RMThreshold

A short introduction

Uwe Menzel, 2016

uwe.menzel@matstat.de

www.matstat.org

RMThreshold

The package RMThreshold attempts to determine an objective threshold which separates signal from noise in large real-valued, symmetric matrices. Such matrices can for instance describe correlation or mutual information between data of various origin, or might represent the set of edges in undirected networks.

RMThreshold takes advantage of the predictions of Random Matrix Theory(RMT) for the distribution of the spacing between the eigenvalues of such matrices. That distribution is usually called Nearest Neighbor Spacing Distribution (NNSD). The predictions of RMT are valid in the limit of large matrix dimensions.

RMT was initiated by Eugene Wigner in the context of nuclear physics in 1955 (Wigner E. P., Annals of Mathematics, 1955).

RMThreshold

RMT predicts two extreme scenarios for the NNSD of eigenvalues:

1.) If the matrix elements are completely random, the NNSD is characterized by Gaussian Orthogonal Ensemble (GOE) statistics, and the shape of the NNSD resembles the Wigner-Dyson distribution ("Wigner surmise"):

$$P_{GOE}(s) = \frac{\pi}{2} \cdot s \cdot \exp\left(-\frac{\pi}{4} s^2\right)$$

where s is the eigenvalue spacing and $P(s)$ it's distribution. This distribution approaches zero for $s = 0$ which can be imagined as if there was some sort of "repulsion" between the eigenvalues.

RMThreshold

2.) If the matrix has a non-random, modular structure (associated with block-like composition), the NNSD comes close to an Exponential distribution:

$$P_{Exp}(s) = \exp(-s)$$

Both functions differ most at $s = 0$, where $P_{GOE} = 0$ and $P_{Exp} = 1$. An imaginary "repulsion" does not occur in the modular case, and zero-spacings between the eigenvalues frequently occur.
This case might apply to the adjacency matrix of a large undirected network consisting of relatively independent clusters with weak connections between them. The connections might possibly just being noise by their nature.
By identifying an appropriate threshold for such matrices, it should be possible to reveal the underlying modular structure of the network, i.e. to identify the genuine clusters.

RMThreshold

Now, if we assume that a matrix or a network actually **has** a modular structure which is hidden by noise, it should be possible to identify a signal-noise separating threshold by finding the threshold at which the NNSD changes from the Wigner-Dyson case to the Exponential case.
Consequently, the main function of the package (rm.get.threshold) increments a suppositional threshold monotonically, thereby recording the eigenvalue spacing distribution of the thresholded matrix.

A typical procedure to infer a signal-noise separating threshold by using the package RMThreshold may consist of the following steps:
1.) checking the conformity of the input matrix using the function rm.matrix.validation,
2.) running the main function rm.get.threshold in order to find a candidate threshold,
3.) optionally repeat running rm.get.threshold on a smaller interval of thresholds, and
4.) applying the identified threshold to the matrix. The thresholded matrix created by the latter step should then represent the real signal. Some important steps of this procedure are described in more detail on the following pages.

1. Checking if a matrix is well-conditioned for the RMT-based algorithm

RMT can be applied to large, real-valued, symmetric matrices. The function rm.matrix.validation can be used to assert if a matrix is suitable for the proposed algorithm:

```
library(RMThreshold)
set.seed(777)
random.mat  <-  create.rand.mat(size = 1000, distrib = "norm")$rand.matr
res <- rm.matrix.validation(random.matrix)
str(res)
```

The function rm.matrix.validation checks the input matrix for several criteria:

1. The matrix must be symmetric and real-valued. These are natural properties of correlation matrices, mutual information matrices, and adjacency matrices for undirected networks.
2. The matrix must be quite large because the results of RMT in the strict sense are only valid in the limit of infinite matrices.
3. The matrix should not have a rank which is much smaller than the matrix dimensions. If the rank of the input matrix is low, one cannot expect a proper eigenvalue spectrum because the rank of the matrix determines the number of non-zero eigenvalues.

Checking if a matrix is well-conditioned for the RMT-based algorithm

The function rm.matrix.validation creates several diagnostic plots. One plot is created on-the-fly, showing 4 diagnostic plots. Two plots are created and saved to disk, the names of which are returned by the function. The (hard-coded) file names are matrix.validation.plot.png and fit.unfold.png, respectively.

RMT examines the local fluctuations between eigenvalues, and ignores the global properties. Therefore, in order to obtain a local eigenvalue spacing distribution, the eigenvalues must be "unfolded", i.e. they must be scaled in such a way that the average (global) spacing is constant over the whole spectrum. The resulting NNSD then reflects the short-range "interaction" between the eigenvalues. In RMThreshold, two methods are provided for unfolding: one method is based on calculation of the Gaussian kernel density of the eigenvalue spectrum; another method is based on fitting a cubic spline function to the cumulative empirical eigenvalue distribution. The parameter unfold.method determines which method is used. In the above code example, Gaussian kernel density – the default - was used.

Checking if a matrix is well-conditioned for the RMT-based algorithm

The magnitudes of the matrix elements (**Figure 1**) suggest that the matrix is superimposed by noise. The example matrix 'random.matrix' used in the code snippet above contains matrix elements that are normally distributed with mean zero (Gaussian matrix). **Figure 2** shows the distribution of the eigenvalues of the matrix. If the matrix is well-conditioned, as in the case of a Gaussian matrix, the eigenvalue distribution resembles a characteristic shape called Wigner semi-circle. In **Figure 3**, a scatterplot of the eigenvalue spacing is presented, together with a linear fit (red dotted line). The linear fit should have a slope of 0 and an intersect of 1, indicating that the average eigenvalue spacing is constantly 1 over the whole range, thereby proving that the unfolding algorithm was completed correctly. **Figure 4** presents the distribution of the eigenvalues spacings (NNSD). For a matrix dominated by noise, as it is the case here, this distribution is close to the Wigner-Dyson distribution, a curve that approaches zero for small eigenvalue spacings. The theoretical shape of the Wigner-Dyson distribution is indicated by the blue solid line in the plot. It should be noted that the diagnostic plots not often look as perfect as those shown in the figures presented here. However, even if so, the algorithm might be able to successfully separate signal and noise.

