Tutorial - Week 3

Please read the related material and attempt these questions before attending your allocated tutorial. Solutions are released on Friday 4pm.

Question 1

"*Every minute of every data, everywhere on the planet, dozens of companies — largely unregulated, little scrutinized — are logging the movements of tens of millions of people with mobile phones and storing the information in gigantic data files*", see **[A]**.

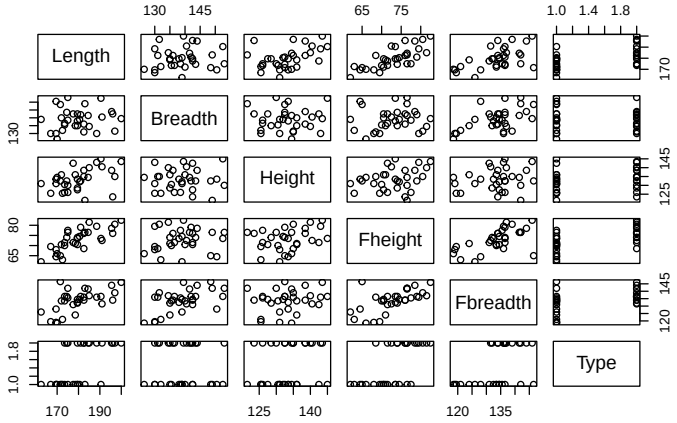
Start by reading the paper **[B]** about errors in GPS movement data. This paper is located in the 'Readings' folder on Wattle under the name *2016-RanacherBrunauerEtAl.pdf*.

(a) Consider the movement data in the file 'path.txt' that contains 100 positions over time and store it in the variable x of dimensions 100 x 2. Plot the path, and calculate the distance between the start of the path **P** and the end of the path **Q**.

Solution: Instead of just loading the data from `data.txt`, I am first going to show you how the data was generated. Synthetic movement data was created using the `trajectories` package.

```
#install.packages('trajectories')
```

```
library(trajectories)
path = rTrack()
plot(path,lwd=2,main="A simulated path")
```



We can extract the data points of the path and store them in x.

```
df = as.data.frame(path)
X = cbind(df$x, df$y)
```

```
dim(X)
```

```
## [1] 100   2
```

The data was then written to the file `path.txt`.

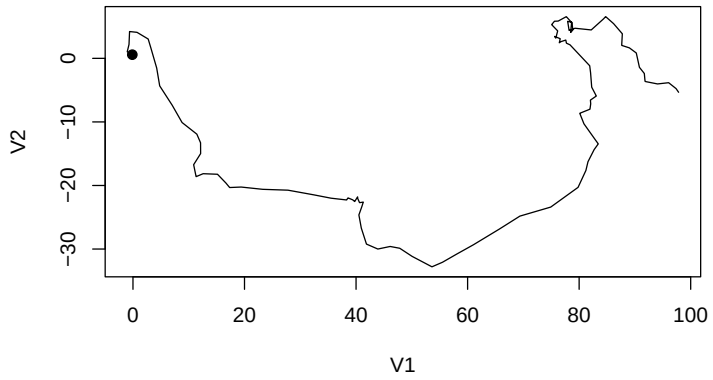
```
write.table(X, 'path.txt')
```

If you have the file `path.txt`, you load it using the following command.

```
X = read.table('path.txt')
```

We can plot these paths ourselves and add the starting point as a black disk.

```
plot(X, type='l')
points(X[1,1],X[1,2], pch=19)
```



(b) Perform a simulation study whereby you first assume your movement data x contains no measurement error, then add measurement noise ε (to each measurement) drawn from a bivariate normal with covariance $\Sigma = \sigma I$ where I is a 2×2 identity matrix and $\sigma > 0$. Vary σ and plot the distance $d(P, Q)$ as a function of σ. What can you conclude?

Solution: We are going to use the `MASS` package to generate bivariate normal noise.

