Example of a Pangenome Statistical Analysis with pasaR

Stelios Batziakas (ampatziakas@gmail.com)

22 August, 2017

Introduction

In this vignette the package functions will be showcased on a dataset containing seven (81) strains of the following bacteria : twelve (12) of Streptococcus pneumoniae, thirteen (13) of Streptococcus Pyogenes, thirty nine (39) of Bacillus cereus and seventeen (17) of Bacillus thuringiensis.

Genome summary

A summary with **panm_summary()** of the genome participation to the clusters allows for an initial overview of the data, something not so useful in cases of bigger datasets.

	Genomes	Clusters
1	1	93296
2	2	23071
3	3	12247
4	4	5615
5	5	3386
6	6	2398
7	7	1681
8	8	1380
9	9	1029
10	10	855
11	11	728
12	12	693
13	13	511
14	14	360
15	15	284
16	16	249
17	17	239
18	18	210
19	19	189
20	20	142
21	21	123
22	22	106
23	23	81
24	24	61
25	25	71
26	26	48
27	27	46
28	28	43
29	29	34
30	30	40

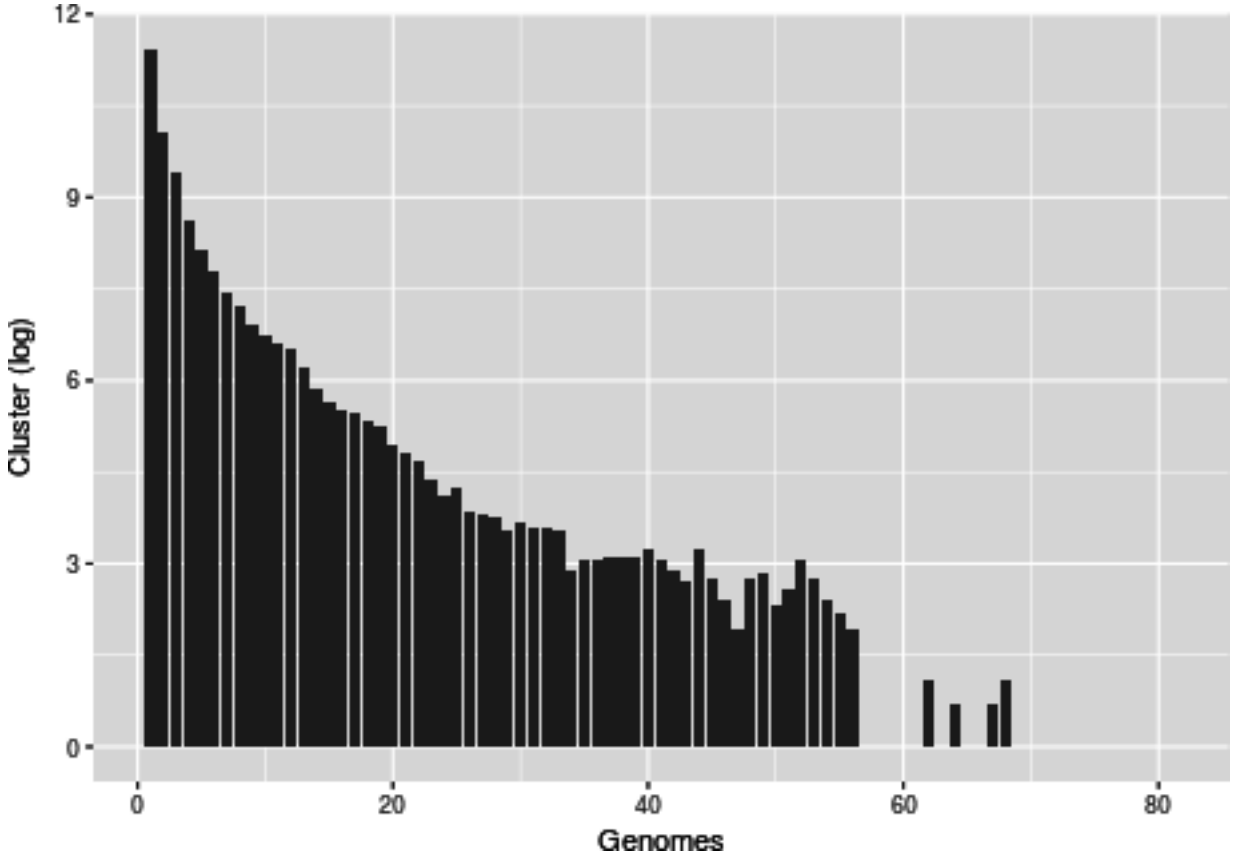
	Genomes	Clusters
31	31	36
32	32	36
33	33	34
34	34	18
35	35	21
36	36	21
37	37	22
38	38	22
39	39	22
40	40	25
41	41	21
42	42	18
43	43	15
44	44	25
45	45	16
46	46	11
47	47	7
48	48	16
49	49	17
50	50	10
51	51	13
52	52	21
53	53	16
54	54	11
55	55	9
56	56	7
57	57	1
58	58	1
60	60	1
61	61	1
62	62	3
64	64	2
65	65	1
67	67	2
68	68	3

