Workshop 4

- Introducing correlation between components
  - Example: Deterministic correlation
  - Cholesky decomposition for the square root
  - Spectral decomposition for the square root
  - Plotting observations in two-dimensional case
- Nonparametric estimate of the LSD
  - Kernel density estimators
  - KDE of the LSD
  - Effect of a correlation matrix
  - Stochastic $\mathbf{T}_n$

In this workshop we are going to look at the *generalised* Marcenko-Pastur distribution which is a description of the behaviour of the eigenvalues (of the sample covariance matrix) in the case where correlation is introduced between the entries of the p-dimensional vector observations.

Part of the workshop will look at reproducing the results in the paper by Jing, Pan, Shao and Zhou (2010) titled “Nonparametric estimate of spectral density functions of sample covariance matrices: a first step” published in The Annals of Statistics.

Introducing correlation between components

The MP situation is very limited as entries of the observations are assumed to be independent with mean zero and variance 1. In other words, the sample covariance matrix is $\mathbf{S}_n = \frac{1}{n}\mathbf{X}\mathbf{X}^*$ where $\mathbf{X}$ is a $p \times n$ data matrix. Here, $\mathbf{S}_n$ is a matrix of size $p \times p$.

If $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is a $p \times n$ data matrix, we can introduce a correlation between i’th and j’th (where $1 \leq i, j \leq p$) component of each observation vector $\mathbf{x}_1, \ldots, \mathbf{x}_n$ by introducing a $p \times p$ nonrandom Hermitian nonnegative definite matrix $\mathbf{T}_n$ and then consider random matrices of the form

$$\mathbf{A}_n = \frac{1}{n}\mathbf{T}_n^{1/2}\mathbf{X}\mathbf{X}^*\mathbf{T}_n^{1/2}.$$

Example: Deterministic correlation

A simple deterministic case would be $\mathbf{T}_n = (\rho^{|i-j|})$ for $1 \leq i, j \leq p$ where ρ is a constant. For example, when $n = 3$ we get

$$\mathbf{T}_n = \begin{pmatrix} 1 & \rho^{|1-2|} & \rho^{|1-3|} \\ \rho^{|2-1|} & 1 & \rho^{|2-3|} \\ \rho^{|3-1|} & \rho^{|3-2|} & 1 \end{pmatrix} = \begin{pmatrix} 1 & \rho & \rho^2 \\ \rho & 1 & \rho \\ \rho^2 & \rho & 1 \end{pmatrix}.$$

For example, set $\rho = 0.4$ and generate the 3×3 matrix by hand.

```
rho <- 0.4
Tn <- matrix(c(1, rho, rho^2, rho, 1, rho, rho^2, rho, 1), 3, 3)
Tn
```

```
##      [,1] [,2] [,3]
## [1,] 1.00  0.4 0.16
## [2,] 0.40  1.0 0.40
## [3,] 0.16  0.4 1.00
```

Of course, when p becomes large it is going to be very hard to create the matrix T_n by hand so let's write a function that generates this "power correlation matrix" for an arbitrary p.

```
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 test it to see if we get the same thing as our hand generated matrix.

```
Tn <- pcor(rho, 3)
Tn
```

```
##      [,1] [,2] [,3]
## [1,] 1.00  0.4 0.16
## [2,] 0.40  1.0 0.40
## [3,] 0.16  0.4 1.00
```

Yep, that looks the same.

Given, the matrix $\mathbf{T}_n$ we want to obtain the "square root matrix" $\mathbf{T}_n^{1/2}$.

Processing linear algebra 20%

In linear algebra, given a positive semidefinite matrix T with complex entries, B is a *square root* of T if

$T = B^*B$, where B^* denotes the Hermitian adjoint of B.

Cholesky decomposition for the square root

One way is to use a matrix $\mathbf{C}$ such that $\mathbf{C}^*\mathbf{C} = \mathbf{T}_n$. This decomposition is called the of $\mathbf{T}_n$ and can be obtained using the `chol` function.

```
Q <- chol(Tn)
```

If you want to be able to recover $\mathbf{T}_n$, you need to fiddle a bit (see documentation for `chol`).

```
Q <- chol(Tn, pivot=TRUE)
pivot <- attr(Q, "pivot")
Q <- Q[, order(pivot)]
```

Now it should work.

```
t(Q) %*% Q
```

```
##      [,1] [,2] [,3]
## [1,] 1.00  0.4 0.16
## [2,] 0.40  1.0 0.40
## [3,] 0.16  0.4 1.00
```

Unfortunately, as you can see by the fiddling above, this is not the *unique non-negative square root*. But it is the more general case and useful for many applications.

Spectral decomposition for the square root

Another approach is to use the spectral decomposition of the matrix $\mathbf{T}_n$. This approach works if and only if $\mathbf{T}_n$ has p eigenvectors (which is our case).

```
eigs <- eigen(Tn)
Q <- eigs$vectors %*% diag(sqrt(eigs$values)) %*% solve(eigs$vectors)
```

Testing that we recover $\mathbf{T}_n$.

```
t(Q) %*% Q
```

```
##      [,1] [,2] [,3]
## [1,] 1.00  0.4 0.16
## [2,] 0.40  1.0 0.40
## [3,] 0.16  0.4 1.00
```

This spectral decomposition approach is nice as we also have

```
Q %*% Q
```

```
##      [,1] [,2] [,3]
## [1,] 1.00  0.4 0.16
## [2,] 0.40  1.0 0.40
## [3,] 0.16  0.4 1.00
```

We can define Q to be our square root, i.e., $Q = \mathbf{T}_n^{1/2}$.

Plotting observations in two-dimensional case

Generate a data matrix with two-dimensional entries.

```
p <- 2
n <- 500
X <- matrix(rnorm(p*n), p, n)
```

Generate the power correlations.

```
rho <- 0.8
Tn <- pcor(rho, 2)
```

This is the (true!) *population* covariance.

```
Tn
```

```
##      [,1] [,2]
## [1,]  1.0  0.8
## [2,]  0.8  1.0
```

Generate the square-root matrix $\mathbf{T}_n^{1/2}$.