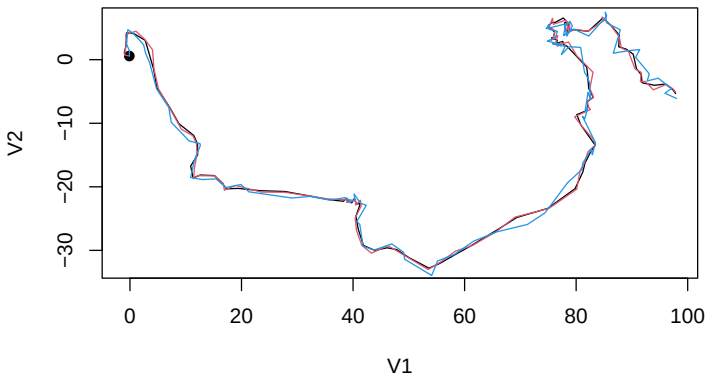
```
library(MASS)
```

We create a function that takes a path, computes its length n, and the samples from a bivariate normal to add noise to $n-1$ of the points (omit the starting point). The function takes σ as a parameter.

```
noisy = function(X, sigma=0.01, p=2) {
  n = dim(X)[1]
  eps = mvrnorm(n-1, mu=rep(0,p), Sigma = sigma * diag(p))
  return(X + rbind(rep(0,p), eps))
}
```

We can see the effect of noise on the path.

```
plot(X, type='l')
points(X[1,1],X[1,2], pch=19)
lines(noisy(X, 0.1), col=2)
lines(noisy(X, 0.5), col=4)
```



We consider the differences of the steps of the path, for example, the first step is:

```
X[2,]-X[1,]
```

```
##          V1        V2
## 2 -2.567129 -0.545436
```

We can successively do this for all points on the paths using `diff`. Notice that the first value matches `X[2,] - X[1,]`. We print the first 3:

```
diffs = apply(X, 2, diff)
diffs[1:3,]
```

```
##          V1        V2
## 2 -2.567129 -0.545436
## 3 -1.871983 -2.545923
## 4 -1.973826 -1.518387
```

The aim is to then to calculate the Euclidean norm of each increment (i.e., the differences) and then sum them all up to get the length of the path. We create a function to do this.

```
pathlen = function(X) {
  diffs = apply(X, 2, diff)
  sum(apply(diffs, 1, function(x) norm(as.matrix(x), type="2")))
}
```

We test it out on our data.

```
pathlen(X)
```

```
## [1] 214.0438
```

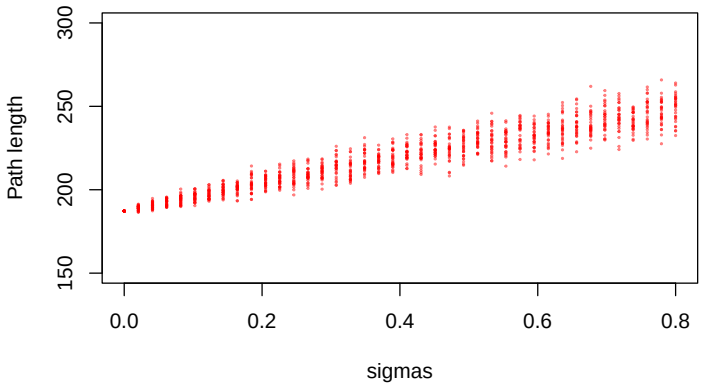
Now we test it on a noisy path.

```
pathlen(noisy(X, 0.1))
```

```
## [1] 219.8352
```

We create a plot where we consider σ varying from 0 to 0.8. For each σ, we generate 30 noisy paths and then plot their path length as a point. This allows us to see the distribution of path lengths at each σ value.

```
k = 40 # steps of sigma
sigmas = seq(0,0.8,length.out=k)
plot(sigmas, 0*sigmas, ylim=c(150,300), type="n", ylab="Path length") # empty plot

for (j in 1:30) {
  plen = numeric(k)
  for (i in 1:k) {
    plen[i] = pathlen(noisy(X, sigmas[i]))
  }
  points(sigmas, plen, cex=0.2, pch=19, col=rgb(red=1, green=0, blue=0, alpha=0.5))
}
```



We notice in the above plot that the mean length of the path $d(P, Q)$ increases as σ increases. This is somewhat unexpected as noise we are adding to the paths at every point has zero mean. Further, as σ increases the variance of the path length also increases (which is expected).

