma, c)
lambda_2 <- phi(a2, gamma, c)
lambda_1
```

```
## [1] 6.43173
```

```
lambda_2
```

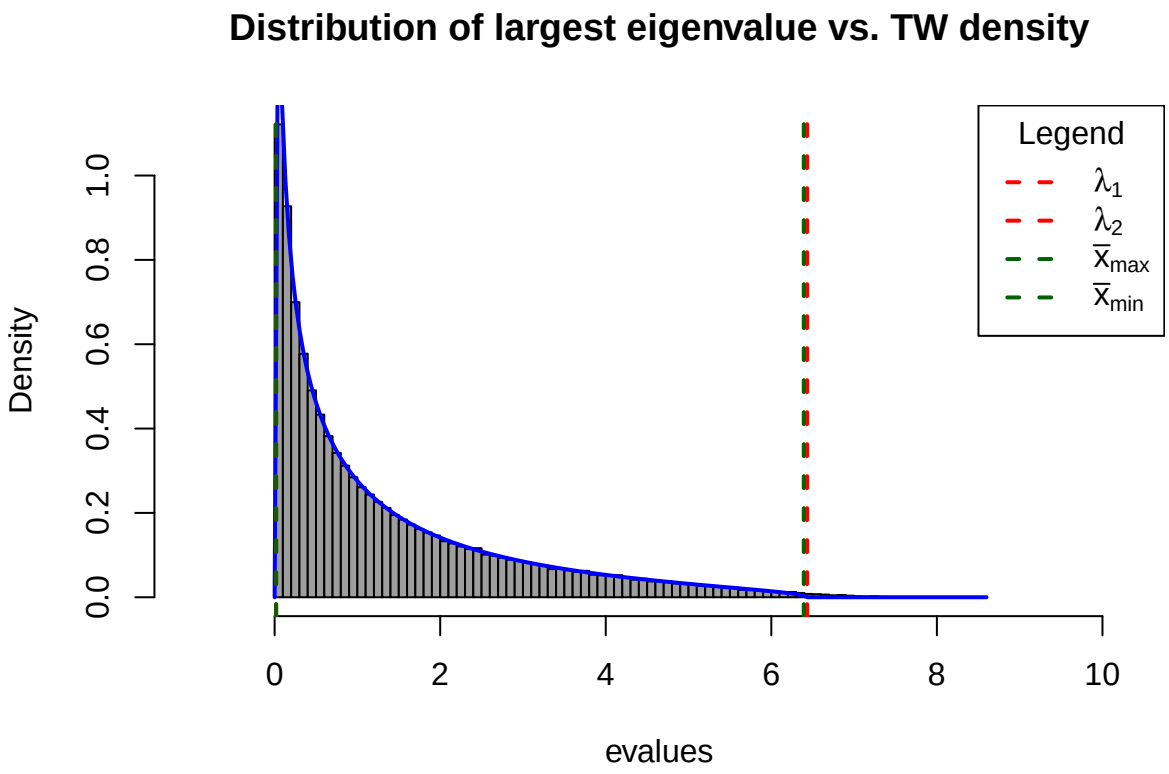
```
## [1] 0.01728772
```

The expected spike locations for δ_1 is 6.43173 and for δ_2 is 0.01728.

```
sim.F.eigenvalues = function(delta_1, delta_2, n = 400, c, y, n_sims = 1) {
  p = n * y
  m = p / c
  evalues = c()
  eval_max = c()
  eval_min = c()
  for (sim in 1:n_sims) {
    # Sigma1 = diag(x = 1, p)
    # Sigma2 = diag(p) + diag(x = c(delta_1, delta_2, rep(0, p - 2)), p)
    Sigma1 = diag(x = c(1/(1 + delta_1), 1/(1 + delta_2), rep(1, p - 2)), p)
    Sigma2 = diag(x = 1, p)
    mu = rep(0, p)
    X = rmvn(m+1, mu, Sigma1)
    Z = rmvn(n+1, mu, Sigma2)
    S1 = t(X) %*% (X)/m
    S2 = t(Z) %*% (Z)/n
    FF = S1 %*% solve(S2)
    eval = eigen(FF, only.values = TRUE)$values
    eval_max = c(eval_max, max(eval))
    eval_min = c(eval_min, min(eval))
    evalues = c(evalues, eval)
  }
  return(list(evalues, eval_max, eval_min))
```

```
}

dfisher = Vectorize(function(x, c, y) {
  h = sqrt(c + y - c*y)
  a = (1 - h)^2 / (1 - y)^2
  b = (1 + h)^2 / (1 - y)^2
  ifelse(x <= a | x >= b, 0, suppressWarnings(sqrt((x - a) * (b - x))*(1-y)/(2 * pi * x *(c+y*x))))
},"x")

eval.list = sim.F.eigenvalues(delta1, delta2, c = c, y = y, n_sims = 1000)
evalues = unlist(eval.list[1])
eval_max = unlist(eval.list[2])
eval_min = unlist(eval.list[3])
```

```
# Create the histogram
hh <- hist(evalues, breaks=100, xlim=c(-1,1.2*max(evalues)), freq=FALSE, col=8, main='')
x <- seq(hh$breaks[1], hh$breaks[length(hh$breaks)], length.out=200)

# Plot the density function
lines(x, dfisher(x, c, y), type='l', lwd=2, col="blue")

# Add vertical lines
abline(v = c(lambda_1, lambda_2, mean(eval_max), mean(eval_min)),
       lty=2, lwd=2, col=c('red', 'red', 'darkgreen', 'darkgreen'))

# Add a legend
legend("topright",
       legend = c(expression(lambda[1]),
                  expression(lambda[2]),
                  expression(bar(x)[max]),
                  expression(bar(x)[min])),
       col = c('red', 'red', 'darkgreen', 'darkgreen'),
       lty = 2,
       lwd = 2,
       title = "Legend",
       bg = 'white')  # background color of the legend box

# Add labels
xlabel <- expression("Eigenvalues")
ylabel <- expression("Density")
title(main="Distribution of largest eigenvalue vs. TW density")
```



We can see that our simulated locations correspond well to expected locations for δ_1 and for δ_2, respectively 6.43173 and 0.01728. We can see their values.

```
lambda_1
```

```
## [1] 6.43173
```

```
lambda_2
```

```
## [1] 0.01728772
```

```
mean(eval_max)
```

```
## [1] 6.391876
```

```
mean(eval_min)
```

```
## [1] 0.01881308
```

(f) Consider the signal detection problem where we are trying to determine the number of signals in observations of the form

$$x_i = Us_i + \varepsilon_i, \quad i = 1, \dots, m, \quad \text{(SD)}$$

where the x_i's are p-dimensional observations, s_i is a $k \times 1$ low dimensional signal ($k \ll p$) with covariance I_k, U is a $p \times k$ mixing matrix, and (ε_i) is an i.i.d. noise with covariance matrix Σ^2. None of the quantities on the right hand side of (SD) are observed. In Section 7.2 of [B], they propose to estimate the number of signals k by

$$\hat{k} := \max\{i : \lambda_i \geq \beta + \log(p/p^{2/3})\},$$

where (λ_i) are the eigenvalues of $\mathbf{S}$. Reproduce Case 1 in Table 1 of [B] for the Gaussian case for values $p = 25, 75, 125, 175, 225, 275$. Fix $y = 1/10$ and $c = 9/10$, further parameters and setup can be found at the bottom of p.436 and on p.437.

Answer to question 1 (f): From the setup for Model 1 of [B], we have that the $cov(\varepsilon_i) = diag(1, .., 1, 2, , , 2)$ with $p/2$ number of 1s and 2s and given $\varepsilon_i s$ are iid distributed and $cov(s_i) = I_k$ we have:

$$\begin{aligned}
\text{cov}(x_i) &= \text{cov}(Us_i + \varepsilon_i, Us_i + \varepsilon_i) \\
&= \text{cov}(Us_i, Us_i) + 2\text{cov}(Us_i, \varepsilon_i) + \text{cov}(\varepsilon_i, \varepsilon_i) \\
&= \text{cov}(Us_i, Us_i) + \text{cov}(\varepsilon_i, \varepsilon_i) \\
&= (c_1\nu_1 nu_1^T + c_2\nu_2 nu_2^T) + cov(\varepsilon_i)
\end{aligned}$$

Also, it is easy to see that $c_1\nu_1\nu_1^T + c_2\nu_2\nu_2^T = diag(c_1, c_2, c_2, 0, ..., 0) \in R^{p\times p}$ in the paper setting. Now we normalized covariance matrix of x_i by that of ε_i, and then we have the following:

$$cov(x_i) * cov(\varepsilon_i)^{-1} = (c_1\nu_1 nu_1^T + c_2\nu_2 nu_2^T) * cov(\varepsilon_i)^{-1} + I_p$$

This is equal to a Fisher statistic with $\Sigma_1 = (c_1\nu_1 nu_1^T + c_2\nu_2 nu_2^T) * cov(\varepsilon_i)^{-1} + I_p$ and $Sigma_2 = I_p$. So the perturbation $\Delta = (c_1\nu_1 nu_1^T + c_2\nu_2 nu_2^T) * cov(\varepsilon_i)^{-1}$ and Σ_1 has eigenvalues $c_1 + 1$ and $c_2 + 1$ with multiplicity of 1 and 2 respectively.

Let's do some simulation in Gaussian case assume the 0 mean and normalized covariance (see above Σ_1, Σ_2) to show this:

```
p_list = c(25, 75, 125, 175, 225, 275)
y = 1/10
c = 9/10
gamma = 1/(1 - y)
h = sqrt(c + y - c*y)
b = (1 + h)^2/(1 - y)^2
critical_value = gamma*(1 + h)
c1 = 10
c2 = 5

sim.F.signal = function(c1, c2, p, c, y, b, n_sims = 1) {
  n = p/y
  m = p/c
  k_hat = c()
  for (sim in 1:n_sims) {
    Sigma1 = diag(x = c(c1, c2, c2, rep(0, p-3)), p) + diag(p)
    Sigma2 = diag(p)
```

```
    mu = rep(0, p)
    X = rmvn(m+1, mu, Sigma1)
    Z = rmvn(n+1, mu, Sigma2)
    S1 = t(X) %*% (X)/m
    S2 = t(Z) %*% (Z)/n
    FF = solve(S2) %*% S1
    eval = eigen(FF, only.values = TRUE)$values
    k_hat = c(k_hat, sum(eval >= b +  log(p/p^{2/3})))
  }
  k_hat
}

probability1 = c()
probability2 = c()
probability3 = c()
probability4 = c()

for(i in 1:length(p_list)){
  k_hat = sim.F.signal(c1, c2, p_list[i], c, y, b, n_sims = 1000)
  probability1 = c(probability1, sum(k_hat==1)/length(k_hat))
  probability2 = c(probability2, sum(k_hat==2)/length(k_hat))
  probability3 = c(probability3, sum(k_hat==3)/length(k_hat))
  probability4 = c(probability4, sum(k_hat==4)/length(k_hat))
}
```

Let's see our version of Table 1:

```
data <- data.frame(
  p = p_list,
  n = p_list/y,
  m = p_list/c,
  `k=1` = probability1,
  `k=2` = probability2,
  `k=3` = probability3,
  `k=4` = probability4
)

kable(data, format = "html", caption = "Frequency of our estimator in Model 1", align = "c")
```

Frequency of our estimator in Model 1

p

n

m

k.1

k.2

k.3

k.4

25

250

27.77778

0.026

0.488

0.486

0

75

750

83.33333

0.000

0.171

0.829

0

125

1250

138.88889

0.000

0.093

0.907

0

175

1750

194.44444

0.000

0.063

0.937

0

225

2250

250.00000

0.000

0.033

0.967

0

275

2750

305.55556

0.000

0.024

0.976

0

Immediately, I can see that the frequency increases as p gets larger, which confirms the consistency of our estimator.

