Individual Assignment 2

Daniel Alonso

May 23th, 2021

Exercise 1

(a) Closeness centrality

The YouTube network: Perhaps a good example of how closeness centrality could be useful is the YouTube network. From using YouTube regularly, I know that there's several content creators with a large following on many areas of entertainment, education etc. For example, if we take a specific group of influencers, like Python programming educators on youtube, we can mentally visualize how the followers of many of these are also the followers of the others.

This way, if we visualize a reasonably large subset of the network (as the network is probably immensely large, and maybe too big to even properly visualize), we can maybe see who are the largest influencers in the network in terms of how close they are to the rest of the nodes (followers), given that these nodes will sometimes be exclusively connected to one of the channels within that specific topic, but more commonly the followers of one channel will also follow multiple creators within that topic.

This way we can identify clusters of content creators which create similar types of content.

(b) Betweenness centrality

Airport/Road network: Betweenness centrality helps identify nodes where a lot of potential traffic could come as a way to move from one node to another in the network. This way we can identify airports with a very large amount of connections, which are often very busy airports. For example, Madrid, Paris, London, Frankfurt, Singapore, Atlanta airports are all notoriously busy, but they're also very important connection hubs to redirect air traffic to other locations. For these, we would find a reasonably large betweenness centrality measure.

Also, the same would apply for a network of roads + counties, where we can see the most important intersections, those that should theoretically be the best link among different counties in a city. As for an example pertaining Spain and other European countries, roundabouts could represent a good analogy for busy intersections, assuming there's significantly larger/busier roundabouts in the city given their presence as connection hubs between different node/location clusters.

(c) Eigenvector centrality or PageRank centrality

Udemy network: If this data were to be publicly available, I would envision it as courses + instructors + buyers of courses type dataset, where we connect nodes with different shapes, buyers could be circles, and a connection between a buyer and a course (squares) means the buyer has purchased the course, then courses could be connected to instructors (triagles), meaning the instructor is the owner/creator of said course. A bit of a complex idea, but this way we would have sort of 2 sub-networks, one of instructors + courses and one of buyers + courses.

The idea in my mind, is that Eigenvector Centrality could allow us to show what instructors are the most influential in the network, given that an instructor with several high-quality courses, which have pulled a significant amount of users towards 1 course, could also probably pull a significant amount of users towards

multiple courses. This way, in my mind, I can envision how the eigenvector centrality measure could tell us the most influential/highest earning instructors within the platform, among clusters of high earning instructors for specific categories.

Pagerank centrality would do something similar, however it would simply show us high earning instructors in general.