(c) Consider the p-dimensional case of Theorem 3.1, that is, assume $\mathbf{P} = 0 \in \mathbb{R}^p$ and $\mathbf{Q} = (d_0, 0, \ldots, 0) \in \mathbb{R}^p$. Reprove the result. What can you conclude?

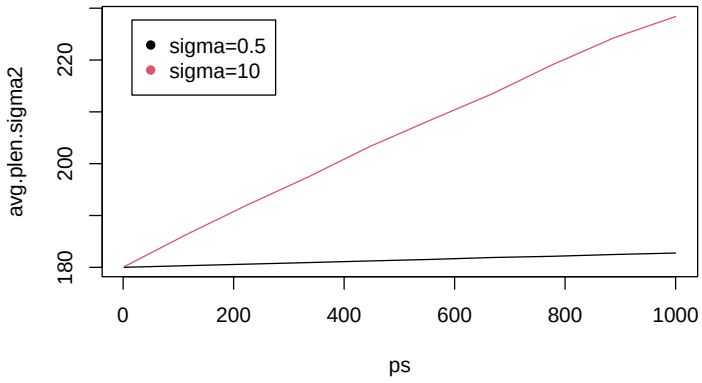
Solution: See handwritten solutions.

(d) Consider distances in a p-dimensional space, take the starting point to be $P = (0, 0, 0, \ldots, 0)$ and end point to be $Q = (d_0, 0, \ldots, 0)$. Add noise to this path and consider how the mean length changes as p increases. Consider the simplified path

$P^m + \varepsilon$ to $Q^m = Q + \eta$. Do this study for various choices of $p = 2, 10, 50, 100$. What can you conclude?

Solution: We consider the distances in a p-dimensional space, we take the starting point to be $P = (0, 0, 0, \ldots, 0)$ and end point to be $Q = (d_0, 0, \ldots, 0)$. We add noise to this path and consider how the mean length changes as p increases. We consider the simplified path $P^m + \varepsilon$ to $Q^m = Q + \eta$.

```r
library(MASS)

d0 = 180

k = 10 # steps of p
ps = seq(2,1000,length.out=k)

sigma = 0.5
avg.plen = numeric(k)
for (i in 1:k) {
  p = ps[i]
  plen = numeric(30)
  for (j in 1:30) {
    Q = matrix(c(d0, rep(0, p-1)), nrow=p)
    eps = mvrnorm(1, mu=rep(0,p), Sigma = sigma * diag(p))
    eta = mvrnorm(1, mu=rep(0,p), Sigma = sigma * diag(p))
    plen[j] = sqrt(d0^2 + sum((eta-eps)^2))
  }
  avg.plen[i] = mean(plen)
}
avg.plen.sigma1 = avg.plen

sigma = 10
avg.plen = numeric(k)
for (i in 1:k) {
  p = ps[i]
  plen = numeric(30)
  for (j in 1:30) {
    Q = matrix(c(d0, rep(0, p-1)), nrow=p)
    eps = mvrnorm(1, mu=rep(0,p), Sigma = sigma * diag(p))
    eta = mvrnorm(1, mu=rep(0,p), Sigma = sigma * diag(p))
    plen[j] = sqrt(d0^2 + sum((eta-eps)^2))
  }
  avg.plen[i] = mean(plen)
}
avg.plen.sigma2 = avg.plen

plot( ps, avg.plen.sigma2, col=2, type='l')
lines(ps, avg.plen.sigma1, col=1, type='l')
legend("topleft", c("sigma=0.5", "sigma=10"), col=c(1,2), inset = .05)
```



Assuming independence of ε and η, we see an increase in the mean path length as the dimensionality p increases depending the noise level σ.

See **[C]**, for further real-world data.

Question 2

Illustrate numerically that the spectral density of large symmetric matrices formed from independent identically distributed random variables with zero mean and finite variance converges to the density of the Wigner Semicircle distribution. That is:

(a) Take $p = 100$ and write a function that generates a $p \times p$ symmetric matrix with entries sampled from the standard Normal distribution. Hint: generate a $p \times p$ matrix `A` with Normal entries and then symmetrise using `A[lower.tri(A)] <- t(A)[lower.tri(A)]`.