Exploring the data

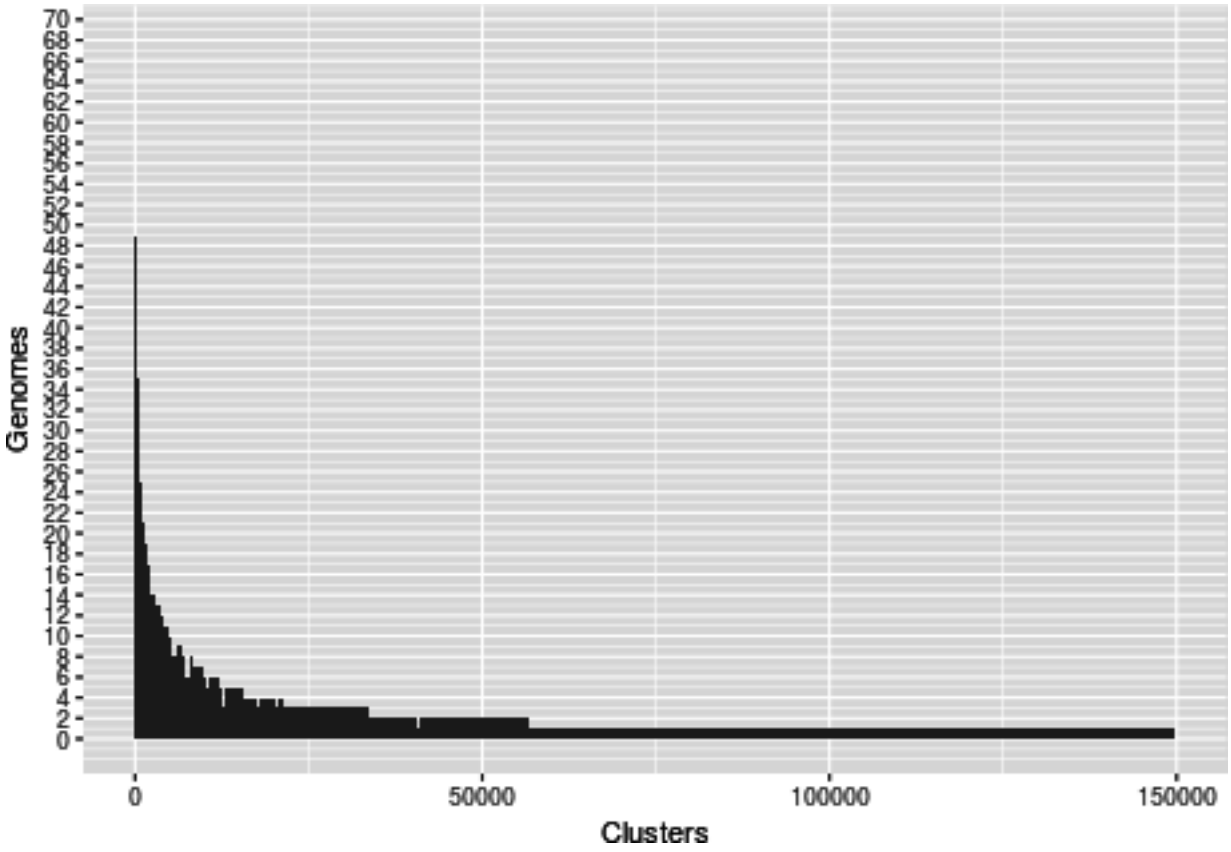
Another way of exploring the genome data is through the use of various types of plots:

- User can plot Genome - Cluster participation, with **pm_plot()**

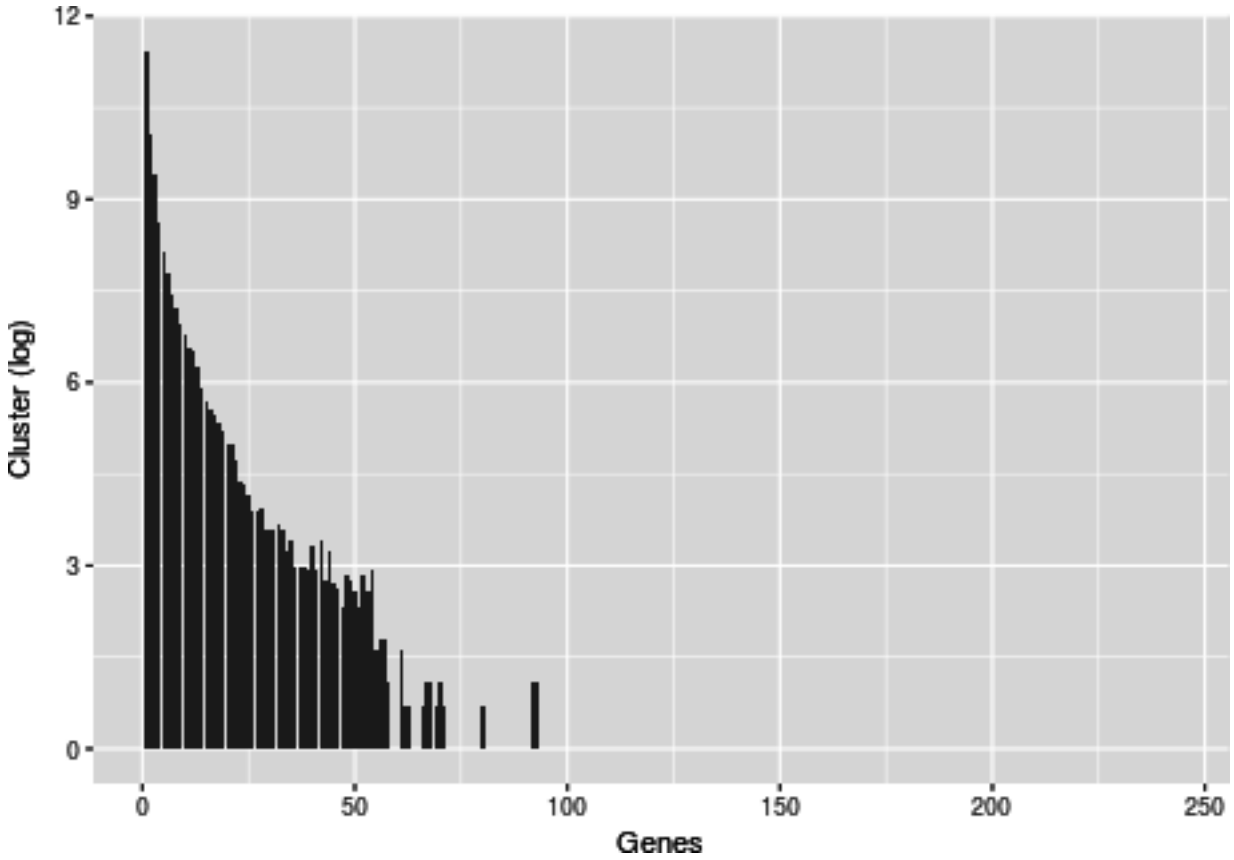
```
## Warning: Removed 16 rows containing missing values (geom_bar).
```