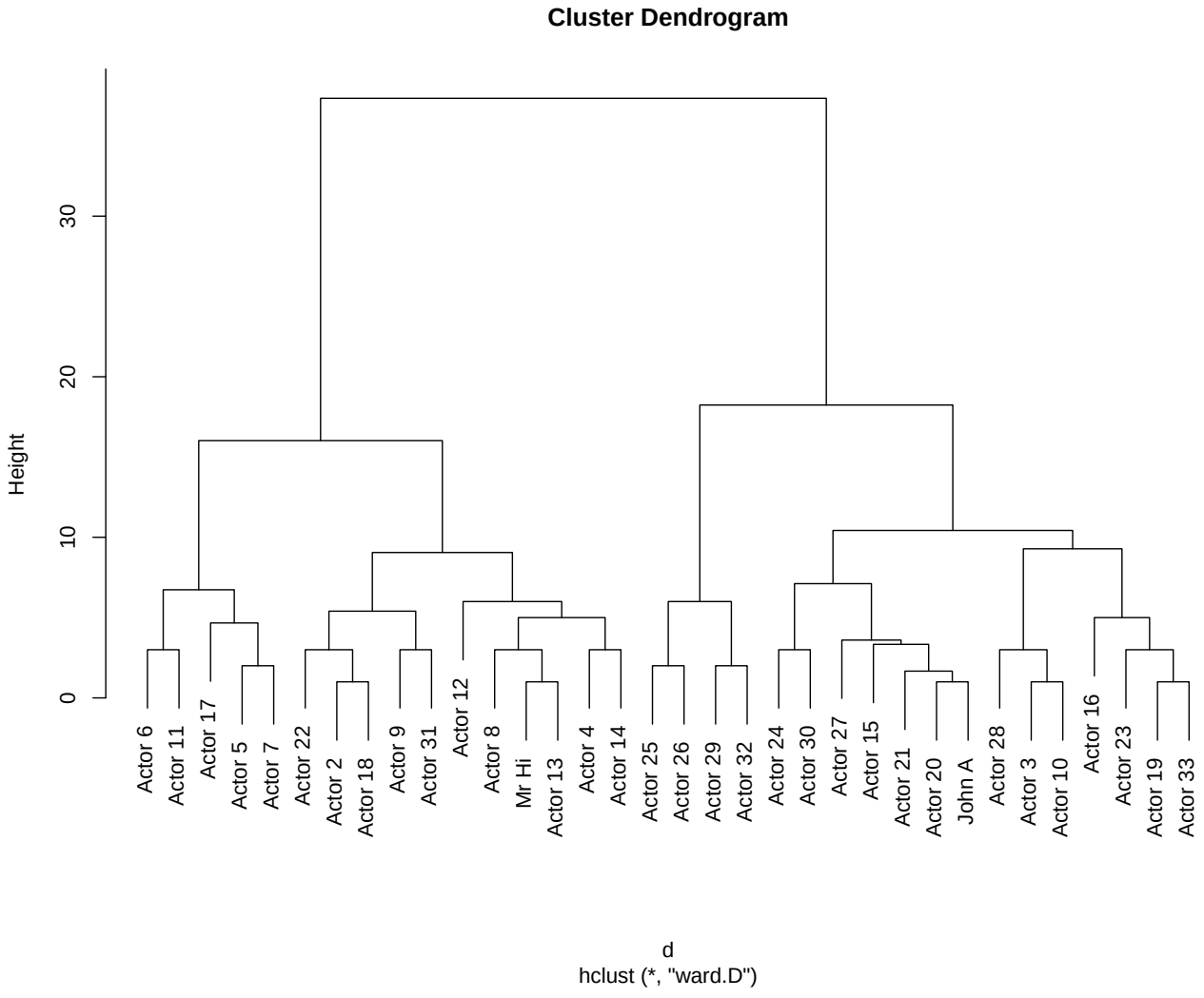
Exercise 2

Hierarchical clustering

First we load up the dataset and compute the matrix of distances as follows:

```
data(karate)
d <- as.dist(distances(karate))
```

Using method='ward.D'



Modularity of clustering

K = 2

```
#> [1] 0.3175542
```

This partition has a decent modularity, slightly worse than that of edge betweenness community detection, but not by much. Worse than the rest of the ones seen in the topic 3 part 2 notebook.

K = 3

```
#> [1] 0.3256903
```

This one is better than using k=2 and than edge betweenness community detection, but worse than the other methods in the other methods as well.

K = 4

```
#> [1] 0.348044
```

There is an improvement here as well, not huge, but reasonable, still worse than the rest of the methods, but improving on the results using k=3.

K = 5

```
#> [1] 0.3048159
```

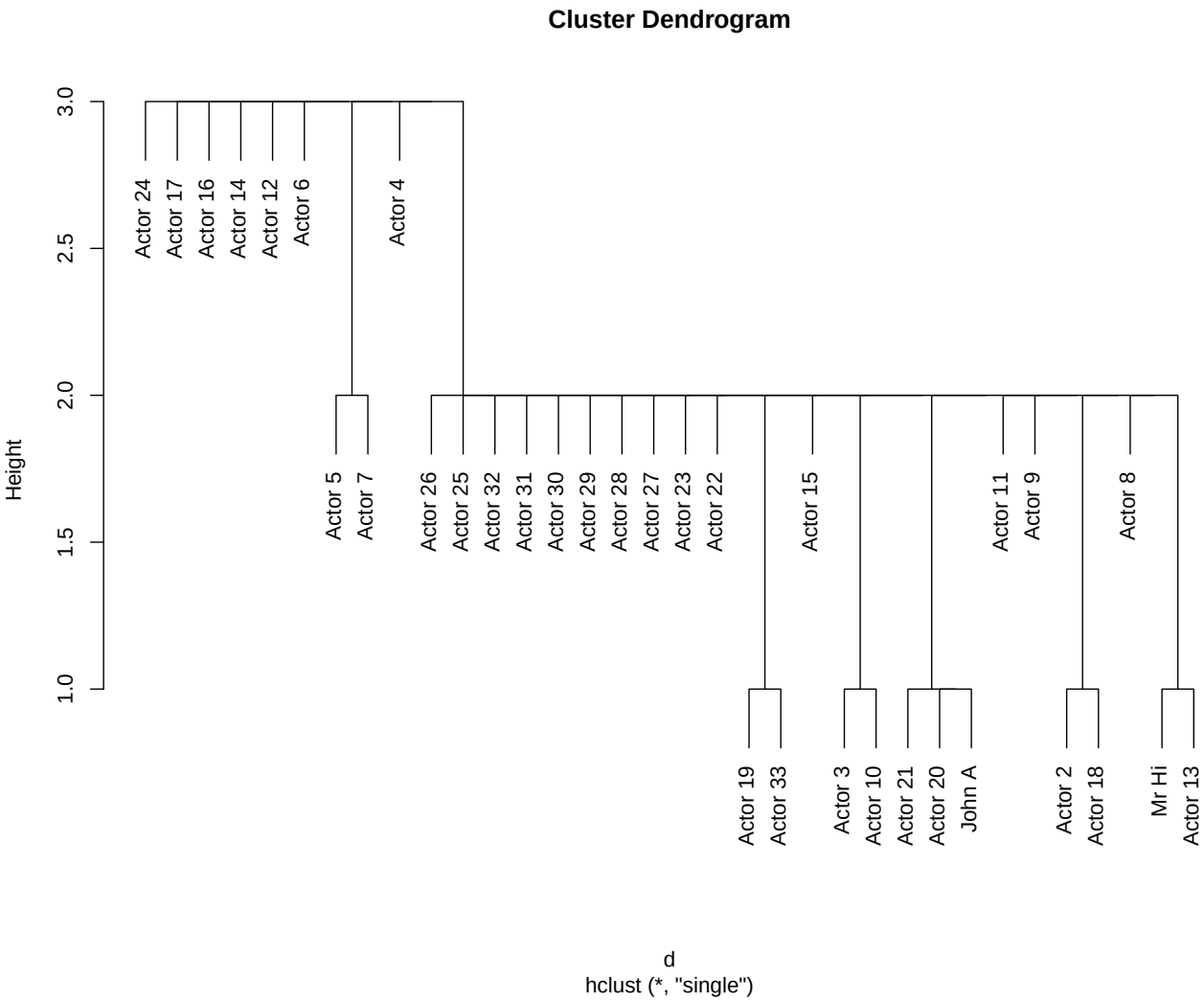
The modularity using k=5 is worse compared to k=2.

K = 6

```
#> [1] 0.315664
```

This one is also worse than k=2.

Using method='single'



Modularity of clustering

K = 2

```
#> [1] -0.002054569
```

K = 3

```
#> [1] -0.003205128
```

K = 4

```
#> [1] -0.004684418
```

K = 5

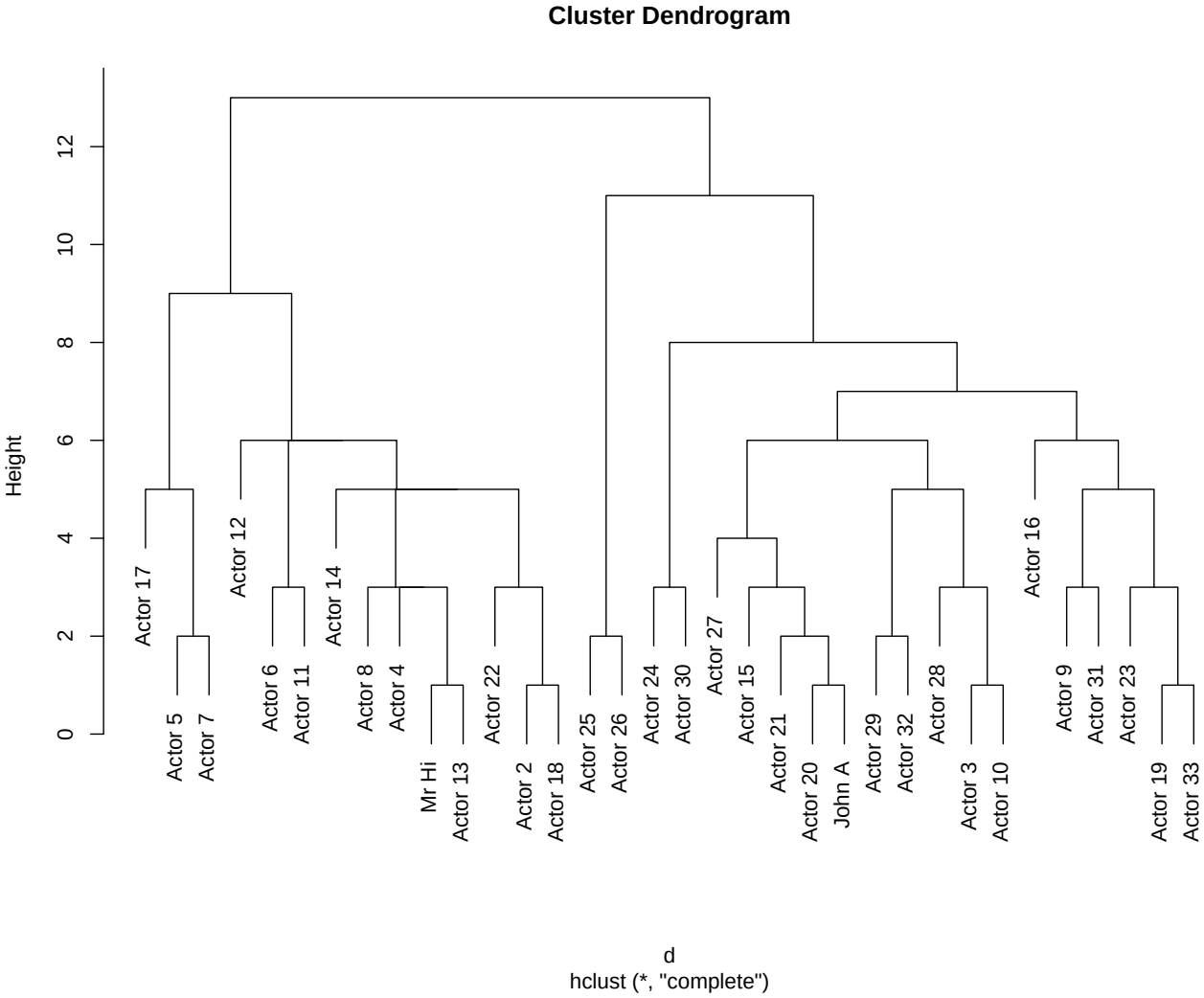
```
#> [1] -0.01043721
```

K = 6

```
#> [1] -0.01166995
```

For all k=2 up to k=6, the models using the *single* method do not improve upon the methods evaluated in class, and also do not improve vs utilizing *ward.D*.

Using method='complete'



Modularity of clustering

K = 2

```
#> [1] 0.3404832
```

The result for this version also improves upon edge betweenness community detection reasonably, but still less good than using k=4 with *ward.D*.

K = 3

```
#> [1] 0.3321006
```

Same as the previous result, improving upon the result from edge betweenness, but still less good than that of using k=4 with *ward.D*, and along with the others, less effective than other methods which optimize modularity.