Solution: We set our parameters

```
p = 100
```

We check the generation of a single matrix.

```
A = matrix(rnorm(p*p), p, p)
A[lower.tri(A)] <- t(A)[lower.tri(A)]
```

We also check that it is symmetric, that is, $A[i,j] = A[j,i]$ for any i, j.

```
A[1,p]
```

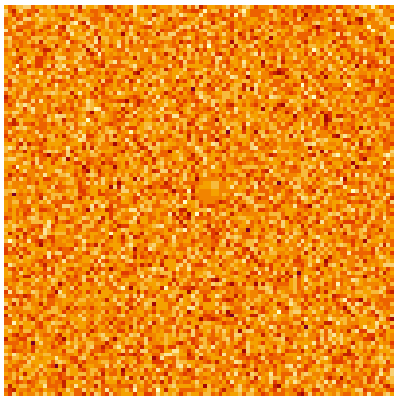
```
## [1] -0.8056531
```

```
A[p,1]
```

```
## [1] -0.8056531
```

We can also plot the matrix.

```
heatmap(A, Rowv=NA, Colv=NA, labRow=NA, labCol=NA)
```



We write a function to generate the random symmetric matrix with Normal entries.

```
rsymmat = function(p) {
  A = matrix(rnorm(p*p), p, p)
  A[lower.tri(A)] <- t(A)[lower.tri(A)]
  A
}
```

(b) Write a simulation that generates n of these matrices, calculates the eigenvalues of each of these matrices and plots the histogram of all these eigenvalues together (i.e., obtained from all the matrices).

Solution:

```
n = 500
Lambdas = c() # empty vector

for (sim in 1:n) {
  A = rsymmat(p)
  # calculate eigenvalues, use symmetric=TRUE, add to vector of Lambdas
  Lambdas = c(Lambdas, eigen(A, TRUE, only.values=TRUE)$values)
}
```

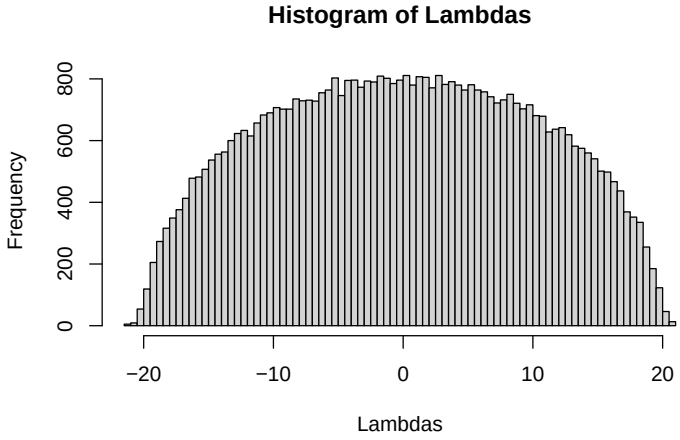
The vector `Lambdas` has the expected number of values ($n * p$).

```
length(Lambdas)
```

```
## [1] 50000
```

Plot the histogram of eigenvalues (with 100 breakpoints). We see a semi-circle shape.

```
hist(Lambdas, 100)
```

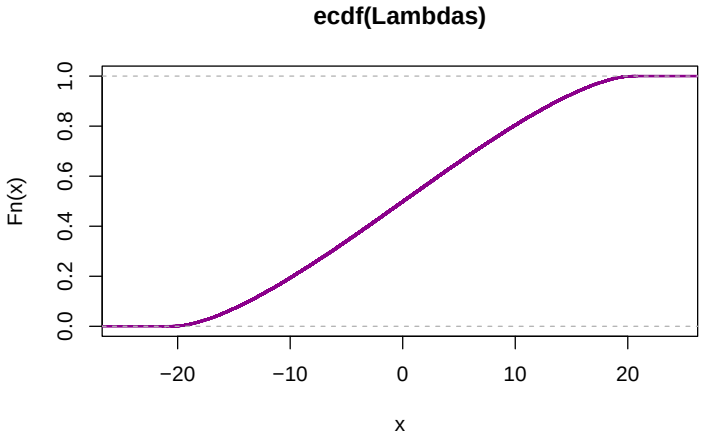


(c) Plot a matching Wigner semicircle distribution over this histogram (you'll need to guess the appropriate parameters of the distribution).