```
rm(list=ls())
```

Question 2

In this question, we shall consider high-dimensional sample covariance matrices of data that is sampled from an elliptical distribution. We say that a random vector $\mathbf{x}$ with zero mean follows an elliptical distribution if (and only if) it has the stochastic representation

$$\mathbf{x} = \xi A\mathbf{u}, \quad (\star)$$

where the matrix $A \in \mathbb{R}^{p\times p}$ is nonrandom and $\text{rank}(A) = p$, $\xi \geq 0$ is a random variable representing the radius of $\mathbf{x}$, and $u \in \mathbb{R}^p$ is the random direction, which is independent of ξ and uniformly distributed on the unit sphere S_{p-1} in $\mathbb{R}^p$, denoted by $\mathbf{u} \sim \text{Unif}(S_{p-1})$. The class of elliptical distributions is a natural generalization of the multivariate normal distribution, and contains many widely used distributions as special cases including the multivariate t-distribution, the symmetric multivariate Laplace distribution, and the symmetric multivariate stable distribution.

(a) Write a function `runifsphere(n,p)` that samples n observations from the distribution $\text{Unif}(S_{p-1})$ using the fact that if $\mathbf{z} \sim N_p(0, I_p)$ then $\frac{\mathbf{z}}{\|\mathbf{z}\|} \sim \text{Unif}(S_{p-1})$. Check your results by:

1. set $p = 25$, $n = 50$ and show that the (Euclidean) norm of each observation is equal to 1.

Answer to question 2 (a) (1): Now we show that the (Euclidean) norm of each observation is equal to 1 in (1):

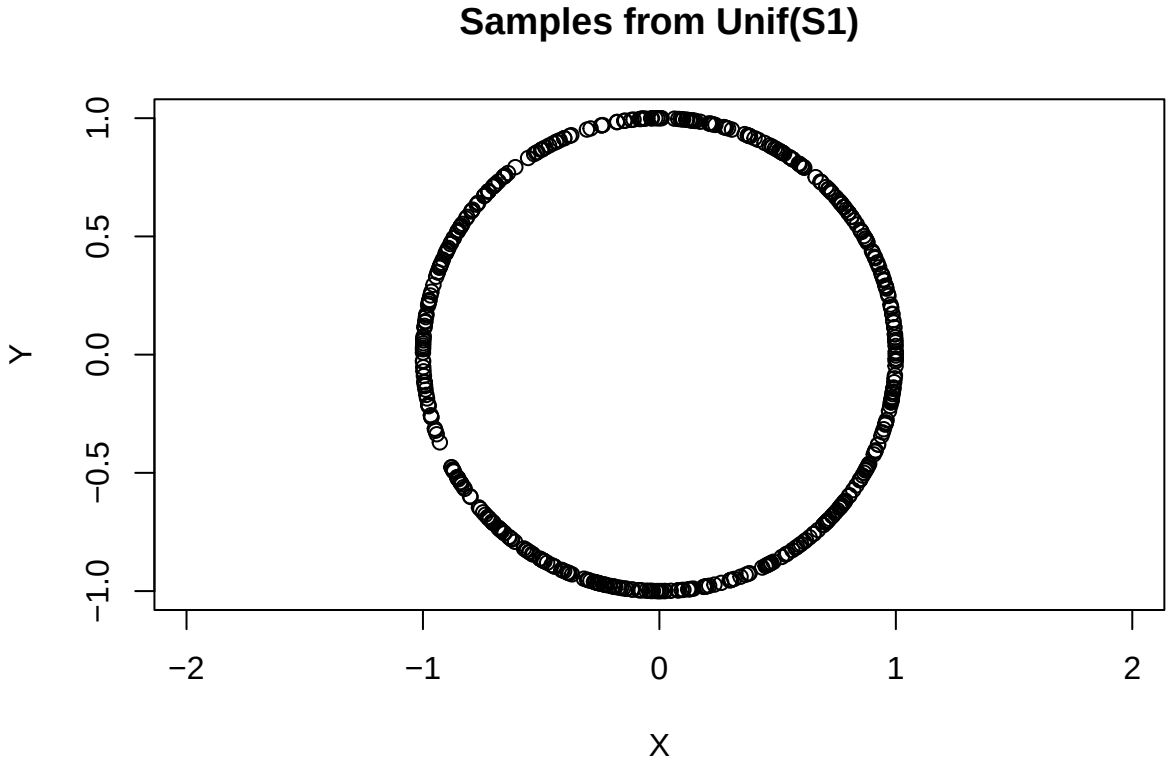
```
runifsphere <- function(n, p) {
  # Generate n samples from N_p(0, I_p)
  mu = rep(0, p)
  Sigma2 = diag(p)
  z = rmvn(n, mu, Sigma2)


  # Normalize each sample to have unit norm
  norms <- sqrt(rowSums(z^2))
  samples_on_sphere <- sweep(z, 1, norms, FUN = "/")

  return(samples_on_sphere)
}

p = 25
n = 50
u <- runifsphere(n, p)
norms <- sqrt(rowSums(u^2))
norms
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1
```

(2) generate a scatter plot in the case $p = 2$, $n = 500$ to show that the samples lie on a circle.

Answer to question 2 (a) (2): Now we show that the observations lie on a circle in (2):

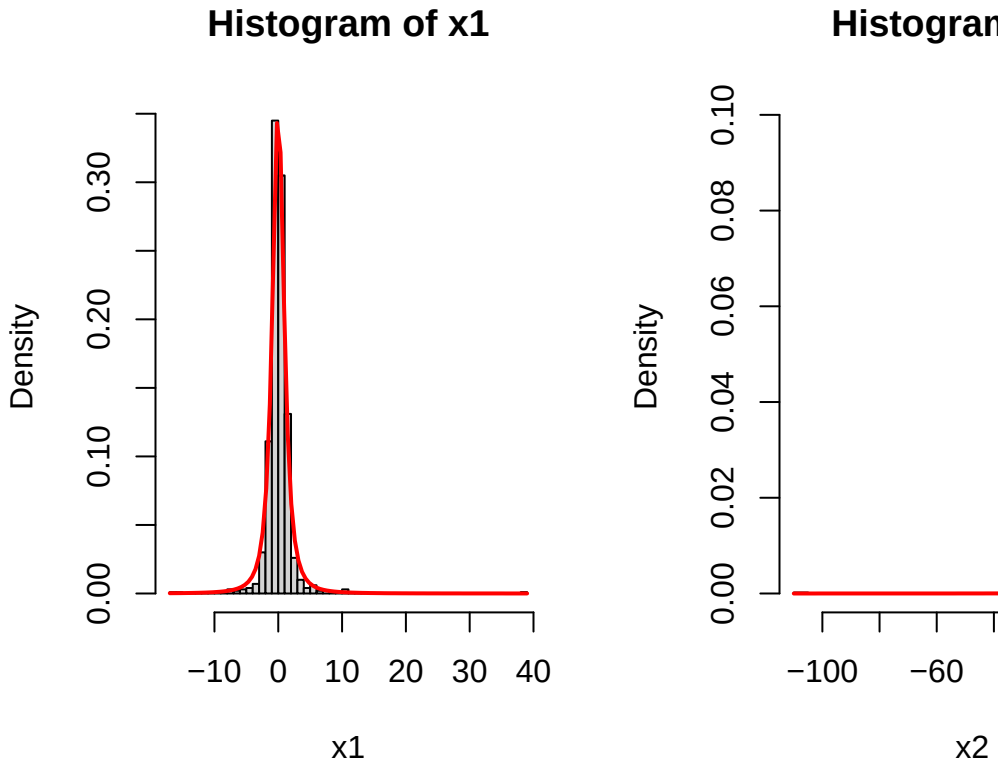
```
p = 2
n = 500
u2 <- runifsphere(n, p)

# Create a scatter plot of the samples
plot(u2[,1], u2[,2], xlim = c(-1, 1), ylim = c(-1, 1), xlab = "X", ylab = "Y", asp = 1, main = "Samples
```



Show that you can simulate a multivariate p t-distribution $t_\nu(0, I_p)$ by setting $\xi \sim \sqrt{\frac{\nu}{C}}$ in $(\star)$ with $A = I_p$ and $C \sim \chi^2_\nu$. Do this by sampling observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and comparing the two marginal histograms of the observations against the density of the univariate t_ν distribution. Take $p = 2$, $n = 1000$, $\nu = 2$.