```
eigs <- eigen(Tn)
Q <- eigs$vectors %*% diag(sqrt(eigs$values)) %*% solve(eigs$vectors)
```

We can generate the correlated observations.

```
x <- Q %*% X
```

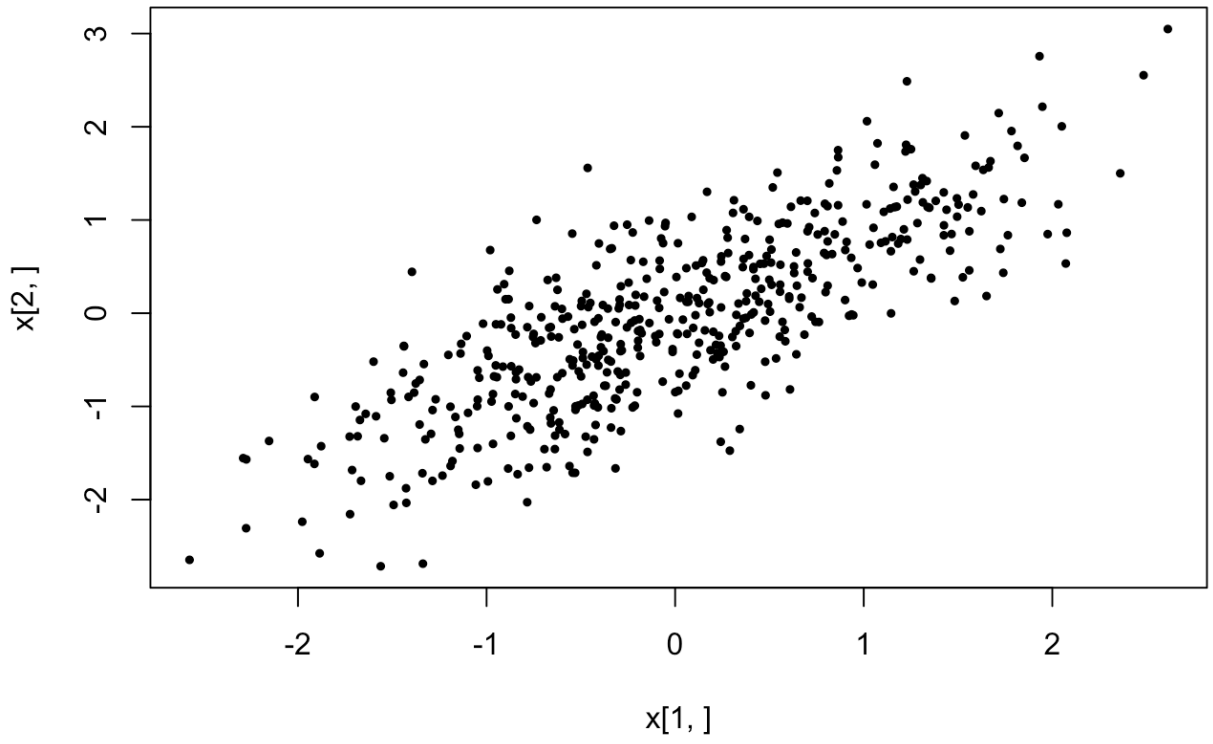
We have a matrix of dimensions $p \times n$.

```
dim(x)
```

```
## [1]   2 500
```

Since we are considering the $p = 2$ case, it is easy to plot the n observations as points in the two-dimensional plane.

```
plot(x[1,], x[2,], pch=19, cex=0.5)
```



Notice how the points are very correlated. Try varying ρ between -1 and 1 and redoing the plot.

We now generate the sample covariance matrix in the correlated case. Note this is a $p \times p$ matrix.

```
An <- Q %*% X %*% t(X) %*% Q / n
An
```

```
##           [,1]      [,2]
## [1,] 0.8802042 0.7110265
## [2,] 0.7110265 0.9298009
```

Notice how this *sample* covariance matrix $\mathbf{A}_n$ is close to the *population* covariance matrix $\mathbf{T}_n$ but not exactly the same. $\mathbf{A}_n$ changes every time we sample new observations and this is why your $\mathbf{A}_n$ might be slightly different to the one outputed above (in my R session).

Our aim (in this course!) is to describe the distribution of the eigenvalues of this sample covariance matrix as p and n become very large with $p/n \to y > 0$.

Of course here in this example we have looked at the simple case where $p = 2$ and $n = 500$. So we only have two eigenvalues (which is pretty boring…).

Nonparametric estimate of the LSD

In the case when $\mathbf{T}_n = I$ (i.e., the identity matrix), we can obtain a characterisation of the limiting spectral distribution (LSD) and this distribution is given by the Marcenko-Pastur (MP) law. In Workshop 2 we implemented the MP density as follows.

