Workshop 2

- Multivariate normals
  - Generating random samples
  - Calculating the variance
  - Calculating correlation and covariance matrices
  - Sampling a multivariate normal distribution with given covariance
  - Densities
- Eigenvalues of sample covariance matrix
  - Case p=50 and n=500
  - Case p=250 and n=500
  - Case p=500 and n=1000
  - Case p=250 and n=1000
  - Conclusion

Multivariate normals

First, we load some packages.

```
library(Matrix)   # matrix operations
library(MASS)     # Multivariate Normal Distribution
```

Generating random samples

Generate a Multivariate Normal Sample as a linear combination of iid standard normal rvs.

```
len<-5
N<-matrix(rnorm(len*2),len,2) # 5x2 iid N(0,1) rvs
A<-matrix(c(1,1,.5,.1),2,2)   # 2x2 matrix of coefficients
X<-N%*%A                      # 5x2 linear combination
```

We can also generate a Multivariate Normal Sample using the R function `mvrnorm`.

```
Sigma <- matrix(c(10,4,4,2),2,2)
mvrnorm(n=1,c(0,0),Sigma) # sample 1x2 with mean [0,0]
```

```
## [1] -6.137565 -2.142496
```

```
mvrnorm(n=5,c(0,0),Sigma) # sample 5x2 with mean [0,0]
```

```
##           [,1]       [,2]
## [1,] -1.114340 -1.4446878
## [2,]  5.847722  2.6706654
## [3,]  2.942172  0.7841130
## [4,] -1.208800 -0.3271502
## [5,] -2.876475 -1.6989882
```

```
mvrnorm(n=5,c(-100,100),Sigma) # sample 5x2 with mean [-100,100]
```

```
##            [,1]      [,2]
## [1,]  -98.74231 100.12753
## [2,] -105.78871  98.65900
## [3,]  -99.38140 100.67298
## [4,] -100.21660 100.01420
## [5,] -105.56765  97.64273
```

Calculating the variance

Here, `Sigma` is the population variance.

```
var(mvrnorm(n=1000, rep(0, 2), Sigma))
```

```
##          [,1]     [,2]
## [1,] 9.553365 3.845611
## [2,] 3.845611 1.962885
```

Here, `Sigma` is the sample variance.

```
var(mvrnorm(n=1000, rep(0, 2), Sigma, empirical = TRUE))
```

```
##      [,1] [,2]
## [1,]   10    4
## [2,]    4    2
```

Calculating correlation and covariance matrices

We calculate the correlation matrices for `N` and `X`.

```
cor(N)
```

```
##           [,1]      [,2]
## [1,]  1.000000 -0.924855
## [2,] -0.924855  1.000000
```

```
cor(X)
```

```
##            [,1]       [,2]
## [1,]  1.0000000 -0.5767201
## [2,] -0.5767201  1.0000000
```

We can calculate the variance-covariance matrices for `N` and `X`.

```
cov(N)
```

```
##            [,1]      [,2]
## [1,]  0.5922935 -1.180500
## [2,] -1.1805002  2.750729
```

```
cov(X)
```

```
##            [,1]        [,2]
## [1,]  0.9820224 -0.13708043
## [2,] -0.1370804  0.05753065
```

These give both the same results as the function `cov`.

```
var(N)
```

```
##            [,1]      [,2]
## [1,]  0.5922935 -1.180500
## [2,] -1.1805002  2.750729
```

```
var(X)
```

```
##            [,1]        [,2]
## [1,]  0.9820224 -0.13708043
## [2,] -0.1370804  0.05753065
```

Sampling a multivariate normal distribution

with given covariance

Setup the variance matrices.

```
Sigma <- matrix(c(10,4,4,2),2,2)
I<-diag(c(1,1)) # identity matrix
```

Generate two large samples with these variances.

```
N<-mvrnorm(n=10000,c(0,0),I)
X<-mvrnorm(n=10000,c(0,0),Sigma)
```

Perform the spectral decomposition and obtain the eigenvectors and eigenvalues.

```
e<-eigen(Sigma)
P<-e$vectors
L<-e$values
```

Generate the *inverse square-root matrix* using $A^k = PD^kP^{-1}$ for $k \in \mathbb{Z}$ where D is a diagonal matrix.

```
Sm <- P%*%sqrt(diag(1/L))%*%t(P)
```

Using the same technique generate the *square-root matrix*.

```
Sp <- P%*%sqrt(diag(L))%*%t(P)
```