```
# Parameters
p <- 2
n <- 1000
nu <- 2
```

```r
# Simulate from multivariate t-distribution
rellipse_t <- function(n, p, nu) {
  u <- runifsphere(n, p)
  A = diag(p)

  # Generate xi
  C <- rchisq(n, df = nu)
  xi <- sqrt(nu / C)

  # Compute x
  x <- xi * t(A %*% t(u))

  return(x)
}

# Generate samples
samples <- rellipse_t(n, p, nu)

# Plot histograms against t-distribution density
par(mfrow = c(1, 2))
hist(samples[,1], breaks = 40, prob = TRUE, main = "Histogram of x1", xlab = "x1")
curve(dt(x, df = nu), col = "red", lwd = 2, add = TRUE)
hist(samples[,2], breaks = 40, prob = TRUE, main = "Histogram of x2", xlab = "x2")
curve(dt(x, df = nu), col = "red", lwd = 2, add = TRUE)
```



Answer to question 2 (a) part 3:

We can see that the plot matches the distribution very well.

(b) Suppose that $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are p-dimensional observations sampled from an elliptic distribution ($\star$). We stack these observations into the data matrix $\mathbf{X}$ and calculate the sample covariance matrix $\mathbf{S}_n := \mathbf{X}\mathbf{X}^T/n$. Theorem 2.2 of the recent paper [C] is a central limit theorem for linear spectral statistics (LSS) of $\mathbf{S}_n$. For example, Eq. (2.10) in [C] provides the case of the joint distribution of the LSS $\phi_1(x) = x$ and $\phi_2(x) = x^2$. Following the notation used there (for all the following terms in this question). Perform a simulation experiment to examine the fluctuations of $\hat{\beta}_{n1}$ and $\hat{\beta}_{n2}$. In the experiment, take $H_p = \frac{1}{2}\delta_1 + \frac{1}{2}\delta_2$ and choose the distribution of $\xi \sim k_1 \text{Gamma}(p, 1)$ with $k_1 = \frac{1}{\sqrt{p+1}}$. Set the dimensions to be $p = 200$ and $n = 400$. Choose the number of simulations based on the computational power of your machine. Similar to Figure 1 in [C], use a QQ-plot to show normality.

Answer to question 2 (b): The population PSD H_p is assumed to be fixed and therefore we have $H_p = H$. Immediately, we have the following conclusion:

$$\gamma_{nj} = \int t^j dH_p(t) = \int t^j dH(t) = \gamma_j$$

In this question, we assume that the $H_p = \frac{1}{2}\delta_1 + \frac{1}{2}\delta_2 \Rightarrow \Sigma = diag(1, .., 1, 2, ..., 2) = AA^T$ with equal number of 1's and 2's so we can compute their respectively values:

$$\gamma_1 = \gamma_{n1} = \int t dH_p(t) = \int t d(\frac{1}{2}\delta_1 + \frac{1}{2}\delta_2) = \frac{1}{2}\int t d\delta_1(t) + \frac{1}{2}\int t d\delta_2(t)$$
$$\gamma_2 = \gamma_{n2} = \int t^2 dH_p(t) = \int t^2 d(\frac{1}{2}\delta_1 + \frac{1}{2}\delta_2) = \frac{1}{2}\int t^2 d\delta_1(t) + \frac{1}{2}\int t^2 d\delta_2(t)$$

By propert

$$\int f(t)\delta_a(t)dt = f(a) \tag{1}$$

We have that:

$$\int t d\delta_2(t) = 1$$
$$\int t d\delta_2(t) = 2$$
$$\int t^2 d\delta_1(t) = 1^2$$
$$\int t^2 d\delta_2(t) = 2^2$$

$$\gamma_1 = \gamma_{n1} = 1.5 \quad \text{and} \quad \gamma_2 = \gamma_{n2} = 2.5 \quad \text{similarly} \quad \gamma_3 = 4.5 \quad \text{and} \quad \gamma_4 = 8.5$$

The value we need to show the CLT assuming $p = 200, \quad n = 400 \quad$ and $\quad \frac{p}{n} := c_n = c = \frac{1}{2}$:

$$
\begin{aligned}
\tau &= 4 \\
\beta_{n1} &= \gamma_{n1} = 1.5, \\
\beta_{n2} &= \gamma_{n2} + c_n \gamma_{n1}^2 = 2.5 + \frac{1}{2} 1.5^2 = 3.625, \\
\nu_1 &= 0, \\
\nu_2 &= c\gamma_2 + c(\tau - 2)\gamma_1, \\
\psi_{11} &= 2c\gamma_2 + c(\tau - 2)\gamma_1^2 = 2.5 + 1.5^2 = 4.75, \\
\psi_{22} &= 8c\gamma_4 + 4c^2\gamma_2^2 + 16c^2\gamma_1\gamma_3 + 8c^3\gamma_1^2\gamma_2 + 4c(\tau - 2)(c\gamma_1^2 + \gamma_2)^2 = 125.4375
\end{aligned}
$$

Let's code up these variables into R:

```
p = 200
n = 400
c = p/n
gamma_1 = 1.5
gamma_2 = 2.5
gamma_3 = 4.5
gamma_4 = 8.5
tau = 4
beta_n1 = gamma_1
beta_n2 = gamma_2 + c*gamma_1^2
nu_1 = 0
nu_2 = c*gamma_2 + c*(tau - 2)*gamma_1
psi_11 = 2 * c * gamma_2 + c * (tau - 2) * gamma_1^ 2
psi_22 = 8*c*gamma_4 + 4*c^2*gamma_2^2 + 16*c^2*gamma_1*gamma_3 + 8*c^3*gamma_1^2*gamma_2 + 4*c*(tau - 2
```

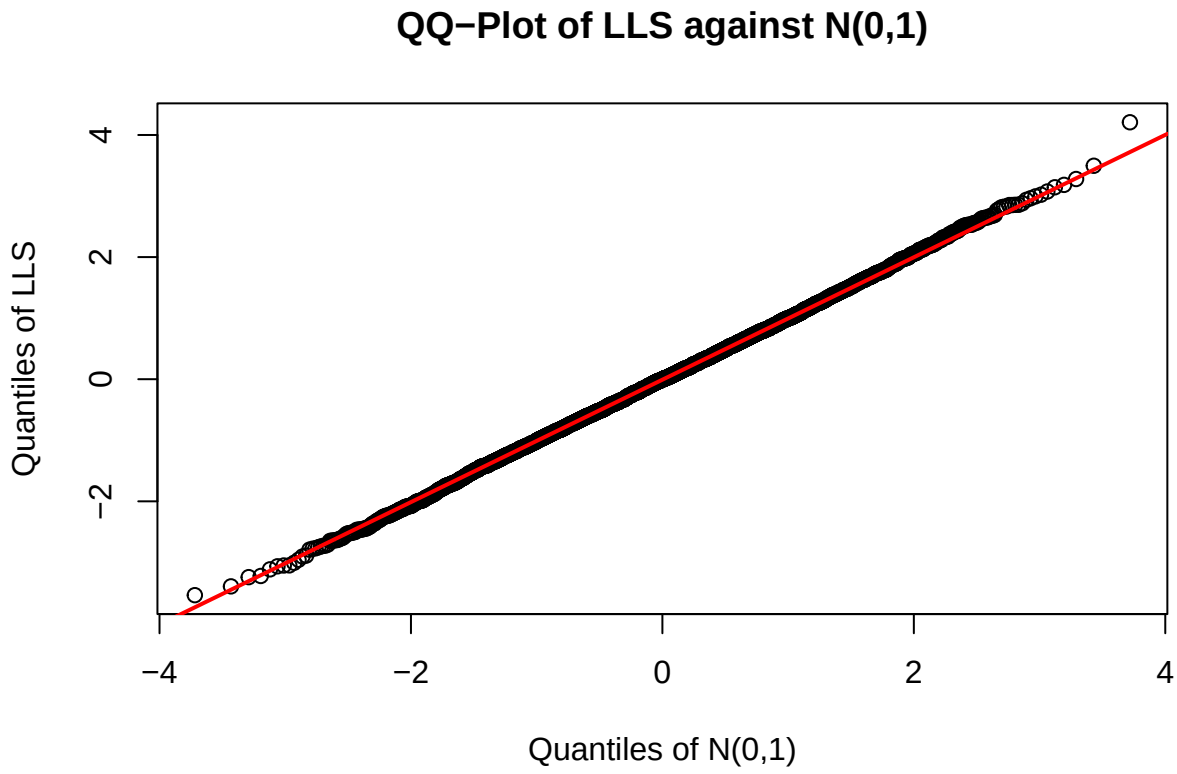
To facilitate our simulation later, we will construct a function that can generate the ellipse distribution under the question setting:

```
rellipse <- function(n, p){
  u <- runifsphere(n, p)
  # A = diag(x = c(rep(1, p/2), rep(2, p/2)), p)
  if(p %% 2 == 0){
    A = diag(x = c(rep(1, p/2), rep(sqrt(2), p/2)), p)
  }else{
    A = diag(x = c(rep(1, floor(p/2)), rep(sqrt(2), p - floor(p/2))), p)
  }

  # Generate xi
  k_1 = 1/sqrt(p+1)
  xi <- k_1 * rgamma(n, shape = p , scale = 1)

  # Compute x
  x <- xi * t(A %*% t(u))

  return(x)
}
```

Now, Let's simulate the fluctuations of $\hat{\beta}_{n1}$, which corresponds to the case where $f(x) = x$ as defined above.

```
plan(multisession, workers = 8)
n_sims = 5000
beta_n1_hat = future_replicate(n_sims, {
  p = 200
  n = 400
  X = rellipse(n, p)
  Sn = cov(X)
  L=eigen(Sn, only.values = TRUE)$values
  sum(L)/p
})
LLS_normalized = p*(beta_n1_hat - beta_n1)/sqrt(psi_11)
```

```
qqnorm(LLS_normalized, main = "QQ-Plot of LLS against N(0,1)", ylab = "Quantiles of LLS", xlab = "Quanti
qqline(LLS_normalized, col = "red", lwd = 2)
```

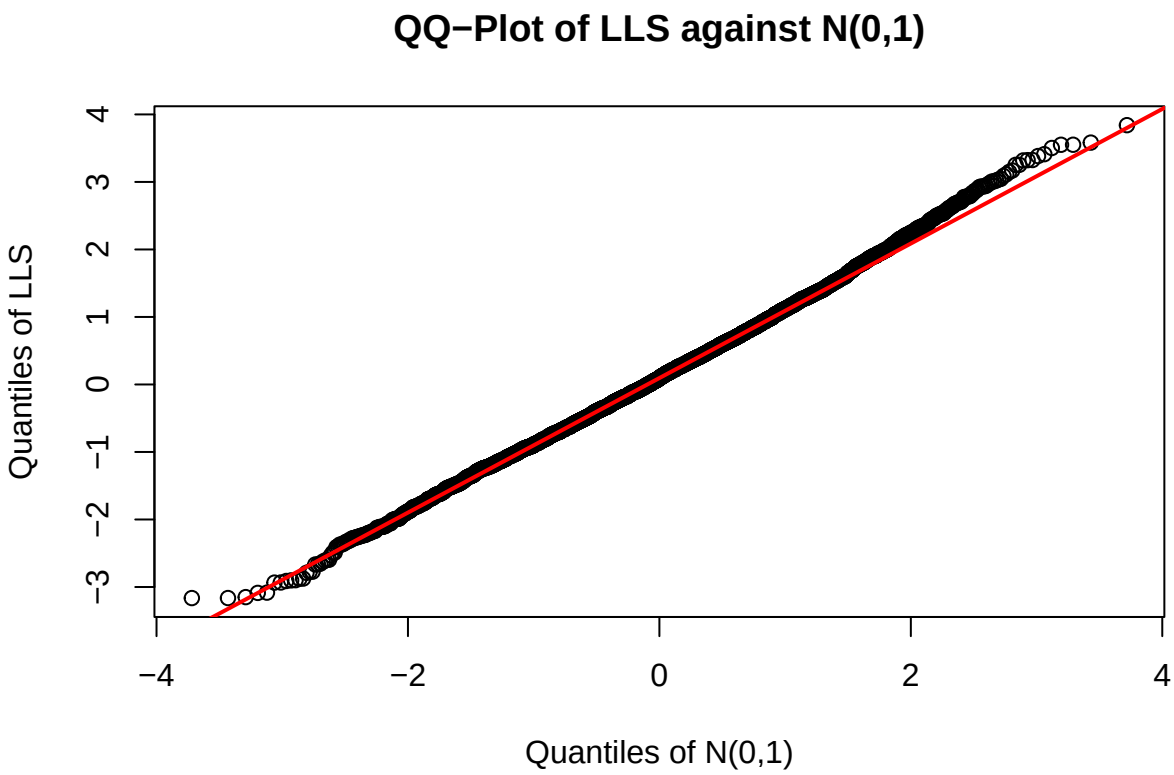


Now, Let's simulate the fluctuations of $\hat{\beta}_{n2}$, which corresponds to the case where $f(x) = x^2$ as defined above.