```
dmp <- function(x, y, sigma=1) {
  a <- (1-sqrt(y))^2
  b <- (1+sqrt(y))^2
  ifelse(x <= a | x >= b, 0, suppressWarnings(sqrt((x - a) * (b - x))/(2 * pi *
sigma * x * y)))
}
```

Unfortunately, for a general $\mathbf{T}_n$ we cannot get an explicit closed-form expression for the LSD. We will now look at one approach to approximating the LSD given in the paper by Jing, Pan, Shao and Zhou.

Kernel density estimators

Let's consider the univariate case. Suppose that obervations $X_1, \ldots, X_n$ are i.i.d. random variables with an unknown density function $f(x)$ and $F_n(x)$ is the empirical distribution function determined by the sample. We call a *kernel density estimator* of $f(x)$ the function

$$\hat{f}_n(x) = \frac{1}{nh} \sum_{j=1}^{n} K\left(\frac{x - X_j}{h}\right).$$

where the function $K(y)$ is the *kernel function* and $h = h(n)$ is the *bandwidth* which tends to 0 as $n \to \infty$. There is a large body of literature on this topic and, under appropriate conditions, it can be shown that

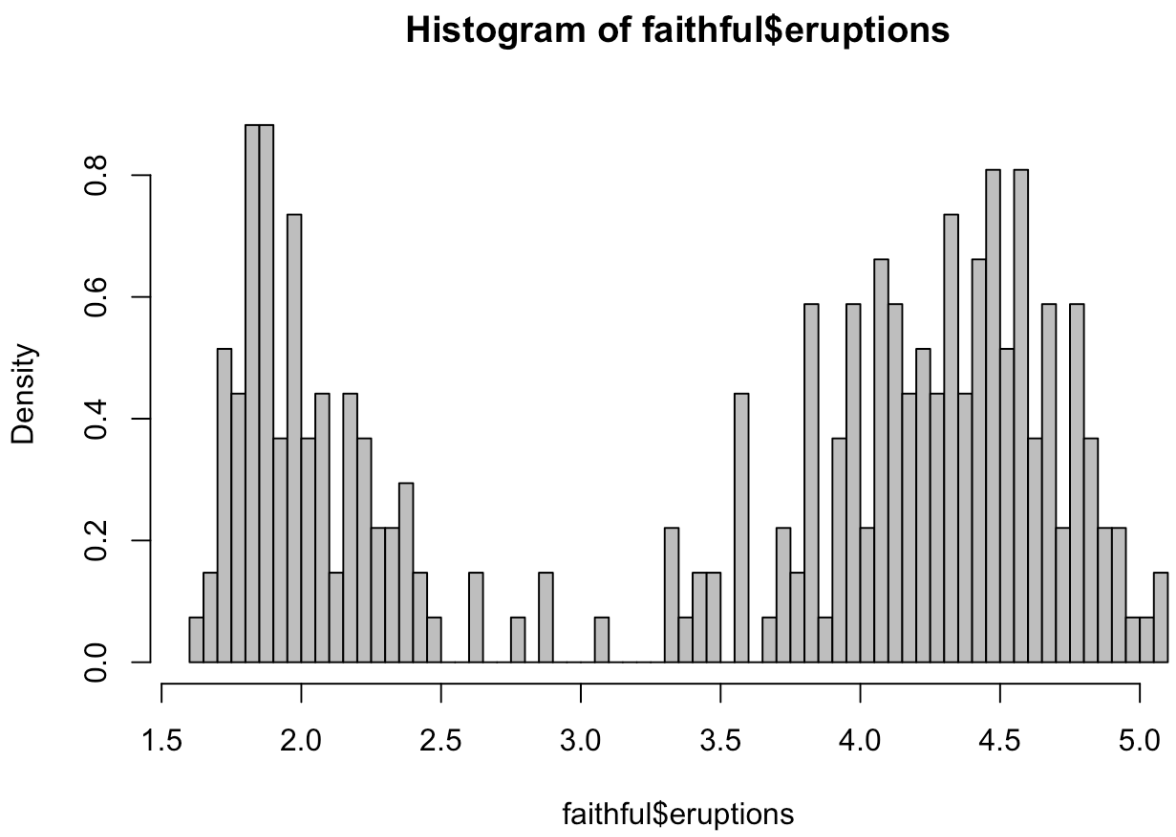
$$\hat{f}_n(x) \to f(x)$$

as $n \to \infty$ (in some appropriate sense).

Since this is such a common technique in statistics, it is directly built into R. The function is called `density`.

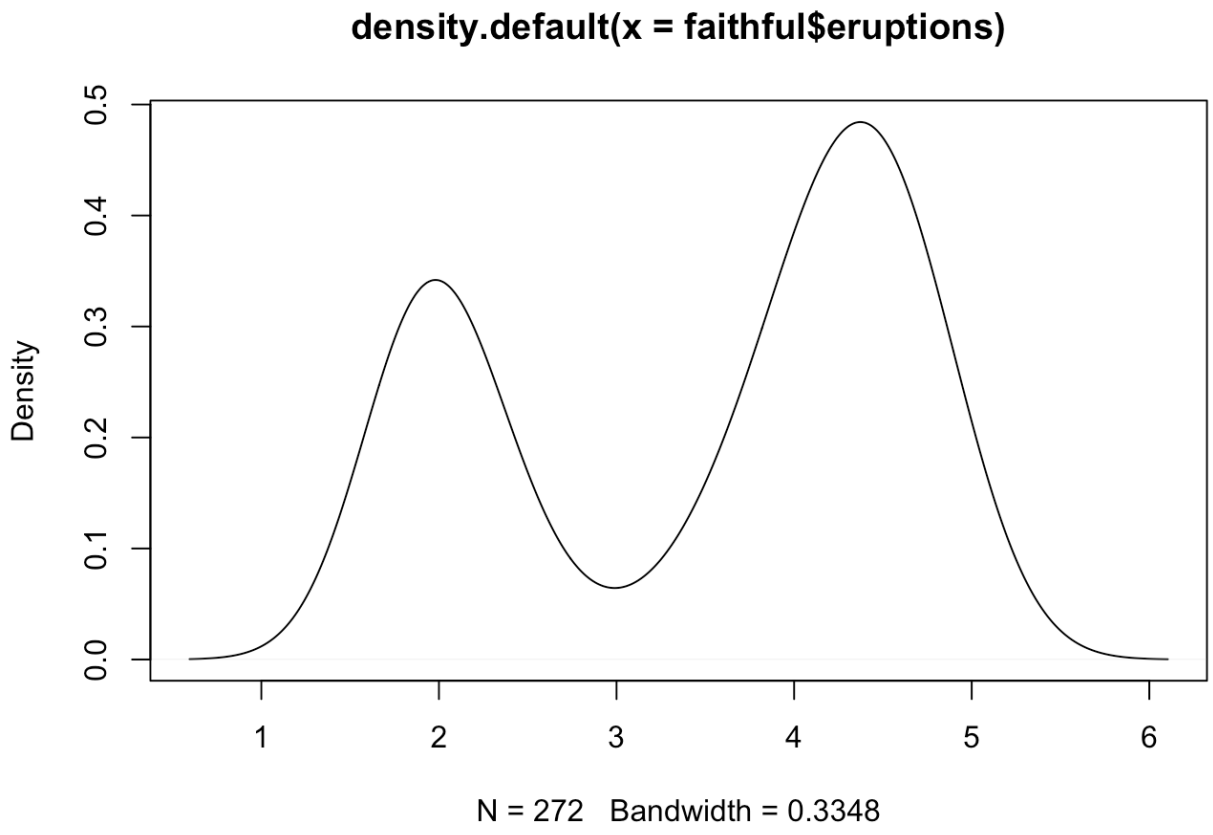
For example, take the `faithful` dataset and plot a histogram of the eruptions.