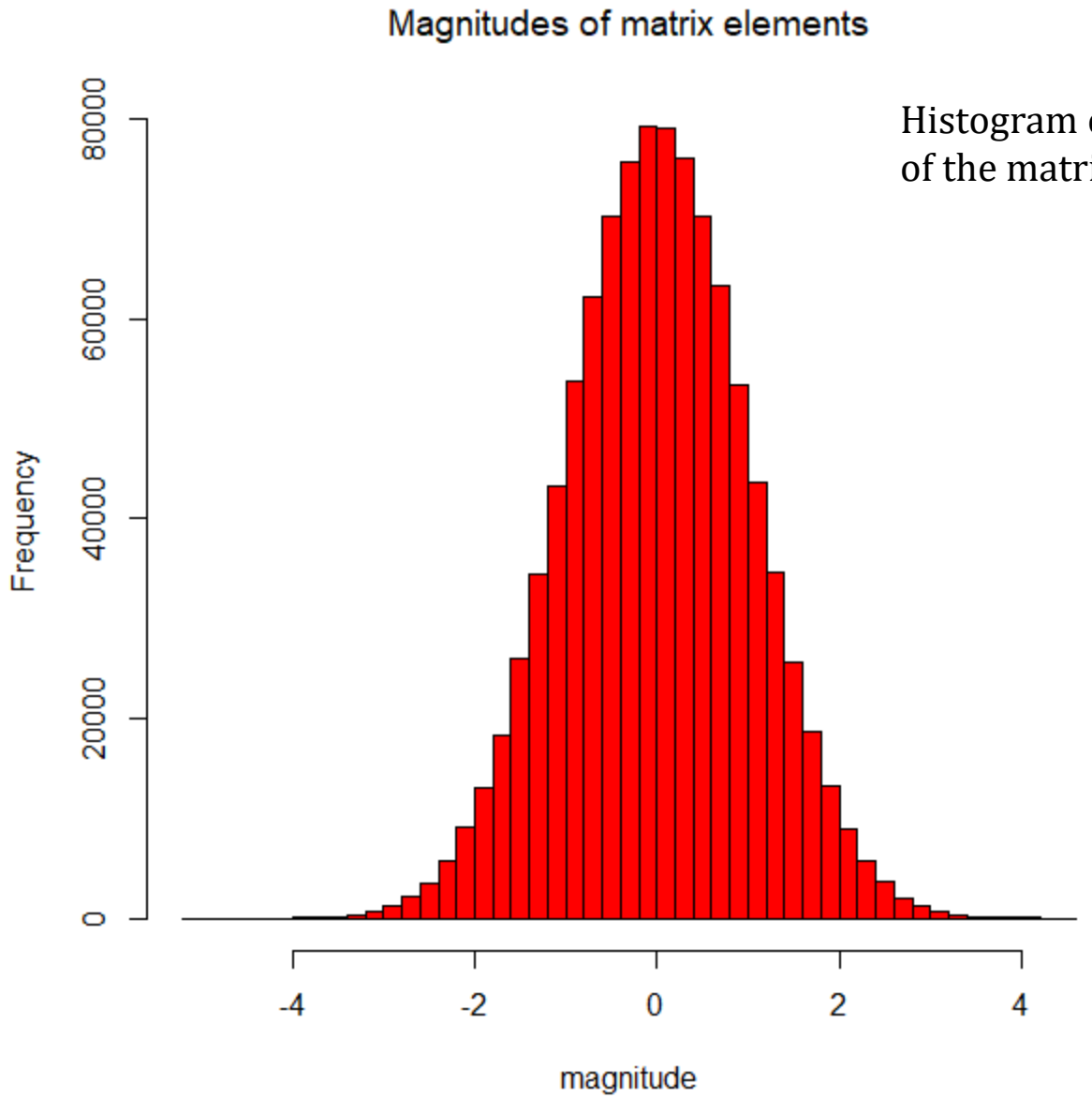


Figure 1

Histogram of the magnitudes of the matrix elements

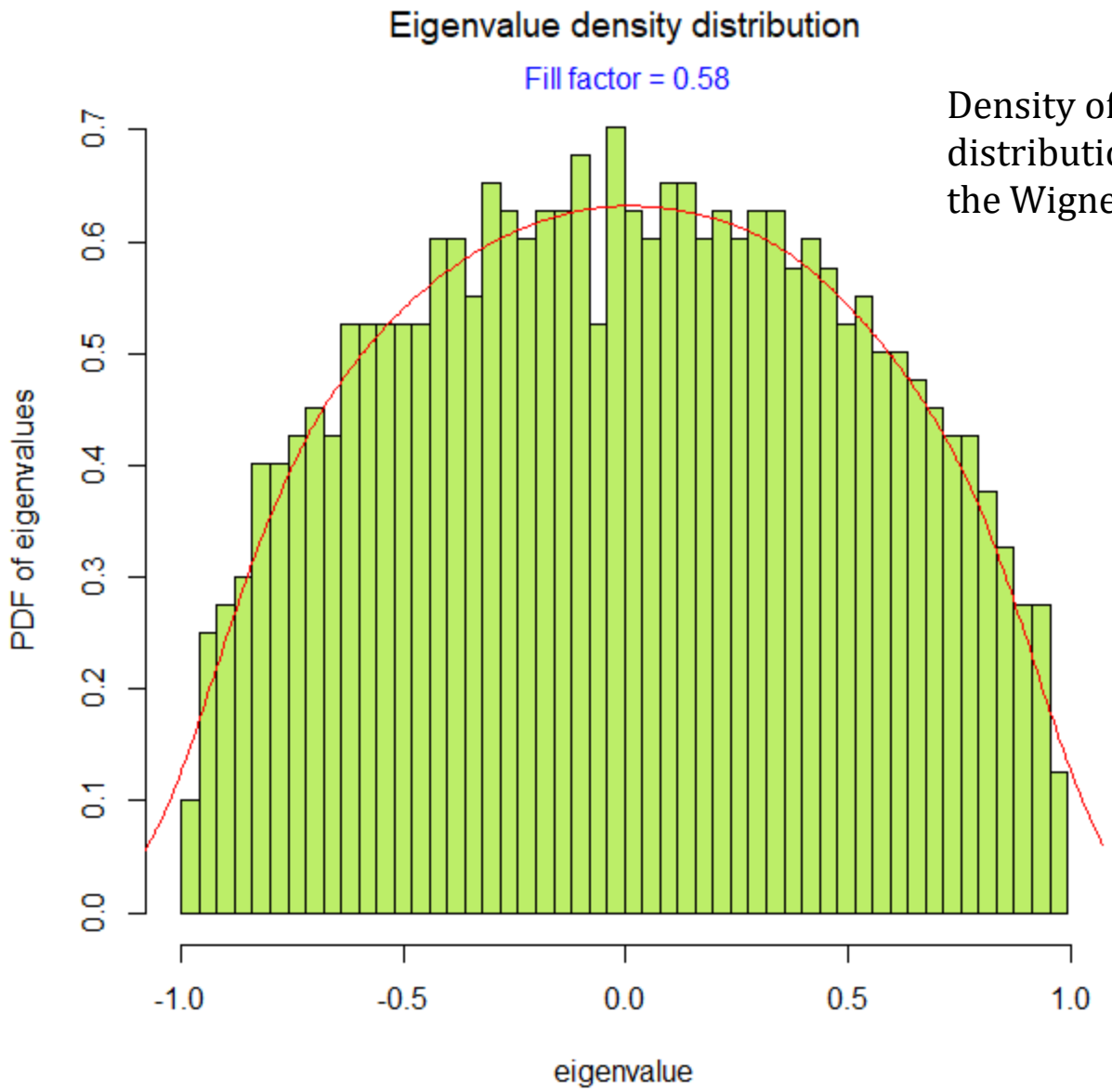


Figure 2

Density of the eigenvalue distribution, compared to the Wigner semi-circle

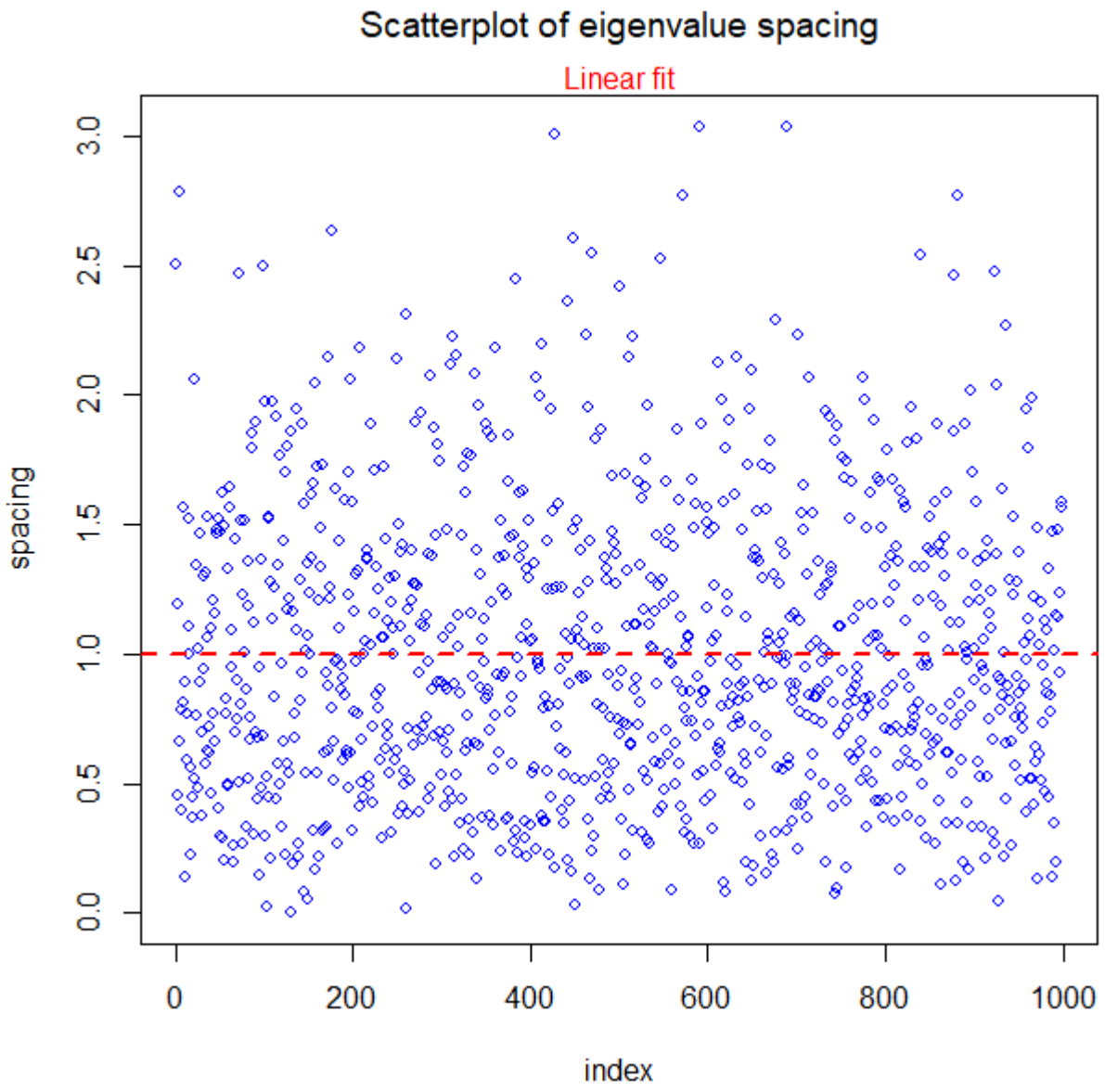
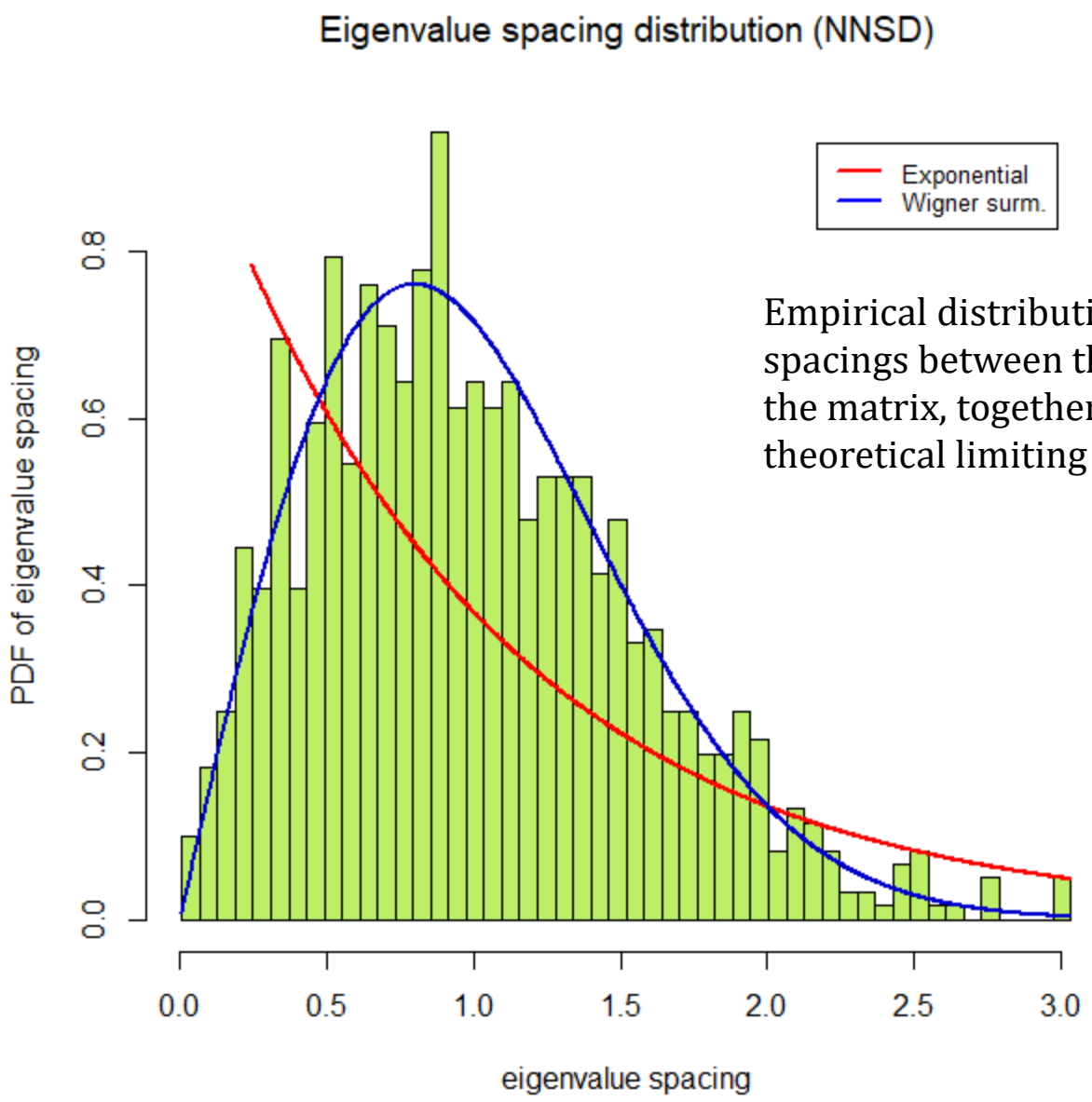


Figure 3

Scatterplot of the eigenvalue spacing including a linear fit

Figure 4



Empirical distribution of the spacings between the eigenvalues of the matrix, together with the two theoretical limiting distributions

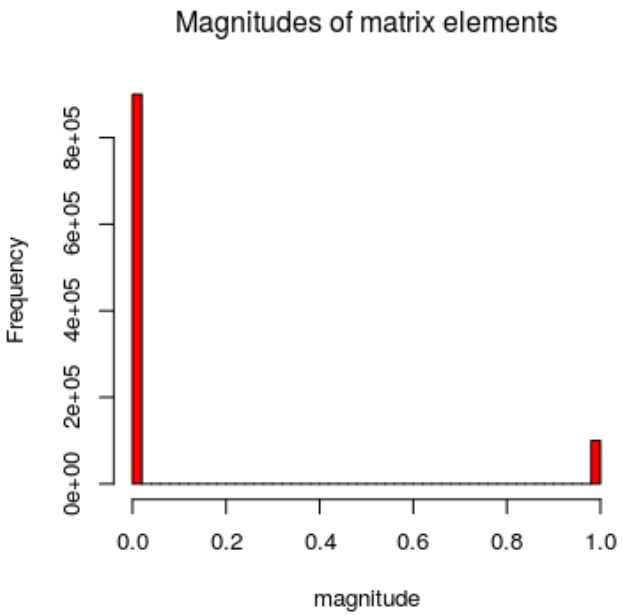
Uwe Menzel, 2016

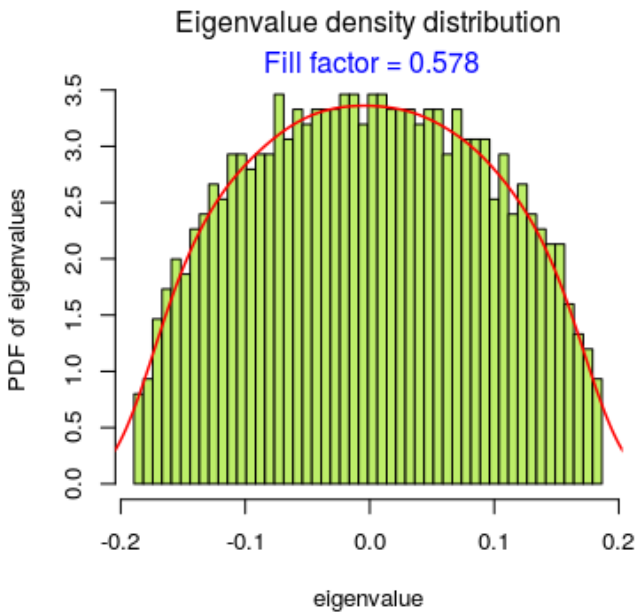
Another example

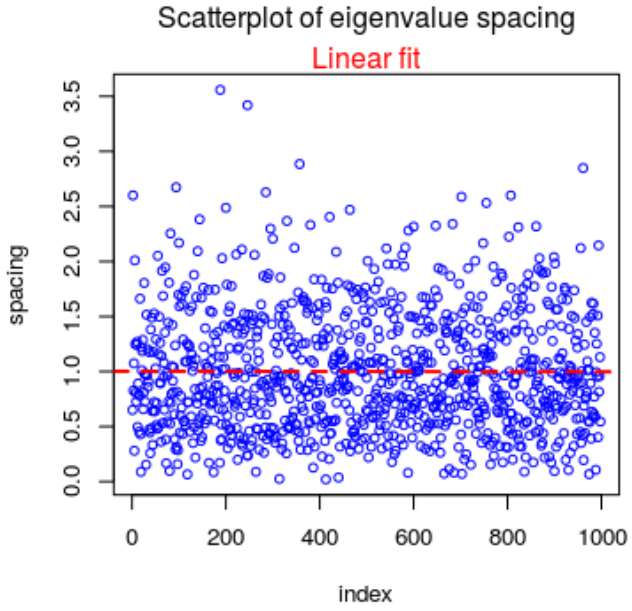
It is very instructive to have a look at another example that illustrates the two extreme scenarios for the NNSD mentioned above. By using the function rm.matrix.validation, it can be demonstrated how the NNSD can be used to distinguish between purely random and modularly composed adjacency matrices representing the edges in undirected networks.
To illustrate the random case, we create a single Erdös - Renyi graph using the igraph package in R. In such a graph, any two nodes are randomly connected with some given probability p. The edges (i.e. the elements of the adjacency matrix) are 1 if two nodes are connected, and 0 if they are not connected:

```
library(igraph); library(Matrix)
g <- erdos.renyi.game(1000, 0.1)
rm.matrix.validation(as.matrix(get.adjacency(g)))
```

The corresponding validation plot is shown in **Figure 5. T**he NNSD is shown in the bottom-right part of the plot. It appears that the randomly distributed non-zero off-diagonal elements, representing spurious interactions between the nodes of the graph, cause the NNSD to be Wigner-Dyson like, where zero-spacings do not occur (“repulsion” of eigenvalues).







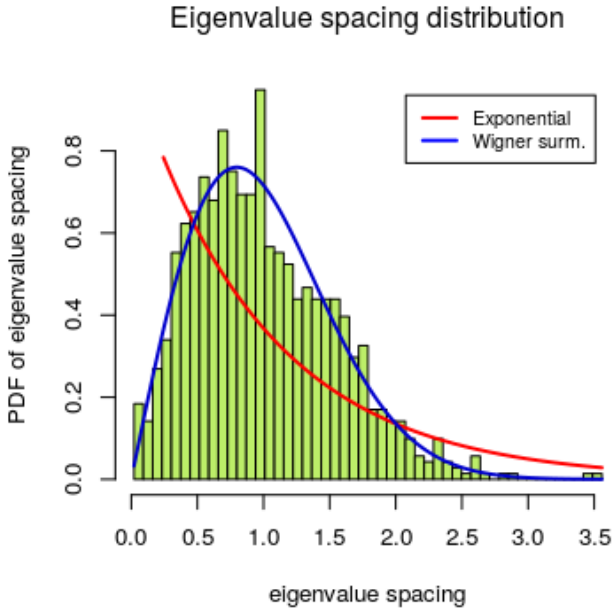


Figure 5

Validation plot for the adjacency matrix obtained for a single Erdös-Renyi graph. The NNSD (bottom right) resembles the Wigner-Dyson distribution, illustrating that the underlying structure of the matrix is completely random. In particular, eigenvalue spacings close to zero rarely occur ("repulsion" of eigenvalues).

