Clustering genome

1. Data Understanding and Cleaning
2. Data Preparation
3. Modelling

Data Understanding

In [1]:
```
#import all the necessary libraries

import pandas as pd
import numpy as np
import pandas as pd

# For Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# To Scale our data
from sklearn.preprocessing import scale

# To perform KMeans clustering
from sklearn.cluster import KMeans

# To perform Hierarchical clustering
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree
```

In [2]:
```python
# read the dataset
dat = pd.read_csv('genomics test dataset.csv')
dat.head()
```

Out[2]:

	ind	Marker	Variation
0	ind1	Marker100	AA
1	ind1	Marker101	AA
2	ind1	Marker129	AA
3	ind1	Marker136	AA
4	ind1	Marker187	AA

In [3]:
```python
dat.shape
```

Out[3]:
```
(13237585, 3)
```

In [4]:
```python
#basic data checks
dat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13237585 entries, 0 to 13237584
Data columns (total 3 columns):
ind          object
Marker       object
Variation    object
dtypes: object(3)
memory usage: 303.0+ MB
```

In [5]:
```python
#basic data cleaning checks
dat.isna().sum()
```

Out[5]:
```
ind          0
Marker       0
Variation    0
dtype: int64
```

Transforming data from column C into numeric format, formating data from string format to matrix format

Dummy Variables

The variable `Variation` has levels. We need to convert these levels into integer as well. For this, we will use something called `dummy variables` .

In [6]:
```
# Get the dummy variables for the feature 'Variation' and store it in a new variable - 'status'

status = pd.get_dummies(dat['Variation'])

# Check what the dataset 'status' looks like
status.head()
```

Out[6]:

	AA	AB	AC	AD	AE	AF	AG	AH	AI	BB	...	FH	FI	FJ	GG	GH	HH	II	IJ	JJ	KK
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 51 columns

Now, you don't need 51 columns. You can drop any one column, as Variation can be identified with just the last 50 columns

In [7]:
```python
# droping the first column from status df using 'drop_first = True'
status = pd.get_dummies(dat['Variation'], drop_first = True)

# Add the results to the original housing dataframe
datm = pd.concat([dat, status], axis = 1)

#  head of our dataframe.
datm.head()
```

Out[7]:

	ind	Marker	Variation	AB	AC	AD	AE	AF	AG	AH	...	FH	FI	FJ	GG	GH	HH	II	IJ	JJ	KK
0	ind1	Marker100	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	ind1	Marker101	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	ind1	Marker129	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	ind1	Marker136	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	ind1	Marker187	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 53 columns

In [8]:
```
# Converting marker categorical variable to numeric
from sklearn.preprocessing import LabelEncoder

lb_make = LabelEncoder()
datm["Marker_code"] = lb_make.fit_transform(datm["Marker"])
datm.head(11)
```

Out[8]:

	ind	Marker	Variation	AB	AC	AD	AE	AF	AG	AH	...	FI	FJ	GG	GH	HH	II	IJ	JJ	KK	Marker_code
0	ind1	Marker100	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
1	ind1	Marker101	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	2
2	ind1	Marker129	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	32
3	ind1	Marker136	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	40
4	ind1	Marker187	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	94
5	ind1	Marker188	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	95
6	ind1	Marker210	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	119
7	ind1	Marker211	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	120
8	ind1	Marker211	BB	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	120
9	ind1	Marker212	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	121
10	ind1	Marker215	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	124

11 rows × 54 columns

In [9]:
```
# generating sampled data set
datm = datm.sample(n = 20000 , replace="False")
datm.shape
```

Out[9]:
```
(20000, 54)
```

In [10]:
```
datm.columns
```

Out[10]:
```
Index(['ind', 'Marker', 'Variation', 'AB', 'AC', 'AD', 'AE', 'AF', 'AG', 'AH',
       'AI', 'BB', 'BC', 'BD', 'BE', 'BF', 'BG', 'BH', 'BI', 'BK', 'CC', 'CD',
       'CE', 'CF', 'CG', 'CH', 'CI', 'CK', 'DD', 'DE', 'DF', 'DG', 'DH', 'DI',
       'DK', 'EE', 'EF', 'EG', 'EH', 'EI', 'EJ', 'FF', 'FG', 'FH', 'FI', 'FJ',
       'GG', 'GH', 'HH', 'II', 'IJ', 'JJ', 'KK', 'Marker_code'],
      dtype='object')
```

Clustering

Hopkins Statistics:

The Hopkins statistic, is a statistic which gives a value which indicates the cluster tendency, in other words: how well the data can be clustered.