Generate a vector of iid $N(0, 1)$ rvs.

```
Z<-t(Sm%*%t(X))
```

Generate a MVN rv with variance `Sigma`.

```
X1<-t(Sp%*%t(N))
```

Check the variances.

```
var(Z)
```

```
##             [,1]       [,2]
## [1,] 1.01825688 0.01466004
## [2,] 0.01466004 1.01941388
```

```
var(X1)
```

```
##           [,1]     [,2]
## [1,] 10.111164 4.023844
## [2,]  4.023844 2.002236
```

```
Sigma
```

```
##      [,1] [,2]
## [1,]   10    4
## [2,]    4    2
```

Densities

We can use the package `mvtnorm`. If you don't have it, you can install it with `install.packages('mvtnorm')` and then load it.

```
library(mvtnorm)
```

```
dmvnorm(x=c(0,0))
```

```
## [1] 0.1591549
```

```
dmvnorm(x=c(0,0), mean=c(1,1))
```

```
## [1] 0.05854983
```

```
sigma <- matrix(c(4,2,2,3), ncol=2)
x <- rmvnorm(n=500, mean=c(1,2), sigma=sigma)
colMeans(x)
```

```
## [1] 0.9830814 2.0199301
```

```
var(x)
```

```
##          [,1]     [,2]
## [1,] 4.220998 2.039222
## [2,] 2.039222 3.019053
```

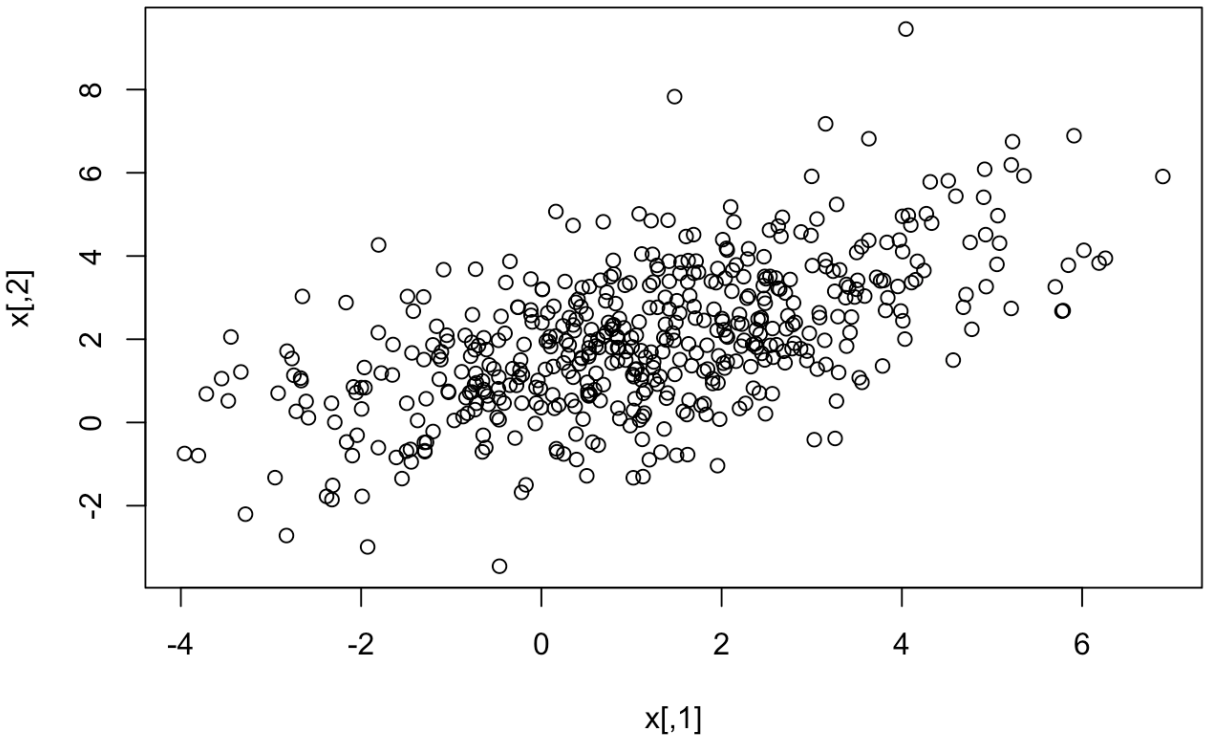
```
x <- rmvnorm(n=500, mean=c(1,2), sigma=sigma,method="chol")
colMeans(x)
```

```
## [1] 1.130779 1.997141
```

```
var(x)
```

```
##          [,1]     [,2]
## [1,] 3.984907 2.014604
## [2,] 2.014604 3.043778
```

```
plot(x)
```



```
mu<-c(5, 10)
P<-cov2cor(V = Sigma)
P
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.8944272
## [2,] 0.8944272 1.0000000
```

