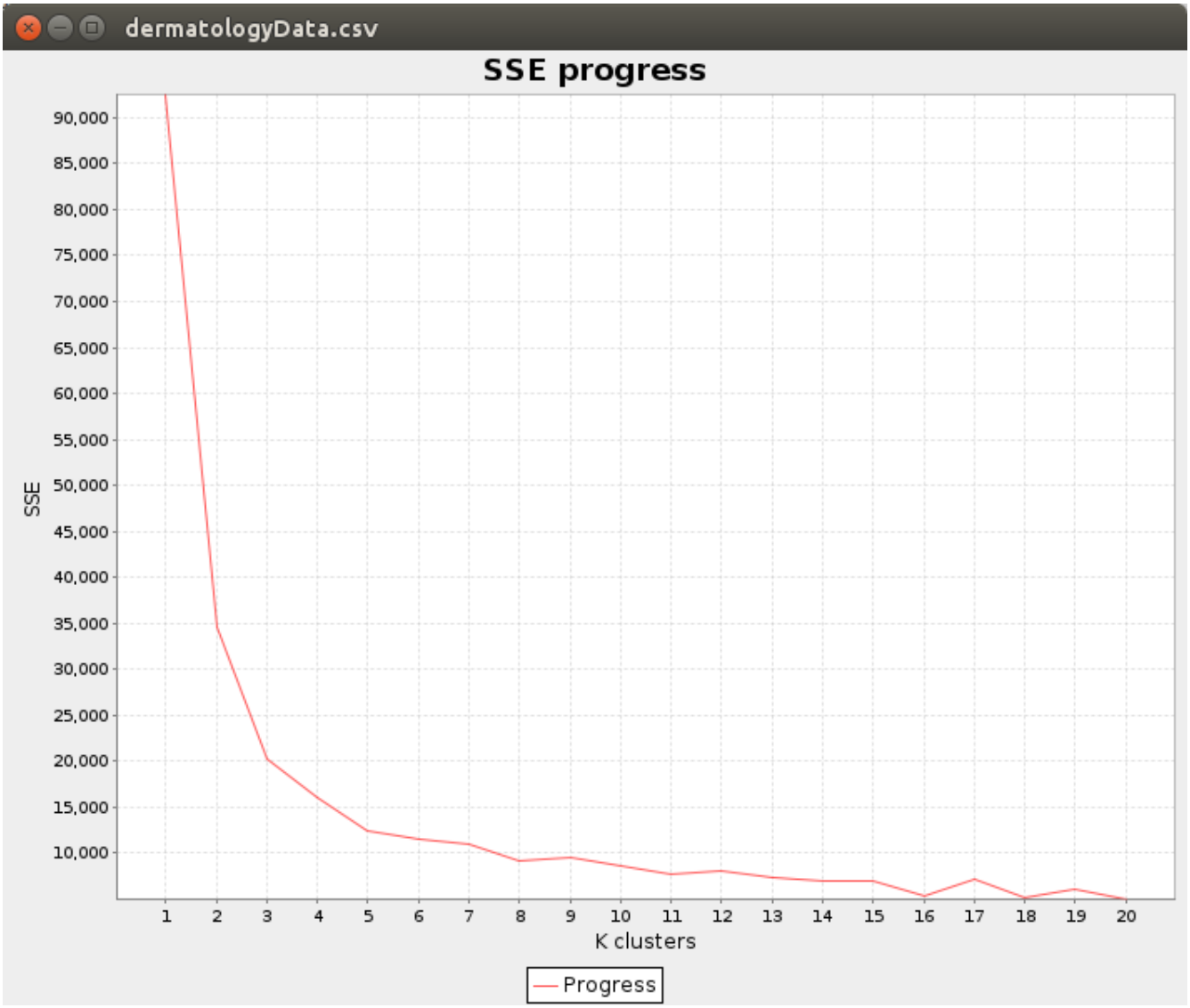
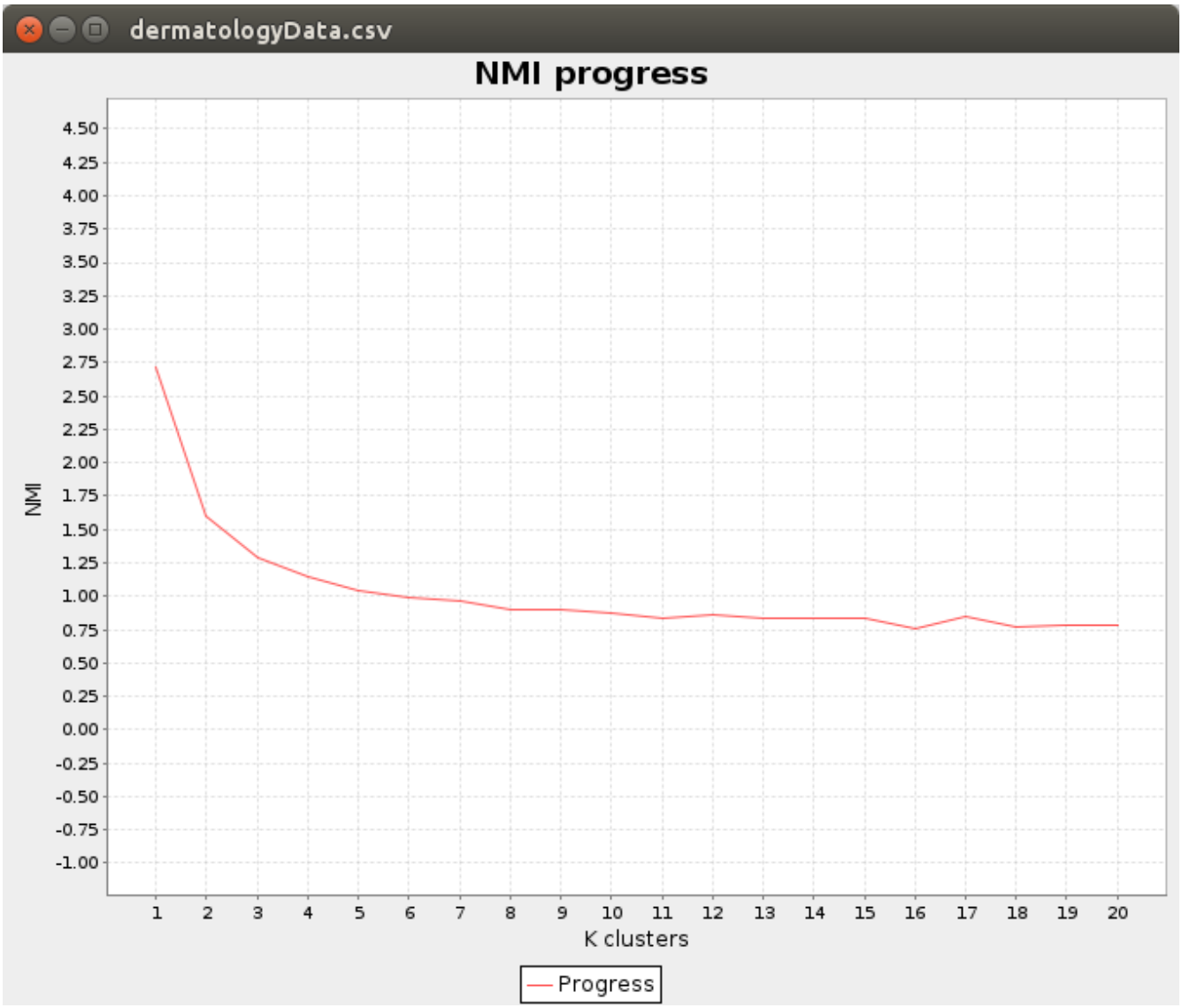
Shreyas Mahimkar