Solution: Probably the best way to match distributions is to consider the empirical CDF and compare it to our target theoretical CDF. The empirical CDF is easy to find in R using the `ecdf` function.

```
Fn = ecdf(Lambdas)
```

```
plot(Fn, col="darkmagenta", lwd=2)
```

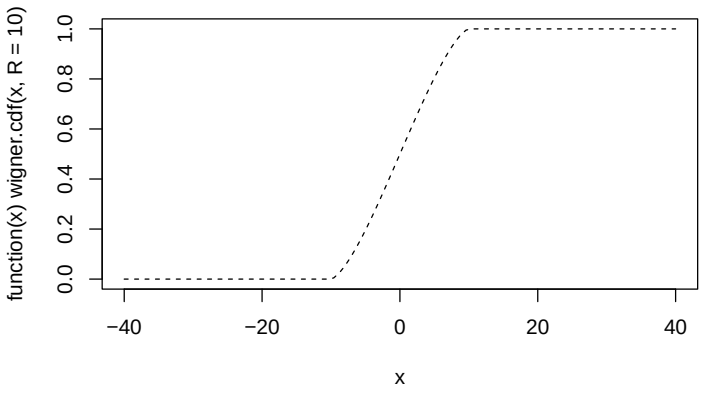


We write a function to calculate the Wigner semicircle distribution, given parameter R. See the wikipedia page: https://en.wikipedia.org/wiki/Wigner_semicircle_distribution

```
wigner.cdf = function(x, R=1) {
  result = suppressWarnings(0.5 + x*sqrt(R^2-x^2)/(pi * R^2) + asin(x/R)/pi)
  result[x<=-R] = 0
  result[x>=R] = 1
  result
}
```

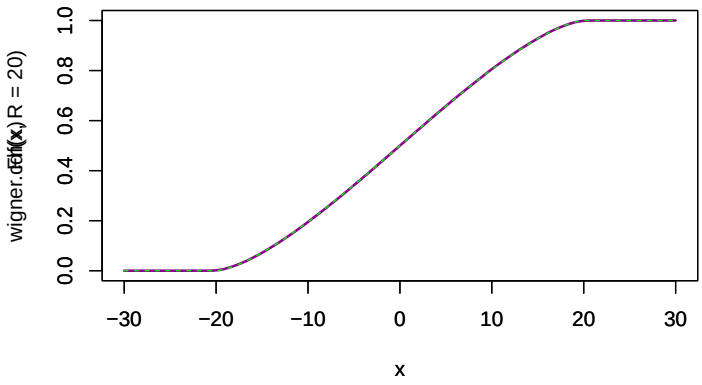
We can plot the CDF with $R = 10$.

```
plot(function(x) wigner.cdf(x, R=10), from=-40, to=40, lty=2)
```



Looking at the definition of the Wigner distribution and the empirical CDF, we can make a very good guess on the appropriate choice of the parameter R. The key is noticing that the Wigner distribution is zero for $|x| > R$. Looking at the empirical CDF, we see it vanishes for $x < -20$. We check our conjecture that $R = 20$.

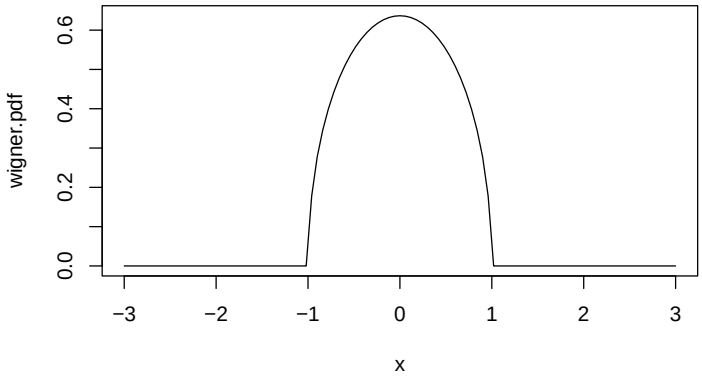
```
x = seq(-30, 30, length.out=100)
plot(x, Fn(x), col="darkmagenta", type='l', lwd=2)
par(new=TRUE) # trick to add on top
plot(x, wigner.cdf(x, R=20), col='green', type='l', lty=2)
```