K = 4

```
#> [1] 0.3079389
```

K = 5

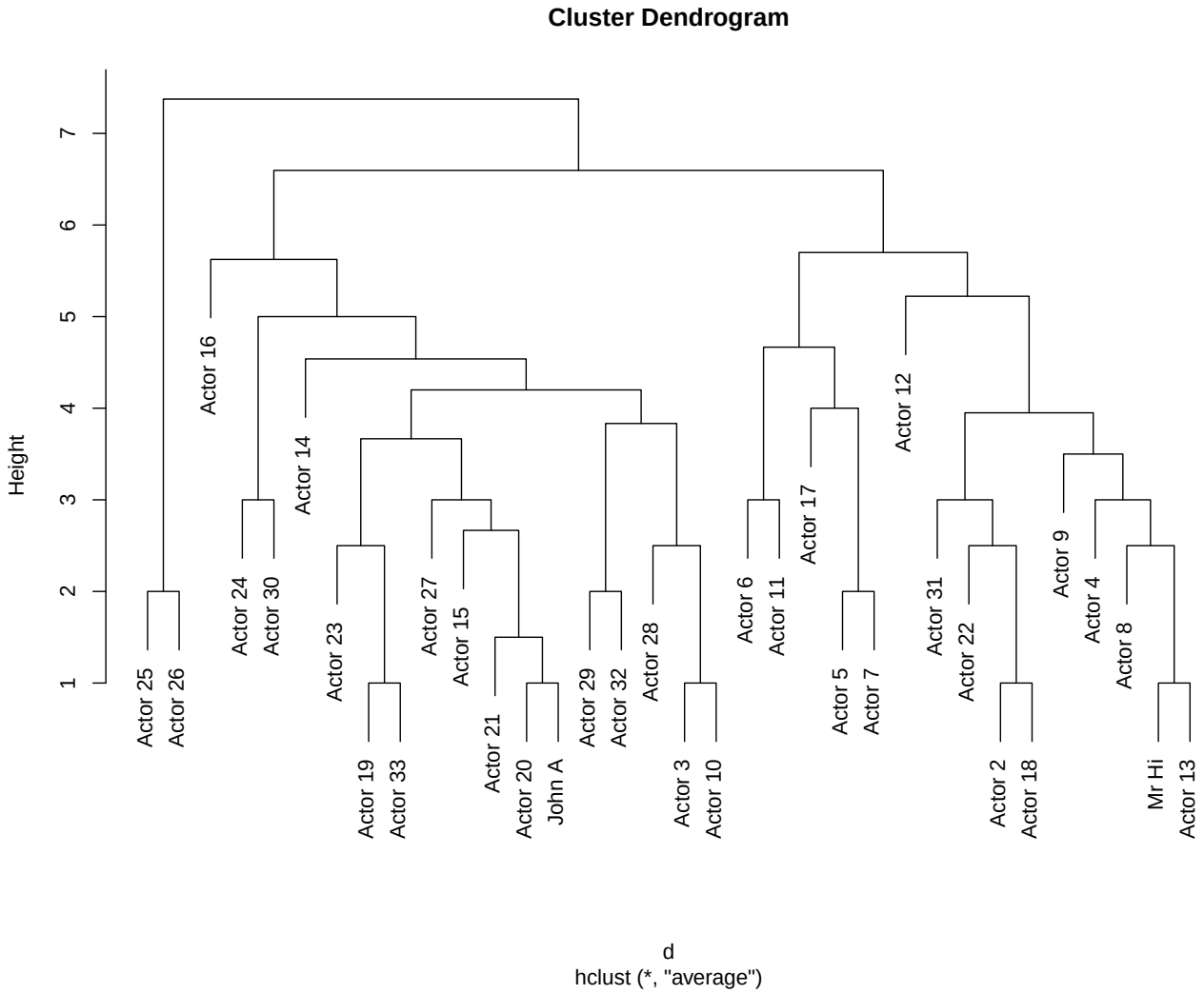
```
#> [1] 0.2887081
```

K = 6

```
#> [1] 0.2608481
```

We can see that the modularity decays for this method as increases as k increases.

Using method='average'



Modularity of clustering

K = 2

```
#> [1] 0.02268245
```

K = 3

```
#> [1] 0.2873932
```

K = 4

```
#> [1] 0.3031723
```

K = 5

```
#> [1] 0.2908448
```

K = 6

```
#> [1] 0.2821335
```

All the results using the *average* method perform worse than those of *ward.D* and the methods previously seen in class.

However, there is a noticeable difference between using k=2 and k>2, returning a significantly worse result for k=2 than the rest.