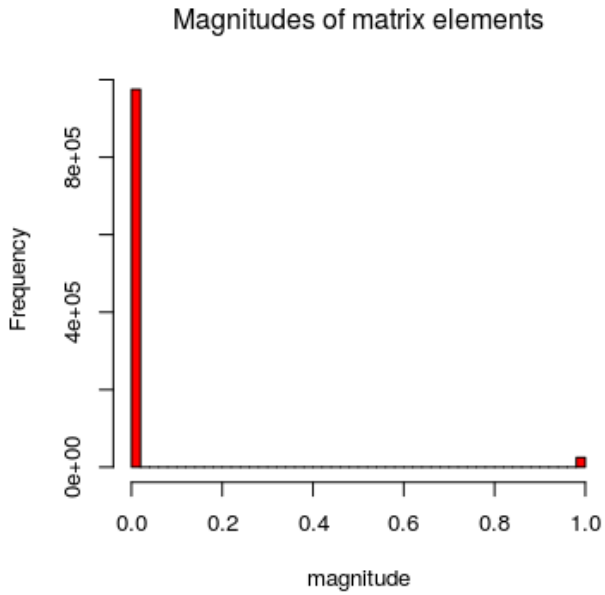
Another example

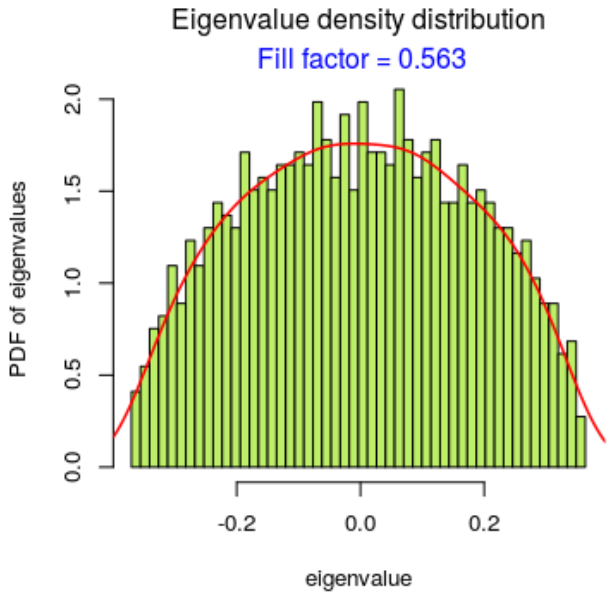
Now, a modular structure is introduced into a network by first creating four smaller Erdös - Renyi graphs, which are thereafter assembled into a block-diagonal adjacency matrix (by using bdiag from the "Matrix" package). Such a matrix represents a modular network, consisting of four clusters:

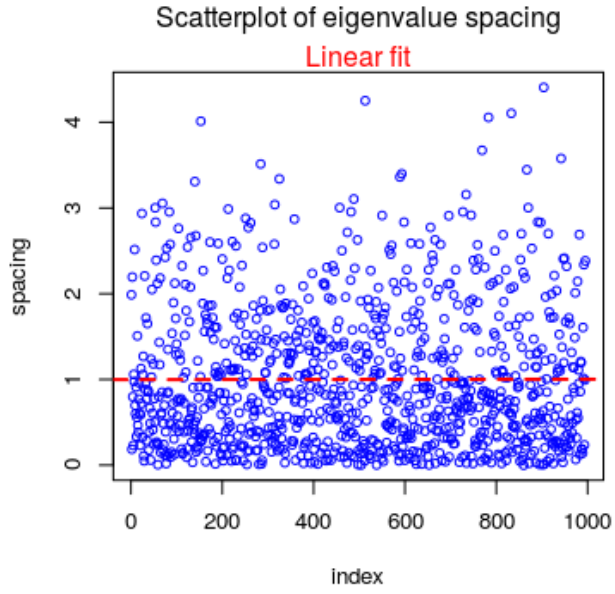
```
matlist = list()
for (i in 1:4) matlist[[i]] = get.adjacency(erdos.renyi.game(250, 0.1))
mat <- bdiag(matlist)
rm.matrix.validation(as.matrix(mat))
```

The validation plot for the second scenario is shown in **Figure 6.** Notably, the NNSD (bottom-right) now resembles the Exponential distribution.

As mentioned above, we can find a proper signal-noise separating threshold by exploiting the fact that the NNSD differs between the two scenarios. In order to do so, the function rm.get.threshold can be used.







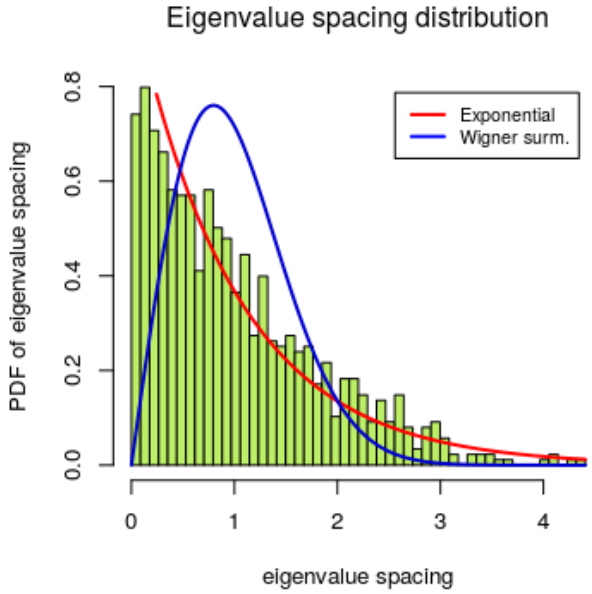


Figure 6

Validation plot for an adjacency matrix composed of four smaller Erdös-Renyi graphs. The matrix exhibits a block-diagonal structure, and therefore represents a modular network. Here, the NNSD (bottom right) resembles an Exponential distribution. Zero-eigenvalue spacings frequently occur for this scenario.

2. Finding a candidate signal-noise separating threshold for the matrix

The function rm.get.threshold is the main function of the package. In the simplest case, it can be used as follows:

```
set.seed(777)
random.mat <- create.rand.mat(size = 1000, distrib = "norm")$rand.matr
res <- rm.get.threshold(random.matrix)
str(res)
```

The function rm.get.threshold takes a random matrix as input and applies a sequence of increasing suppositional thresholds on it by setting all matrix elements to zero whose absolute magnitude is below the actual threshold. The eigenvalue spectrum is then calculated and unfolded by one of the two methods mentioned above, again controlled by the parameter unfold.method. From the unfolded eigenvalues, the NNSD is derived. According to RMT, that distribution undergoes a characteristic change when the threshold properly separates signal from noise. For low thresholds, if the assumed modular structure is still covered by noise, the NNSD is close to the Wigner-Dyson distribution. This means that the applied threshold was too low, leaving spurious matrix elements which should also be set to zero.

Finding a candidate signal-noise separating threshold for the matrix

When the threshold has reached sufficiently high levels, the assumed modular (block-like) structure of the matrix starts to prevail. The NNSD then switches to an Exponential distribution. During program execution, this change is monitored in a plot window showing the NNSD and the two limiting distributions (read and blue lines, respectively). **Figure 7a** shows the NNSD for a low threshold, when the thresholded matrix is still ruled by noise. The modular case is depicted in **Figure 7b**, where the histogram shape comes close to the Exponential distribution.

By thresholding the matrix, rows and columns exclusively containing zeros in the off-diagonal matrix elements likely emerge. The parameter min.mat.dim determines the minimum allowed number of non-zero rows and columns of the probed matrix during the loop over increasing thresholds. The loop is terminated if this number is getting below the chosen value for min.mat.dim. The default for that parameter is 40.

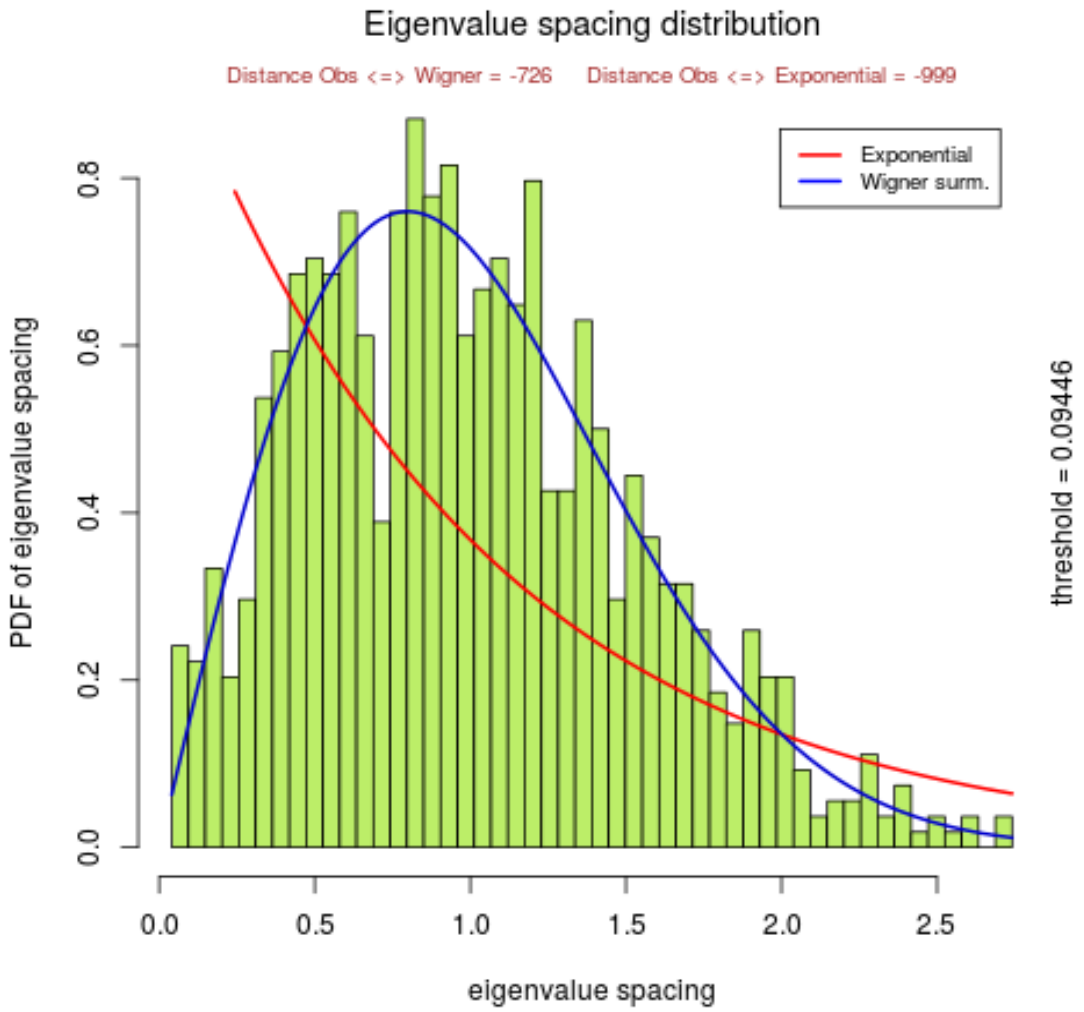


Figure 7a

Eigenvalue spacing distribution for a low threshold of 0.095. The matrix is ruled by noise, and the NNSD resembles the Wigner-Dyson distribution.

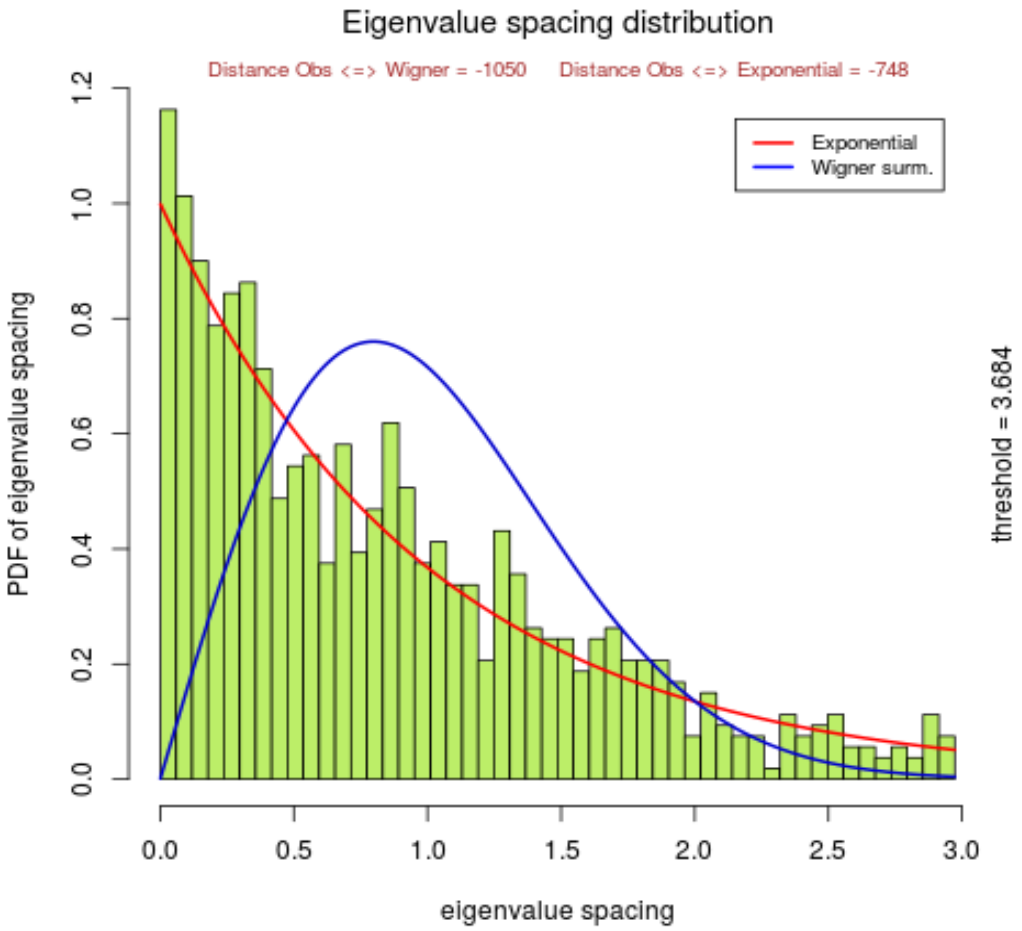


Figure 7b

Eigenvalue spacing distribution for a high threshold of 3.684. Now, the matrix has a block structure because spurious matrix elements induced by noise have been removed.

Finding a candidate signal-noise separating threshold for the matrix

For each threshold, a distance between the empirical eigenvalue spacing distribution and both limiting distributions is estimated. Two methods of distance computation are implemented: a method based on computation of the log-likelihood of the empirical eigenvalue spacing, and a method based on calculation of the Kullback-Leibler divergence between the NNSD and the limiting distributions. The parameter dist.method determines which method is used (default is "LL" which uses log-likelihood).

Two additional parameters are critical for proper functioning of the algorithm. Depending on the structure of the matrix, it might occur that the eigenvalue spectrum includes outlier eigenvalues located far away from the bulk of the spectrum. For some types of input matrices, it might be necessary to remove such outliers, in order to ensure that the unfolding process works correctly. This is achieved by the setting discard.outliers = TRUE, which is the default setting. In some other cases, it might not be necessary to remove the outliers.

Another critical parameter is discard.zeros. If set to TRUE, rows and columns exclusively containing zeros outside the main diagonal are removed from the matrix at each threshold. This causes the matrix to shrink during the loop over thresholds. Setting discard.zeros = TRUE is especially recommended when the eigenvalue spacing distribution piles up at the left tail of the NNSD during thresholding.

Finding a candidate signal-noise separating threshold for the matrix

The distance of the empirical NNSD to the limiting distributions is not calculated over the whole range of eigenvalue spacings, but over an interval (0, max.ev.spacing). At a spacing of zero, the difference between the Wigner-Dyson and the Exponential distribution is biggest. The maximum spacing considered is determined by the parameter max.ev.spacing. This parameter should not be lower than $\sqrt{2/\pi}$, where the peak of the Wigner-Dyson distribution resides. On the other hand, is does not make sense to choose too high values for max.ev.spacing, because both the Wigner-Dyson and the Exponential distribution assume rather low values at the right tail, not worth to compare with empirical results.

If the matrix contains an inherent modular structure which is covered by noise, a relatively sharp transition from the Wigner-Dyson case to the Exponential case takes place when the threshold is increased gradually. This is confirmed in a plot showing up after completion of the loop over thresholds, as presented in **Figure 8**. The plot shows the calculated distance between NNSD and both limiting distributions versus threshold. At a certain threshold, a striking change becomes apparent: while the log-likelihood belonging to the Wigner-Dyson limit increases sharply (red curve), the log likelihood based on the Exponential case decreases (blue curve). Note that the plot actually shows the negative log-likelihood per eigenvalue.

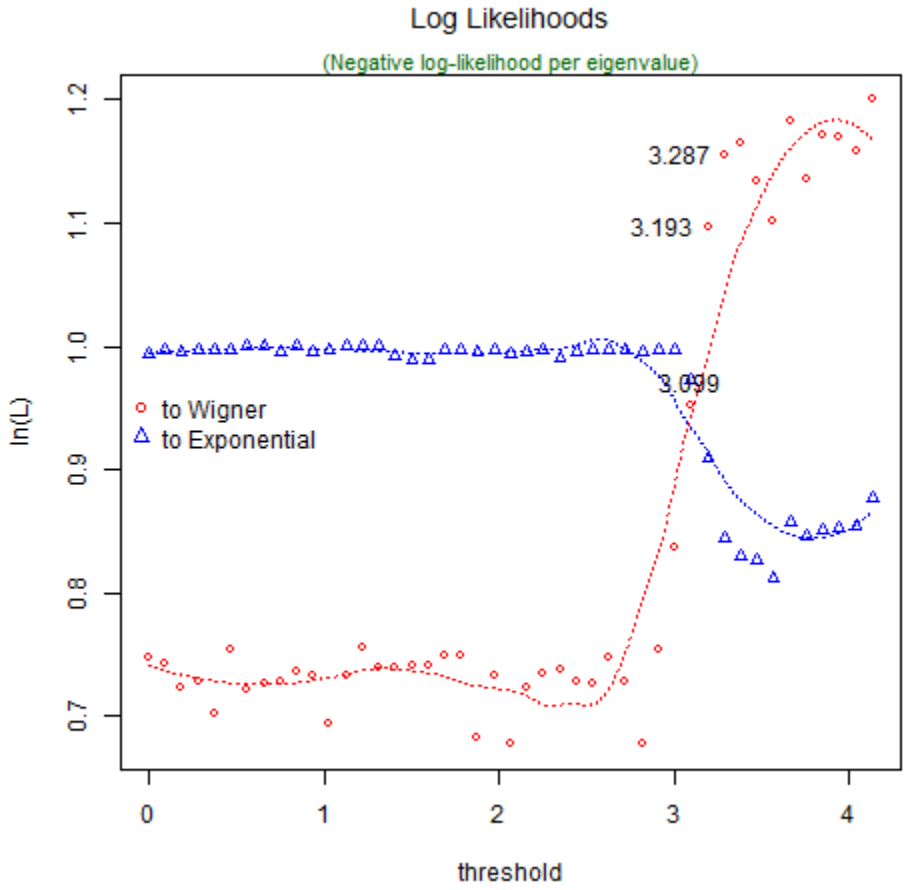


Figure 8

A plot showing the distances of the NNSD to both limiting distributions (Wigner-Dyson and Exponential distribution). Log-likelihood was used as a distance measure. At a certain threshold (of about 3.2), a distinct change occurs. Candidate thresholds have been marked in this plot using the left mouse button (indicated by the labels close to the points).

Finding a candidate signal-noise separating threshold for the matrix

In the plot depicted in **Figure 8,** the user can choose candidate thresholds by clicking with the left mouse button on the points of the red-colored curve. The selection is terminated by a right mouse-click somewhere on the plot. The hereby chosen candidate thresholds are returned by the function.

Two additional plots pop up after completion of the loop (if interactive = TRUE was chosen). The first one shows the p-values for the Kolmogorov-Smirnov test for significance of difference between empirical NNSD and Exponential distribution versus chosen thresholds. The NNSD can be considered close to the Exponential distribution if the p-values rise. The second plot shows some sort of Sum of Squared Errors (SSE) between empirical NNSD and Exponential function versus thresholds. The NNSD comes close to the Exponential function if the SSE drops considerably. Likewise, in both plots, candidate thresholds can be marked using the mouse.

Finding a candidate signal-noise separating threshold for the matrix

In **Figure 8,** the (somewhat scaled) log likelihood was used as a distance measure. We can also use the Kullback-Leibler divergence to estimate the distance to the limiting distributions:

```
res <- rm.get.threshold(random.matrix, dist.method = "KLD")
```

which results in a similar plot (not shown).

In the next example, we refrain from removing outliers in the eigenvalue spectrum:

```
res <- rm.get.threshold(random.matrix, discard.outliers = F)
```

For the random input matrix created above, retaining outliers is not a problem. However, it must be noted that outliers actually can cause problems for other types of matrices.

3. Uncovering the modular structure of a network by thresholding the adjacency matrix

In order to show how the function rm.get.threshold can be used to find a threshold which removes noise from a matrix or network, it is again worthwhile to look at a random graph created using the igraph package. As in section 1, four Erdös - Renyi graphs are created and subsequently assembled into a block-diagonal matrix, representing a modular network with four clusters:

```
matlist = list()
set.seed(979)
for (i in 1:4) matlist[[i]] = get.adjacency(erdos.renyi.game(250, 0.1))
mat <- bdiag(matlist)
rm.matrix.validation(as.matrix(mat))
```

As expected, the NNSD (bottom right of the plot) is exponential because the matrix is modular. Furthermore, we can reveal the clustered structure of the corresponding graph with the clusters command of the igraph package:

```
m <- mat != 0
g  <- graph.adjacency(m, mode = "undirected")
clusters(g)    # 4 clusters of size 250, as expected
```

Uncovering the modular structure of a network by thresholding the adjacency matrix

Now, we hide the modular structure of the matrix by adding Gaussian noise to it using the function add.Gaussian.noise from the RMThreshold package:

```
set.seed(979)
mat1 = add.Gaussian.noise(as.matrix(mat), mean = 0, stddev = 0.1)
rm.matrix.validation(mat1)
```

A validation of that new, noisy matrix (mat1) reveals that it is completely random – the NNSD is Wigner-like (bottom-right of the validation plot). A modular structure is no longer visible, and the whole matrix seems to consist of a single large cluster, because spurious matrix elements have been generated which now connect the true clusters:

```
m1 <- mat1 != 0
g1  <- graph.adjacency(m1, mode = "undirected")
clusters(g1)      # a single big cluster with 1000 nodes
```

Uncovering the modular structure of a network by thresholding the adjacency matrix

Next, we try to find a threshold for the adjacency matrix that removes the noise, i.e. we try to set those matrix elements to zero that are noise by their nature:

```
res <- rm.get.threshold(mat1)    # noisy matrix as input
```

We can assume the threshold somewhere between 0.52 and about 1 where a plateau in the distance plot appears. Trying with 0.6, we should be able to reconstruct the original clusters by using the command rm.denoise.mat which simply applies the above identified threshold to the noisy matrix:

```
cleaned <- rm.denoise.mat(mat1, 0.6)
matr <- cleaned != 0
g  <- graph.adjacency(matr, mode = "undirected")
clusters(g)    # 4 clusters reconstructed !
```

It appears that the four original clusters have been successfully reconstructed.

4. Running the main algorithm on a smaller interval of thresholds

The main function of RMThreshold records the eigenvalue spacing distribution at each suppositional threshold. By default, the interval of searched thresholds ranges from the minimum absolute value of all matrix elements to the maximum absolute value of all matrix elements. This interval can be narrowed down by adjusting the parameter interval:

```
res <- rm.get.threshold(random.matrix, interval = c(2.5, 3.5))
```

The above command only searches thresholds between 2.5 and 3.5 (recall that the threshold found for the matrix 'random.matrix' was about 3). The result of the refined search is shown in **Figure 9**. However, in most of the cases, searching a smaller interval of thresholds might not be necessary.

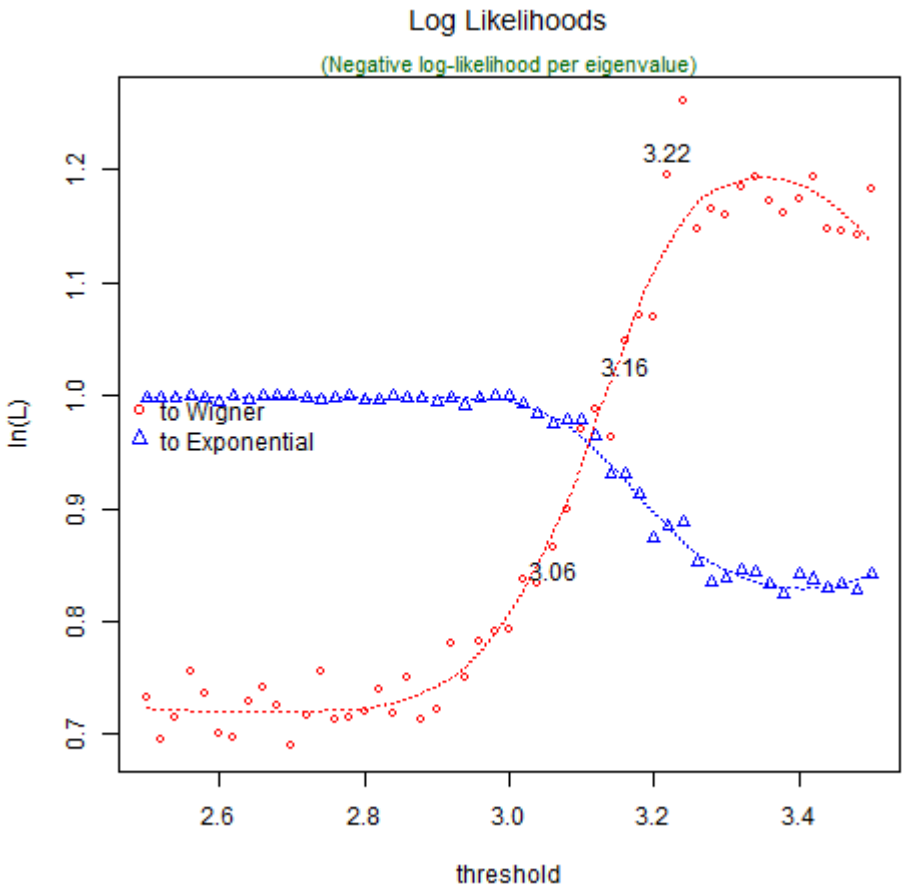


Figure 9

A plot showing the distances between NNSD and both limiting distributions. A smaller interval of thresholds was searched by setting the interval parameter to (2.5, 3.5).

5. Applying the identified threshold to the matrix

We have already seen in section 3 how the command rm.denoise.mat can be used to remove spurious matrix elements from adjacency matrices. For the matrix 'random.matrix', we had identified a threshold of about 3.2 (section 2, **Figure 8**):

```
cleaned.matrix <- rm.denoise.mat(random.matrix, threshold = 3.2)
cleaned.matrix <- rm.discard.zeros(cleaned.matrix)
dim(cleaned.matrix)
```

The first command sets all matrix elements whose absolute value is below 3.2 to zero. The second command can be used if discard.zeros = FALSE had been chosen during execution of the main function. It removes those rows/columns of the matrix that exclusively contain zeros outside of the main diagonal (with regard to networks, it removes isolated nodes).

Applying the identified threshold to the matrix

Finally, let us try to find clusters in the graph corresponding to this matrix, i.e. let's assume that it was an adjacency matrix:

```
m3 <- cleaned.matrix != 0
g3  <- graph.adjacency(m3, mode = "undirected")
clusters(g3)
```

Here, we cannot expect a clear clustered structure, because the matrix is exclusively built up from Gaussian noise (by using the function create.rand.mat).

References

- The package RMThreshold is available in the **CRAN** https://cran.r-project.org/web/packages/
- **Wikipedia** on Random Matrices: https://en.wikipedia.org/wiki/Random_matrix
- Wigner semi-circle law at **Wolfram MathWorld**: https://en.wikipedia.org/wiki/Random_matrix
- Wigner, E. P. , **Characteristic vectors of bordered matrices with infinite dimensions**, Ann. Math. 62, 548–564, 1955.
- Mehta, M., **Random Matrices**, 3nd edition. Academic Press, 2004.
- Furht, B. and Escalante, A. (eds.), **Handbook of Data Intensive Computing**, Springer Science and Business Media, 2011.
- Luo, F. et al., **Constructing gene co-expression networks and predicting functions of unknown genes by random matrix theory**. BMC Bioinformatics, 2007.
- Jonoska, N. Saito M. (Eds.), **Discrete and topological models in Molecular Biology**. Springer 2014.