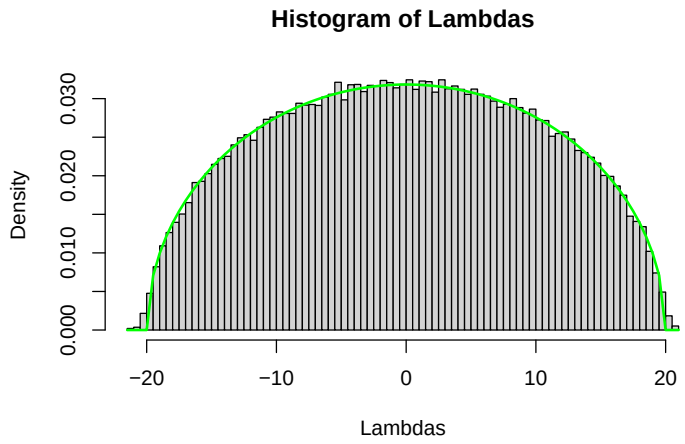
Alternatively, we could have spotted the $R = 20$ conjecture with the density as well.

```
wigner.pdf = function(x, R=1) {
  result = suppressWarnings(2/(pi * R^2) * sqrt(R^2 - x^2))
  result[x<=-R] = 0
  result[x>=R] = 0
  result
}
```

```
plot(wigner.pdf, from=-3, to=3)
```



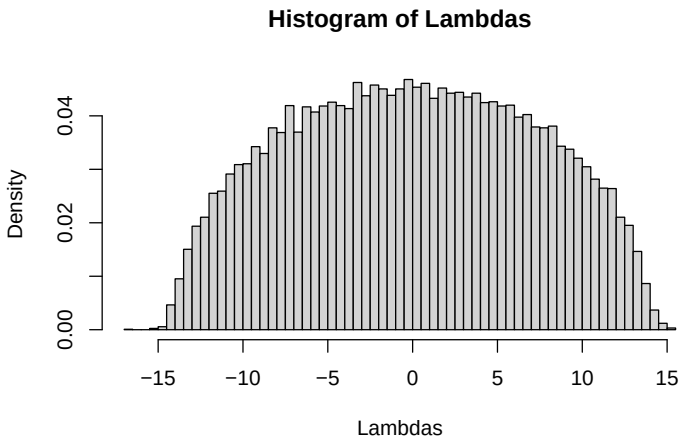
```
hh = hist(Lambdas, 100, probability = TRUE)
x = hh$breaks
lines(x, wigner.pdf(x, R=20), col="green", lwd=2)
```



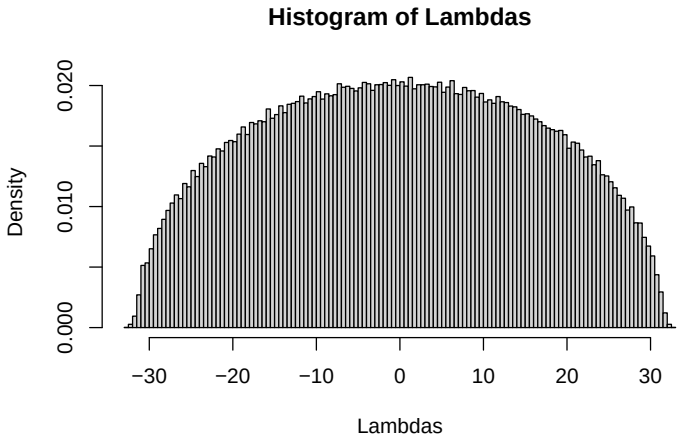
(d) Repeat the experiment three times with p and n larger and larger with the ratio p/n fixed. What do you observe?

Solution: We set $y = p/n$ and then vary y.