- If the value is between {0.01, ...,0.3}, the data is regularly spaced.
- If the value is around 0.5, it is random.
- If the value is between {0.7, ..., 0.99}, it has a high tendency to cluster.

In [11]:

```
#Calculating the Hopkins statistic
from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
import numpy as np
from math import isnan

def hopkins(X):
    d = X.shape[1]
    #d = len(vars) # columns
    n = len(X) # rows
    m = int(0.1 * n)
    nbrs = NearestNeighbors(n_neighbors=1).fit(X.values)

    rand_X = sample(range(0, n, 1), m)

    ujd = []
    wjd = []
    for j in range(0, m):
        u_dist, _ = nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),d).reshape(1, -1), 2, return_distance=True)
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1, -1), 2, return_distance=True)
        wjd.append(w_dist[0][1])

    H = sum(ujd) / (sum(ujd) + sum(wjd))
    if isnan(H):
        print(ujd, wjd)
        H = 0

    return H
```

```
In [12]: #Let's check the Hopkins measure
         hopkins(datm.drop(['ind','Marker','Variation'],axis=1))
```

```
Out[12]: 0.986085146829514
```

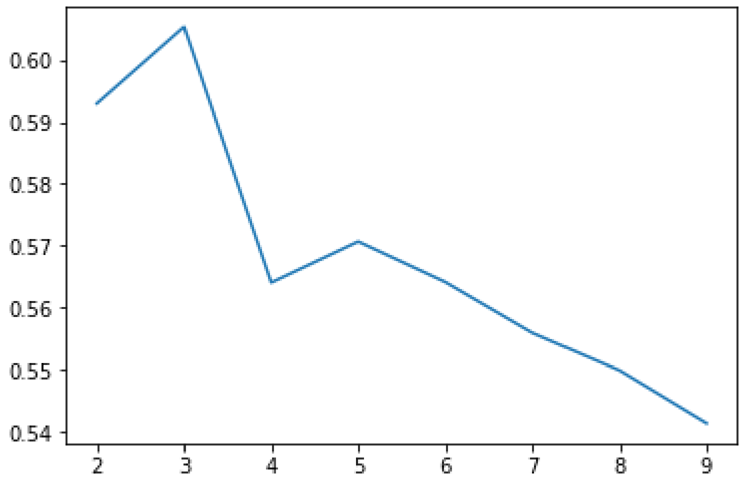
0.98 is a good Hopkins score. Hence the data is suitable for clustering. Preliminary check is now done.

```
In [13]: dat3=datm.drop(['ind','Marker','Variation'],axis=1)
```

K-means Clustering

```
In [14]: #Let's check the silhouette score first to identify the ideal number of clusters
         from sklearn.metrics import silhouette_score
         sse_ = []
         for k in range(2, 10):
             kmeans = KMeans(n_clusters=k).fit(dat3)
             sse_.append([k, silhouette_score(dat3, kmeans.labels_)])
```

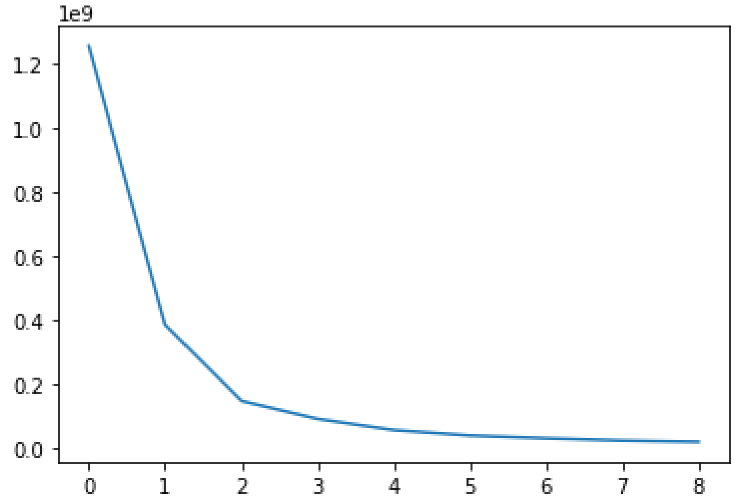
```
In [15]: plt.plot(pd.DataFrame(sse_)[0], pd.DataFrame(sse_)[1]);
```