Explicit calculation at mean.

```
dmvnorm(x = mu, mean = mu, sigma = Sigma)
```

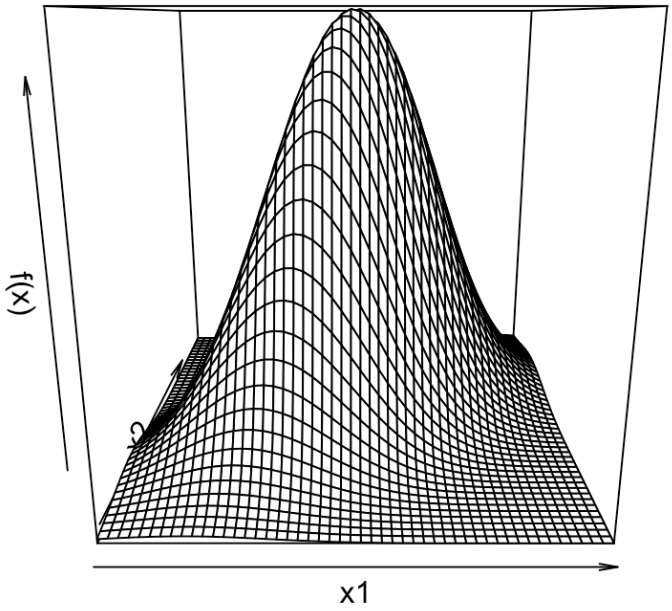
```
## [1] 0.07957747
```

Generate points where we shall evaluate.

```
x1<-seq(from = mu[1]-5, to = mu[1]+5, length=50)
x2<-seq(from = mu[2]-5, to = mu[2]+5, length=50)
all.x <- expand.grid(x1, x2)
```

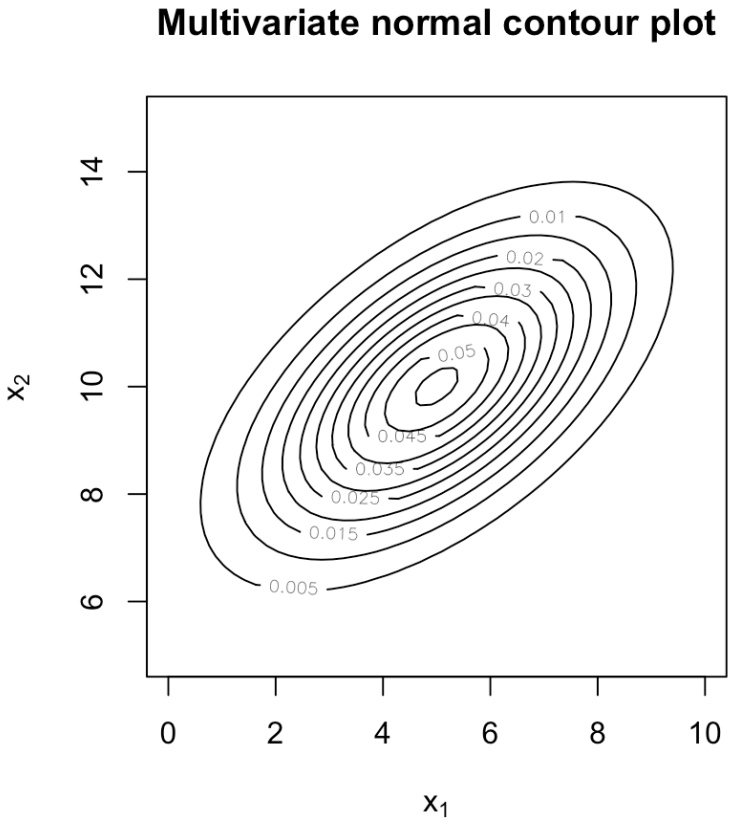
```
fx <- matrix(data = dmvnorm(x = all.x, mean = mu, sigma = sigma), nrow = length
(x1), ncol = length(x2), byrow = FALSE)
```

```
persp(x = x1, y = x2, z = fx, col = "white", xlab = "x1", ylab = "x2", zlab = "
f(x)")
```