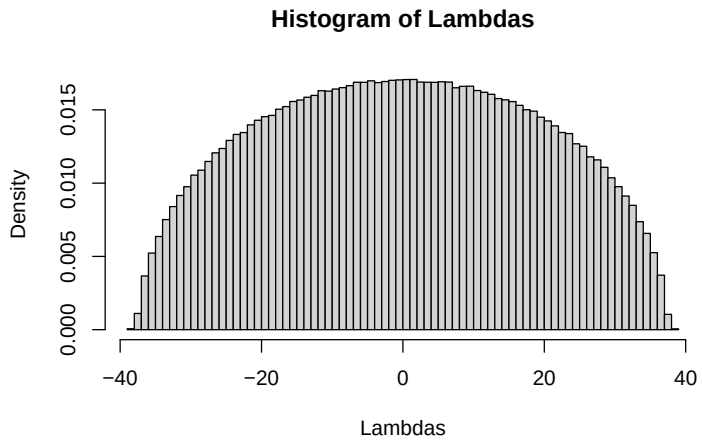
```
y = 0.1
n = 500
p = y * n

Lambdas = c()
for (sim in 1:n) {
  A = rsymmat(p)
  Lambdas = c(Lambdas, eigen(A, TRUE, only.values=TRUE)$values)
}
hist(Lambdas, 100, probability = TRUE)
```




```
y = 0.5
n = 500
p = y * n

Lambdas = c()
for (sim in 1:n) {
  A = rsymmat(p)
  Lambdas = c(Lambdas, eigen(A, TRUE, only.values=TRUE)$values)
}
hist(Lambdas, 100, probability = TRUE)
```



```
y = 0.7
n = 500
p = y * n

Lambdas = c()
for (sim in 1:n) {
  A = rsymmat(p)
  Lambdas = c(Lambdas, eigen(A, TRUE, only.values=TRUE)$values)
}
hist(Lambdas, 100, probability = TRUE)
```



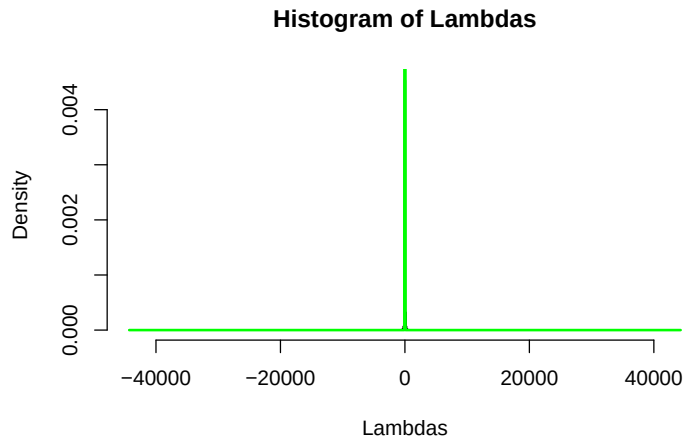
(e) Now, repeat the experiment with the entries sampled from a Student-T distribution with parameter $1 < \nu \leq 2$. What do you observe and what can you conclude?

Solution: We create a new function that generates a symmetric matrix with Student-T entries.

```
rsymmat.student = function(p, nu=1.5) {
  A = matrix(rt(p*p, df=nu, ncp=0), p, p)
  A[lower.tri(A)] <- t(A)[lower.tri(A)]
  A
}
```

We redo the simulation with the appropriate change to the matrix sampling part. We notice that we have extreme outliers for eigenvalues and the Wigner PDF no longer fits.

```
p = 100
n = 500

Lambdas = c()
for (sim in 1:n) {
  A = rsymmat.student(p) # CHANGED
  Lambdas = c(Lambdas, eigen(A, TRUE, only.values=TRUE)$values)
}
hh=hist(Lambdas, breaks=1000, probability = TRUE)
x = hh$breaks
lines(x, wigner.pdf(x, R=20), col="green", lwd=2)
```



References

[A] https://www.nytimes.com/interactive/2019/12/19/opinion/location-tracking-cell-phone.html

[B] Ranachera, Brunauer, Trutschnig, Van der Spek, and Reich (2016). *Why GPS makes distances bigger than they are*. International Journal of Geographical Information Science. Vol 30, No 2, 316 – 333.

[C] https://archive.ics.uci.edu/ml/datasets/GPS+Trajectories