```
n_sims = 5000
beta_n2_hat = future_replicate(n_sims, {
  p = 200
  n = 400
  X = rellipse(n, p)
  Sn = cov(X)
  L=eigen(Sn, only.values = TRUE)$values
  sum(L^2)/p
```

```
})
LLS_normalized = ((p*(beta_n2_hat - beta_n2)) - nu_2)/sqrt(psi_22)
```

```
qqnorm(LLS_normalized, main = "QQ-Plot of LLS against N(0,1)", ylab = "Quantiles of LLS", xlab = "Quanti
qqline(LLS_normalized, col = "red", lwd = 2)
```



We can see that the linear spectral statistic (LSS) corresponds very well to the standard normal distribution, and only shows occansionally off at the tail.

(c) In the recent paper [E], it is shown that if $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are p-dimensional observations sampled from an elliptic distribution ($\star$) then the largest eigenvalue λ_1 of the sample covariance $\mathbf{S}_n$ (appropriately scaled) converges to the Tracy-Widom distribution as long as a certain condition on the tail of the distribution holds (Condition 2.7 in the paper). Perform a simulation to show that this holds true in the case of a double exponential distribution but not in the case of a multivariate student-t distribution. That is, simulate the largest eigenvalue of the sample covariance matrix and compare it to the Tracy-Widom distribution in each case. In the first case it should match (double exponential) and in the second case it shouldn't (multivariate student-t).

Answer to question 2 (c): Firstly, let's follow the above assumption $A = I_p$. By making this critical assumption, we achieve $\Sigma = AA^T = I_p$ and thus the population spectral distribution (PSD) H_p of matrix Σ converge to the Dirac measure at 1, namely $H(t) = \delta_1(t)$. By Theorem 2.1 of [C], we have that the emperical spectual distribution of F^{S_n} , with $S_n = \frac{1}{n}XX^T \in R^{p\times p}$ converge weakly to the $F^{c,H}$, which is a (not the general case of) Marcenko-Pastur distributions with variance 1.

Thus, immediately we get that the upper end points of this distribution, denoted as λ_+ is $\lambda_+ = (1 + \sqrt{c})^2$, *where* $c = p/n$. Thus, by Claim 1 and Corollary 3.3 in [E], we have $\gamma n^{2/3}(\lambda_1(B_n) - \lambda_+)$ should distribute as a Tracy-Widom distribution. Firstly, let's consider the γ.

$$\frac{1}{\gamma^3} = \frac{1}{x^3}\left(1 + c\int(\frac{\lambda x}{1-\lambda x})^3\pi(d\lambda)\right)$$

When we suppose that we have the PSD equal to $\delta_1(t)$. By property (1) (see above), this formula simplifies to:

$$\gamma = \left(\frac{1}{x^3} + \frac{c}{(1-x)^3}\right)^{-\frac{1}{3}}$$

From equation 2.9 in the paper [E], we define f and the relational formula between λ and x: $\lambda_+ := f(-x)$

$$f(x) := -\frac{1}{x} + c\int\frac{w\pi(dw)}{1+wx}$$

By property (1) (see above), this function simplifies to:

$$f(-x) = \frac{1}{x} + c\frac{1}{1-x} = \lambda_+$$

The first derivative and by 2.9 in [E]:

$$f'(-x) = \frac{1}{x^2} - \frac{c}{(1-x)^2} = 0$$

We get that:

$$x_1 = \frac{1}{1-\sqrt{c}}, \quad x_2 = \frac{1}{1+\sqrt{c}}$$

for $x_1, x_2 \in (0, \sigma_1^{-1})$. The σ_1 is the largest eigenvalue in matrix $\Sigma = I_p$ in our case so $\sigma_1 = 1$. When $c = \frac{1}{2}$, we choose x_2 to be the valid x in our case.

Moreover, we will need ξ to satisfy the condition 2.7 in [E]. It part (b), we see that the setting $\xi \sim k_1\text{Gamma}(p, 1)$ and $k_1 = \frac{1}{\sqrt{p+1}}$ is the same as the condition 2.7. We proof a little bit here:

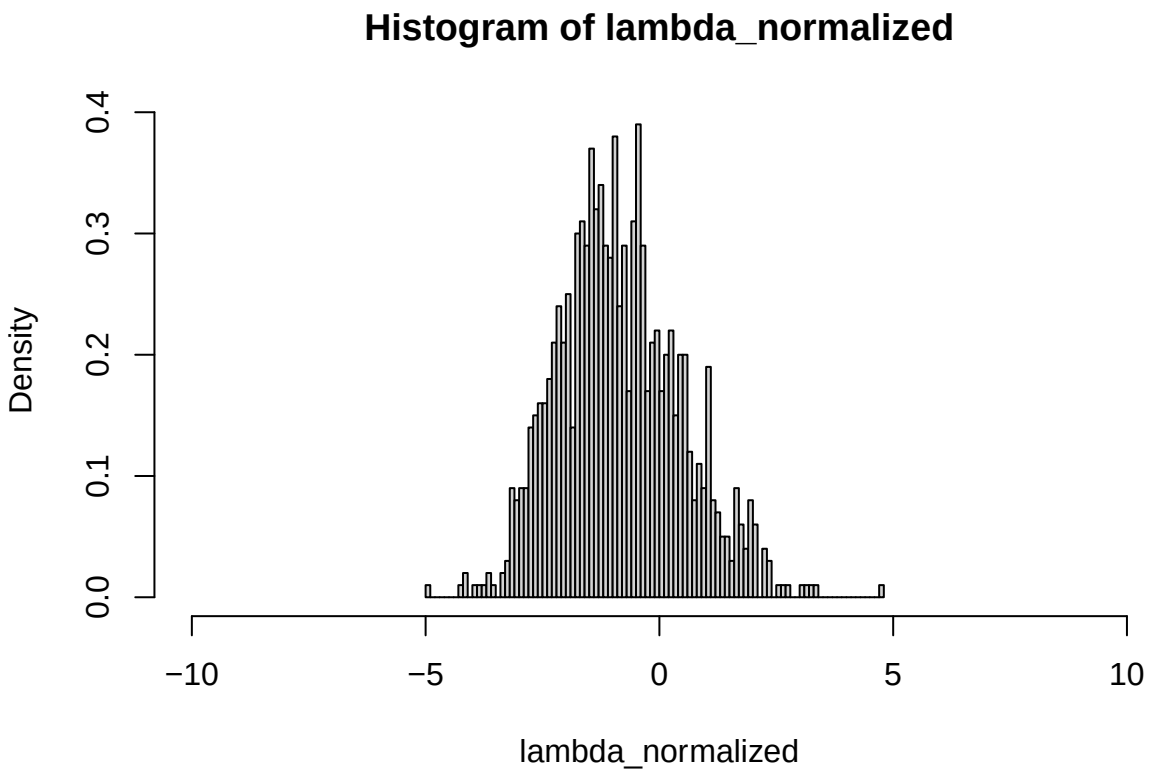
$$\lim_{s\to\infty}\limsup_{N\to\infty} s^2 P\left(|\hat{\xi}_i^2 - M| > \sqrt{Ms}\right) = 0,\ \xi_i^2 = \hat{\xi}_i^2/\sqrt{N}$$

The $\hat{\xi}_i^2$ is in our case ξ_i^2. This generally says that the $\hat{\xi}_i^2$ should be closed to M, which is p in our notation. Also, condition 2.7 also says that $E\xi_i^2 = \phi$. Let's substitute the $\xi_i^2 = \hat{\xi}_i^2/\sqrt{N}$ and get $E\frac{\hat{\xi}_i}{N}^2 = \phi$ and it is the same as $E\hat{\xi}_i^2 = \phi * N = M$, which is equal to $E{\xi_i}^2 = p$ in [c]. This condition has been check when in part (b).

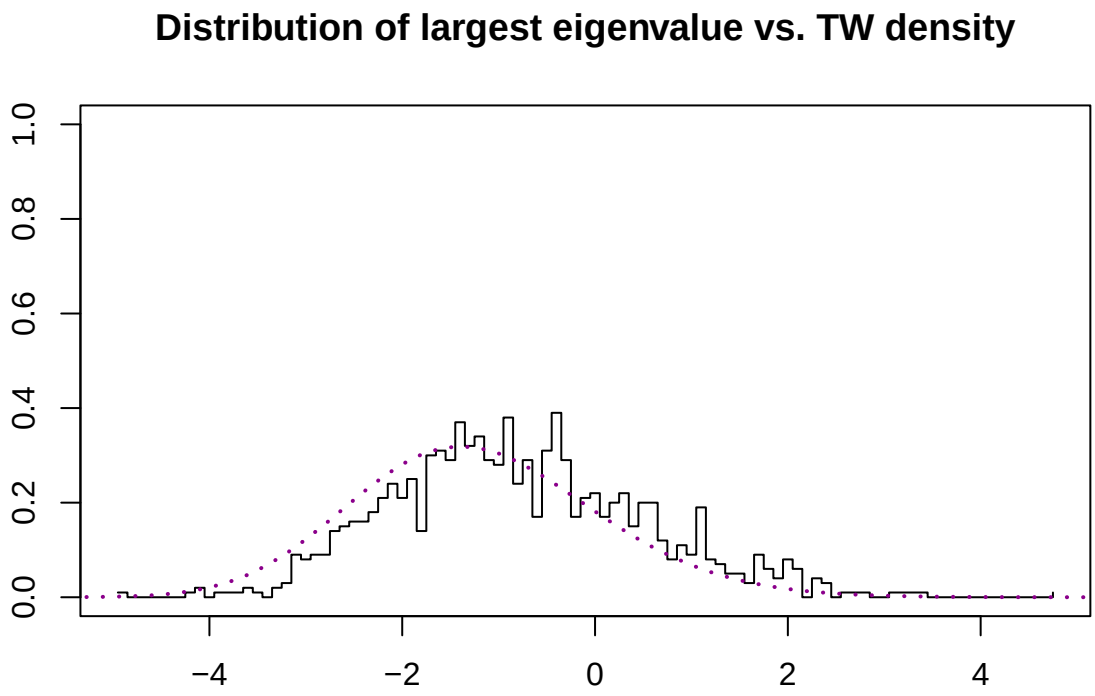
Let's write some codes to show this result. Firstly, let's show the case where x follows double exponential distribution. This is to say $\xi \sim k_1\text{Gamma}(p, 1)$ and $k_1 = \frac{1}{\sqrt{p+1}}$ (to satisfy condition 2.7).