Contour plot - purposely made x and y-axes the same length so that one can judge variability.

```
par(pty = "s")
contour(x = x1, y = x2, z = fx, main = "Multivariate normal contour plot", xlab
= expression(x[1]), ylab = expression(x[2]))
```



Two variables - Show eigenvectors on contour plot.

```
mu<-c(5, 10)
sigma<-matrix(data = c(1, 0.5, 0.5, 1.25), nrow = 2, ncol = 2, byrow = TRUE)

x1<-seq(from = 0, to = 15, by = 0.1)
x2<-seq(from = 0, to = 15, by = 0.1)
all.x<-expand.grid(x1, x2)
fx<-matrix(data = dmvnorm(x = all.x, mean = mu, sigma = sigma), nrow = length(x
1), ncol = length(x2), byrow = FALSE)

par(pty = "s")
contour(x = x1, y = x2, z = fx, main = expression(paste("Multivariate normal co
ntour plot with eigenvectors for ", Sigma)), xlab = expression(x[1]), ylab = ex
pression(x[2]), levels = c(0.01, 0.001), xlim=c(-5, 15))

abline(h = seq(from = -10, to = 30, by = 10), lty = "dotted", col = "lightgray"
)
abline(v = seq(from = -10, to = 30, by = 10), lty = "dotted", col = "lightgray"
)

abline(h = 0, lwd = 2)
abline(v = 0, lwd = 2)

save.eig<-eigen(sigma)
save.eig$values
```

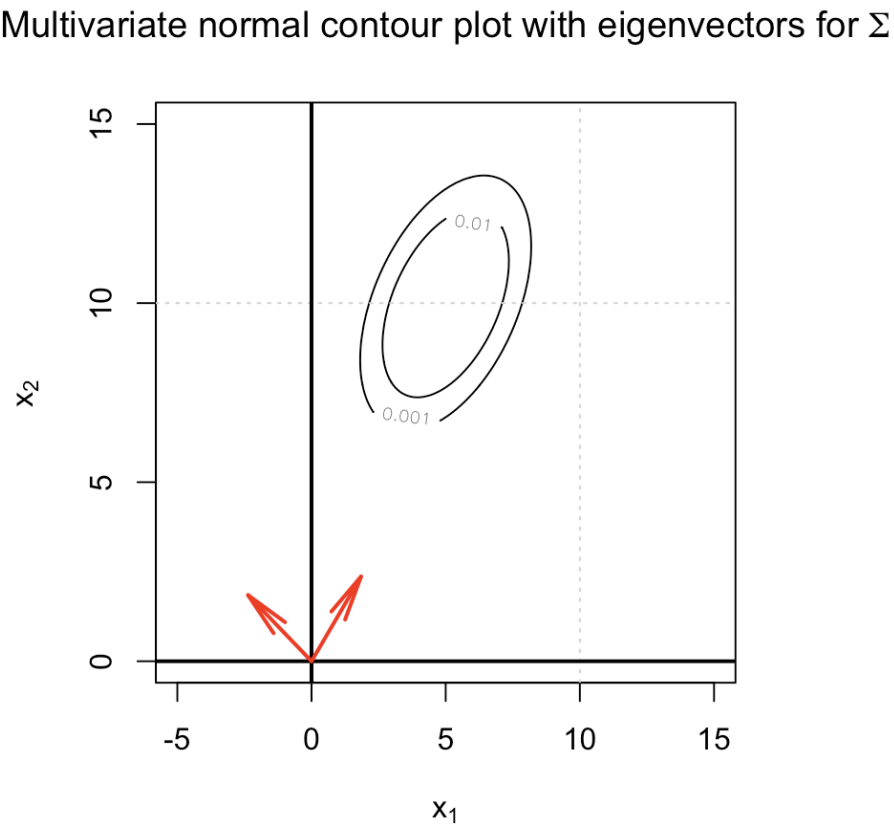
```
## [1] 1.6403882 0.6096118
```

```
save.eig$vectors
```

```
##           [,1]       [,2]
## [1,] 0.6154122 -0.7882054
## [2,] 0.7882054  0.6154122
```

```
arrows(x0 = 0, y0 = 0, x1 = 3*save.eig$vectors[1,1], y1 = 3*save.eig$vectors[2,
1], col = "red", lty = "solid",angle=10,lwd=2)

arrows(x0 = 0, y0 = 0, x1 = 3*save.eig$vectors[1,2], y1 = 3*save.eig$vectors[2,
2], col = "red", lty = "solid",angle=10,lwd=2)
```



Eigenvalues of sample covariance matrix

Sample a covariance based on $\mathbb{S}_n = \frac{1}{n}\mathbb{X}\mathbb{X}^*$ where $\mathbb{X}$ is a $p \times n$ matrix with Gaussian entries with mean zero and variance 1. This gives a matrix $\mathbb{S}_n$ of size $p \times p$.

