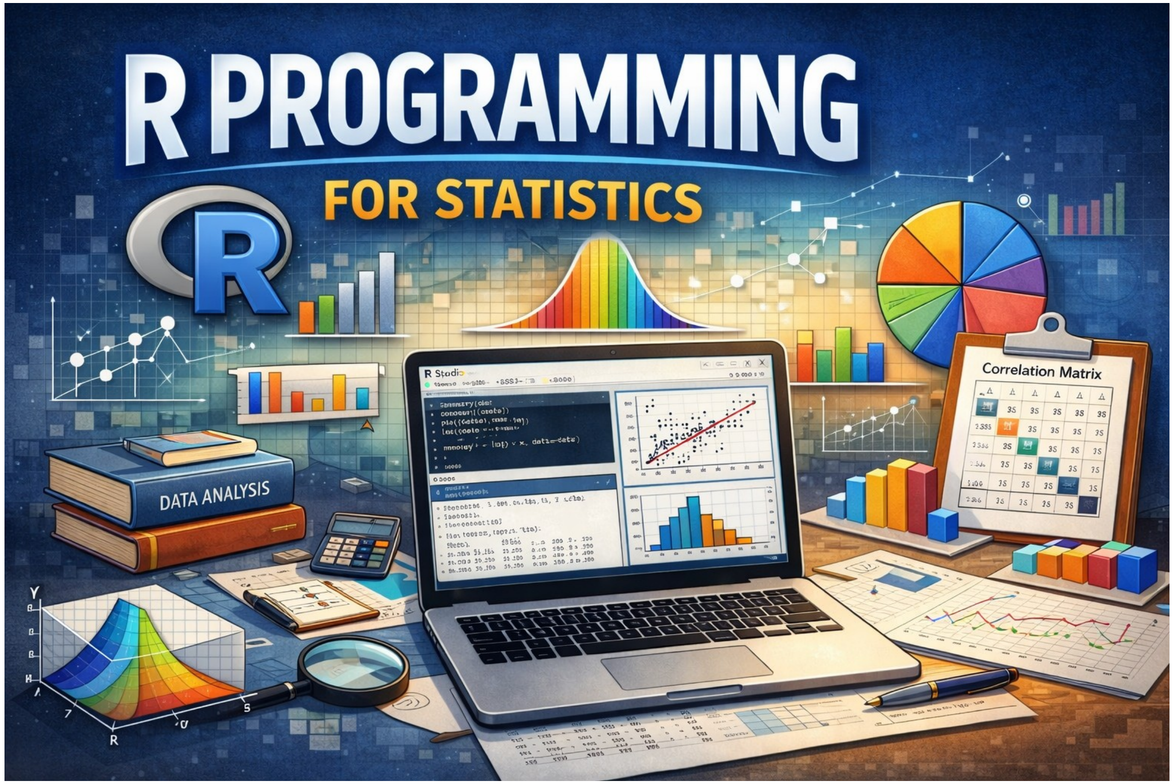
Advanced Clustering in R

Nikolay Oskolkov, MRG Group Leader, LIOS, Riga, Latvia
R course, 12.02.2026



@NikolayOskolkov

@oskolkov.bsky.social

Personal homepage:
https://nikolay-oskolkov.com

Image generated by ChatGPT

Session content

Topics we'll cover in this session:

1) Spectral clustering and Graph Laplacian

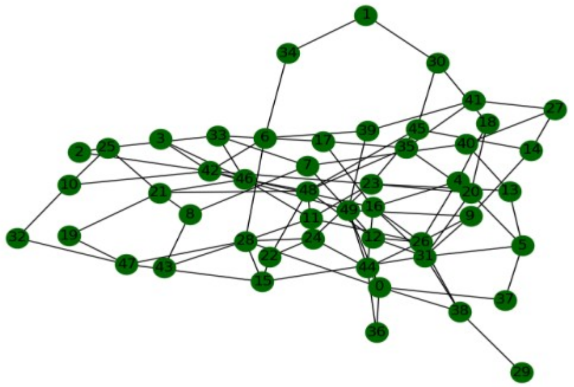
2) Density-based clustering (DBSCAN, HDBSCAN)

3) Graph-based clustering (SNN, Louvain, Leiden)

Spectral clustering

Spectral clustering: Graph Laplacian

Graph Laplacian, Laplacian Eigenmap, spectral clustering, diffusion maps, spectral dimension reduction methods etc.