```
rellipse <- function(n, p){
  u <- runifsphere(n, p)
  A = diag(p)

  # Generate xi
  k_1 = 1/sqrt(p+1)
  xi <- k_1 * rgamma(n, shape = p , scale = 1)

  # Compute x
  x <- xi * t(A %*% t(u))

  return(x)
}

n_sims = 1000
lambda_1 = future_replicate(n_sims, {
  p = 200
  n = 400
  X = rellipse(n, p)
  Sn = X %*% t(X)/n
  L=eigen(Sn, only.values = TRUE)$values
  max(L)
})

lambda_plus = (1 + sqrt(c))^2
x = 1/(1+sqrt(c))
gamma = (1/x^3 + c/(1 - x)^3)^{-1/3}
lambda_normalized = gamma*n^{2/3}*(lambda_1 - lambda_plus)
```

```
h <- hist(lambda_normalized, breaks=100, freq = FALSE, xlim=c(-10,10))
```




```
plot(h$mids, h$density, type="s", xlab="", ylab="", ylim=c(0,1))
curve(dtw, -10, 10, lwd=2, lty=3, col="darkmagenta", add=TRUE)
title(main="Distribution of largest eigenvalue vs. TW density")
```

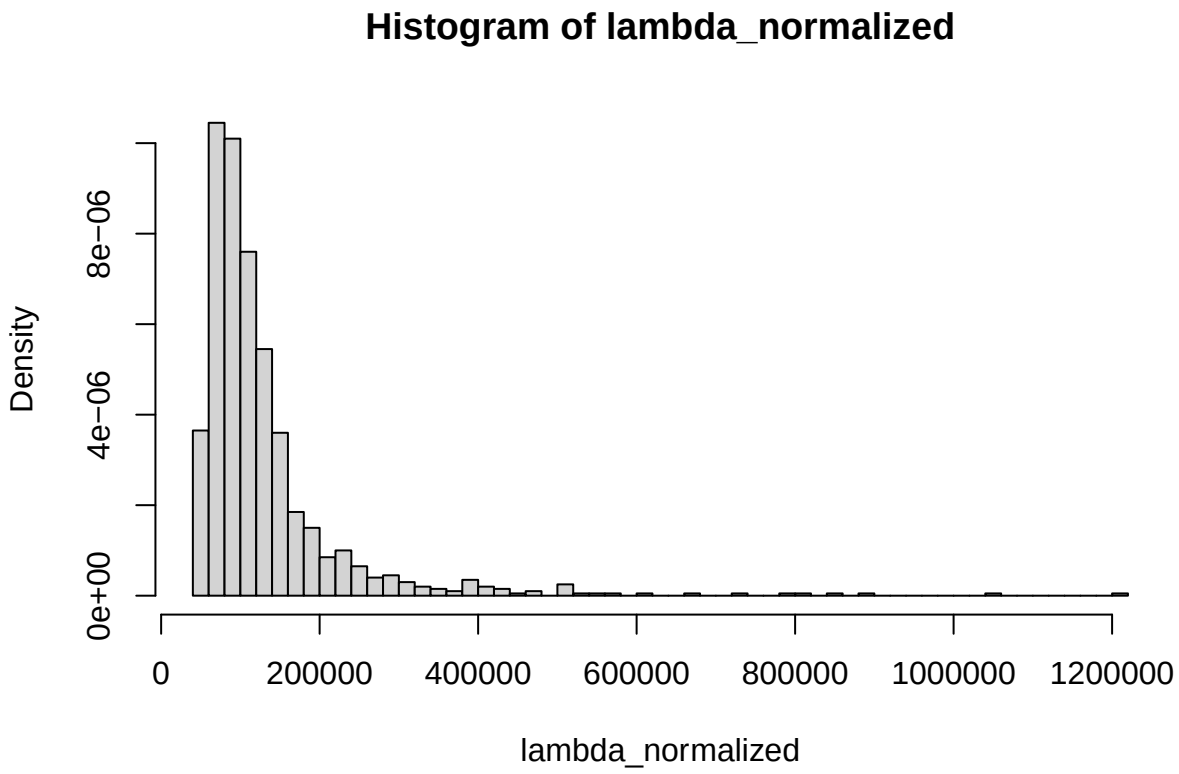


The above case follows the Tracy-Widom distribution well enough. Secondly, let's show the case where x follows multivariate t distribution.

```
rellipse <- function(n, p, nu){
  u <- runifsphere(n, p)
  A = diag(p)

  # Generate xi
  xi <- p*rf(n, p, nu)

  # Compute x
  x <- xi * t(A %*% t(u))

  return(x)
}

n_sims = 1000
lambda_1 = future_replicate(n_sims, {
  p = 200
  n = 400
  X = rellipse(n, p, nu = 10)
  Sn = cov(X)
  L=eigen(Sn, only.values = TRUE)$values
  max(L)
})
```

```
lambda_plus = (1 + sqrt(c))^2
x = 1/(1+sqrt(c))
gamma = (1/x^3 + c/(1 - x)^3)^{-1/3}
lambda_normalized = gamma*n^{2/3}*(lambda_1 - lambda_plus)
```

```
hist(lambda_normalized, breaks=80, freq = FALSE)
```




```
# plot(h$mids, h$density, type="s", xlab="", ylab="", ylim=c(0,1))
# curve(dtw, -10, 10, lwd=2, lty=3, col="darkmagenta", add=TRUE)
# title(main="Distribution of largest eigenvalue vs. TW density")
```

We can see that this histogram does not follow the Tracy-Widom distribution even in the normalized case.

(d) A nice property of elliptic distributions ($\star$) is that the mixture coefficient ξ can feature heteroskedasticity and the overall distribution of $\mathbf{x}$ can exhibit heavy tails. Both are properties that are widely observed in financial and economic data, for example. In the recent paper [F], they proposed a more generalized setting whereby the observations

$$\mathbf{x}_i = \xi_i A \mathbf{u}_i, \quad i = 1, \dots, n.$$

may exhibit the situation that

- ξ_i's can depend on each other and on $\{\mathbf{u}_i : i = 1, \dots, n\}$ in an arbitrary way, and

- ξ_i's do not need to be stationary.

The trick to dealing with these kind of observations is to self-normalize them. That is, we consider the new observations $\tilde{\mathbf{x}}_1, \ldots, \tilde{x}_n$ where

$$\tilde{\mathbf{x}}_i := \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}.$$

The paper introduces two tests (LR-SN and JHN-SN) to consider the sphericity test

$$H_0 : \Sigma \propto I_p \quad \text{v.s.} \quad \Sigma \not\propto I_p$$

where $\propto$ means "proportional to". Reproduce the simulation experiment shown in Table 5 of [F] for the case $p/n = 0.5$ and only for LR-SN and JHN-SN for $p = 100, 200, 500$. Do this in the case of 1,000 replications.

Answer to question 2 (d): Notice that [F] assumes the *iid* $u_i \sim N(0, 1)$ to bring out the LR-SN test (see Proposition 1 and Assumption A'). Let's first write functions to generate the self-normalized observations and to generate AR-type covariance matrix:

```
rellipse_d <- function(n, p, Sigma){
  mu = rep(0, p)
  u <- rmvn(n, mu, Sigma)

  omega <- abs(rnorm(n, 0, 1))

  # Compute x
  x <- omega*u # (n, p)

  # Compute the Euclidean norm for each row
  row_norms <- apply(x, 1, function(x_i) sqrt(sum(x_i^2)))

  # Normalize each row
  normalized_x <- x/row_norms

  return(normalized_x)
}
pcor = function(rho, p) {
  Tn = matrix(0, p, p)
  for (i in 1:p) {
    for (j in 1:p) {
      Tn[i,j] = rho^abs(i-j)
    }
  }
  return(Tn)
}
```

Let's define some parameter and a list of p values we would like to test on.

```
y = 1/2
p_list = c(100, 200, 500)
alpha <- 0.05
```

Let's conduct simulation for LR-SN test first. To evaluate power, sample data under some alternative $\Sigma \propto I_p$ and calculate the proportion of rejections of H_0. This can be achieved by computing the probability of failure to reject the null hypothesis, in other words, the type II error denoted as β. The power of the test is equal to the $1 - \beta$.

```
plan(multisession, workers = 8)
n_sims = 1000
LRSN_simulation <- function(n_sims, p, y){
  t_LRSN = future_replicate(n_sims, {
    n = p/y
    Sigma = pcor(0.1, p)
    X = rellipse_d(n, p, Sigma)
    Sn = sum(diag(Sigma))*t(X)%*%X/n
    L = eigen(Sn, only.values = TRUE)$values
    L_ = sum(log(L))
    mu_LRSN = p*(((y - 1)/y) *log(1 - y) - 1) + log(1 - y)/2 + y
    sd_LRSN = sqrt(-2*log(1 - y) - 2*y)
    LRSN = (L_ - mu_LRSN)/sd_LRSN
    })
}

for(i in 1:length(p_list)){
  t_LRSN <- LRSN_simulation(n_sims, p_list[i], y)
  critical_left = qnorm(alpha/2)
  critical_right = qnorm(1 - alpha/2)
  mean.beta = mean(t_LRSN >= critical_left & t_LRSN <= critical_right)
  cat("The test power is", (1 - mean.beta), "when the dimension is", p_list[i],"\n")
}
```

```
## The test power is 0.341 when the dimension is 100
## The test power is 0.871 when the dimension is 200
## The test power is 1 when the dimension is 500
```

Let's do the JHN-SN test secondarily. The process is very similar.

```
n_sims = 1000
JHNSN_simulation <- function(n_sims, p, y){
  t_JHNSN = future_replicate(n_sims, {
    n = p/y
    Sigma = pcor(0.1, p)
    X = rellipse_d(n, p, Sigma)
    Sn = sum(diag(Sigma))*t(X)%*%X/n
    L = eigen(Sn, only.values = TRUE)$values
    L_ = sum(L^2)
    Tn = L_/y - n - p
    JHNSN = (Tn+1)/2
    })
}

for(i in 1:length(p_list)){
  t_JHNSN <- JHNSN_simulation(n_sims, p_list[i], y)
  critical_left = qnorm(alpha/2)
  critical_right = qnorm(1 - alpha/2)
```

```
  mean.beta = mean(t_JHNSN >= critical_left & t_JHNSN <= critical_right)
  cat("The test power is", (1 - mean.beta), "when the dimension is", p_list[i],"\n")
}
```

```
## The test power is 0.504 when the dimension is 100
## The test power is 0.977 when the dimension is 200
## The test power is 1 when the dimension is 500
```

It confirms the paper's findings: the tests, LR-SN and JHN-SN enjoy a blessing of dimensionality: for a fixed ratio p/n, the higher the dimension p, the higher the power.