```
hist(faithful$eruptions, breaks=50, freq=FALSE, col=8)
```



We can use the `density` function to get a kernel density estimator.

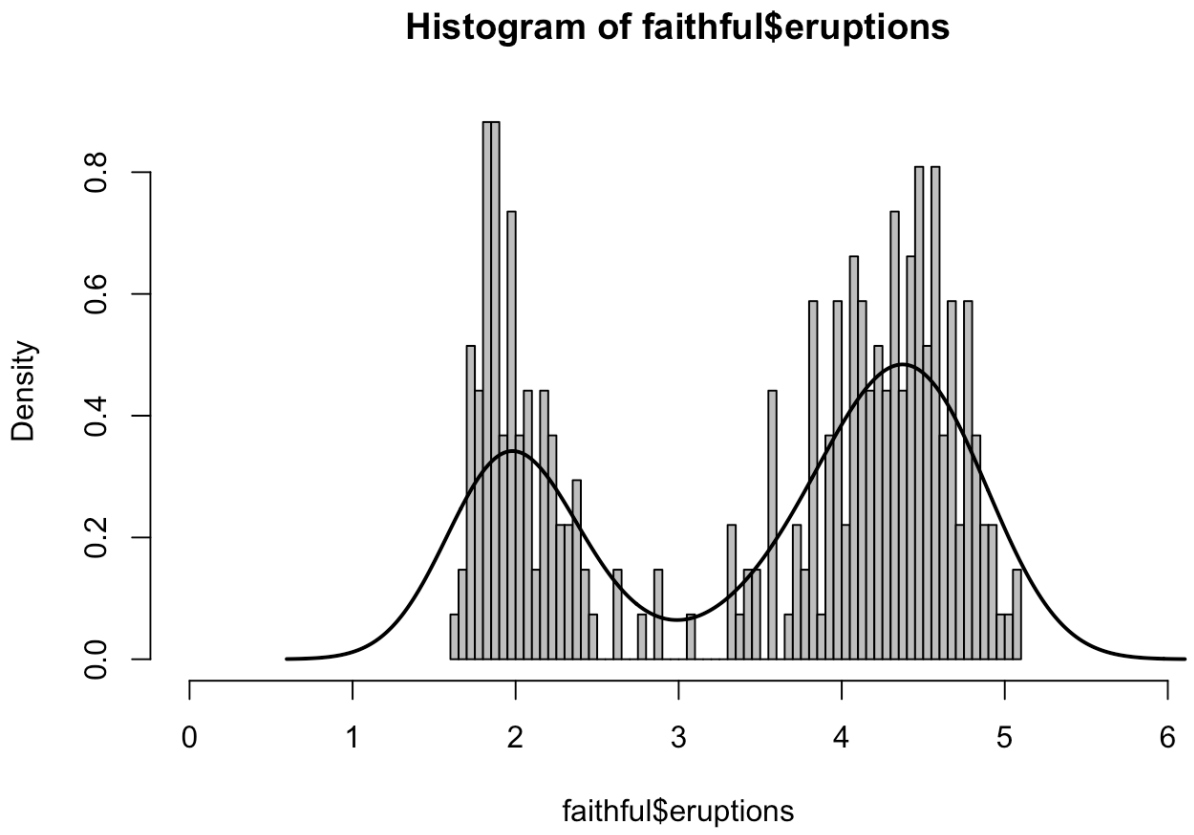
```
d <- density(faithful$eruptions)
plot(d)
```



Or superimposing on top of the histogram.

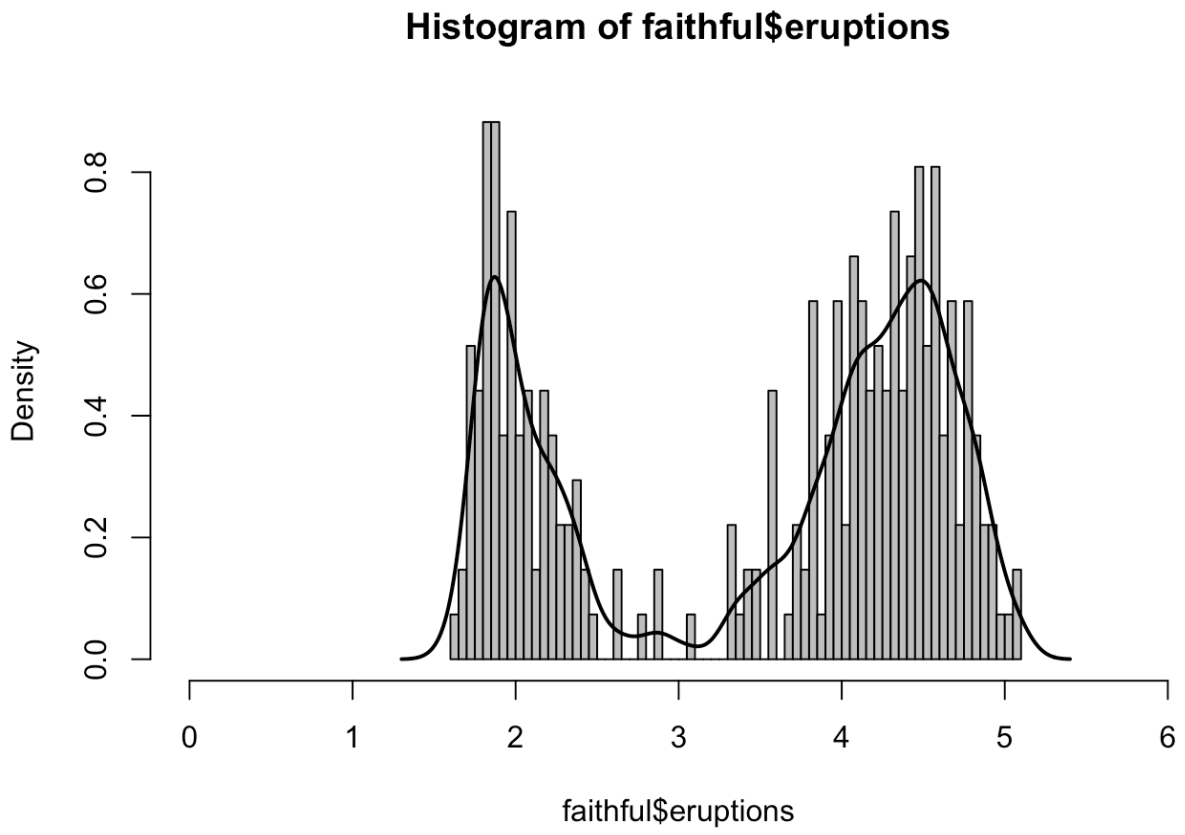
```
hist(faithful$eruptions, breaks=50, freq=FALSE, col=8, xlim = c(0, 6))
lines(d, col=1, lwd=2)
```

Processing math: 100%



There are many kernels to choose from (see documentation for `density`). This is using a Gaussian kernel with bandwidth $h = 0.1$.