In [16]:
```python
#The sihouette score reaches a peak at around 3 clusters indicating that it might be the ideal number of clusters.
#Let's use the elbow curve method to identify the ideal number of clusters.
ssd = []
for num_clusters in list(range(1,10)):
    model_clus = KMeans(n_clusters = num_clusters, max_iter=50)
    model_clus.fit(dat3)
    ssd.append(model_clus.inertia_)

plt.plot(ssd)
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x1581303f0b8>]
```



In [17]:
```python
#A distinct elbow is formed at around 2-3 clusters. Let's finally create the clusters and see for ourselves which ones fare bet
ter
#K-means with k=3 clusters
model_clus5 = KMeans(n_clusters = 3, max_iter=50)
model_clus5.fit(dat3)
```

Out[17]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=50,
    n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [18]:
```python
dat4=datm
dat4.index = pd.RangeIndex(len(dat4.index))
dat_km = pd.concat([dat4, pd.Series(model_clus5.labels_)], axis=1)
dat_km.columns = ['ind', 'Marker', 'Variation', 'AB', 'AC', 'AD', 'AE', 'AF', 'AG', 'AH',
       'AI', 'BB', 'BC', 'BD', 'BE', 'BF', 'BG', 'BH', 'BI', 'BK', 'CC', 'CD',
       'CE', 'CF', 'CG', 'CH', 'CI', 'CK', 'DD', 'DE', 'DF', 'DG', 'DH', 'DI',
       'DK', 'EE', 'EF', 'EG', 'EH', 'EI', 'EJ', 'FF', 'FG', 'FH', 'FI', 'FJ',
       'GG', 'GH', 'HH', 'II', 'IJ', 'JJ', 'KK', 'Marker_code','ClusterID']
dat_km.head()
```

Out[18]:

	ind	Marker	Variation	AB	AC	AD	AE	AF	AG	AH	...	FJ	GG	GH	HH	II	IJ	JJ	KK	Marker_code	ClusterID
0	ind4858	Marker481	BB	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	416	0
1	ind5876	Marker761	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	700	1
2	ind5668	Marker869	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	809	1
3	ind2807	Marker955	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	897	1
4	ind3888	Marker812	AA	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	749	1

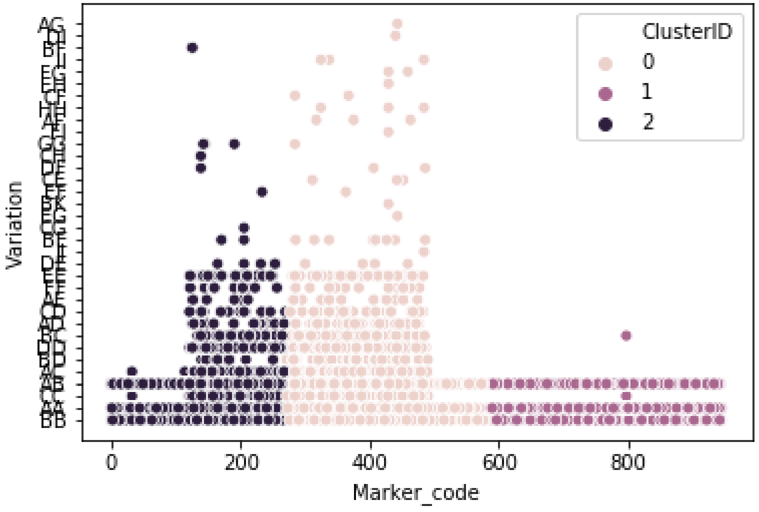
5 rows × 55 columns

In [19]:
```python
dat_km['ClusterID'].value_counts()
```

Out[19]:
```
0    7880
2    7798
1    4322
Name: ClusterID, dtype: int64
```

In [20]:
```python
#Each cluster has a good number
#Let's do some further visualizations.
#We'll be visualising the clusters on the original principal components
sns.scatterplot(x='Marker_code',y='Variation',hue='ClusterID',legend='full',data=dat_km)
```

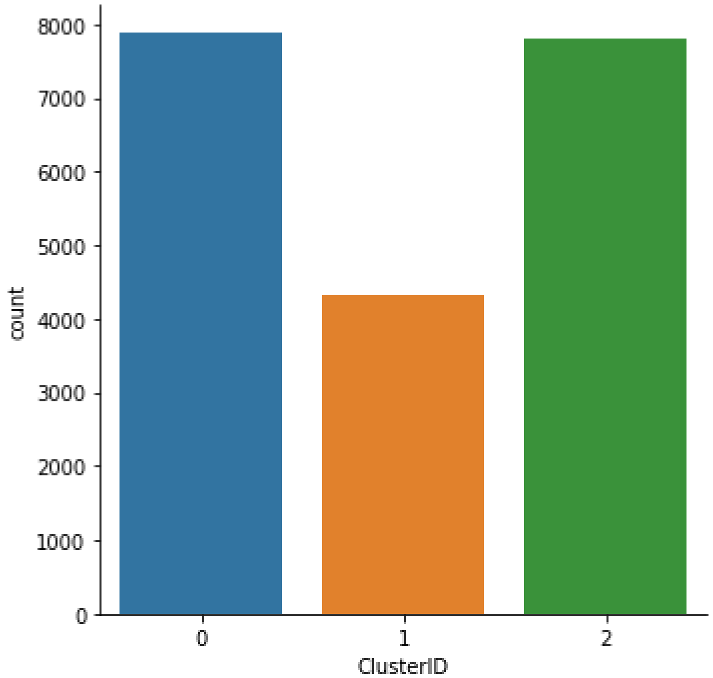
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x15812ff2ef0>



In [21]:
```
sns.factorplot(x ='ClusterID' ,data = dat_km, kind = "count")
```

```
C:\Users\hp\Anaconda3\lib\site-packages\seaborn\categorical.py:3666: UserWarning: The `factorplot` function has been renamed to
`catplot`. The original name will be removed in a future release. Please update your code. Note that the default `kind` in `fac
torplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x15812fcc0f0>
```



In [22]:
```
#let's take a look at those features clusters and try to make sense if the clustering process worked well.
# features in cluster 0
cluster0=dat_km[dat_km['ClusterID']==0]
cluster0[['ind','Marker','Variation']]
```

Out[22]:

	ind	Marker	Variation
0	ind4858	Marker481	BB
6	ind8222	Marker599	AA
14	ind8084	Marker461	AA
16	ind6274	Marker51	AA
17	ind1412	Marker602	BB
18	ind2300	Marker469	AB
22	ind8145	Marker516	AA
23	ind4831	Marker376	BB
25	ind3162	Marker509	AA
29	ind3414	Marker425	AA
30	ind6122	Marker454	BD
31	ind6440	Marker492	AA
32	ind8074	Marker417	BB
33	ind3065	Marker494	AA
34	ind7643	Marker640	BB
35	ind6705	Marker405	AA
39	ind7204	Marker402	BB
40	ind1399	Marker408	DD
44	ind3652	Marker542	AA
45	ind867	Marker367	BB
50	ind2439	Marker548	AA
55	ind1223	Marker379	AA
56	ind6990	Marker502	BB
57	ind6301	Marker371	AA
62	ind5386	Marker523	AA
68	ind2437	Marker393	BB
69	ind4903	Marker548	AA
70	ind2668	Marker392	BB

	ind	Marker	Variation
74	ind339	Marker647	AA
76	ind6090	Marker421	BB
...	...	...	...
19930	ind8170	Marker530	AA
19932	ind2959	Marker494	AB
19936	ind5886	Marker522	AA
19937	ind1523	Marker585	AA
19938	ind1117	Marker615	AB
19943	ind4349	Marker398	BB
19944	ind8219	Marker453	BB
19945	ind1843	Marker535	BB
19946	ind7129	Marker52	BB
19948	ind4834	Marker423	BB
19949	ind3066	Marker527	BB
19952	ind4291	Marker366	BB
19956	ind5707	Marker541	BB
19958	ind3155	Marker592	AA
19959	ind5483	Marker388	BB
19961	ind3790	Marker540	AA
19963	ind7791	Marker407	BB
19964	ind825	Marker394	AA
19966	ind720	Marker360	AC
19968	ind5070	Marker476	AB
19969	ind5470	Marker395	AA
19971	ind5871	Marker574	AA
19978	ind5828	Marker403	BB
19980	ind2694	Marker534	AA
19986	ind2223	Marker576	BB
19990	ind6542	Marker418	DD

	ind	Marker	Variation
19991	ind6831	Marker519	BB
19995	ind289	Marker405	AA
19996	ind6122	Marker528	CD
19998	ind2175	Marker488	BB

7880 rows × 3 columns

In [23]:
```
#unique variation in cluster 0
cluster0['Variation'].unique()
```

Out[23]:
```
array(['BB', 'AA', 'AB', 'BD', 'DD', 'AC', 'CC', 'BC', 'AD', 'CD', 'AE',
       'FF', 'EE', 'DE', 'JJ', 'BE', 'EG', 'BK', 'EF', 'CE', 'FJ', 'AF',
       'HH', 'CF', 'EH', 'FG', 'II', 'DI', 'DF', 'GG', 'AG'], dtype=object)
```

In [24]:
```
#unique ind in cluster 0
cluster0['ind'].unique()
```

Out[24]:
```
array(['ind4858', 'ind8222', 'ind8084', ..., 'ind2223', 'ind6542',
       'ind6831'], dtype=object)
```

In [25]:
```
# features in cluster 1
cluster1=dat_km[dat_km['ClusterID']==1]
cluster1[['ind','Marker','Variation']]
```

Out[25]:

	ind	Marker	Variation
1	ind5876	Marker761	AA
2	ind5668	Marker869	AA
3	ind2807	Marker955	AA
4	ind3888	Marker812	AA
5	ind4574	Marker859	BB
7	ind7442	Marker885	BB
8	ind6635	Marker681	AA
11	ind7687	Marker900	AA
19	ind664	Marker649	AA
26	ind2598	Marker869	BB
28	ind6503	Marker904	AA
38	ind5552	Marker704	AA
41	ind6699	Marker991	AA
51	ind973	Marker953	AA
52	ind3032	Marker656	BB
59	ind5118	Marker948	AA
64	ind4527	Marker944	BB
65	ind1370	Marker662	AB
67	ind2089	Marker813	BB
71	ind1527	Marker821	AA
72	ind4996	Marker874	AB
73	ind2851	Marker927	BB
75	ind7640	Marker882	BB
77	ind4385	Marker852	AB
80	ind4441	Marker962	AA
81	ind674	Marker691	BB
82	ind1013	Marker947	AA
84	ind7826	Marker704	BB

	ind	Marker	Variation
86	ind4748	Marker871	AA
89	ind5498	Marker899	AA
...	...	...	...
19836	ind3531	Marker933	AA
19846	ind3215	Marker677	AA
19852	ind4345	Marker953	AA
19858	ind743	Marker775	AA
19861	ind1231	Marker751	AA
19862	ind2212	Marker842	AA
19869	ind1778	Marker780	AA
19872	ind3271	Marker772	AA
19879	ind2371	Marker708	AA
19884	ind5216	Marker654	AA
19886	ind6445	Marker814	AA
19902	ind3288	Marker932	BB
19904	ind2549	Marker828	AA
19912	ind5638	Marker832	BB
19913	ind6765	Marker86	AA
19919	ind1152	Marker737	AA
19921	ind8178	Marker787	AA
19925	ind6711	Marker962	AB
19926	ind1864	Marker860	AA
19934	ind3518	Marker744	BB
19947	ind4576	Marker804	AA
19954	ind2785	Marker982	AA
19955	ind5097	Marker750	BB
19957	ind3787	Marker699	AA
19960	ind3769	Marker951	AA
19962	ind2777	Marker69	BB

	ind	Marker	Variation
19976	ind7878	Marker838	AA
19982	ind6403	Marker697	AA
19985	ind1597	Marker753	AA
19999	ind899	Marker911	BB

4322 rows × 3 columns

In [26]:
```
#unique variation in cluster 1
cluster1['Variation'].unique()
```

Out[26]:
```
array(['AA', 'BB', 'AB', 'BC', 'CC'], dtype=object)
```

In [27]:
```
#unique ind in cluster 1
cluster1['ind'].unique()
```

Out[27]:
```
array(['ind5876', 'ind5668', 'ind2807', ..., 'ind7878', 'ind1597',
       'ind899'], dtype=object)
```

In [28]:
```
#features in cluster 2
cluster2=dat_km[dat_km['ClusterID']==2]
cluster2[['ind','Marker','Variation']]
```

Out[28]:

	ind	Marker	Variation
9	ind4513	Marker231	CC
10	ind1883	Marker150	BB
12	ind2517	Marker216	AB
13	ind6417	Marker128	AA
15	ind2935	Marker162	AA
20	ind6520	Marker258	BB
21	ind6772	Marker255	AC
24	ind1722	Marker228	AA
27	ind5102	Marker131	BB
36	ind733	Marker156	AA
37	ind3320	Marker337	AA
42	ind3493	Marker329	AA
43	ind3433	Marker339	BB
46	ind8103	Marker194	AB
47	ind7440	Marker171	AB
48	ind3759	Marker194	BB
49	ind6560	Marker242	BB
53	ind1657	Marker258	BB
54	ind7523	Marker341	BB
58	ind4526	Marker310	BB
60	ind8138	Marker170	AB
61	ind8187	Marker265	AA
63	ind6454	Marker210	BB
66	ind2530	Marker231	AA
78	ind553	Marker337	BB
79	ind2173	Marker156	AB
83	ind773	Marker256	DD
85	ind506	Marker101	AA

	ind	Marker	Variation
88	ind4789	Marker323	AB
93	ind2185	Marker164	BB
...	...	...	...
19929	ind4604	Marker127	AA
19931	ind4888	Marker157	AB
19933	ind4892	Marker286	BB
19935	ind5845	Marker129	BB
19939	ind4058	Marker321	AB
19940	ind2322	Marker144	AB
19941	ind3480	Marker274	BB
19942	ind690	Marker32	AA
19950	ind7137	Marker245	CC
19951	ind8070	Marker123	AB
19953	ind5220	Marker197	BB
19965	ind729	Marker271	AA
19967	ind5976	Marker308	BB
19970	ind3577	Marker251	AA
19972	ind8023	Marker140	BB
19973	ind4494	Marker282	AC
19974	ind4745	Marker296	BC
19975	ind667	Marker222	BB
19977	ind3535	Marker218	AA
19979	ind1469	Marker282	CC
19981	ind5362	Marker309	CC
19983	ind6313	Marker290	AA
19984	ind4847	Marker288	DD
19987	ind2808	Marker222	BB
19988	ind5329	Marker158	BB
19989	ind7001	Marker322	BB

	ind	Marker	Variation
19992	ind869	Marker14	AA
19993	ind1383	Marker283	CC
19994	ind5649	Marker294	AB
19997	ind2683	Marker164	AA

7798 rows × 3 columns

In [29]:
```
#unique variation in cluster 2
cluster2['Variation'].unique()
```

Out[29]:
```
array(['CC', 'BB', 'AB', 'AA', 'AC', 'DD', 'BC', 'EE', 'BD', 'CD', 'AD',
       'BE', 'FF', 'CG', 'DE', 'DF', 'CH', 'AE', 'GG', 'BF', 'EF'],
      dtype=object)
```

In [30]:
```
#unique ind in cluster 2
cluster2['ind'].unique()
```

Out[30]:
```
array(['ind4513', 'ind1883', 'ind2517', ..., 'ind5329', 'ind869',
       'ind2683'], dtype=object)
```