Case p=50 and n=500

```
p <- 50
n <- 500
X <- matrix(rnorm(p*n), p, n)
Sn <- X %*% t(X) / n
dim(Sn)
```

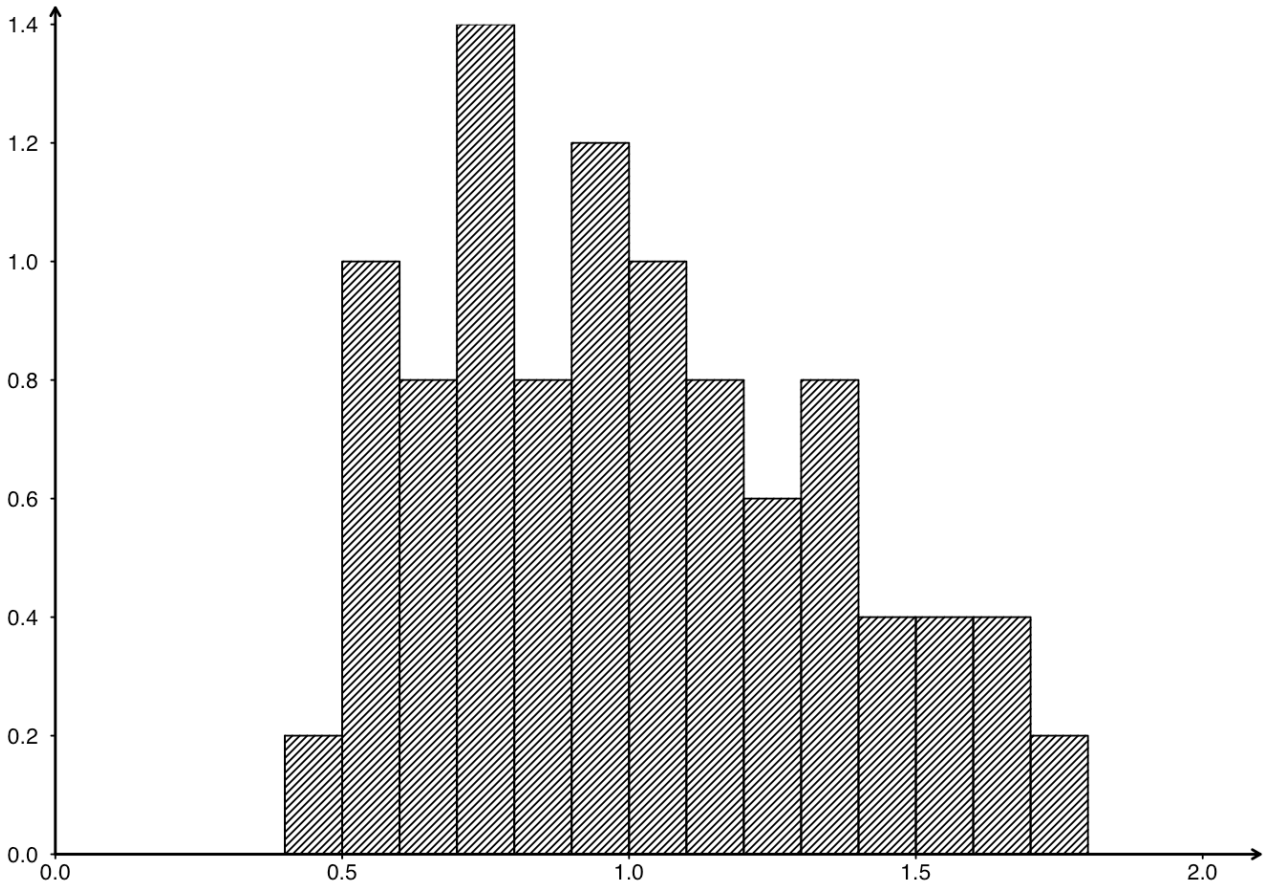
```
## [1] 50 50
```

Calculate the eigenvalues.

```
e<-eigen(Sn)
L<-e$values
```

Plot a histogram of the eigenvalues.

```
hist(L, breaks=p/3, xlim=c(0,1.2*max(L)), freq=FALSE, col=1, main='')
```



Case p=250 and n=500

```
p <- 250
n <- 500
X <- matrix(rnorm(p*n), p, n)
Sn <- X %*% t(X) / n
dim(Sn)
```

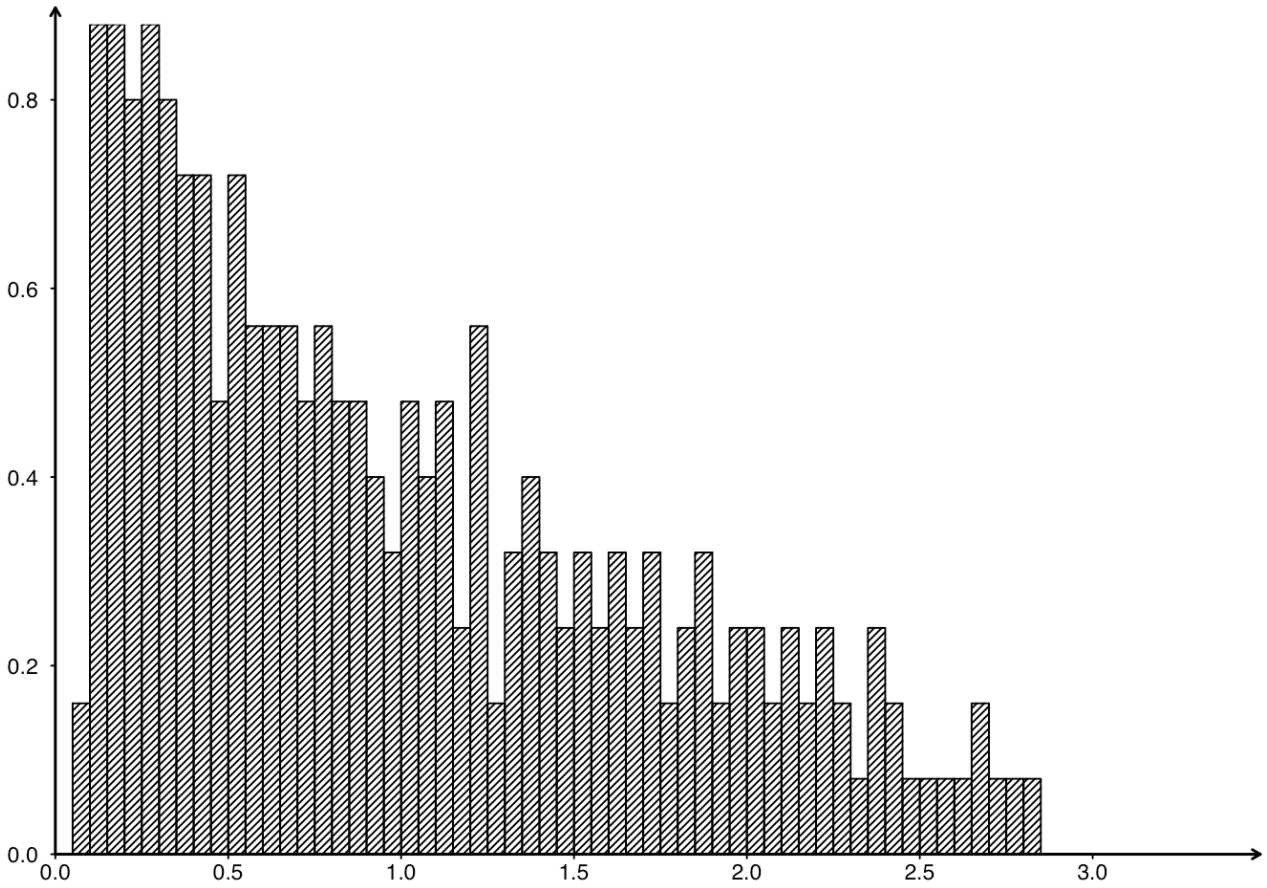
```
## [1] 250 250
```

Calculate the eigenvalues.

```
e<-eigen(Sn)
L<-e$values
```

Plot a histogram of the eigenvalues.

```
hist(L, breaks=p/3, xlim=c(0,1.2*max(L)), freq=FALSE, col=1, main='')
```