```
hist(faithful$eruptions, breaks=50, freq=FALSE, col=8, xlim = c(0, 6))
d <- density(faithful$eruptions, kernel="gaussian", bw=0.1)
lines(d, col=1, lwd=2)
```



KDE of the LSD

We shall now look at obtaining the kernel density estimator for the limiting spectral distribution. In their paper, the authors propose, estimating the density of the LSD by

$$\hat{f}_n(x) = \frac{1}{ph}\sum_{i=1}^{p} K\left(\frac{x-\mu_i}{h}\right),$$

where μ_i, $i = 1, \ldots, p$ are the eigenvalues of $\mathbf{A}_n$.

To check that this works let's look at the MP case first, i.e. $\mathbf{T}_n = I$, and compare it to the closed-form MP density which is given by

```
dmp <- function(x, y, sigma=1) {
  a <- (1-sqrt(y))^2
  b <- (1+sqrt(y))^2
  ifelse(x <= a | x >= b, 0, suppressWarnings(sqrt((x - a) * (b - x))/(2 * pi *
sigma * x * y)))
}
```

```
p <- 100
n <- 500
X <- matrix(rnorm(p*n), p, n)
Sn <- X %*% t(X) / n
dim(Sn)
```

```
## [1] 100 100
```

Calculate the eigenvalues.

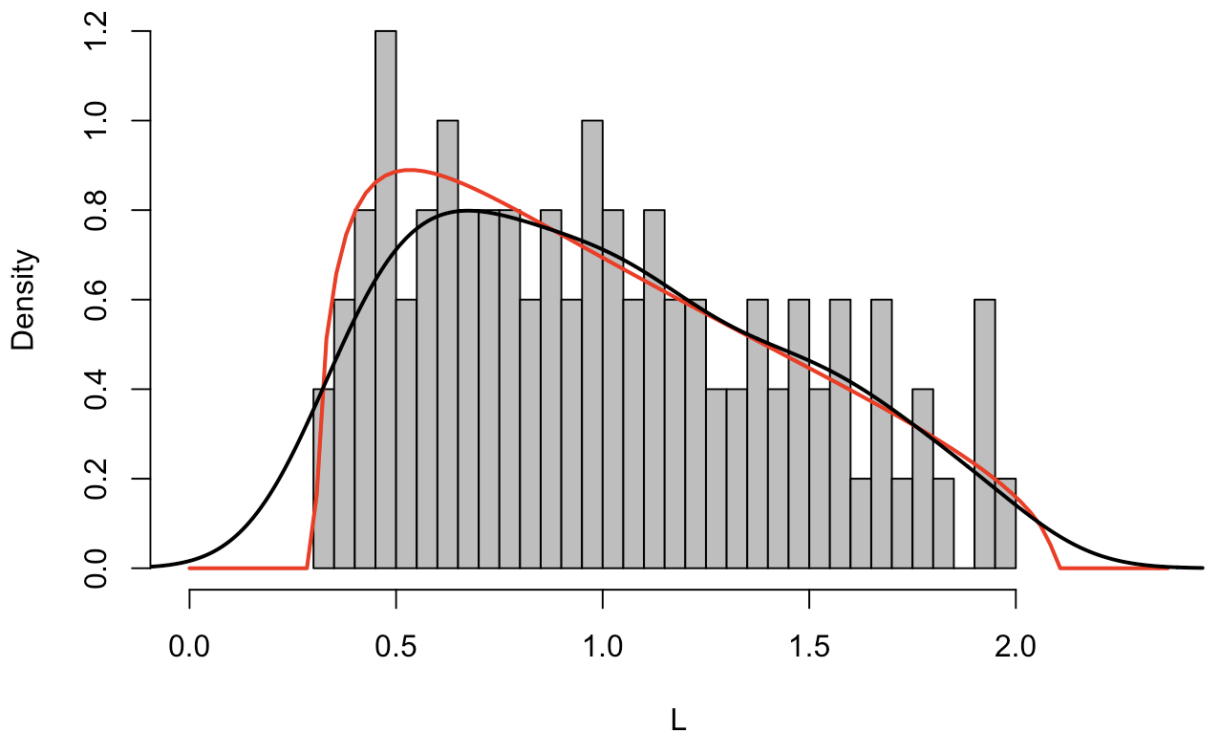
```
e<-eigen(Sn)
L<-e$values
```

Generate the KDE of the eigenvalues with the default choice of bandwidth.