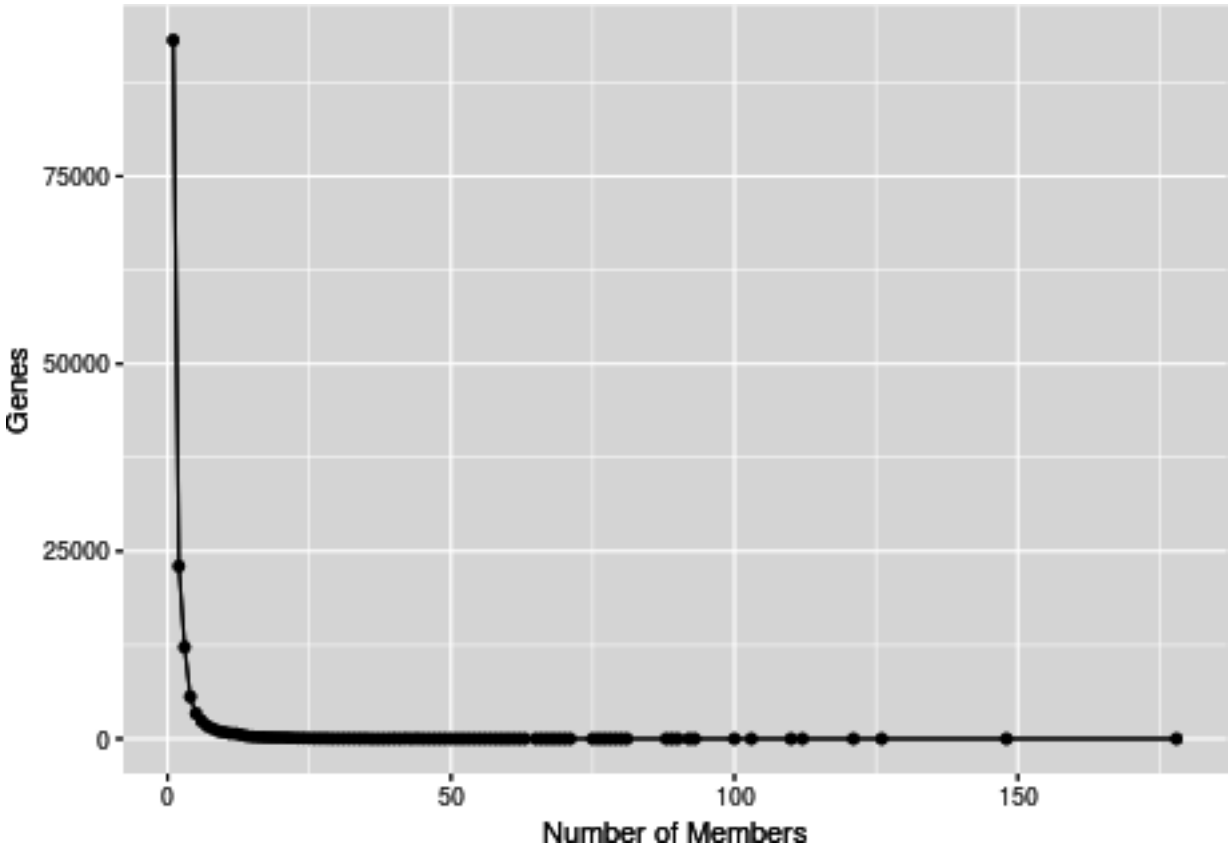
- Cluster participation - Genomes, with **cp_plot()**:



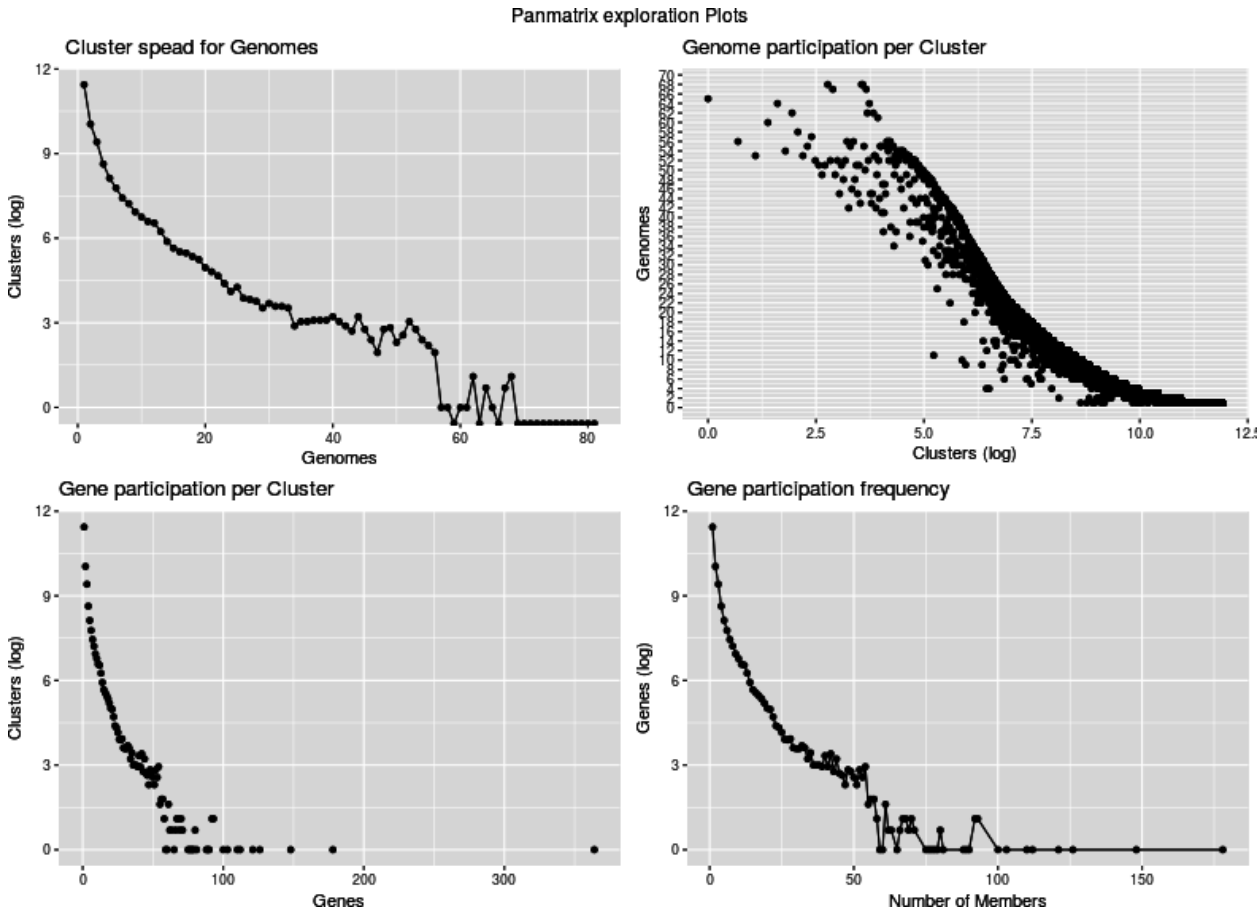
- Genes - Cluster participation, with **gp_plot()**:



- Genes - Cluster participation frequency, with **mg_plot()**:



These plots can be be conviniently grouped by **grid_plot()**:



Heaps Law fitting

A power law can be fitted to the data with **pm_heaps()** to decribe the fit of the data , two estimated parameters, intercept and decay parameter a. If a<1, like this case then the pangenome is considered to be open, meaning that at any point adding new strains to our data will add new genes.

```
##    Intercept        alpha
## 4179.0305225    0.2492299
```

This is an optimized version of the **heaps()** function from package micropan.

Binomial mixture fitting

We can estimate the pangenome size and the pangenome core size through binomial mixtures as proposed by Snipen, Almoy & Ussery[1] and implemented in package **micropan**. One must search the model with the optimal number of components. This is determined comparing the Bayesian Information criterion scores of the models and chosing the one with the smaller score. In our case the optimal pangenome composition seems to be nine components.

```
## $BIC.table
##               Core.size Pan.size      BIC
```