Case p=500 and n=1000

```
p <- 500
n <- 1000
X <- matrix(rnorm(p*n), p, n)
Sn <- X %*% t(X) / n
dim(Sn)
```

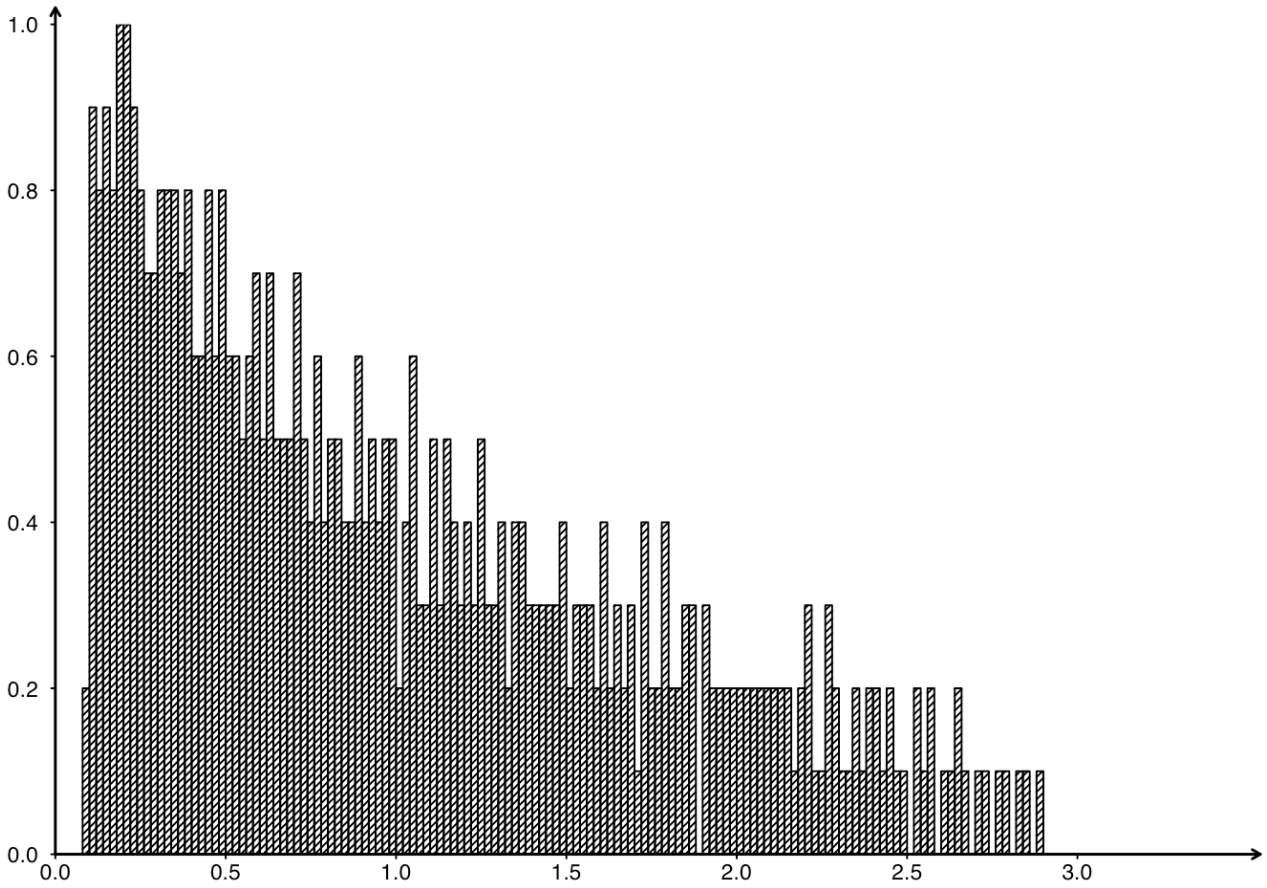
```
## [1] 500 500
```

Calculate the eigenvalues.

```
e<-eigen(Sn)
L<-e$values
```

Plot a histogram of the eigenvalues.

```
hist(L, breaks=p/3, xlim=c(0,1.2*max(L)), freq=FALSE, col=1, main='')
```