$$s(x_i, x_j) = exp(-\alpha \|x_i - x_j\|^2)$$

$$L = I - D^{-1} * S$$
$$L * u = \lambda * u$$

Laplacian Eigenmap

$$P = D^{-1} * S$$
$$P^t * u = \lambda^t * u$$

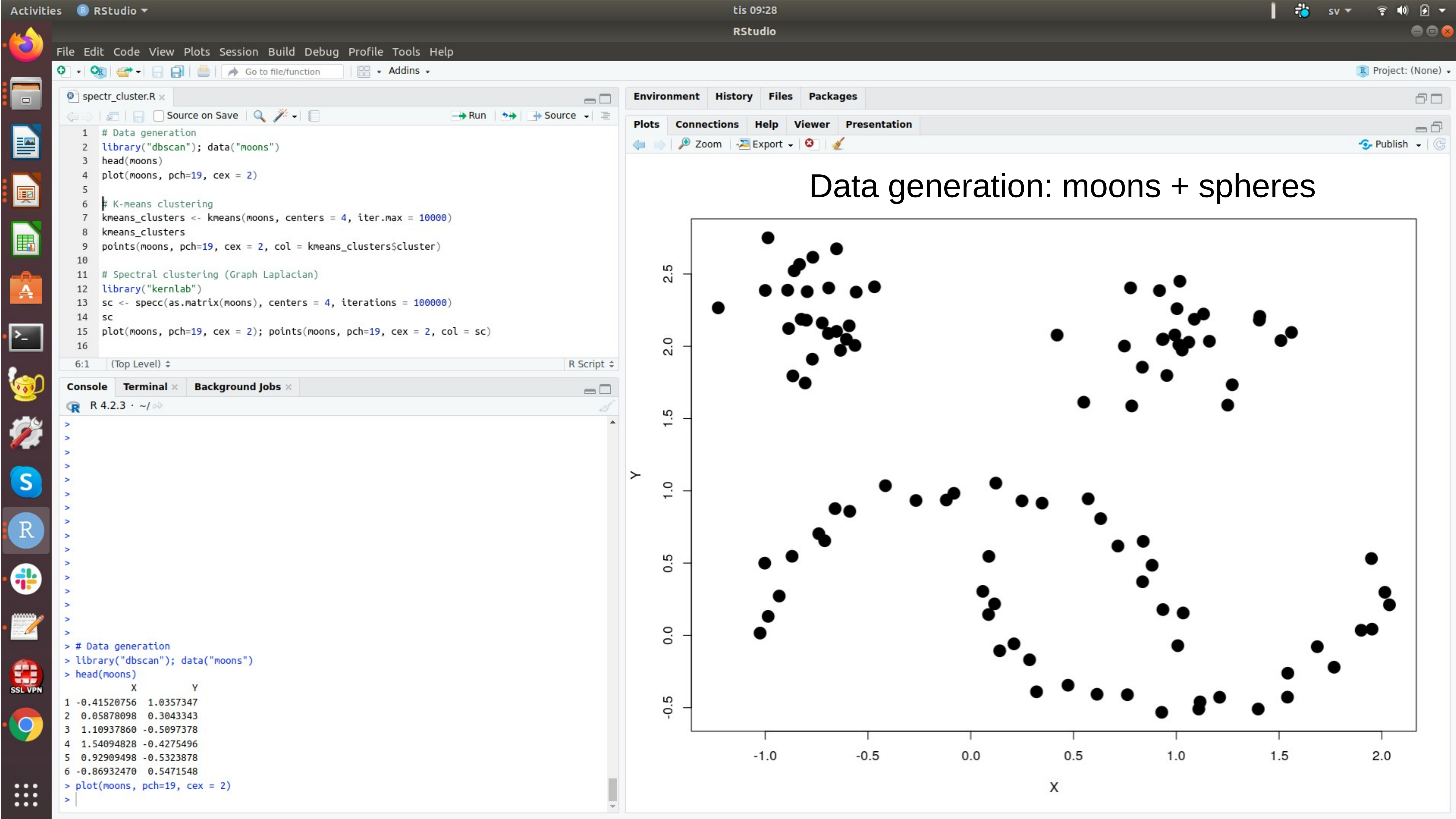
Diffusion Maps

S=

```
        [,1] [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
[1,] 1.0000000    0 0.7429016 0.6319343 0.0000000 0.0000000 0.0000000
[2,] 0.0000000    1 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
[3,] 0.7429016    0 1.0000000 0.0000000 0.0000000 0.0000000 0.6657756
[4,] 0.6319343    0 0.0000000 1.0000000 0.0000000 0.0000000 0.7195922
[5,] 0.0000000    0 0.0000000 0.0000000 1.0000000 0.7765565 0.0000000
[6,] 0.0000000    0 0.0000000 0.0000000 0.7765565 1.0000000 0.0000000
[7,] 0.0000000    0 0.6657756 0.7195922 0.0000000 0.0000000 1.0000000
[8,] 0.0000000    0 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
```

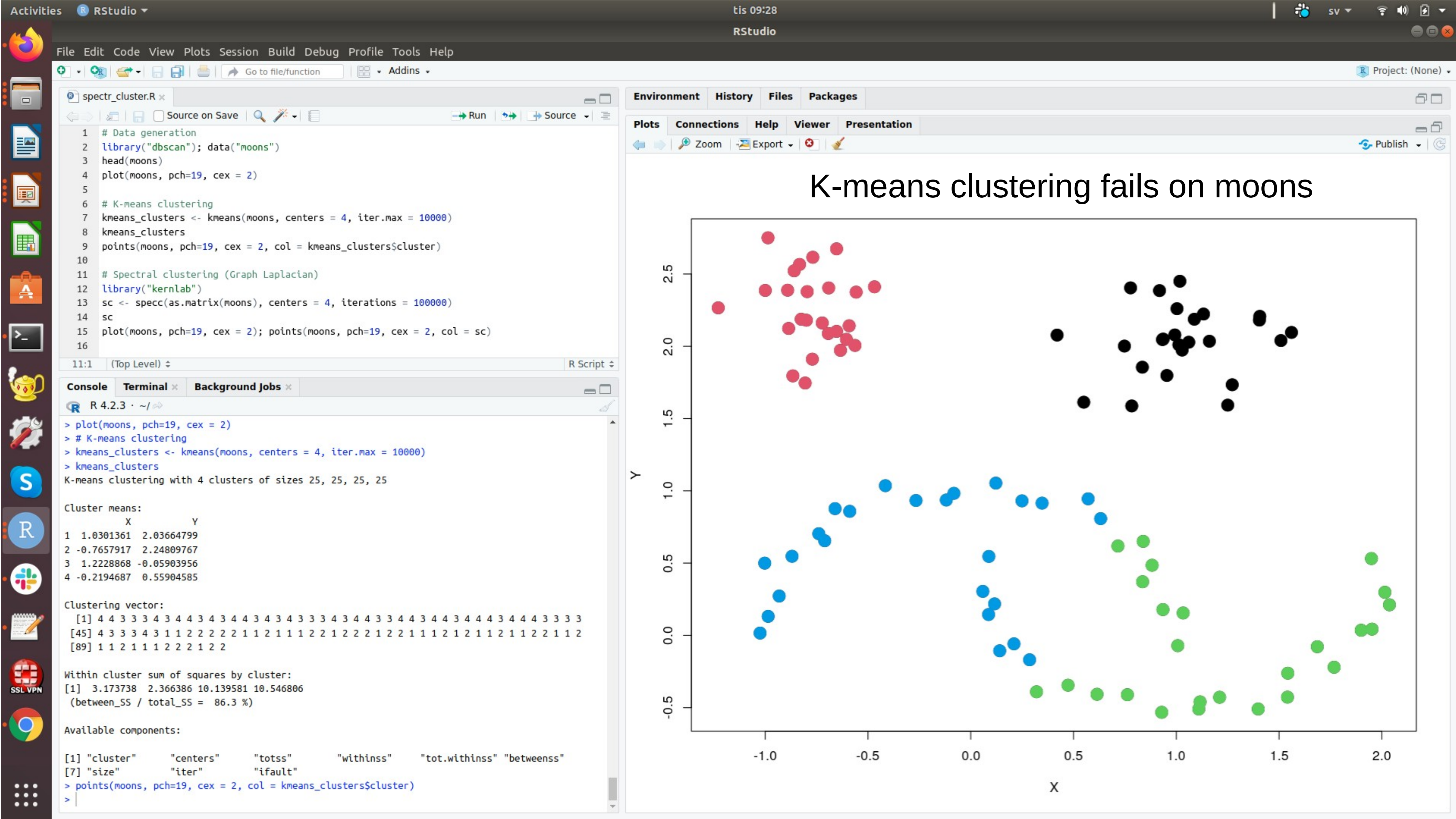
D=

```
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
[1,] 2.374836 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
[2,] 0.000000 2.597451 0.000000 0.000000 0.000000 0.000000 0.000000
[3,] 0.000000 0.000000 2.408677 0.000000 0.000000 0.000000 0.000000
[4,] 0.000000 0.000000 0.000000 2.351526 0.000000 0.000000 0.000000
[5,] 0.000000 0.000000 0.000000 0.000000 2.523175 0.000000 0.000000
[6,] 0.000000 0.000000 0.000000 0.000000 0.000000 2.519936 0.000000
[7,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3.170424
[8,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
```


Activities
RStudio
tis 09:28
sv
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
spectr_cluster.R
Source on Save
Run
Source
Environment History Files Packages
Plots Connections Help Viewer Presentation
Zoom
Export
Publish
6:1
(Top Level)
R Script
Console Terminal Background Jobs
R 4.2.3 · ~/
SSL VPN
Data generation: moons + spheres
Y
X
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
2.5

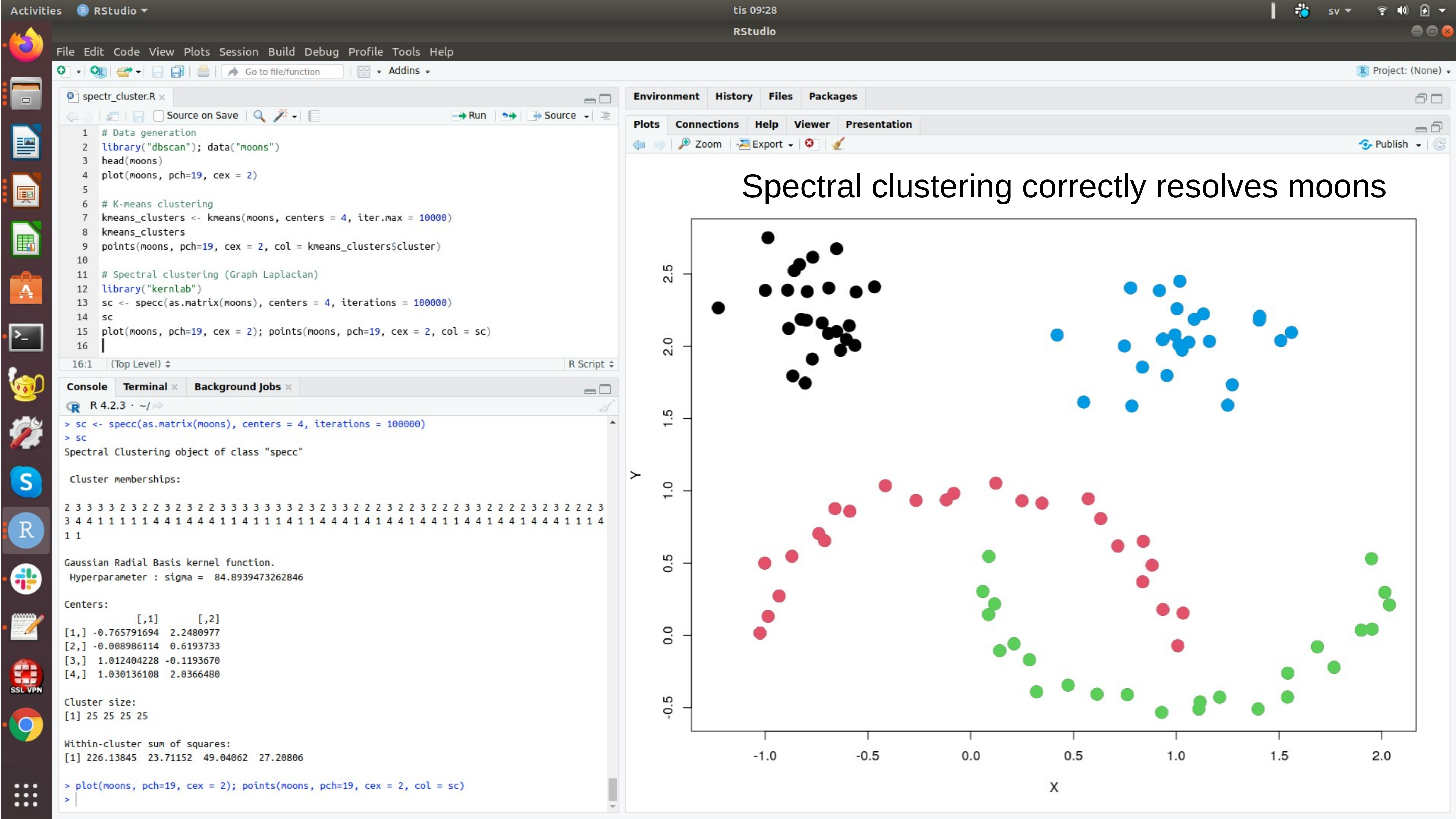
```r
# Data generation
library("dbscan"); data("moons")
head(moons)
plot(moons, pch=19, cex = 2)

# K-means clustering
kmeans_clusters <- kmeans(moons, centers = 4, iter.max = 10000)
kmeans_clusters
points(moons, pch=19, cex = 2, col = kmeans_clusters$cluster)

# Spectral clustering (Graph Laplacian)
library("kernlab")
sc <- specc(as.matrix(moons), centers = 4, iterations = 100000)
sc
plot(moons, pch=19, cex = 2); points(moons, pch=19, cex = 2, col = sc)
```

```
> # Data generation
> library("dbscan"); data("moons")
> head(moons)
            X          Y
1 -0.41520756  1.0357347
2  0.05878098  0.3043343
3  1.10937860 -0.5097378
4  1.54094828 -0.4275496
5  0.92909498 -0.5323878
6 -0.86932470  0.5471548
> plot(moons, pch=19, cex = 2)
>
```



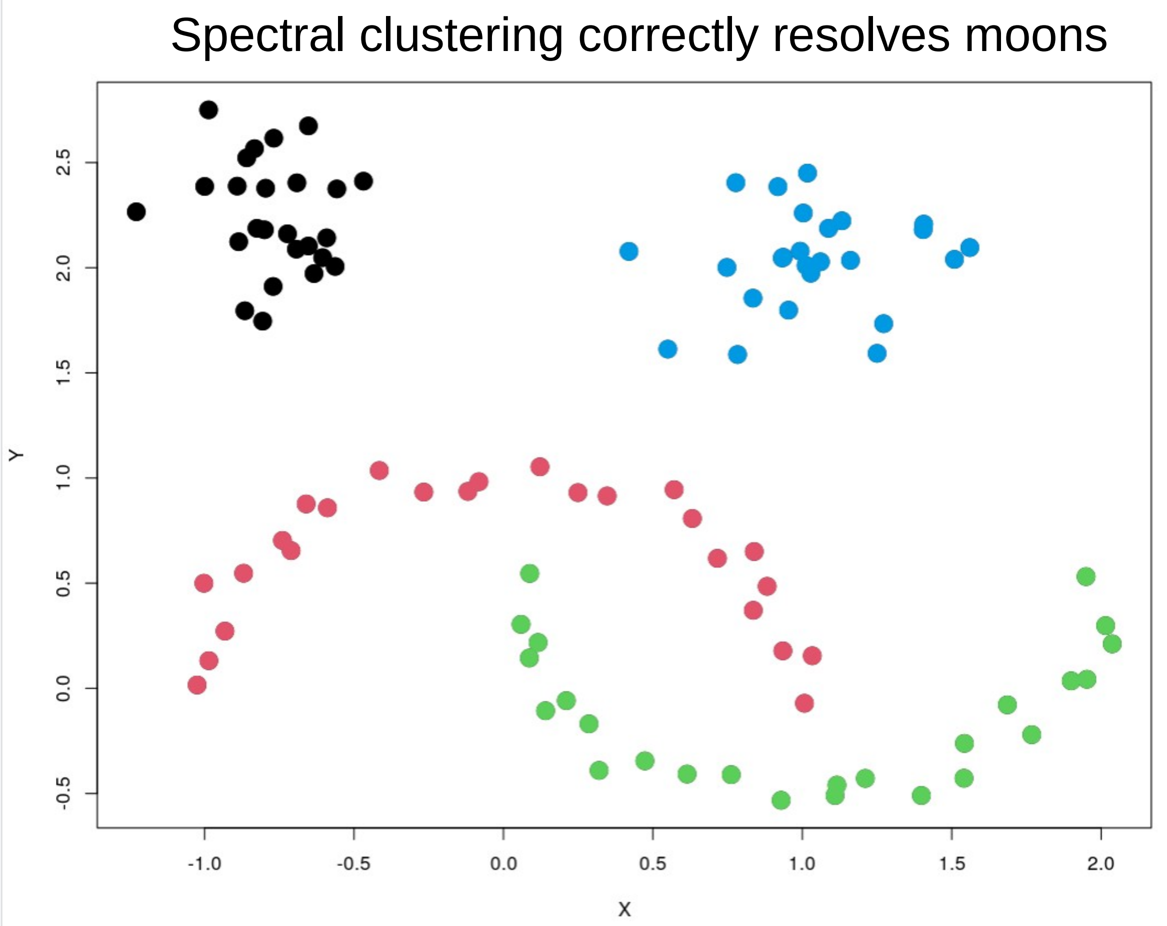
```r
# Data generation
library("dbscan"); data("moons")
head(moons)
plot(moons, pch=19, cex = 2)

# K-means clustering
kmeans_clusters <- kmeans(moons, centers = 4, iter.max = 10000)
kmeans_clusters
points(moons, pch=19, cex = 2, col = kmeans_clusters$cluster)

# Spectral clustering (Graph Laplacian)
library("kernlab")
sc <- specc(as.matrix(moons), centers = 4, iterations = 100000)
sc
plot(moons, pch=19, cex = 2); points(moons, pch=19, cex = 2, col = sc)
```

```
> plot(moons, pch=19, cex = 2)
> # K-means clustering
> kmeans_clusters <- kmeans(moons, centers = 4, iter.max = 10000)
> kmeans_clusters
K-means clustering with 4 clusters of sizes 25, 25, 25, 25

Cluster means:
           X           Y
1  1.0301361  2.03664799
2 -0.7657917  2.24809767
3  1.2228868 -0.05903956
4 -0.2194687  0.55904585

Clustering vector:
  [1] 4 4 3 3 3 4 3 4 4 3 4 3 4 4 3 4 3 4 3 3 3 4 3 4 4 3 3 4 4 3 4 4 3 4 4 4 3 4 4 4 3 3 3 3
 [45] 4 3 3 3 4 3 1 1 2 2 2 2 2 1 1 2 1 1 1 2 2 1 2 2 2 1 2 2 1 1 1 2 1 2 1 1 2 1 1 2 2 1 1 2
 [89] 1 1 2 1 1 1 2 2 2 1 2 2

Within cluster sum of squares by cluster:
[1]  3.173738  2.366386 10.139581 10.546806
 (between_SS / total_SS =  86.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> points(moons, pch=19, cex = 2, col = kmeans_clusters$cluster)
>
```


Activities
RStudio
tis 09:28
sv
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
spectr_cluster.R
Source on Save
Run
Source
Environment History Files Packages
Plots Connections Help Viewer Presentation
Zoom
Export
Publish
16:1
(Top Level)
R Script
Console Terminal Background Jobs
R 4.2.3 · ~/

```
# Data generation
library("dbscan"); data("moons")
head(moons)
plot(moons, pch=19, cex = 2)

# K-means clustering
kmeans_clusters <- kmeans(moons, centers = 4, iter.max = 10000)
kmeans_clusters
points(moons, pch=19, cex = 2, col = kmeans_clusters$cluster)

# Spectral clustering (Graph Laplacian)
library("kernlab")
sc <- specc(as.matrix(moons), centers = 4, iterations = 100000)
sc
plot(moons, pch=19, cex = 2); points(moons, pch=19, cex = 2, col = sc)

```

```
> sc <- specc(as.matrix(moons), centers = 4, iterations = 100000)
> sc
Spectral Clustering object of class "specc"

 Cluster memberships:

2 3 3 3 3 2 3 2 2 3 2 3 2 2 3 3 3 3 3 3 3 2 3 2 3 3 2 2 2 3 2 2 3 2 2 2 3 3 2 2 2 2 3 2 3 2 2 2 3
3 4 4 1 1 1 1 1 4 4 1 4 4 4 1 1 4 1 1 1 4 1 1 4 4 4 1 4 1 4 4 1 4 4 1 1 4 4 1 4 4 1 4 4 4 1 1 1 4
1 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  84.8939473262846

Centers:
             [,1]       [,2]
[1,] -0.765791694  2.2480977
[2,] -0.008986114  0.6193733
[3,]  1.012404228 -0.1193670
[4,]  1.030136108  2.0366480

Cluster size:
[1] 25 25 25 25

Within-cluster sum of squares:
[1] 226.13845  23.71152  49.04062  27.20806

> plot(moons, pch=19, cex = 2); points(moons, pch=19, cex = 2, col = sc)
> 
```



Density-based clustering
(Mean Shift, DBSCAN, HDBSCAN)

Density idea, advantages and advantages

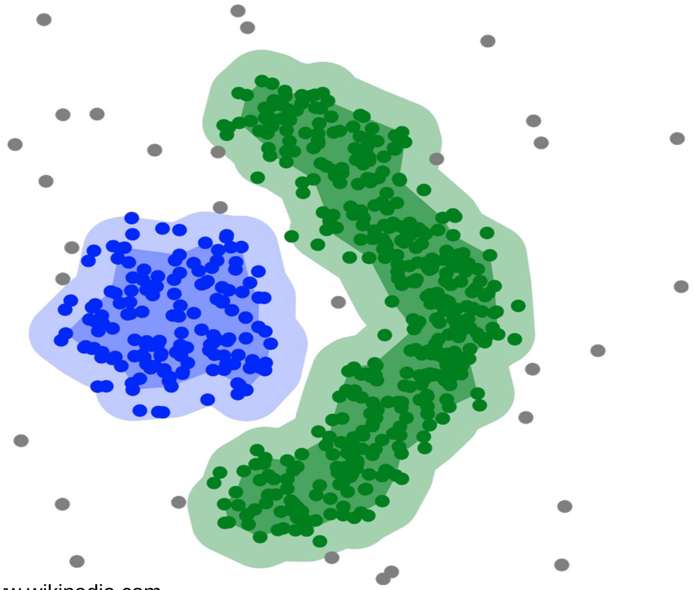


Image source: www.wikipedia.com

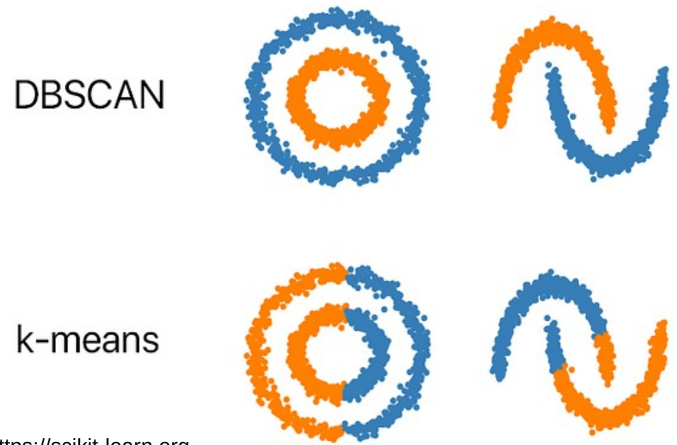


Image source: https://scikit-learn.org

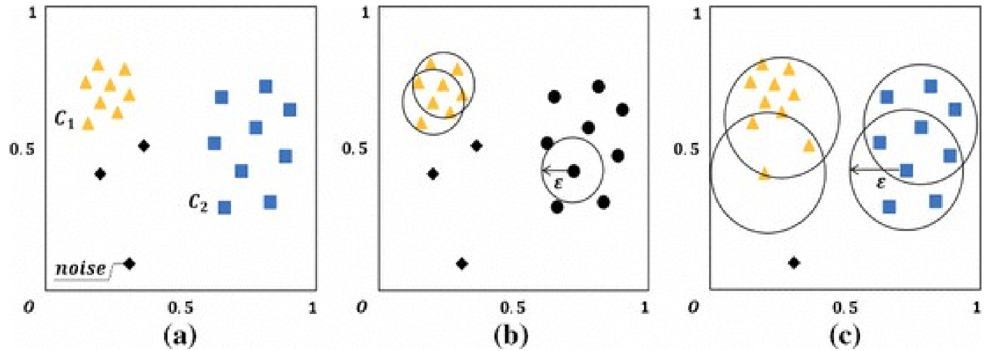
Advantages:

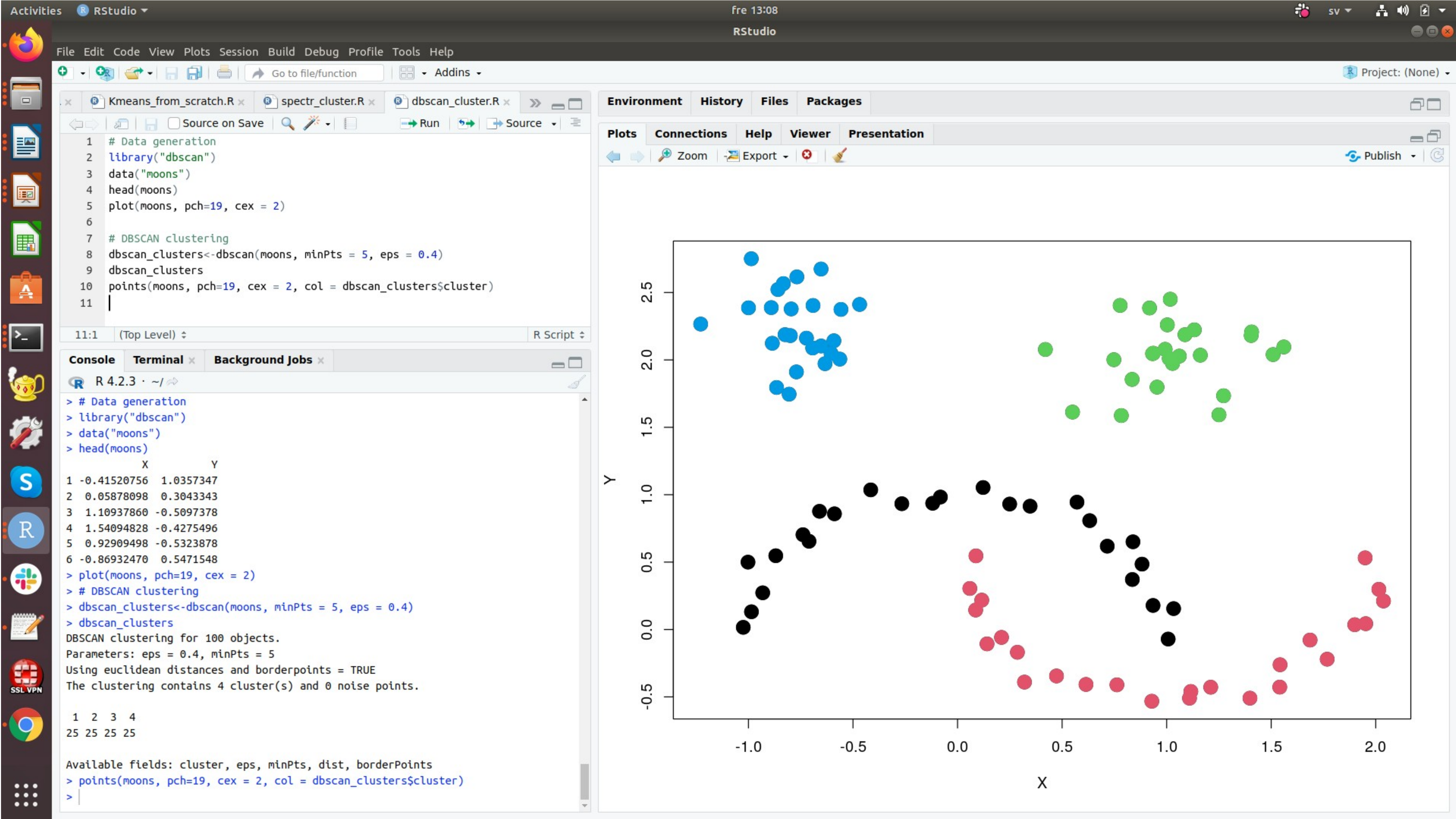
1) Independent of cluster shapes

2) Detects number of clusters automatically

Disadvantages:

Density-based clustering cannot handle clusters of different density



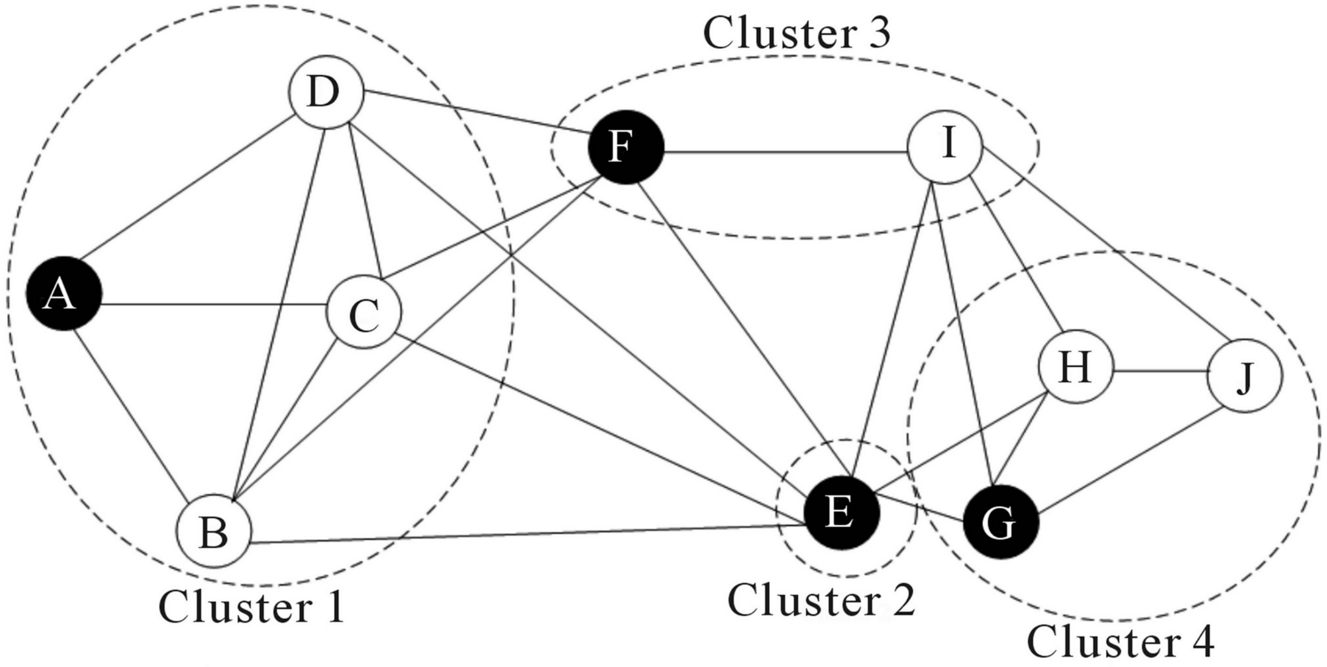


```r
# Data generation
library("dbscan")
data("moons")
head(moons)
plot(moons, pch=19, cex = 2)

# DBSCAN clustering
dbscan_clusters<-dbscan(moons, minPts = 5, eps = 0.4)
dbscan_clusters
points(moons, pch=19, cex = 2, col = dbscan_clusters$cluster)
```

```
> # Data generation
> library("dbscan")
> data("moons")
> head(moons)
            X          Y
1 -0.41520756  1.0357347
2  0.05878098  0.3043343
3  1.10937860 -0.5097378
4  1.54094828 -0.4275496
5  0.92909498 -0.5323878
6 -0.86932470  0.5471548
> plot(moons, pch=19, cex = 2)
> # DBSCAN clustering
> dbscan_clusters<-dbscan(moons, minPts = 5, eps = 0.4)
> dbscan_clusters
DBSCAN clustering for 100 objects.
Parameters: eps = 0.4, minPts = 5
Using euclidean distances and borderpoints = TRUE
The clustering contains 4 cluster(s) and 0 noise points.

 1  2  3  4
25 25 25 25

Available fields: cluster, eps, minPts, dist, borderPoints
> points(moons, pch=19, cex = 2, col = dbscan_clusters$cluster)
>
```

Graph-based clustering (SNN, Louvain, Leiden)

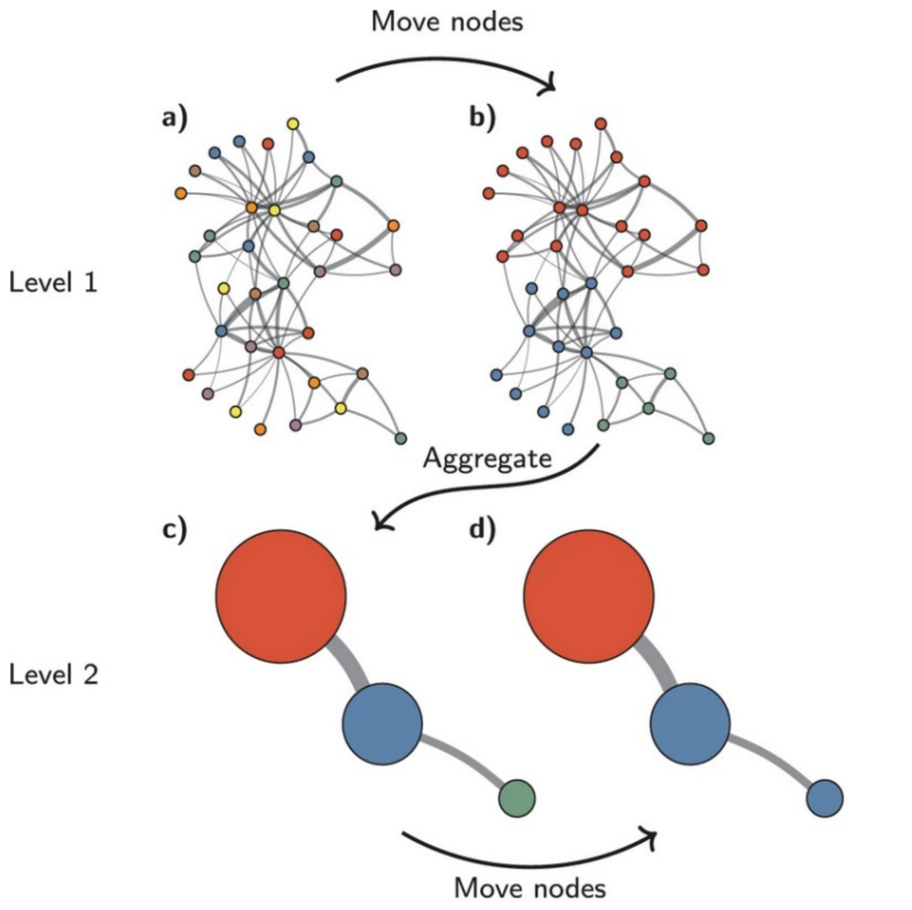
Shared Nearest Neighbors (SNN)



C and D belong to the same cluster because they share 4 neighbors (A, B, E and F)

C and F belong to different clusters because they share 3 neighbors (D, B and E)

Louvain clustering



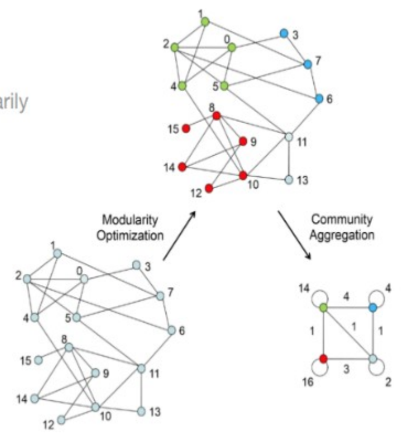
Louvain algorithm. The Louvain algorithm starts from a singleton partition in which each node is in its own community (**a**). The algorithm moves individual nodes from one community to another to find a partition (**b**). Based on this partition, an aggregate network is created (**c**). The algorithm then moves individual nodes in the aggregate network (**d**). These steps are repeated until the quality cannot be increased further.

Phase 1: greedy modularity optimisation

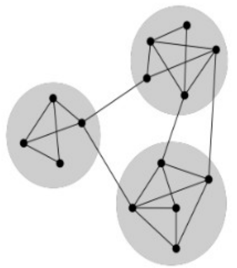
1. Start with 1n/community
2. Compute Q by moving i to the community of j
3. If ΔQ>1, node is placed in community
4. Repeat 1-3 until no improvement is found. Ties solved arbitrarily

Phase 2: coarse grained community aggregation

5. Link nodes in a community into single node.
6. Self loops show intra-community associations
7. Inter-community weights kept
8. Repeat phase 1 on new network



Other methods:
Walktrap
Label propagation
...
(benchmarking)



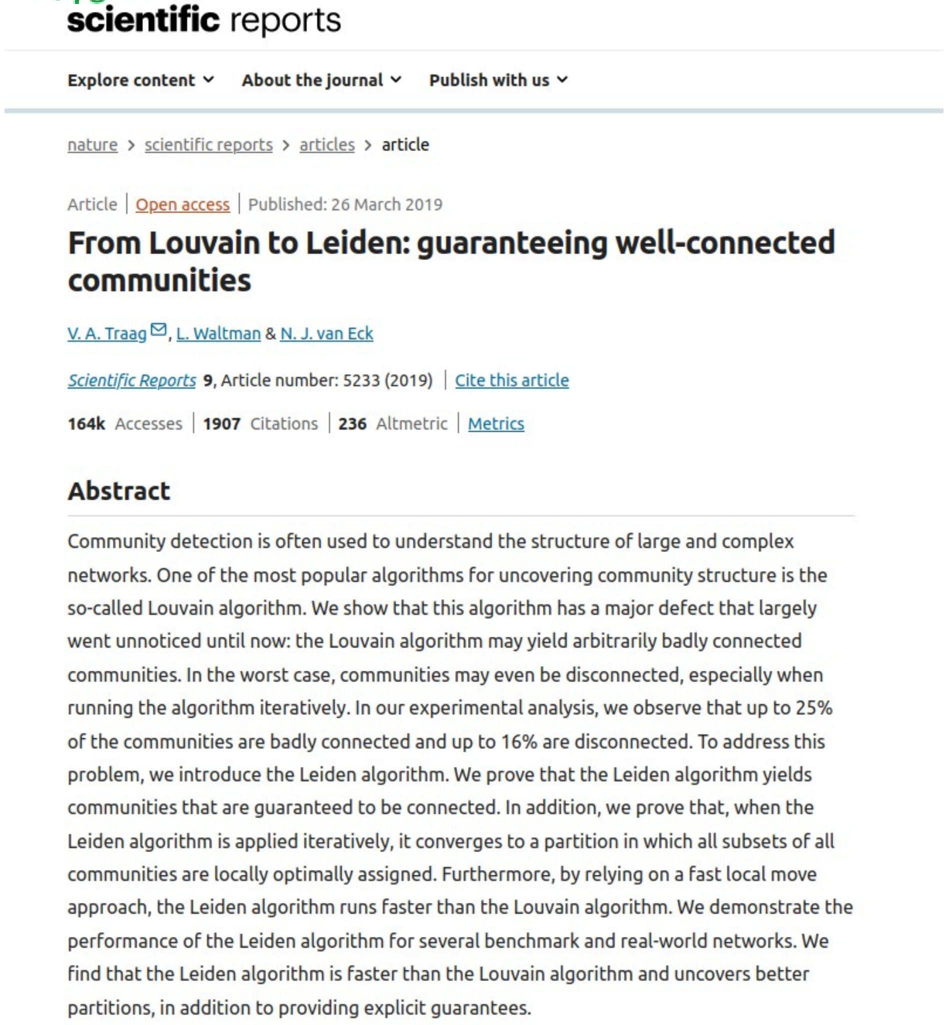
Campigotto 2014
Traag 2019

Source: https://nbisweden.github.io/workshop_omics_integration/session_topology

$$Q = \frac{1}{2m}\sum_c \left(e_c - \gamma\frac{K_c^2}{2m}\right)$$ - modularity optimization

Leiden clustering

Latvian Institute of Organic Synthesis



scientific reports

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature > scientific reports > articles > article

Article | Open access | Published: 26 March 2019

From Louvain to Leiden: guaranteeing well-connected communities

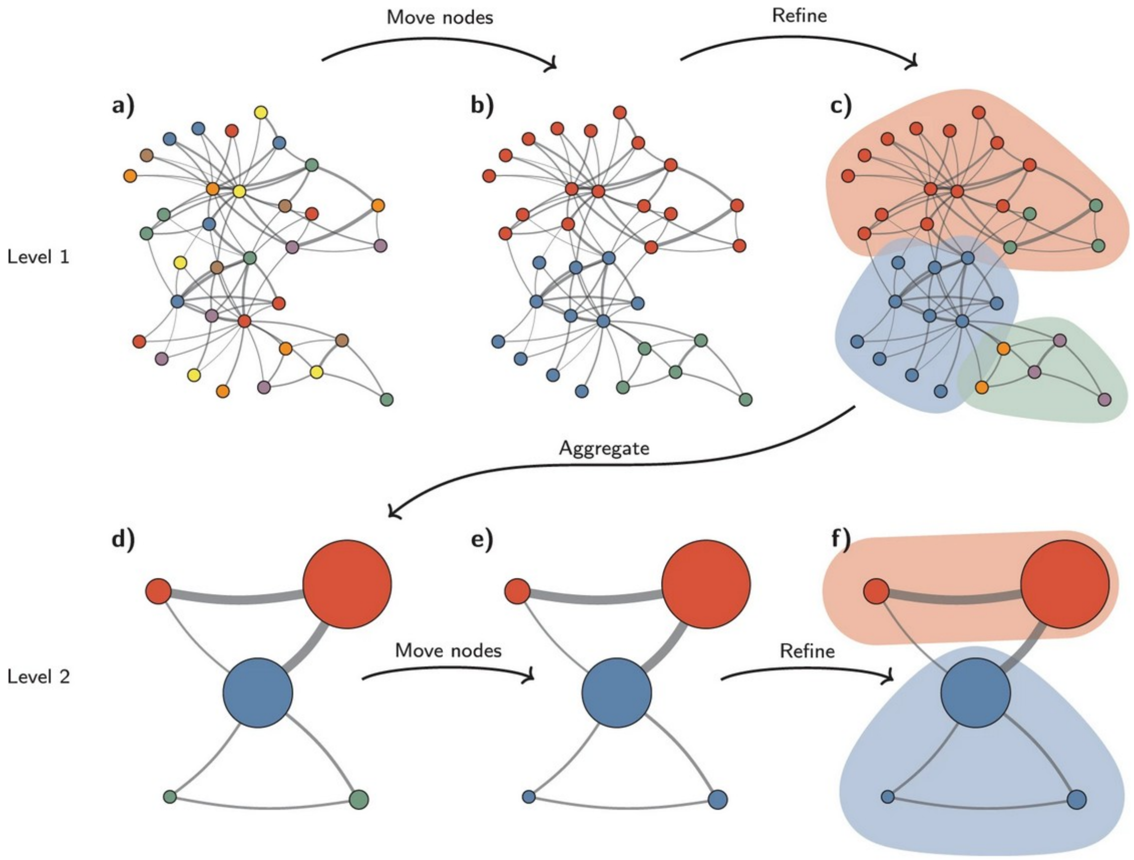
V. A. Traag, L. Waltman & N. J. van Eck

Scientific Reports **9**, Article number: 5233 (2019) | Cite this article

164k Accesses | **1907** Citations | **236** Altmetric | Metrics

Abstract

Community detection is often used to understand the structure of large and complex networks. One of the most popular algorithms for uncovering community structure is the so-called Louvain algorithm. We show that this algorithm has a major defect that largely went unnoticed until now: the Louvain algorithm may yield arbitrarily badly connected communities. In the worst case, communities may even be disconnected, especially when running the algorithm iteratively. In our experimental analysis, we observe that up to 25% of the communities are badly connected and up to 16% are disconnected. To address this problem, we introduce the Leiden algorithm. We prove that the Leiden algorithm yields communities that are guaranteed to be connected. In addition, we prove that, when the Leiden algorithm is applied iteratively, it converges to a partition in which all subsets of all communities are locally optimally assigned. Furthermore, by relying on a fast local move approach, the Leiden algorithm runs faster than the Louvain algorithm. We demonstrate the performance of the Leiden algorithm for several benchmark and real-world networks. We find that the Leiden algorithm is faster than the Louvain algorithm and uncovers better partitions, in addition to providing explicit guarantees.



Leiden algorithm. The Leiden algorithm starts from a singleton partition (**a**). The algorithm moves individual nodes from one community to another to find a partition (**b**), which is then refined (**c**). An aggregate network (**d**) is created based on the refined partition, using the non-refined partition to create an initial partition for the aggregate network. For example, the red community in (**b**) is refined into two subcommunities in (**c**), which after aggregation become two separate nodes in (**d**), both belonging to the same community. The algorithm then moves individual nodes in the aggregate network (**e**). In this case, refinement does not change the partition (**f**). These steps are repeated until no further improvements can be made.

Session summary

Take home messages of the session:

1) Spectral clustering methods are more flexible in resolving clusters of various shapes

2) Density-based clustering does not need to know the number of clusters a-priori and allows for unassigned points

3) Graph-based clustering methods define a special distance between data points on a graph of pair-wise similarities. This is perhaps the most robust approach

Acknowledgments: LIOS + TARGETWISE

Latvian Institute of Organic Synthesis

2027
National Development Plan

Funded by the European Union