```
d <- density(L, kernel="gaussian")
```

Plot a histogram of the eigenvalues against the MP density.

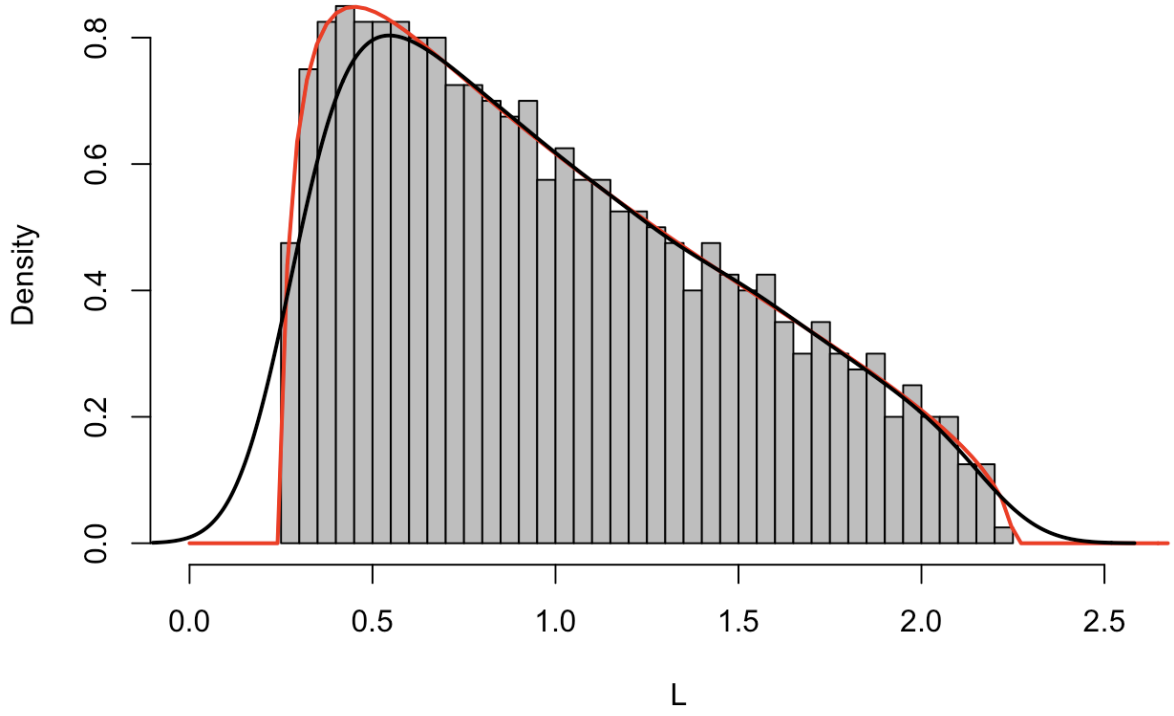
```
hist(L, breaks=50, xlim=c(0,1.2*max(L)), freq=FALSE, col=8, main='')
curve(dmp(x, y=p/n), from = 0, to = 1.2*max(L), lty=1, lw=2, col=2, add=TRUE)
lines(d, col=1, lwd=2)
```



We can see that the estimator (black) is only approximate in this case of $p = 100$ and $n = 500$.

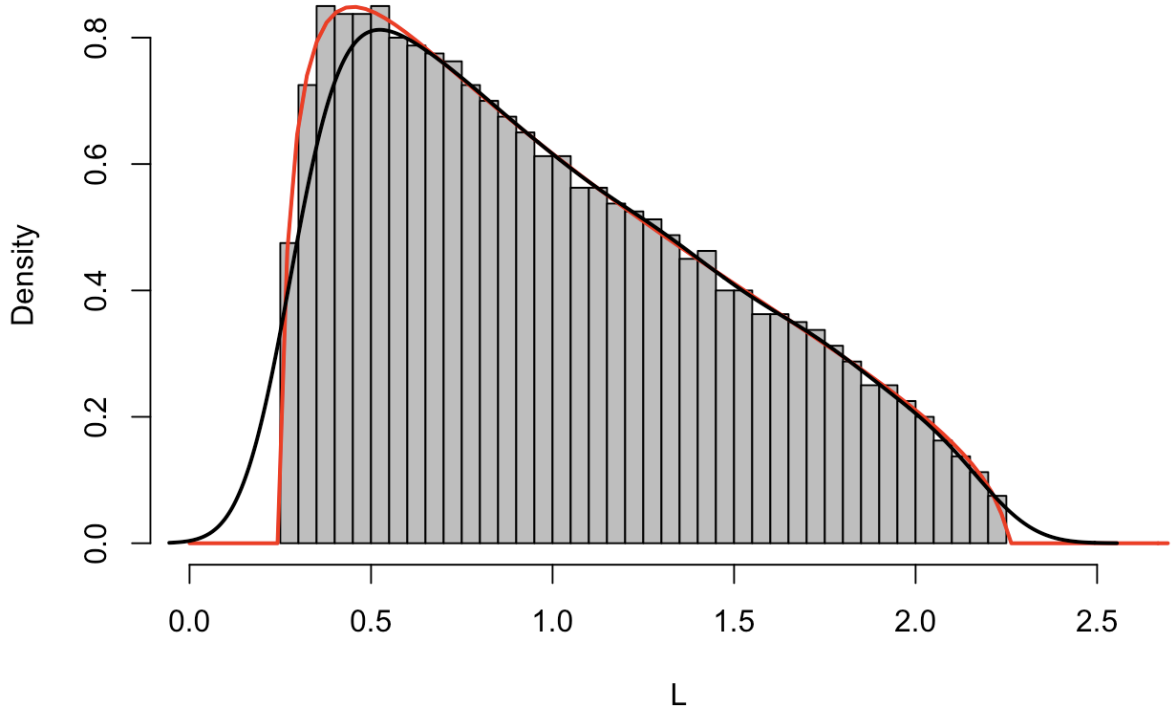
Let's redo the whole thing again for a larger p and n.

```
p <- 800
n <- 3200
X <- matrix(rnorm(p*n), p, n)
Sn <- X %*% t(X) / n
L<-eigen(Sn)$values
d <- density(L, kernel="gaussian")

hist(L, breaks=50, xlim=c(0,1.2*max(L)), freq=FALSE, col=8, main='')
curve(dmp(x, y=p/n), from = 0, to = 1.2*max(L), lty=1, lw=2, col=2, add=TRUE)
lines(d, col=1, lwd=2)
```



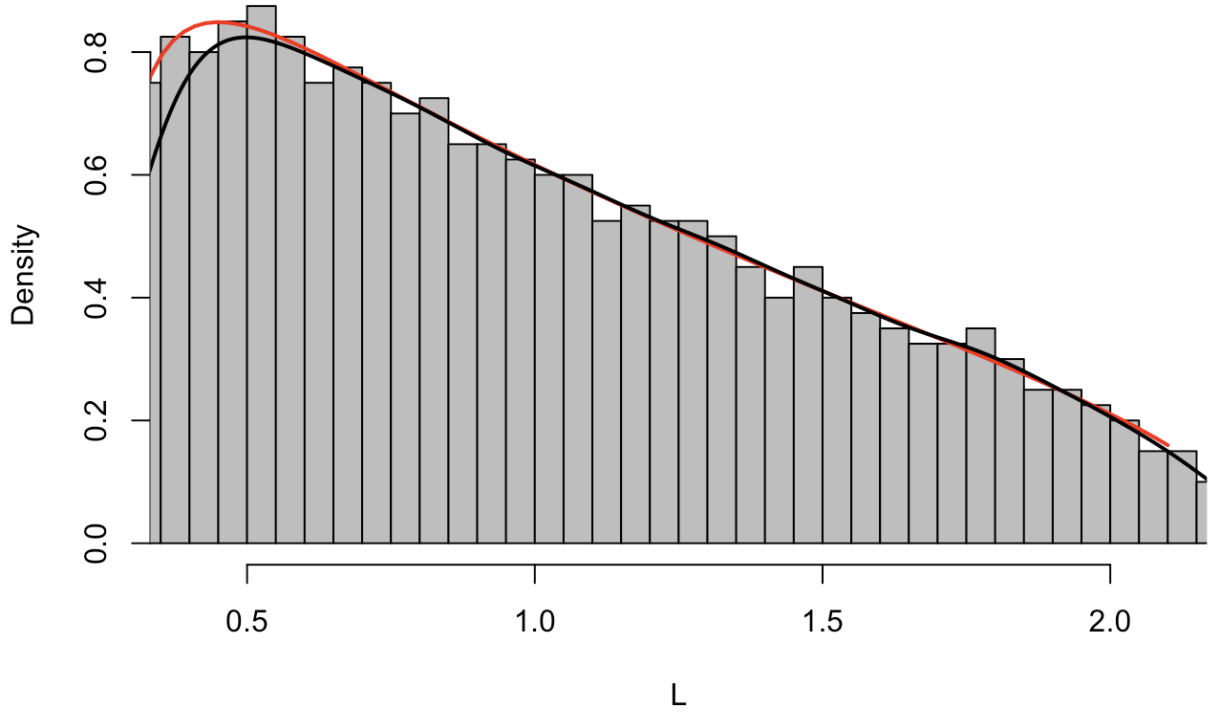
And again.

```
p <- 2*800
n <- 2*3200
X <- matrix(rnorm(p*n), p, n)
Sn <- X %*% t(X) / n
L<-eigen(Sn)$values
d <- density(L, kernel="gaussian")

hist(L, breaks=50, xlim=c(0,1.2*max(L)), freq=FALSE, col=8, main='')
curve(dmp(x, y=p/n), from = 0, to = 1.2*max(L), lty=1, lw=2, col=2, add=TRUE)
lines(d, col=1, lwd=2)
```



As you can see in the paper, the authors are a bit dodgy as they make their figure look good by only plotting from roughly 0.4 to 2.1 like so:

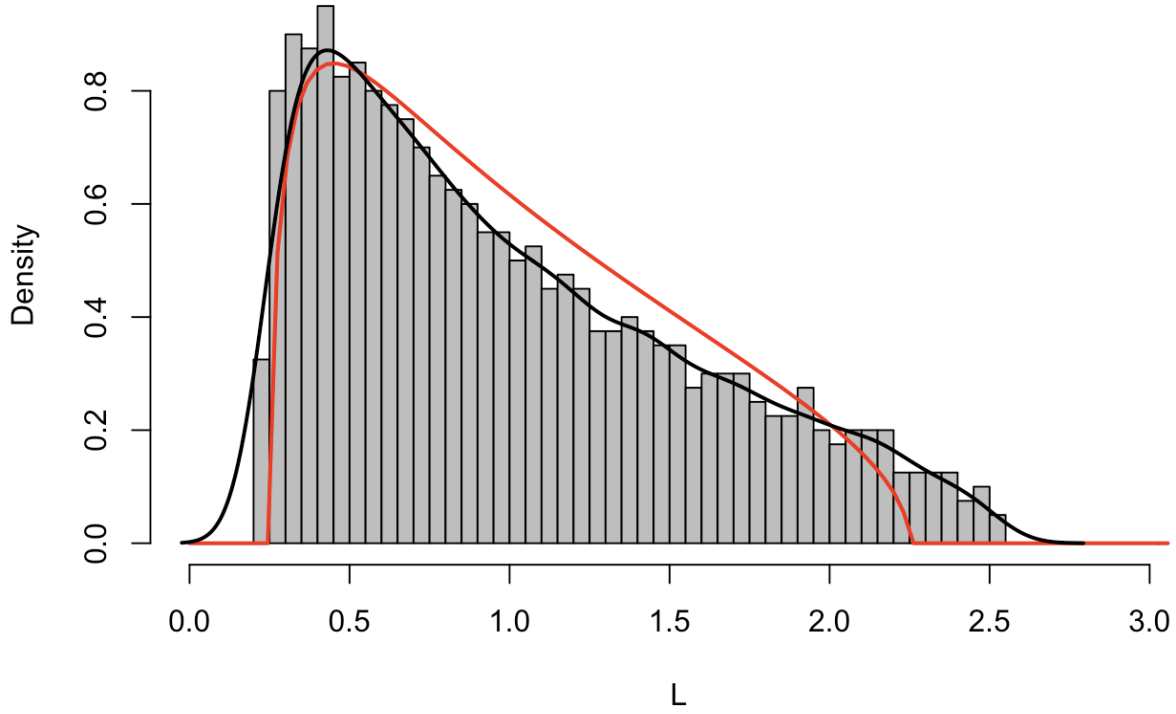
```
p <- 800
n <- 3200
X <- matrix(rnorm(p*n), p, n)
Sn <- X %*% t(X) / n
L<-eigen(Sn)$values
d <- density(L, kernel="gaussian", bw="SJ")