[1]Snipen, L., Almoy, T., Ussery, D.W. (2009). Microbial comparative pan-genomics using binomial mixture models. BMC Genomics, 10:385

```
## 5 components          0   247436 450917.8
## 6 components          0   272479 440018.5
## 7 components          0   309737 443035.0
## 8 components          0   269465 438248.0
## 9 components          0   282357 435058.2
## 10 components         0   262235 439882.8
##
## $Mix.list
## $Mix.list[[1]]
##                     Comp_1     Comp_2      Comp_3       Comp_4       Comp_5
## Detection.prob 0.01029749 0.10029314 0.495694243 9.824501e-01 1.000000e+00
## Mixing.prop    0.91327383 0.08349356 0.003232586 2.058710e-08 1.303819e-10
##
## $Mix.list[[2]]
##                      Comp_1     Comp_2      Comp_3      Comp_4       Comp_5
## Detection.prob 0.008651804 0.07545903 0.280147098 0.728566481 9.975195e-01
## Mixing.prop    0.910455749 0.07967613 0.008847991 0.001020112 2.858171e-10
##                      Comp_6
## Detection.prob 1.000000e+00
## Mixing.prop    1.479747e-08
##
## $Mix.list[[3]]
##                      Comp_1     Comp_2     Comp_3       Comp_4       Comp_5
## Detection.prob 0.006112562 0.04455264 0.19535028 0.7555908504 8.866967e-01
## Mixing.prop    0.843261136 0.13763919 0.01823922 0.0008604543 9.624409e-11
##                      Comp_6       Comp_7
## Detection.prob 9.373485e-01 1.000000e+00
## Mixing.prop    1.689292e-09 2.001045e-09
##
## $Mix.list[[4]]
##                      Comp_1     Comp_2      Comp_3    Comp_4       Comp_5
## Detection.prob 0.009138688 0.08289578 0.275626996 0.5907757 7.940315e-01
## Mixing.prop    0.934672985 0.05523322 0.008092013 0.0019070 9.191325e-05
##                      Comp_6       Comp_7       Comp_8
## Detection.prob 8.023198e-01 8.750605e-01 1.000000e+00
## Mixing.prop    1.252778e-10 2.867320e-06 1.366635e-09
##
## $Mix.list[[5]]
##                     Comp_1     Comp_2      Comp_3      Comp_4       Comp_5
## Detection.prob 0.00826765 0.07222532 0.238393225 0.469551522 0.6578093929
## Mixing.prop    0.91994231 0.06976095 0.008389999 0.001377228 0.0004400223
##                      Comp_6       Comp_7       Comp_8       Comp_9
## Detection.prob 8.014819e-01 8.584994e-01 9.540360e-01 1.000000e+00
## Mixing.prop    8.864611e-05 1.161816e-07 4.843358e-08 6.823578e-07
##
## $Mix.list[[6]]
##                      Comp_1     Comp_2     Comp_3      Comp_4       Comp_5
## Detection.prob 0.009357372 0.08504484 0.31172936 0.613885571 7.730922e-01
## Mixing.prop    0.918717951 0.07333134 0.00658785 0.001263963 6.423900e-07
##                      Comp_6       Comp_7       Comp_8       Comp_9
## Detection.prob 8.224842e-01 8.313253e-01 9.779939e-01 9.998897e-01
## Mixing.prop    9.822721e-05 1.441558e-09 4.094539e-11 2.041044e-08
##                     Comp_10
## Detection.prob 1.000000e+00
```

```
## Mixing.prop    2.831901e-09
```

Chao population estimator

Another way to estimate the pangenome size using the Chao lower bound estimator. The unbiased version of the estimator is provided, so results will vary in comparison with the results of function **chao()** from **micropan** .

```
## Estimated pangenome size       Estimator Variance    CI (95%) -Lower Bound
##                 338349.0                3256678.0                 156401.2
##    CI (95%)- Upper Bound
##                5475997.2
```

Fluidity with sampling

Another measure of interest is fluidity takes values in [0,1], with 1 denoting no common genes. Function **pm_fluidity()** computes fluidity with sampling, as implemented on package micropan, optimized for speed.

Mean Fluidity:
0.911768580110044
Standard Deviation:
0.100383914259587

Fluidity without sampling

The function **pm_fluidity_all()** computes the exact value of fluidity, the standard deviation, the values for every pair and a mean value for each genome paired to all other genomes.

Mean Fluidity:
0.916214549806488
Standard Deviation:
0.109222533362184

Genome_1	Fluidity
12	0.9672711
35	0.9595129
28	0.9480181
63	0.9465690
16	0.9455543
36	0.9437201
17	0.9435082
64	0.9420878
70	0.9414760
61	0.9412805
73	0.9412223
68	0.9406928
57	0.9405811
76	0.9399116

Genome_1	Fluidity
75	0.9392763
74	0.9391107
66	0.9390526
72	0.9382855
67	0.9381773
21	0.9379885
20	0.9376909
80	0.9373979
59	0.9371620
71	0.9370709
58	0.9366042
81	0.9363416
69	0.9359690
65	0.9358752
78	0.9350251
60	0.9350203
77	0.9349032
62	0.9338044
79	0.9337385
25	0.9310333
32	0.9294763
19	0.9285100
29	0.9281519
30	0.9278813
33	0.9267057
8	0.9162972
49	0.9124936
50	0.9113071
54	0.9099685
45	0.9091068
5	0.9081647
14	0.9073649
23	0.9057996
26	0.9052023
39	0.9040531
38	0.9039971
15	0.9034770
43	0.9022424
48	0.9000201
56	0.8974598
52	0.8968733
46	0.8966392
9	0.8959874
51	0.8952177
6	0.8947326
34	0.8927763
13	0.8926932
31	0.8924778
53	0.8920903
1	0.8915605
3	0.8914830
55	0.8913552