3) Kmeans Clustering::

SSE vs k and NMI vs k plots for each dataset ::

1) Dermatology Data



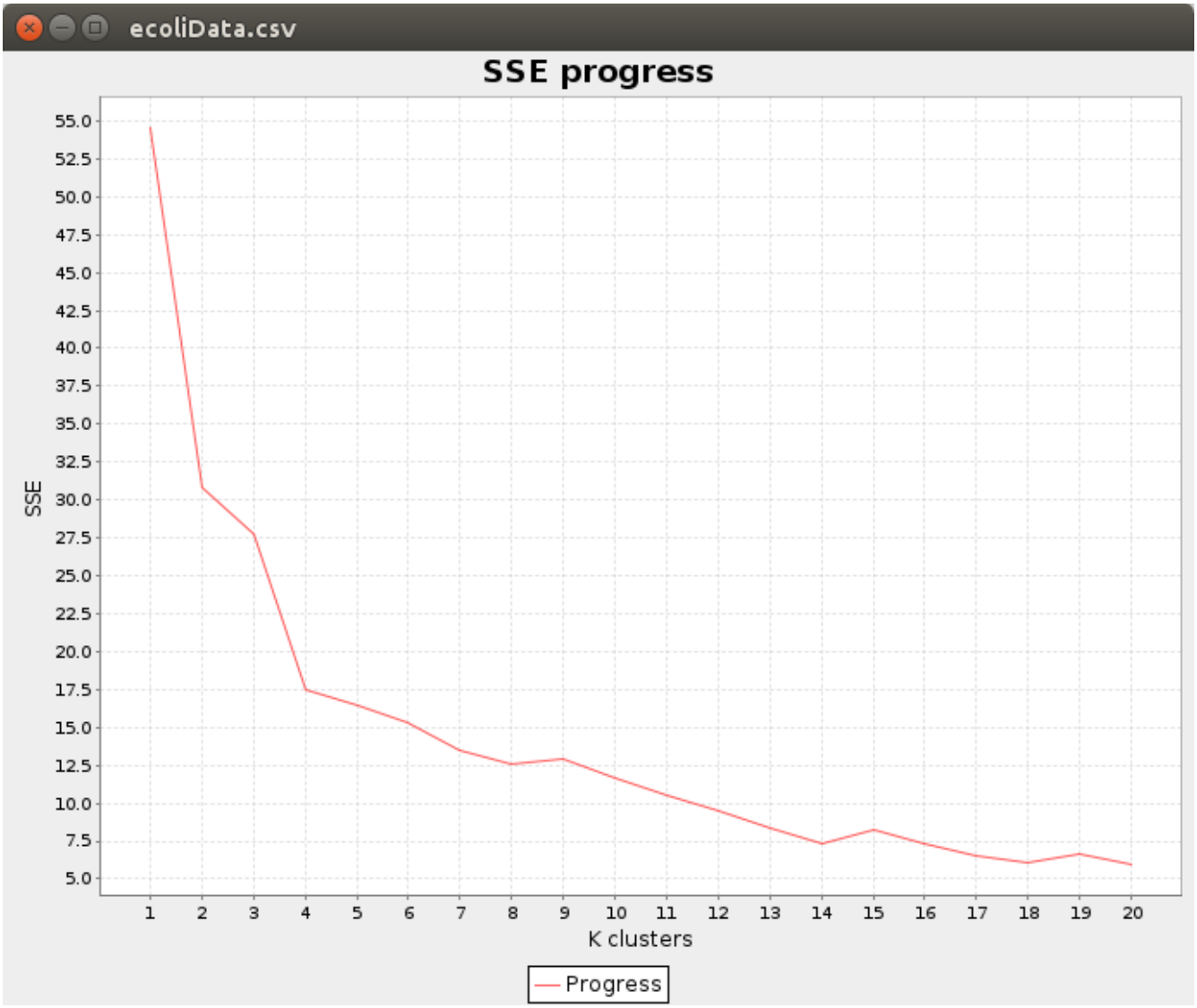


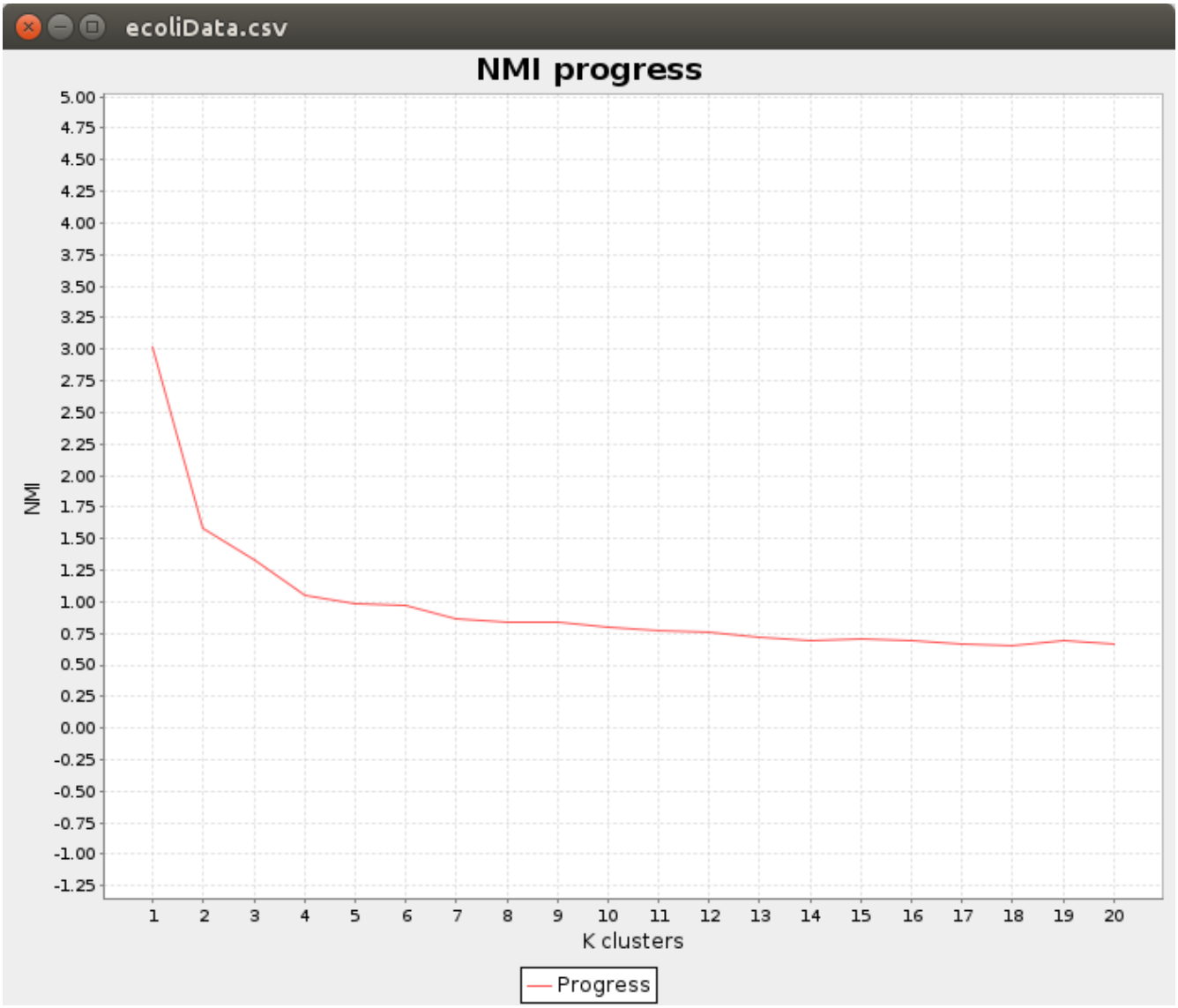
For k = 6

Optimal k base on the above plot is 5 (Both NMI and SSE).

`NMI:0.9896762049031579 sse:11449.901306748103 iteration:38`

2) Ecoli Data


ecoliData.csv
SSE progress
SSE
55.0
52.5
50.0
47.5
45.0
42.5
40.0
37.5
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
K clusters
Progress

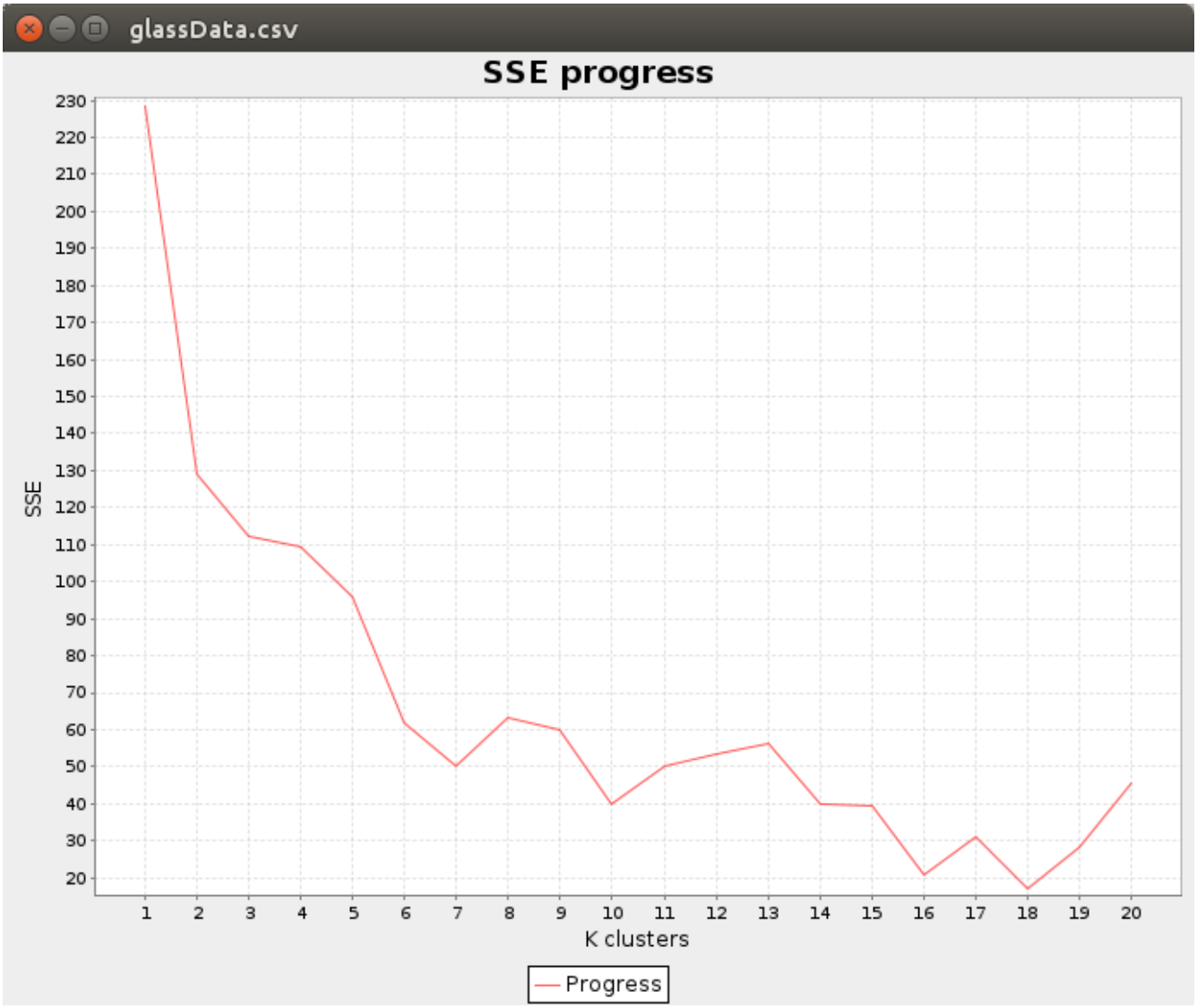


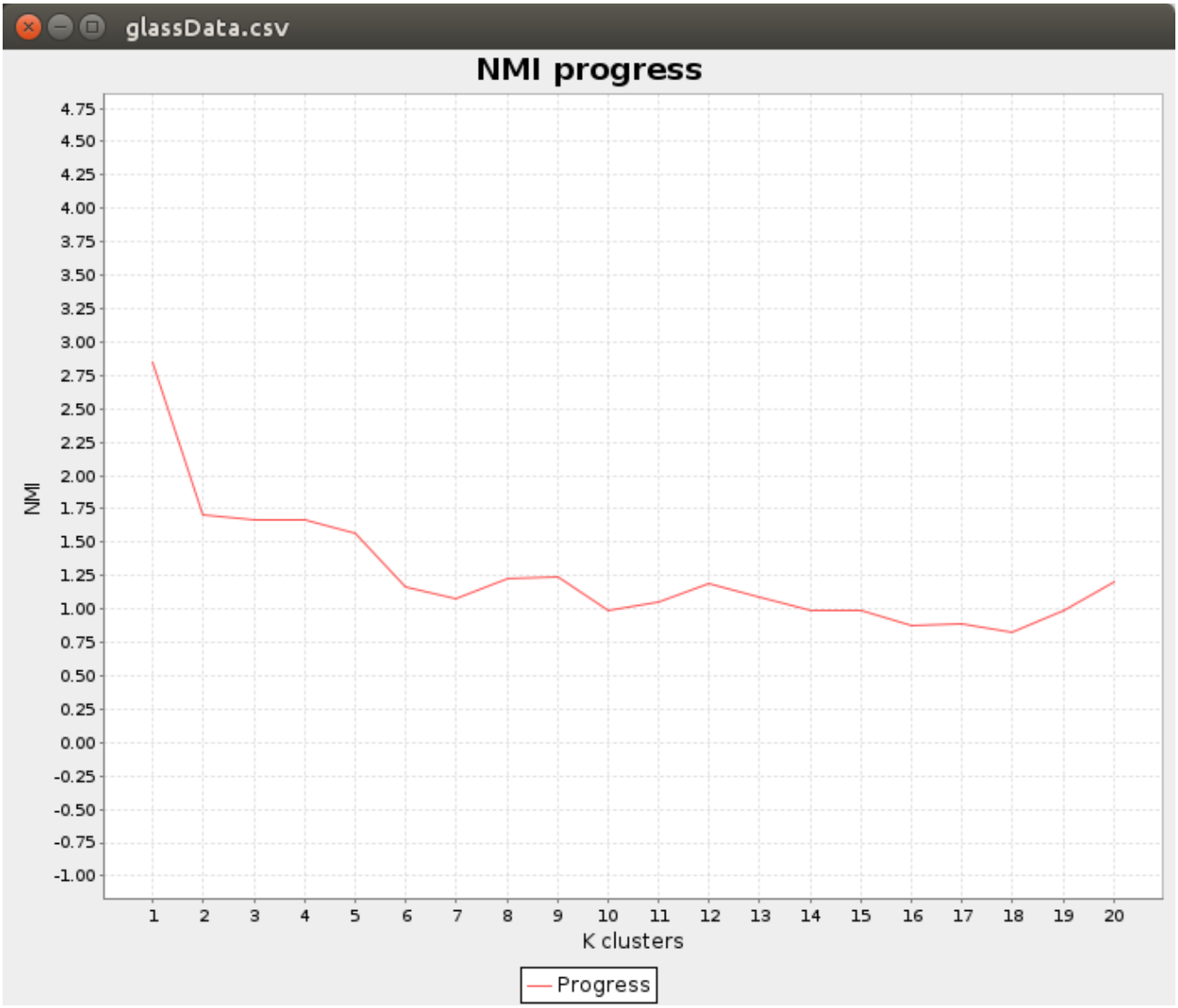
Optimal K base on SSE and NMI :: between 4 and 8. As we can see after k = 4 the exponential drop (knee) is not exponential anymore.

K = 5 – Class number

```
NMI:0.9696986830920746 sse:15.3123443054913 iteration:17
```

3) GlassData



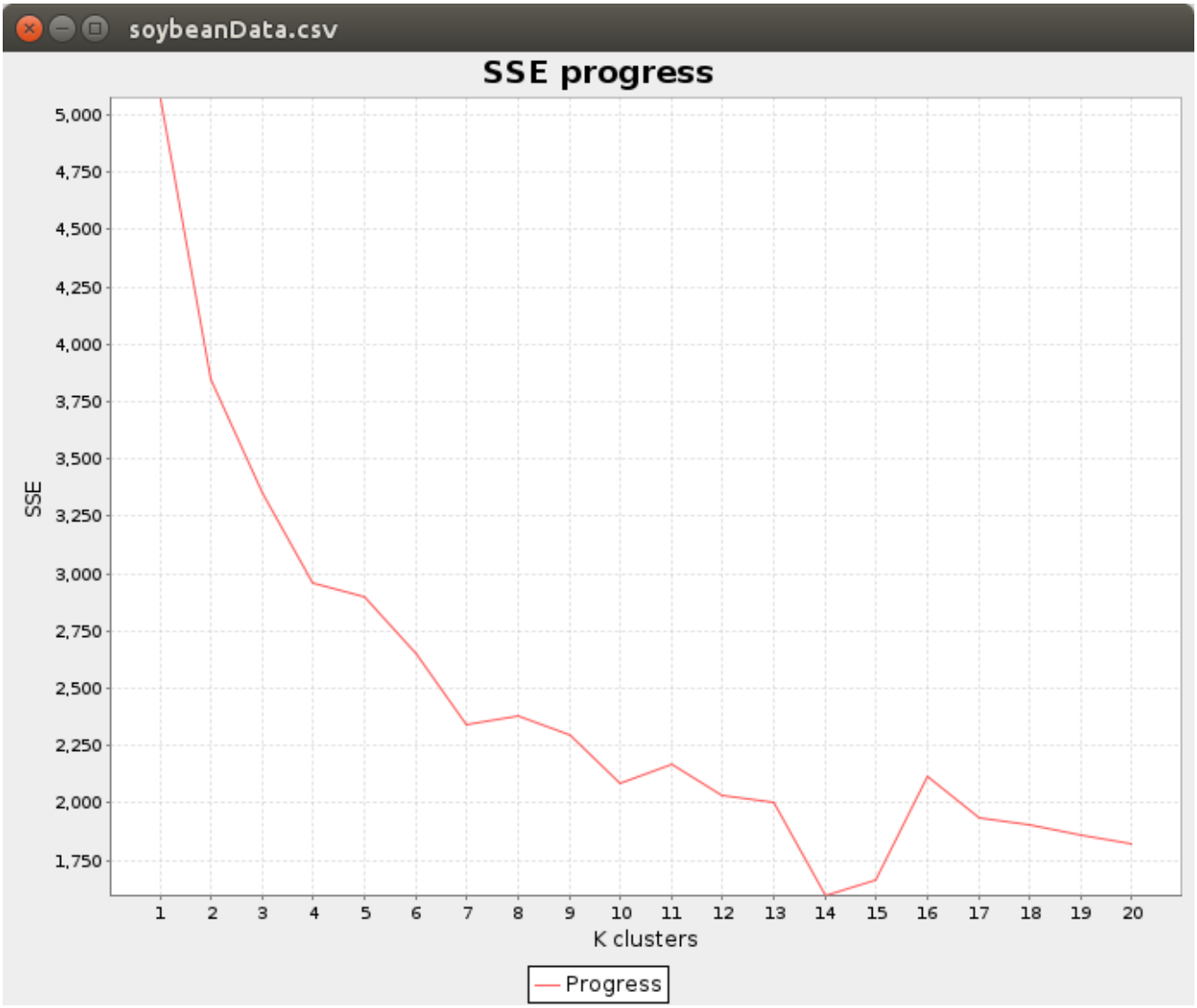


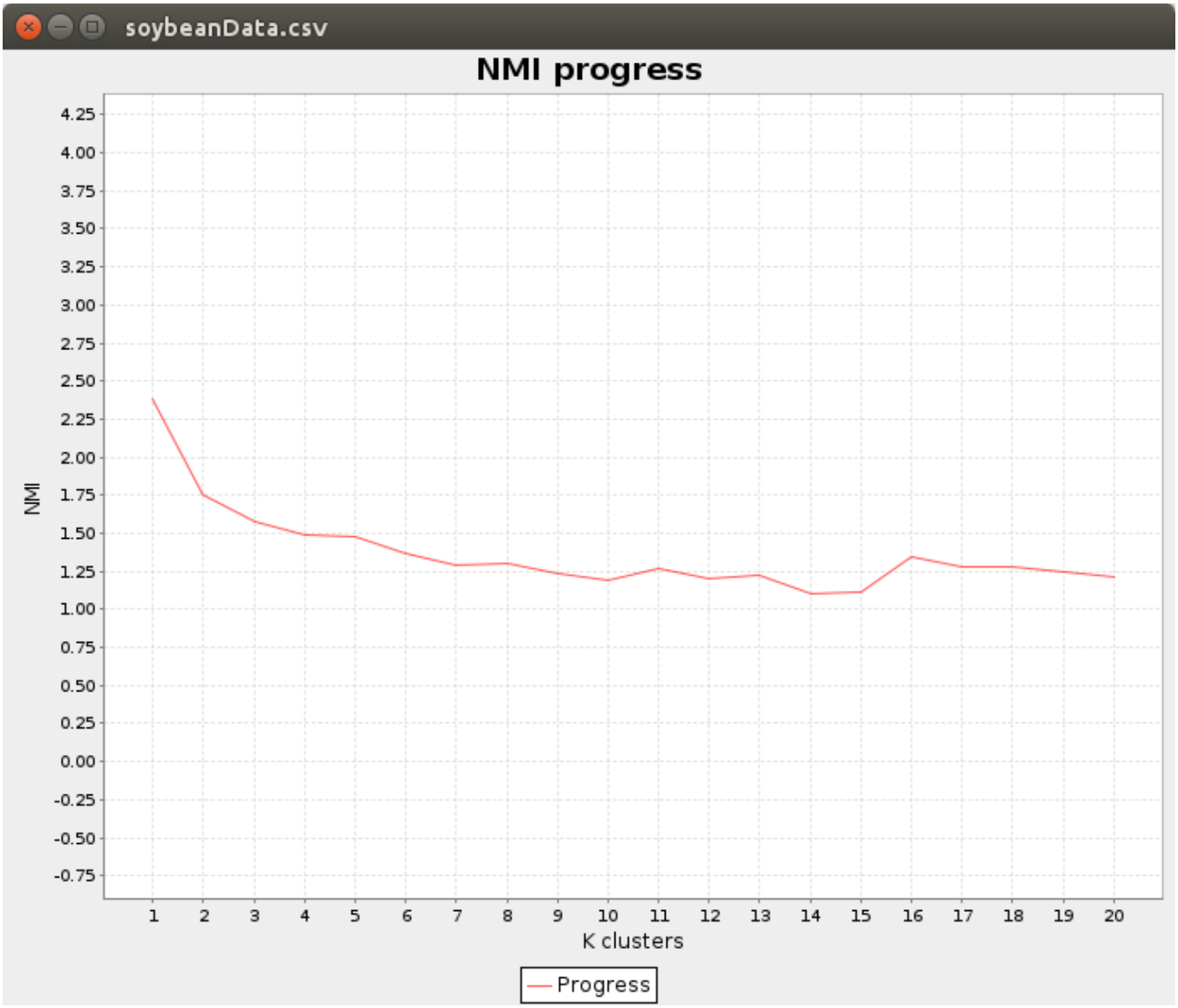
Based on SSE : Best k = 7. Exponential drop stops at 7.

Based on NMI : k=2.

```
k = 6
NMI:1.1622090104655665 sse:61.81117417861708 iteration:16
```

4)SoyBean Data



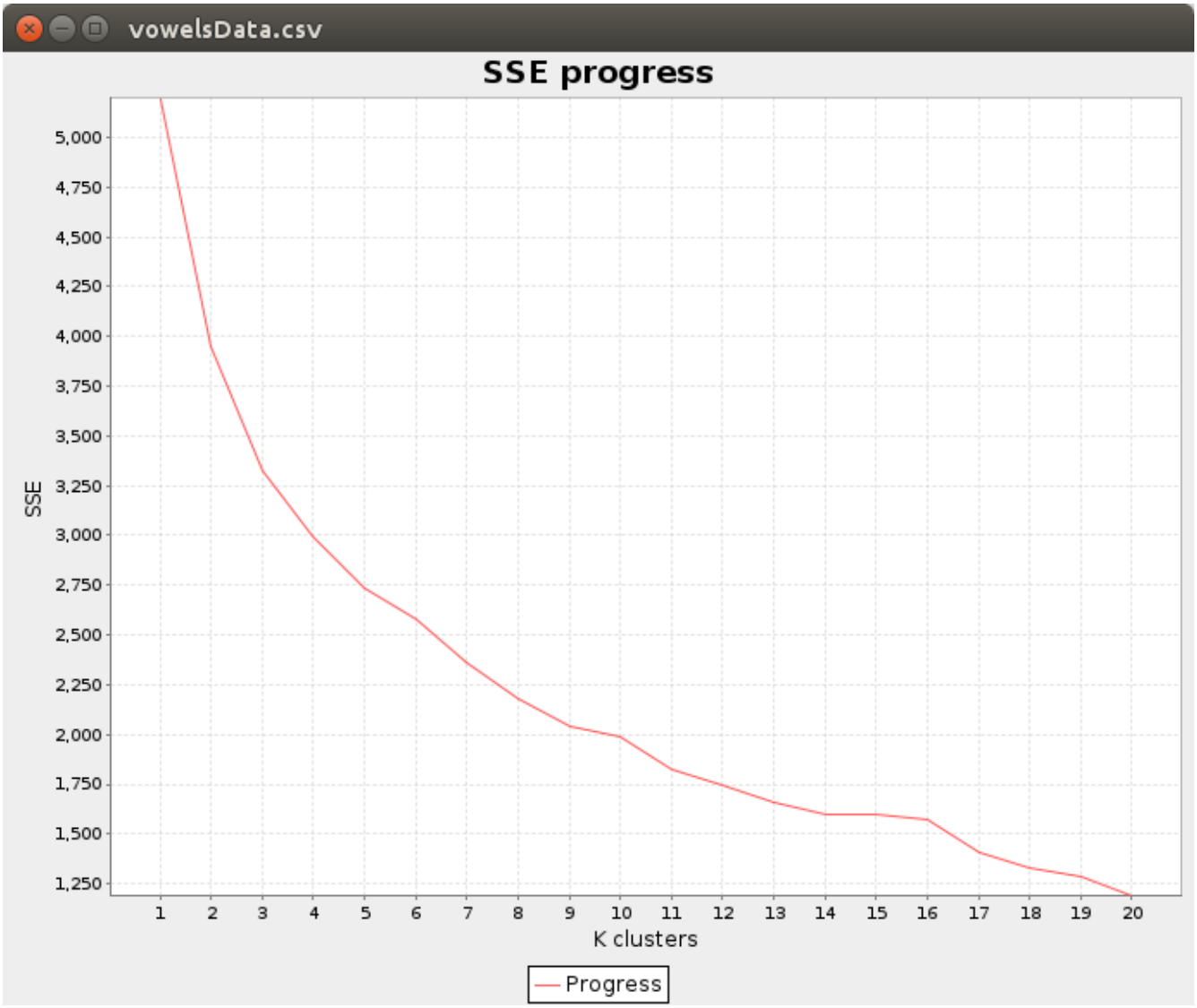


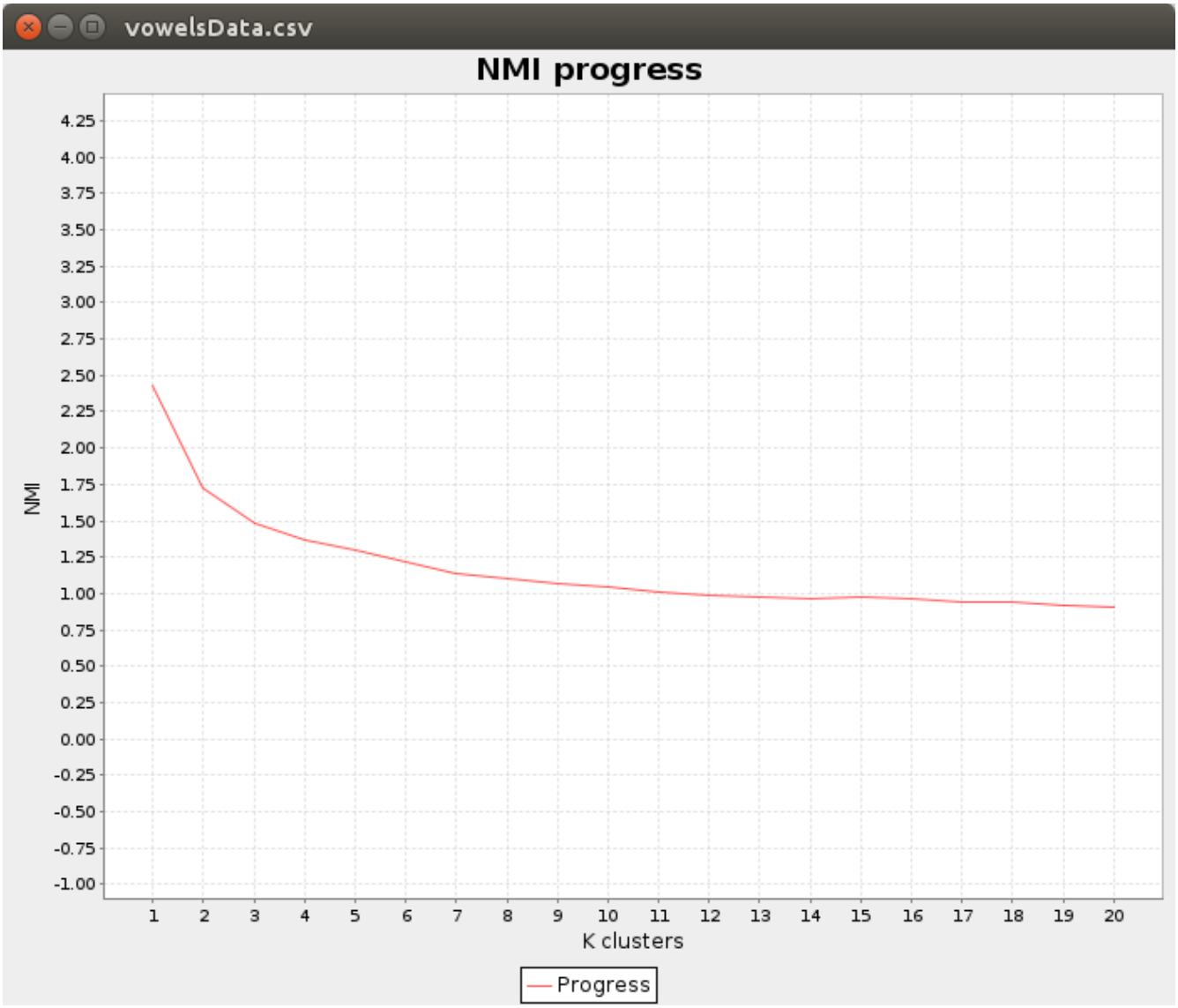
k = 14 according to SSE:

```
k = 15

NMI:1.1184126663796718 sse:1665.668453373021 iteration:24
```

5) Vowels Data

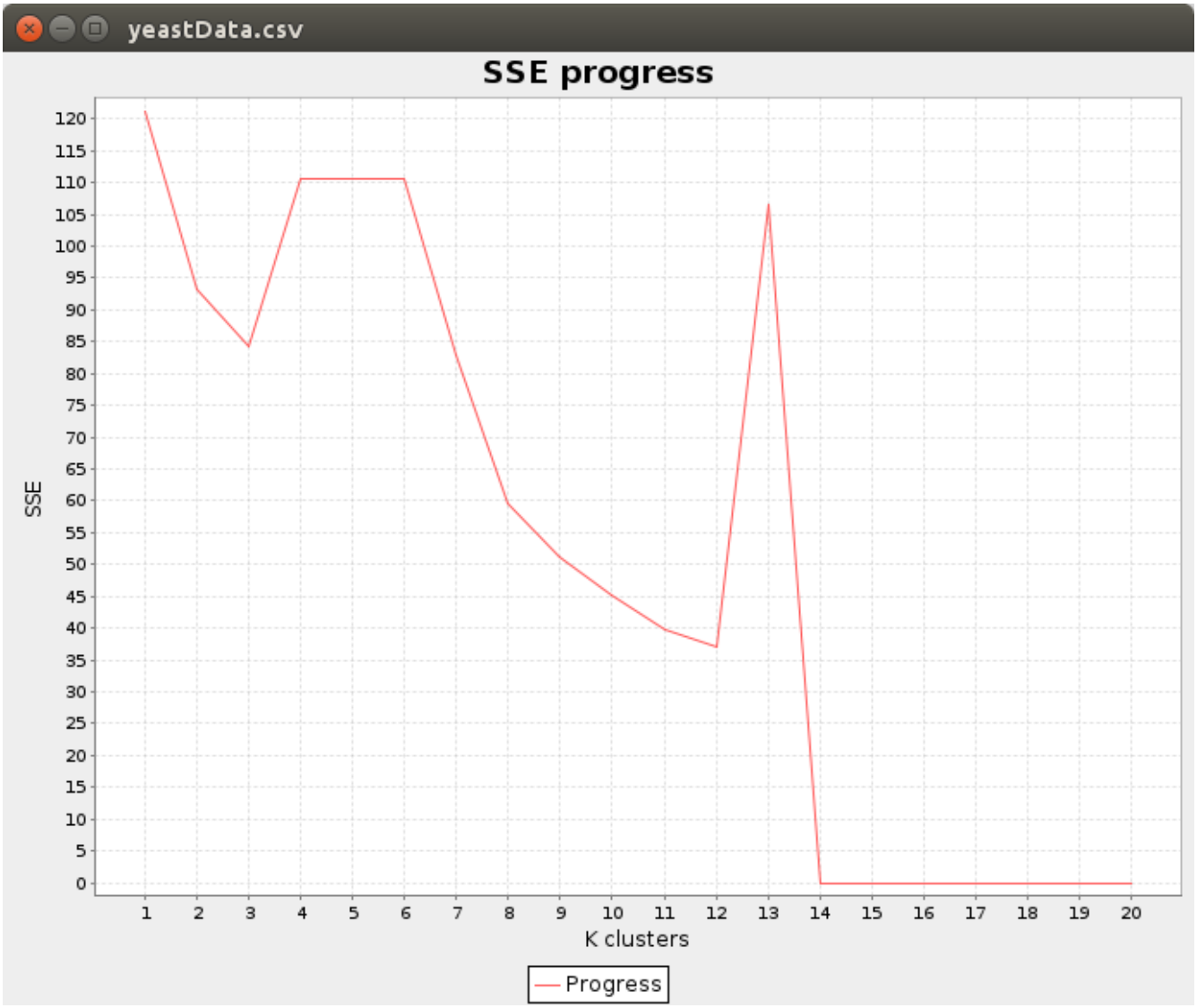


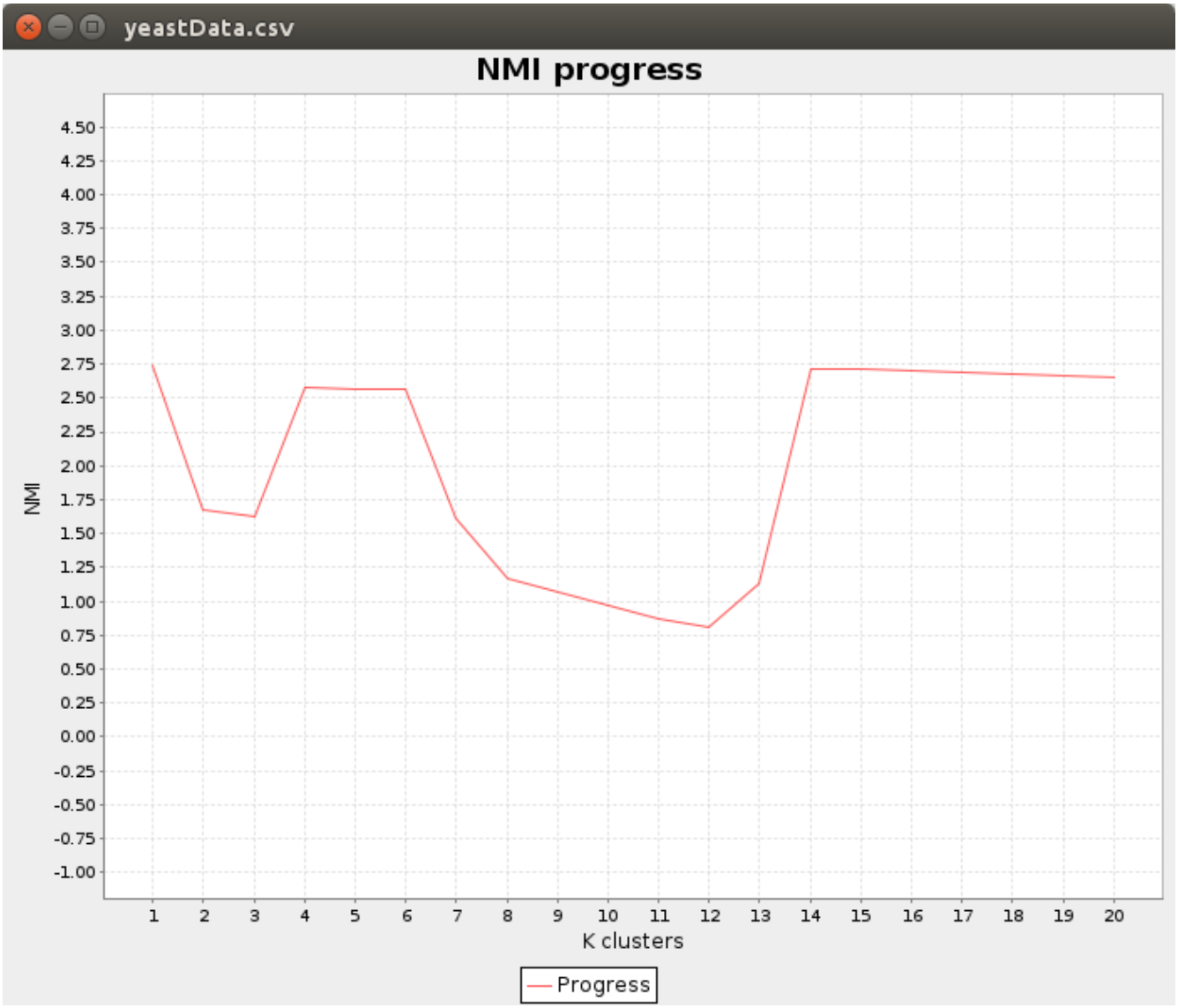


k = 11

NMI:1.0118774989164114 sse:1825.882806346687 iteration:64

6) Yeast Data





k = 9

NMI:1.0730482121277236 sse:51.03846248432911 iteration:46

Based on only Kmeans Algorithm

Datasets	Optimal k based on SSE	Optimal k based on NMI	Num classes	NMI value for k = numClasses	SSE value for k= numClasses
Dermatology	4	4	5	0.96	15.31
Ecoli	5	5	6	0.98	11449.90
Glass	7	2	6	1.16	61.81
SoyBean	14	5	15	1.11	1665.66
Vowels	10	4	11	1.01	1825.88
Yeast	3	2	9	1.07	51.03

Extra Credit for initializing cluster centroids.

Initiate cluster centroids such that if k = 2, select the two centroids in the dataset which are farthest from each other.
http://cs.nyu.edu/courses/spring11/G22.2580-001/lec5.html

Furthest Point First (FPF) Algorithm

[Geraci et al., 2006]

```
{ V := random point in S;
  CENTERS := { V };
  for (U in S) {
    LEADER[U] := V;
    D[U] = d(U,V);
   }
  for (I = 1 to K) {
     X := the point in S-CENTERS with maximal value of D[X];
     C := emptyset;
     for (U in S-CENTERS)
        if (d(U,X) < D[U]) add X to C;
     X := medoid(C,X);
     add X to CENTERS;
     for (U in S-CENTERS) {
       if (d(U,X) < D[U]) {
         LEADER[U] := X;
         D[U] := d(U,X)
        }
       }
    }
return(CENTERS)}
```