hist(L, breaks=50, xlim=c(0.4,2.1), freq=FALSE, col=8, main='')
curve(dmp(x, y=p/n), from = 0.2, to = 2.1, lty=1, lw=2, col=2, add=TRUE)
lines(d, col=1, lwd=2)
```



Effect of a correlation matrix

Let's look at the case of our power correlation matrix with $\rho = 0.2$ first and compare it to the MP case (i.e., $\rho = 0$).

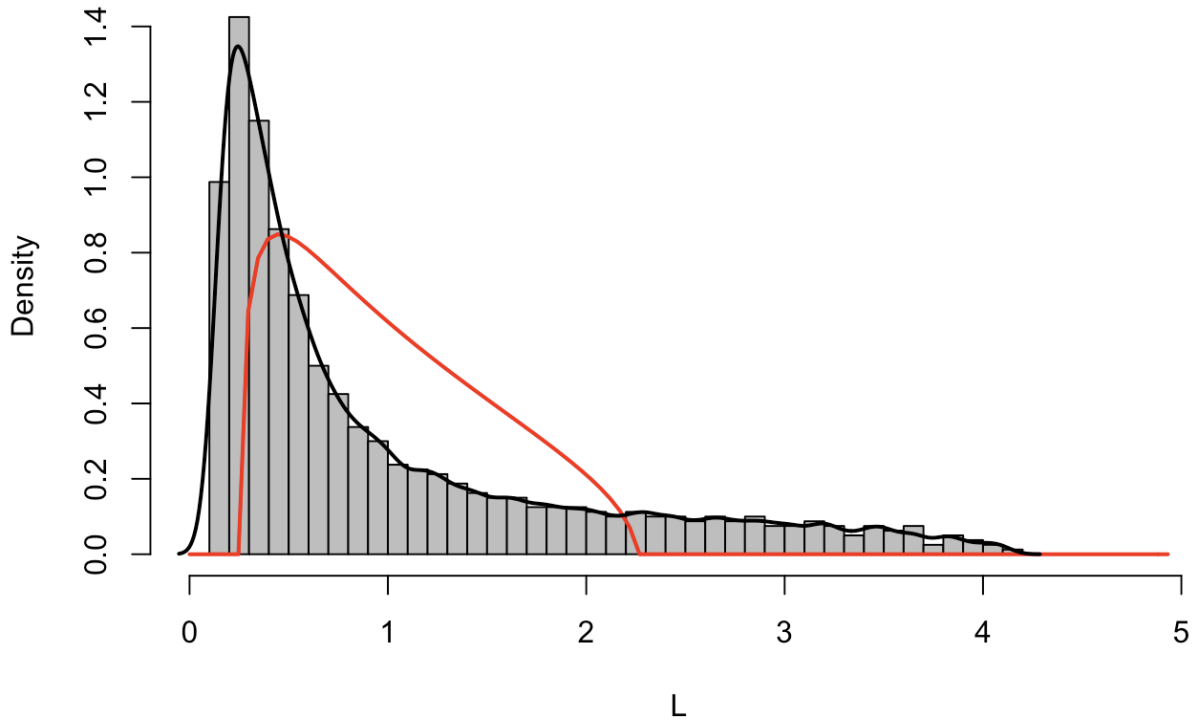
```
p <- 800
n <- 3200
rho <- 0.2 # correlation

Tn <- pcor(rho, p) # population covar
X <- matrix(rnorm(p*n), p, n)

# generate sqroot of Tn
eigs <- eigen(Tn)
Q <- eigs$vectors %*% diag(sqrt(eigs$values)) %*% solve(eigs$vectors)

# generate sample covariance
An <- Q %*% X %*% t(X) %*% Q / n

# KDE of eigenvalues
L<-eigen(An)$values
d <- density(L, kernel="gaussian", bw="SJ")

# plot histogram
hist(L, breaks=50, xlim=c(0,1.2*max(L)), freq=FALSE, col=8, main='')
curve(dmp(x, y=p/n), from = 0, to = 1.2*max(L), lty=1, lw=2, col=2, add=TRUE)
lines(d, col=1, lwd=2)
```



Notice how the shape changes!

Let's take a higher correlation and look at how the shape changes again.

```
p <- 800
n <- 3200
rho <- 0.5 # correlation

Tn <- pcor(rho, p) # population covar
X <- matrix(rnorm(p*n), p, n)

# generate sqroot of Tn
eigs <- eigen(Tn)
Q <- eigs$vectors %*% diag(sqrt(eigs$values)) %*% solve(eigs$vectors)

# generate sample covariance
An <- Q %*% X %*% t(X) %*% Q / n

# KDE of eigenvalues
L<-eigen(An)$values
d <- density(L, kernel="gaussian", bw="SJ")

# plot histogram
hist(L, breaks=50, xlim=c(0,1.2*max(L)), freq=FALSE, col=8, main='')
curve(dmp(x, y=p/n), from = 0, to = 1.2*max(L), lty=1, lw=2, col=2, add=TRUE)
lines(d, col=1, lwd=2)
```



Stochastic $\mathbf{T}_n$

This approach can handle a stochastic $\mathbf{T}_n$. Have a go at reproducing the cases done in the paper, see p. 3732 and formula (4.2).