```
rm(list=ls())
```

Question 3

(a) Unfortunately, the results of [C] do not cover all elliptic distributions due to a moment condition on the distribution, see Table 1 in [C]. The results in [D] extend their results to more general elliptic distributions such as multivariate Gaussian mixtures[1]. A p-dimensional vector $\mathbb{x} \in \mathbb{R}^p$ is a multivariate Gaussian mixture with k subpopulations if its density function has the form:

$$f(\mathbb{x}) = \sum_{j=1}^{k} p_j \phi(\mathbb{x}; \mu_j, \Sigma_j)$$

where (p_j) are the k mixing weights and $\phi(\cdot; \mu_j, \Sigma_j)$ denote the density function of the jth subpopulation with mean vector μ_j and covariance Σ_j. In the case where $\mu_1 = \mu_2 = ... = \mu_k = 0 \in \mathbb{R}^p$ and $\Sigma_j = v_j\Sigma$ for some $v_j > 0$ with $j = 1, ..., k$.

Write an R function to sample from such a distribution using the representation from Eq. (11) in [D].

Answer to question 3 (a): Let's write a function to sample from the multivariate Gaussian mixture. However, we do not assume we know the ξ distribution (see paper), so we will pass a `rxi(n, p)`. to function `GMM(n,p, A, rxi)`.

```
runifsphere <- function(n, p) {
  # Generate n samples from N_p(0, I_p)
  mu = rep(0, p)
  Sigma2 = diag(p)
  z = rmvn(n, mu, Sigma2)


  # Normalize each sample to have unit norm
  norms <- sqrt(rowSums(z^2))
  samples_on_sphere <- sweep(z, 1, norms, FUN = "/")

  return(samples_on_sphere)
}

GMM <- function(n, p, A, rxi){
  # generate dimensional p standard normal distribution
  u <- runifsphere(n, p)
```

```
  # u <- rmvn(n, rep(0,p), diag(p)) #

  # Generate xi
  xi <- rxi(n, p)

  # Compute x
  x <- xi * t(A %*% t(u))

  return(x)
}
```

(b) Using your code from (a), perform a simulation experiment to simulate fluctuations of $\hat{\beta}_2 := \int x^2 dF^{\mathbb{S}_n}(x)$ under a Gaussian scale mixture model where the variable ξ has a discrete distribution with two mass points. Specifically, the probabilities are given by:

$$P(\xi = 1.8\sqrt{p}) = 0.8 \quad \text{and} \quad P(\xi = 1.5\sqrt{p}) = 0.2$$

Consider the following cases:

$$1.\ p = 100, n = 150$$

$$2.\ p = 600, n = 900$$

In each case, plot a histogram of the distribution of $\hat{\beta}_2$ against the theoretical limiting density. Additionally, create a QQ-plot similar to Figure 1 in [D].

Note: this experiment is described just above Section 3 in [D].

Answer to question 3 (b): Let's code up our `rxi(n)` function as it illustrated before.

```
rxi <- function(n, p){
  # Discrete values
  values <- c(1.8*sqrt(p), 1.5*sqrt(p))

  # Associated probabilities
  probs <- c(0.8, 0.2)

  # Sample from the distribution
  sampled_values <- sample(values, size = n, replace = TRUE, prob = probs)

  return(sampled_values)
}
```

Let's calculate some parameters. Let's assume the limiting ratio $p/n \Rightarrow c = \frac{2}{3}$ and $A = I_p = \Sigma$. The limiting distribution $\tilde{H}_p$ for ξ^2/p is given in our context and it equals:

$$\xi^2/p \sim \begin{cases} 1.8^2, & 0.8 \\ 1.5^2, & 0.2 \end{cases}$$

Thus, the limiting distribution of $\tilde{H}_p$ will converge to a deterministic measure: $\delta = 0.2*\delta_{1.5^2} + 0.8*\delta_{1.8^2}$. Also, the limiting distribution H will converge to δ_1. Also, the population spectral distributions are known. In [D], they consider the $\hat{\beta}_2 := \int x^2 dG^{\mathbb{S}_n}(x) = \int x^2 dF^{\mathbb{S}_n}(x) - \int x^2 dF^{c,H,\tilde{H}}(x)$, assuming that the $p/n \to c = 2/3$ in our case and $F^{c,H,\tilde{H}}$ is the limiting spectral distribution.

Given:

$$\int f(x)dG_n(x) := \int f(x)dG_{n1}(x) + \int f(x)dG_{n2} = \int x^2 dF^{\mathbb{S}_n}(x) - \int x^2 dF^{c,H,\tilde{H}}(x)$$

Let's denote the first term on the right is denoted as $\hat{\beta}_2$ (same as the question settings) and the second term on the right is denoted as β_2 so that $\hat{\beta}_2 - \beta_2 = \int f(x)dG_n(x)$

Given the following relationships:

$$p\int f(x)dG_{n1}(x) \to N(\mu_1, \sigma_1^2) \Rightarrow \sqrt{p}(\int f(x)dG_{n1}(x)) \to N(\frac{\mu_1}{\sqrt{p}}, \frac{\sigma_1^2}{p})$$

$$\sqrt{p}\int f(x)dG_{n2}(x) \to N(0, \sigma_2^2)$$

$$\Rightarrow \sqrt{p}(\int f(x)dG_{n1}(x) + \int f(x)dG_{n2}(x)) \sim N(\frac{\mu_1}{\sqrt{p}}, \frac{\sigma_1^2}{p} + \sigma_2^2)$$

In the limiting, this distribution reduces to:

$$\lim_{p\to\infty} \sqrt{p}(\int f(x)dG_{n1}(x) + \int f(x)dG_{n2}(x)) \sim N(0, \sigma_2^2)$$

By the formula given in [D], we can show some results in [D] by computing the parameter, eg γ_i in [D].

$$\gamma_1 = \int td(0.8\delta_{1.8^2} + 0.2\delta_{1.5^2}) = 0.8 * 1.8^2 + 0.2 * 1.5^2$$

$$\gamma_2 = \int t^2 d\delta_1(t) = 0.8 * (1.8^2)^2 + 0.2 * (1.5^2)^2$$

$$\gamma_3 = \int t^3 d\delta_1(t) = 0.8 * (1.8^2)^3 + 0.2 * (1.5^2)^3$$

$$\gamma_4 = \int t^4 d\delta_1(t) = 0.8 * (1.8^2)^4 + 0.2 * (1.5^2)^4$$

```
gamma1 = 0.8*1.8^2 + 0.2*1.5^2
gamma2 = 0.8*(1.8^2)^2 + 0.2*(1.5^2)^2
gamma3 = 0.8*(1.8^2)^3 + 0.2*(1.5^2)^3
gamma4 = 0.8*(1.8^2)^4 + 0.2*(1.5^2)^4
c = 2/3
```

The respective parameters are the following:

```
mu1 = c*gamma2
Sigma1 = 4*c^2*gamma2^2
mu2 = 0
Sigma2 = c^3*(gamma4 - gamma2^2) + 4*c^2*gamma1*gamma3 + 4*c*(1 - c)*gamma1^2*gamma2 - 4*c*gamma1^4
```

The limiting distribution will have mean and variance:

```
0
```

```
## [1] 0
```

```
Sigma2
```

```
## [1] 9.925986
```

```
p = 100
n = 150
mu = mu1/sqrt(p) + mu2
Sigma = Sigma1/p + Sigma2
mu
```

```
## [1] 0.627372
```

```
Sigma
```

```
## [1] 11.50037
```

Now we can get the asymptotic distribution of $\sqrt{p}\int f(x)dG_n(x) = \sqrt{p}(\int x^2 dF^{\mathbb{S}_n}(x) - \int x^2 dF^{c,H,\tilde{H}}(x))$ and we get can $\int x^2 dF^{\mathbb{S}_n}(x)$ easily by simulation. The critical value that has not been uncovered is $\int x^2 dF^{c,H,\tilde{H}}(x)$, which required calculations.

Given the following relationships:

$$z = -\frac{1}{m(z)} + \int \frac{t}{1+ctm(z)} dH(t)$$

$$z = -\frac{1}{m(z)} + \int \frac{t}{1+ctm(z)} d(0.8\delta_{1.8^2} + 0.2\delta_{1.5^2})$$

$$z = -\frac{1}{m(z)} + 0.8\int \frac{t}{1+ctm(z)} \delta_{1.8^2} + 0.2\int \frac{t}{1+ctm(z)} \delta_{1.5^2}$$

$$z = -\frac{1}{m(z)} + 0.8\frac{1.8^2}{1+1.8^2 cm(z)} + 0.2\frac{1.5^2}{1+1.5^2 cm(z)}$$

Denote $m(z)$ as m

$$z = -\frac{1}{m} + 0.8\frac{1.8^2}{1+1.8^2 cm} + 0.2\frac{1.5^2}{1+1.5^2 cm}$$

Code up this in R will give us the stieltjes transform $m(z)$ of $F^{c,H,\tilde{H}}(x)$. By the property of:

$$f_{c,H,\tilde{H}}(x) = \frac{1}{\pi}\lim_{\epsilon \to 0_+} \Im m(x + i\epsilon)$$

to get the theoretical limiting spectral distribution (LSD) and take the integral to get the second moment of the LSD. Let's code this up and use the numerical method:

```
# Our previously defined functions
implicit_equation <- function(m, z, c = 2/3) {
  return(z + 1/m - 0.8*1.8^2/(1 + 1.8^2*c*m) - 0.2*1.5^2/(1 + 1.5^2*c*m))
}

derivative <- function(fn, m, z, h = 1e-5) {
  (fn(m + h, z) - fn(m - h, z)) / (2 * h)
}

newton_raphson_complex <- function(fn, start, z, tol = 1e-6, max_iter = 100) {
  m <- start
  for (i in 1:max_iter) {
    f_val <- fn(m, z)
    f_prime <- derivative(fn, m, z)
    m_next <- m - f_val / f_prime

    # Check for NA or NaN in m_next
    if (is.na(m_next) || is.nan(m_next)) {
      warning("Iteration produced NA or NaN values.")
      return(NA)  # Exit early
    }

    # Check for convergence
    diff <- Mod(m_next - m)
    if (is.na(diff) || is.nan(diff)) {
      warning("Difference computation produced NA or NaN values.")
      return(NA)  # Exit early
    }

    if (diff < tol) {
      return(m_next)
    }
    m <- m_next
  }
  warning("Maximum iterations reached without convergence.")
  return(NA)  # Return NA if doesn't converge
}
```

```
m_stieltjes <- function(z, tol = 1e-6, max_iter = 1000) {
  compute_m_for_z <- function(z_val) {
    m <- 1
    for (i in 1:max_iter) {
      m_next <- 1 / (-z_val - 1/m + 0.8*1.8^2/(1 + 1.8^2*2/3*m) + 0.2*1.5^2/(1 + 1.5^2*2/3*m))
      if (is.na(m_next) || is.nan(m_next) || Mod(m_next - m) < tol) {
        return(m_next)
      }
      m <- m_next
    }
    return(m)
  }

  return(sapply(z, compute_m_for_z))
}
```

```r
# Define your LSD density function as before
lsd_density <- function(x, epsilon = 1e-6) {
  z <- complex(real = x, imaginary = epsilon)
  m_val <- m_stieltjes(z)
  return(Im(m_val) / pi)
}

# Visualization
xs <- seq(0, 6, length.out = 400)  # Adjust the range as necessary
ys <- sapply(xs, lsd_density)

plot(xs, ys, type = "l", xlab = "x", ylab = "Density", main = "Limiting Spectral Distribution")
```

```r
# Function to compute the integrand for the second moment
second_moment_integrand <- function(x) {
  return(x^2 * lsd_density(x))
}

# Numerically compute the integral to get the second moment
result <- integrate(second_moment_integrand, lower = -Inf, upper = Inf)

# Print the result
print(result$value)
```

```r
library(complexplus)

# Your implicit function
implicit_equation <- function(m, z, c = 2/3) {
  return(z + 1/m - 0.8*1.8^2/(1 + 1.8^2*c*m) - 0.2*1.5^2/(1 + 1.5^2*c*m))
}

# Newton-Raphson method for complex numbers
m_stieltjes <- function(z) {
  m <- 0 + 1i*0  # starting value
  tol <- 1e-6
  max_iter <- 100
  iter <- 0

  while(iter < max_iter) {
    f_val <- implicit_equation(m, z)
    f_prime_val <- -1/m^2 + 0.8*1.8^2*c/(1 + 1.8^2*c*m)^2 + 0.2*1.5^2*c/(1 + 1.5^2*c*m)^2
    m_next <- m - f_val/f_prime_val

    # Check for NA or NaN values
    if(is.na(m_next) || is.nan(m_next)) {
      return(NA)
    }

    if(Mod(m_next - m) < tol) {
      break
    }
    m <- m_next
```

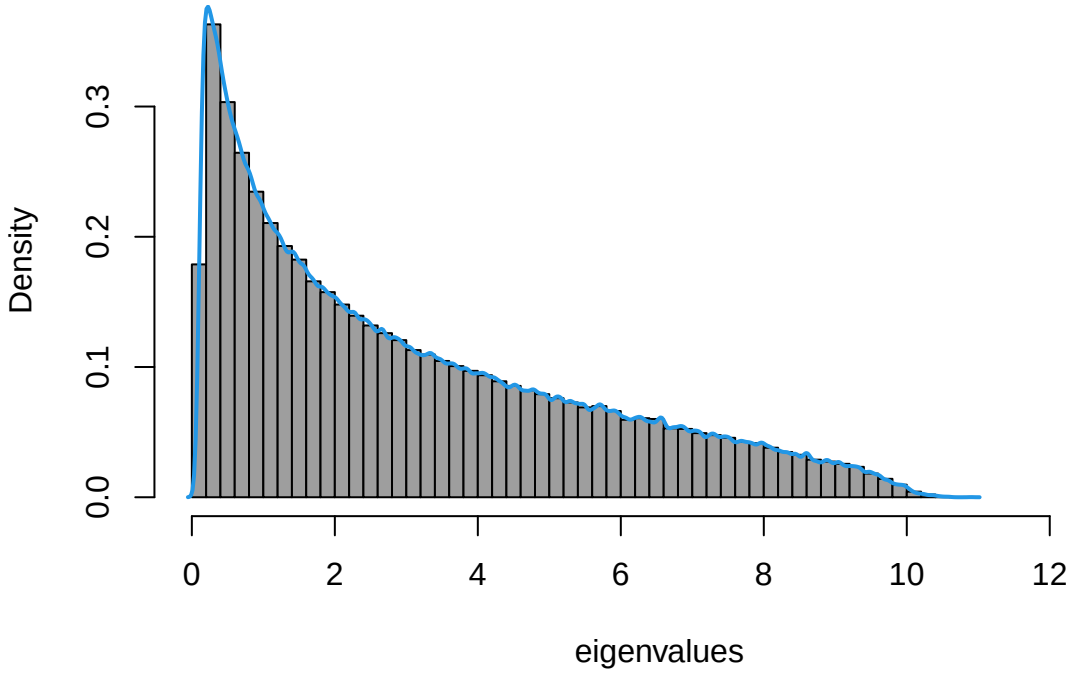
```
    iter <- iter + 1
  }

  return(m)
}


second_moment <- function(z_values, delta_z) {
  f_x <- function(x) {
    z = x + 1i*1e-6
    m = m_stieltjes(z)
    return (-1/pi) * Im(m)
  }

  # Density values for the given z_values
  densities <- sapply(z_values, f_x)

  # Compute second moment using the trapezoid rule
  integral = sum(0.5 * (z_values^2 * densities + c(tail(z_values^2 * densities, -1), 0)) * delta_z)

  return(integral)
}

# Define the z_values over which you believe most of the distribution lies
z_values = seq(0, 10, by = 0.01)  # Adjust as needed
second_moment_val = second_moment(z_values, 0.01)
```

The above method does not work due to unforeseen reasons. We will try the density estimation.

Now we can simulation the GMM in case where $p = 100, n = 150$.

```
n_sims = 1000
eigenvalues = future_replicate(n_sims, {
  X = GMM(n, p, diag(p), rxi)
  Sn = cov(X)
  L=eigen(Sn, only.values = TRUE)$values
})
```

```
d = density(eigenvalues, bw="SJ", kernel="gaussian")
hist(eigenvalues, breaks=50, xlim=c(0,1.2*max(eigenvalues)), freq=FALSE, col=8, main='')
lines(d, col=4, lwd=2)
```

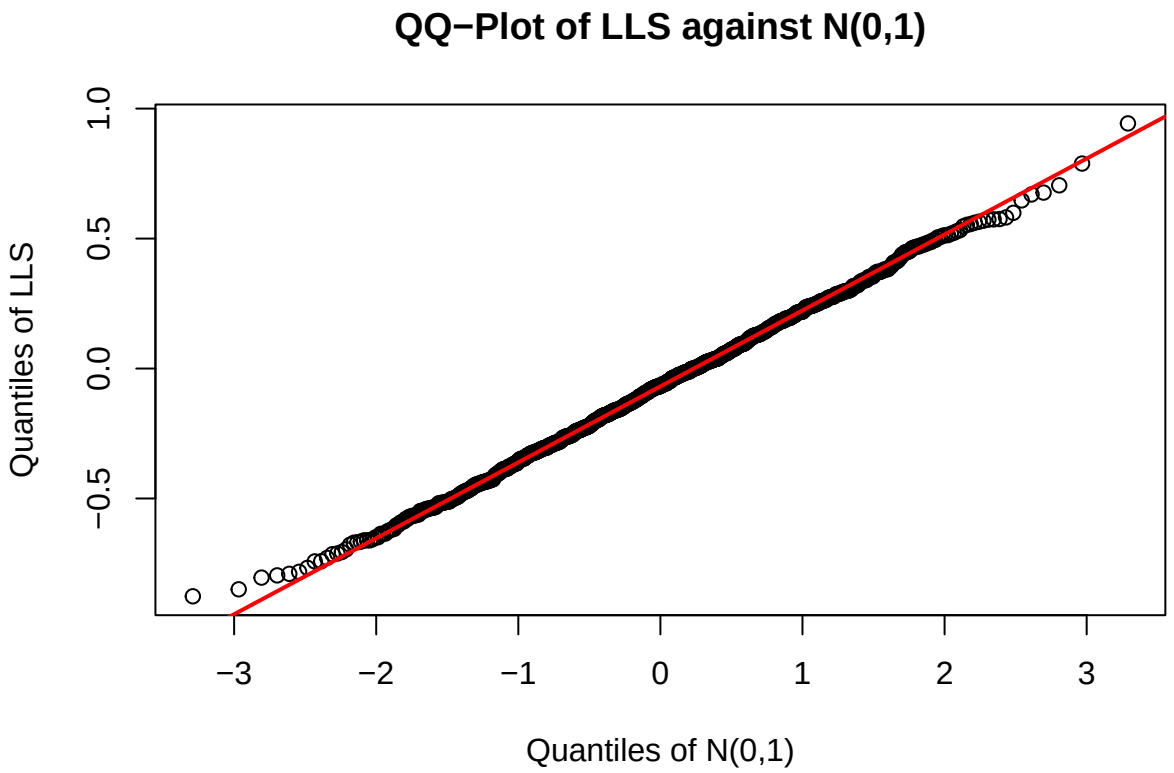



```r
# Define a function representing the kernel density estimate
kde_func <- approxfun(d$x, d$y, method="linear", rule=2)

# Define the integrand for the second moment
integrand <- function(x) {
  x^2 * kde_func(x)
}

# Integrate over the range of the data
second_moment_result <- quadgk(integrand,
                   a = min(eigenvalues), # We consider the positive (semi)definite matrix YY^T/m
                   b = max(eigenvalues))

# Print the result
print(second_moment_result)
```

```
## [1] 15.51741
```

```r
n_sims = 1000
beta_2_hat = future_replicate(n_sims, {
  X = GMM(n, p, diag(p), rxi)
  Sn = cov(X)
  L=eigen(Sn, only.values = TRUE)$values
  LSS = sum(L^2)/p
})
```

```
LLS_normalized = (sqrt(p)*(beta_2_hat -second_moment_result) - mu)/Sigma
qqnorm(LLS_normalized, main = "QQ-Plot of LLS against N(0,1)", ylab = "Quantiles of LLS", xlab = "Quanti
qqline(LLS_normalized, col = "red", lwd = 2)
```



Now we can simulation the GMM in case where $p = 600, n = 900$.

```
p = 600
n = 900
mu = mu1/sqrt(p) + mu2
Sigma = Sigma1/p + Sigma2
mu
```

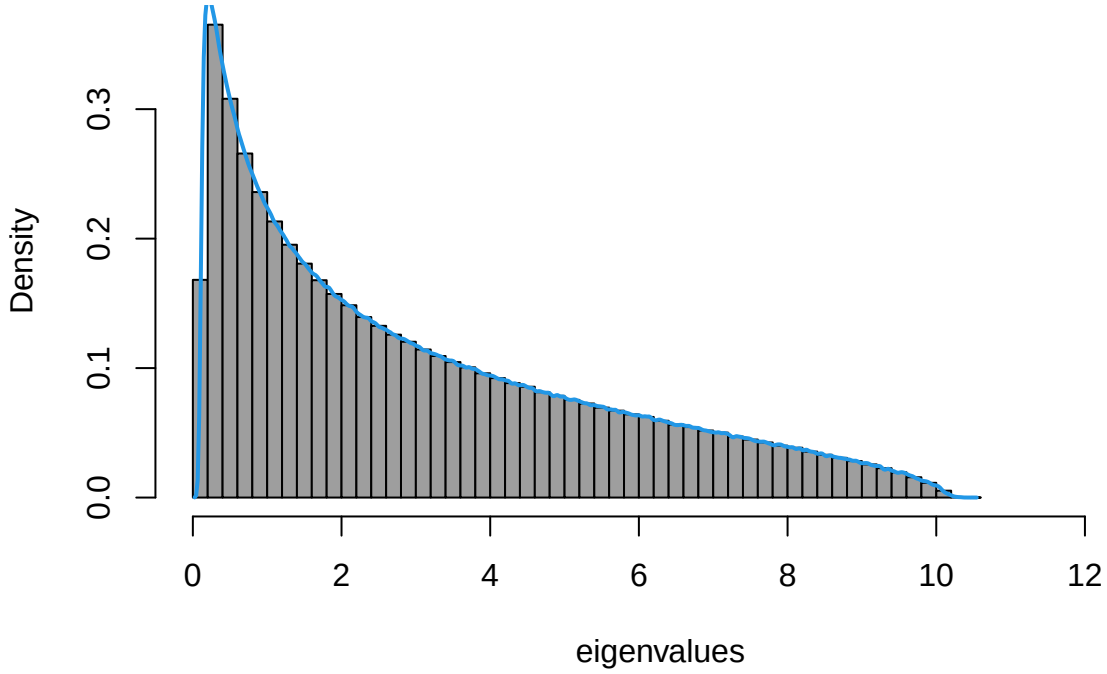
```
## [1] 0.2561235
```

```
Sigma
```

```
## [1] 10.18838
```

```
n_sims = 1000
eigenvalues = future_replicate(n_sims, {
  X = GMM(n, p, diag(p), rxi)
  Sn = cov(X)
  L=eigen(Sn, only.values = TRUE)$values
})
```

```
d = density(eigenvalues, bw="SJ", kernel="gaussian")
hist(eigenvalues, breaks=50, xlim=c(0,1.2*max(eigenvalues)), freq=FALSE, col=8, main='')
lines(d, col=4, lwd=2)
```

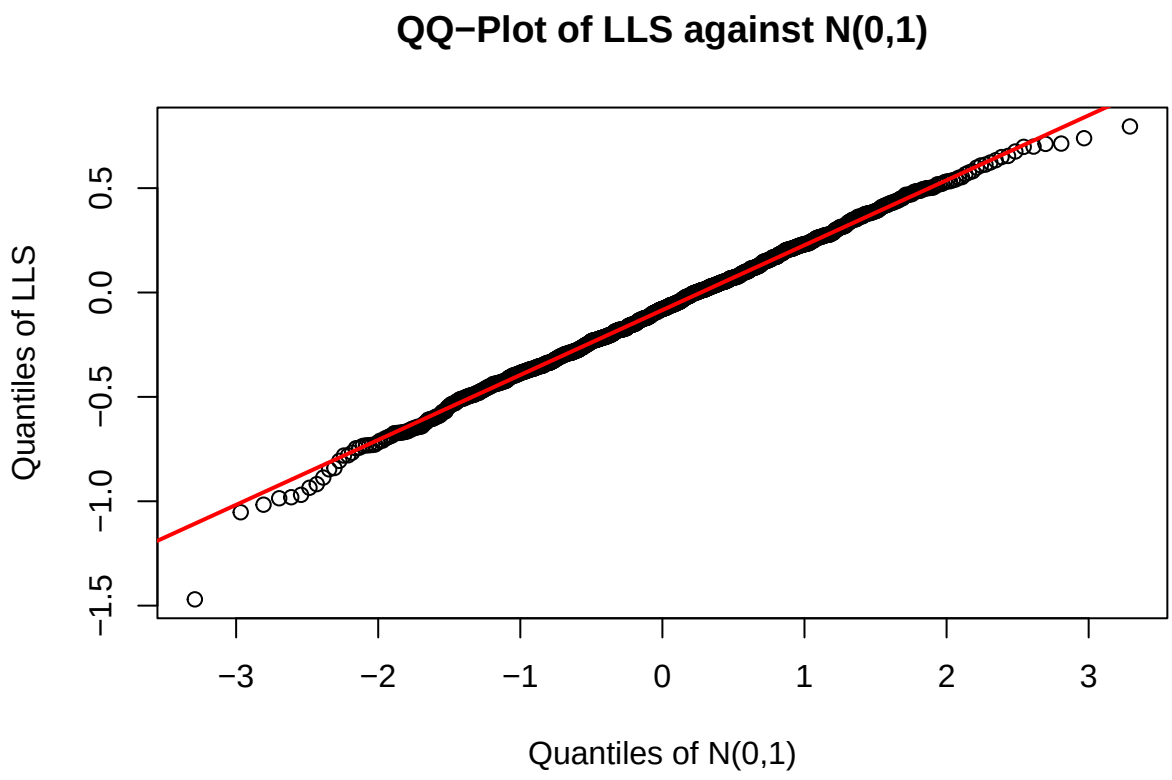



```
# Define a function representing the kernel density estimate
kde_func <- approxfun(d$x, d$y, method="linear", rule=2)

# Define the integrand for the second moment
integrand <- function(x) {
  x^2 * kde_func(x)
}

# Integrate over the range of the data
second_moment_result <- quadgk(integrand,
                    a = min(eigenvalues), # We consider the positive (semi)definite matrix YY^T/m
                    b = max(eigenvalues))

# Print the result
print(second_moment_result)
```

```
## [1] 15.54385
```

```
n_sims = 1000
beta_2_hat = future_replicate(n_sims, {
```

```
  X = GMM(n, p, diag(p), rxi)
  Sn = cov(X)
  L=eigen(Sn, only.values = TRUE)$values
  LSS = sum(L^2)/p
})
LLS_normalized = (sqrt(p)*(beta_2_hat -second_moment_result) - mu)/Sigma
qqnorm(LLS_normalized, main = "QQ-Plot of LLS against N(0,1)", ylab = "Quantiles of LLS", xlab = "Quant
qqline(LLS_normalized, col = "red", lwd = 2)
```




```
rm(list=ls())
```

(c) In addition to Question 2 (d), also reproduce the simulation experiment shown in Table 7 of [F] for the case $\frac{p}{n} = 0.5$ and only for LR-SN and JHN-SN for $p = 100, 200, 500$. Do this in the case of 1,000 replications.

```
library(pracma)
rellipse_normalised <- function(n, p, Sigma = diag(p)){
  # Calculate Z_ij from t with dof 4
  Z <- matrix(rt(n * p, df = 4), ncol = p)

  # Calculate omega_i
  omega <- numeric(n)
```

```
  omega[1] <- 0.01  # assuming initial value of omega is 1, can be changed
  for (i in 2:n) {
    omega_squared <- 0.01 + 0.85 * omega[i - 1]^2 + 0.1 * sum(Z[i - 1, ]^2) / sum(diag(Sigma))
    omega[i] <-  sqrt(omega_squared)
  }

  # Compute x
  A <- sqrtm(Sigma)
  Y <- omega*t(A$B%*%t(Z)) # (n, p)

  # Compute the Euclidean norm for each row
  row_norms <- apply(Y, 1, function(Y_i) sqrt(sum(Y_i^2)))

  # Normalize each row
  normalized_Y <- Y/row_norms

  return(normalized_Y)
}
pcor = function(rho, p) {
  Tn = matrix(0, p, p)
  for (i in 1:p) {
    for (j in 1:p) {
      Tn[i,j] = rho^abs(i-j)
    }
  }
  return(Tn)
}
```

```
y = 1/2
p_list = c(100, 200, 500)
alpha <- 0.05
```

```
plan(multisession, workers = 8)
n_sims = 1000
LRSN_simulation <- function(n_sims, p, y){
  t_LRSN = future_replicate(n_sims, {
    n = p/y
    Sigma = pcor(0.1, p)
    X = rellipse_normalised(n, p, Sigma)
    Sn = sum(diag(Sigma))*t(X)%*%X/n
    L = eigen(Sn, only.values = TRUE)$values
    L_ = sum(log(L))
    mu_LRSN = p*(((y - 1)/y) *log(1 - y) - 1) + log(1 - y)/2 + y
    sd_LRSN = sqrt(-2*log(1 - y) - 2*y)
    LRSN = (L_ - mu_LRSN)/sd_LRSN
    })
}

for(i in 1:length(p_list)){
  t_LRSN <- LRSN_simulation(n_sims, p_list[i], y)
```

```
  critical_left = qnorm(alpha/2)
  critical_right = qnorm(1 - alpha/2)
  mean.beta = mean(t_LRSN >= critical_left & t_LRSN <= critical_right)
  cat("The test power is", (1 - mean.beta), "when the dimension is", p_list[i],"\n")
}
```

Answer to question 3 (c):

```
## The test power is 0.347 when the dimension is 100
## The test power is 0.859 when the dimension is 200
## The test power is 1 when the dimension is 500
```

```
n_sims = 1000
JHNSN_simulation <- function(n_sims, p, y){
  t_JHNSN = future_replicate(n_sims, {
    n = p/y
    Sigma = pcor(0.1, p)
    X = rellipse_normalised(n, p, Sigma)
    Sn = sum(diag(Sigma))*t(X)%*%X/n
    L = eigen(Sn, only.values = TRUE)$values
    L_ = sum(L^2)
    Tn = L_/y - n - p
    JHNSN = (Tn+1)/2
    })
}

for(i in 1:length(p_list)){
  t_JHNSN <- JHNSN_simulation(n_sims, p_list[i], y)
  critical_left = qnorm(alpha/2)
  critical_right = qnorm(1 - alpha/2)
  mean.beta = mean(t_JHNSN >= critical_left & t_JHNSN <= critical_right)
  cat("The test power is", (1 - mean.beta), "when the dimension is", p_list[i],"\n")
}
```

```
## The test power is 0.474 when the dimension is 100
## The test power is 0.974 when the dimension is 200
## The test power is 1 when the dimension is 500
```

```
rm(list=ls())
```