Case p=250 and n=1000

```
p <- 250
n <- 1000
X <- matrix(rnorm(p*n), p, n)
Sn <- X %*% t(X) / n
dim(Sn)
```

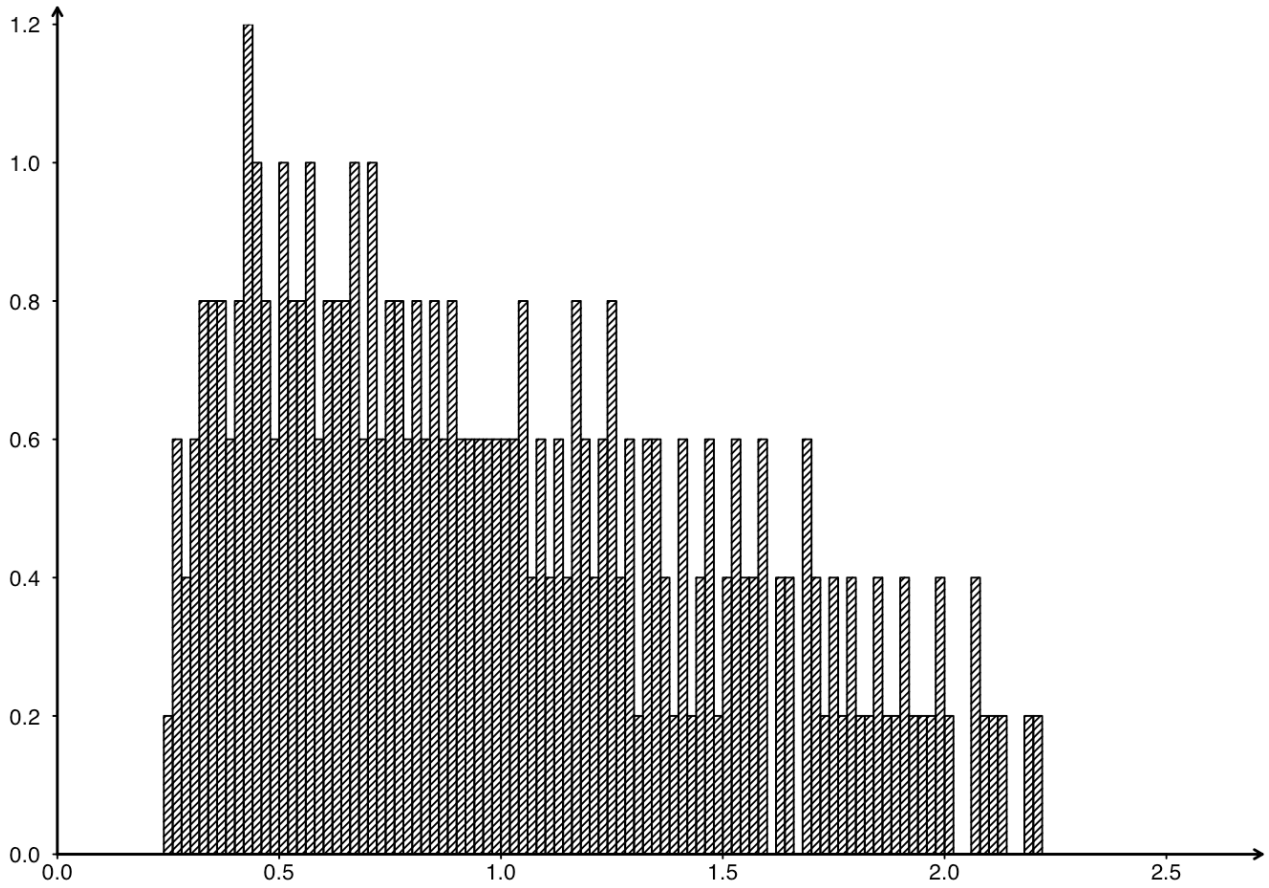
```
## [1] 250 250
```

Calculate the eigenvalues.

```
e<-eigen(Sn)
L<-e$values
```

Plot a histogram of the eigenvalues.

```
hist(L, breaks=p/3, xlim=c(0,1.2*max(L)), freq=FALSE, col=1, main='')
```



Conclusion

Do you notice something about the shape of the histograms as p and n change? Go back and think about what the ratio p/n is for each plot. What do you think will happen to the histogram as $n \to \infty$ and $p \to \infty$ while keeping $0 < p/n < 1$.