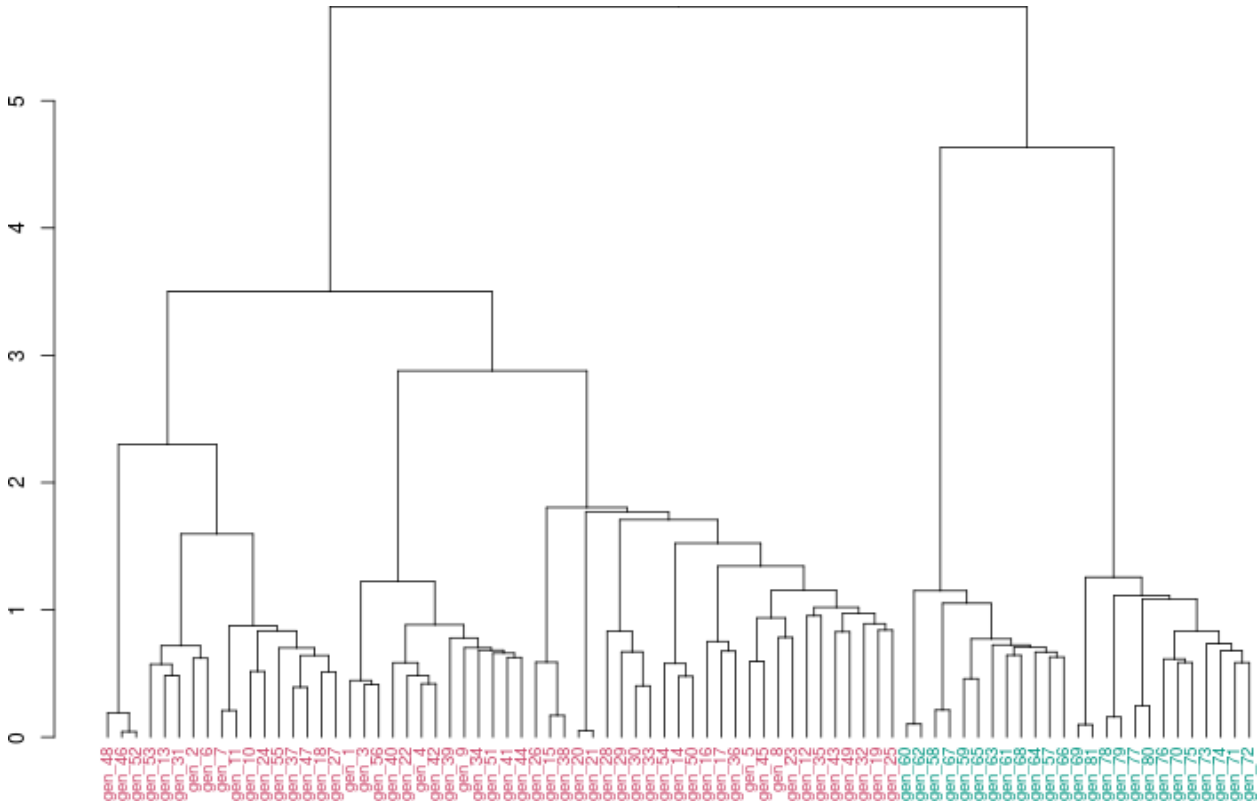
Genome_1	Fluidity
40	0.8912997
2	0.8907398
44	0.8901362
41	0.8894619
42	0.8875548
7	0.8873971
22	0.8866126
4	0.8852710
10	0.8845964
24	0.8843594
47	0.8832681
18	0.8827130
27	0.8815095
37	0.8782722
11	0.8776879

Clustering with fluidity

We can use the fluidity results as a distance measure and **cluster_number()** provides a proposal for the number of clusters found in dataset through hierarchical clustering. Results can be plotted with the help of **dendextend** package.

Clusters	Index	Value
10	Average Silhuette Width	0.24944
2	Gap Statistic	0.96782
3	Dunn	1.00219
7	Entropy	0.66971

Gap statistic proposes 2 clusters:



Dunn index proposed 3 clusters: