5-clustering

February 25, 2021

#

Ciencia de Datos

Víctor Muñiz Sánchez

Maestría en Cómputo Estadístico

Enero a junio 2021

1 Clustering jerárquico

1.1 Ejemplo primates (Izenman)

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import matplotlib as mpl
import os
os.chdir('/home/victor/cursos/ciencia_de_datos_2021/')

sns.set()
%matplotlib inline
```

[2]:
```
primate_scapulae = pd.read_csv('data/primate.scapulae.csv')
primate_scapulae.head()
```

[2]:
```
   genus  AD.BD  AD.CD  EA.CD  Dx.CD  SH.ACR  EAD  beta  gamma  class_str  \
0     54  65.56  166.0  50.55  12.80    70.3  115    14   45.0  Hylobates
1     54  50.91   93.9  61.82  13.09    75.0  121    20   54.0  Hylobates
2     54  46.15   80.8  64.10  11.80    70.0  120    25   61.0  Hylobates
3     54  70.29  220.5  50.00  12.75    61.1  113    12   45.0  Hylobates
4     54  63.16  144.0  57.89  12.98    64.9  115    14   46.0  Hylobates

   classdigit
0           1
1           1
```

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

Quitamos la categoría `homo`, ya que no cuenta con la medición del ángulo γ

[3]:
```
data = primate_scapulae.drop(primate_scapulae[primate_scapulae.class_str ==␣
 ↪'Homo'].index)
data.drop(['genus','gamma','class_str','classdigit'],1,inplace = True)
data.head()
```

[3]:

	AD.BD	AD.CD	EA.CD	Dx.CD	SH.ACR	EAD	beta
0	65.56	166.0	50.55	12.80	70.3	115	14
1	50.91	93.9	61.82	13.09	75.0	121	20
2	46.15	80.8	64.10	11.80	70.0	120	25
3	70.29	220.5	50.00	12.75	61.1	113	12
4	63.16	144.0	57.89	12.98	64.9	115	14

Clustering jerárquico con scipy

[4]:
```
from scipy.cluster.hierarchy import linkage, fcluster, dendrogram

link = 'single' #'complete' 'average' (ver otros en la documentacion del módulo)
link_mat = linkage(data, method=link)
print('size data',data.shape)
print('clustering',link_mat.shape)
link_mat[:10,]
```

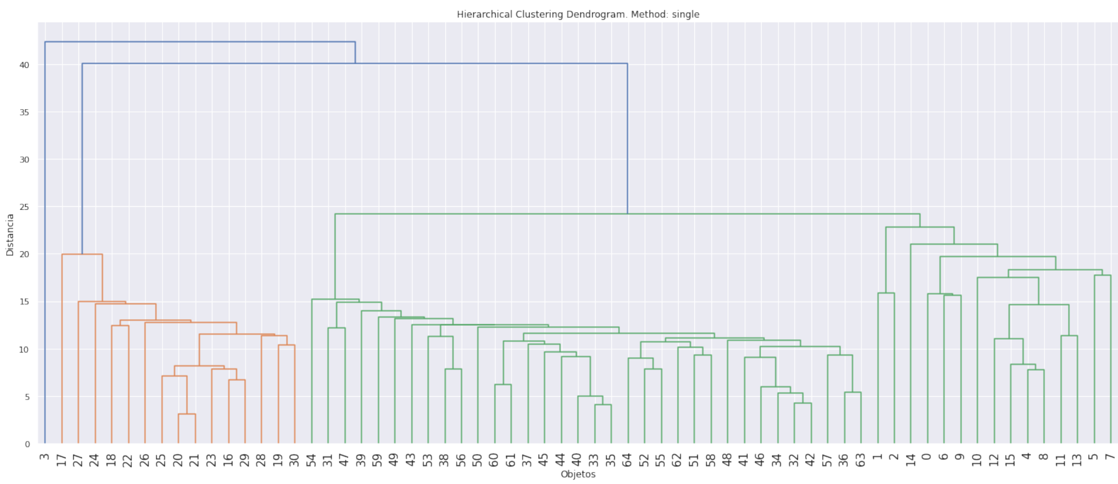
```
size data (65, 7)
clustering (64, 4)
```

[4]:
```
array([[20.        , 21.        ,  3.13261871,  2.        ],
       [33.        , 35.        ,  4.07410113,  2.        ],
       [32.        , 42.        ,  4.2483997 ,  2.        ],
       [40.        , 66.        ,  4.9819173 ,  3.        ],
       [34.        , 67.        ,  5.32959661,  3.        ],
       [36.        , 63.        ,  5.43829937,  2.        ],
       [46.        , 69.        ,  6.00983361,  4.        ],
       [60.        , 61.        ,  6.23247142,  2.        ],
       [16.        , 29.        ,  6.6892152 ,  2.        ],
       [25.        , 65.        ,  7.10786184,  3.        ]])
```

`link_mat` contiene el resultado del clústering jerárquico. Las primeras dos columnas indican el índice de los objetos agrupados. La tercera la distancia (disimilaridad) del clúster y la cuarta, el número de objetos en tal clúster. Cuando el índice de las primeras dos columnas es $> n$, indica un clúster agregado como un nuevo objeto. Ver documentación de `linkage` en https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.

Dendograma básico

```
[6]: plt.figure(figsize=(25, 10))
plt.title('Hierarchical Clustering Dendrogram. Method: '+link)
plt.xlabel('Objetos')
plt.ylabel('Distancia')
dendrogram(
    link_mat,
    leaf_rotation=90.,  # rotates the x axis labels
    leaf_font_size=15.,  # font size for the x axis labels
)
plt.show()
```

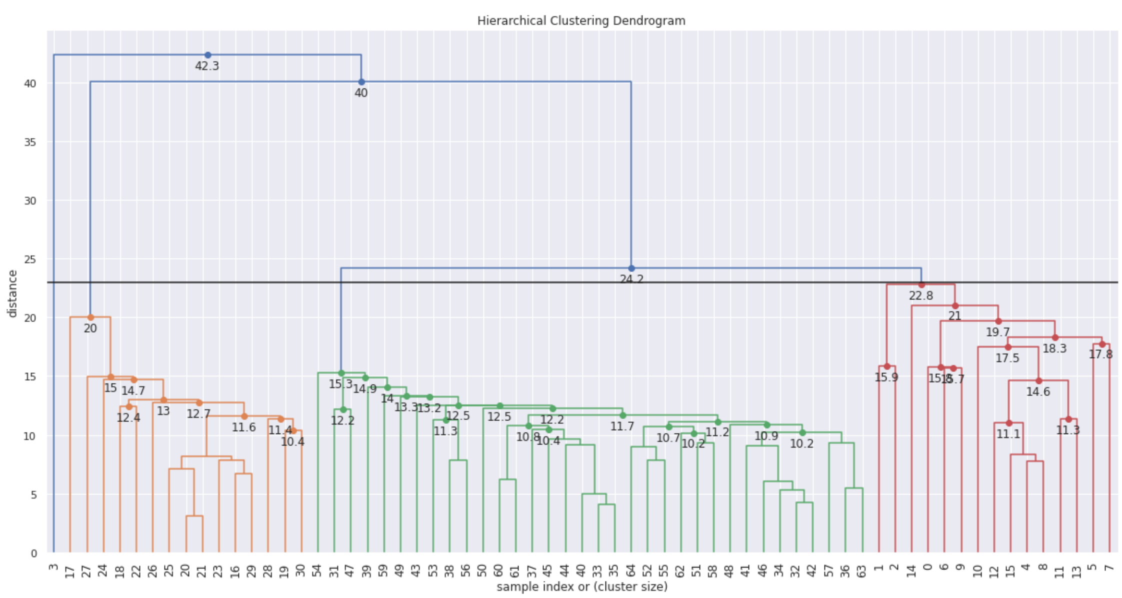


Dendograma con más información y un punto de corte para seleccionar clústers

```
[21]: def fancy_dendrogram(*args, **kwargs):
    max_d = kwargs.pop('max_d', None)
    if max_d and 'color_threshold' not in kwargs:
        kwargs['color_threshold'] = max_d
    annotate_above = kwargs.pop('annotate_above', 0)

    ddata = dendrogram(*args, **kwargs)

    if not kwargs.get('no_plot', False):
        plt.title('Hierarchical Clustering Dendrogram')
        plt.xlabel('sample index or (cluster size)')
        plt.ylabel('distance')
        for i, d, c in zip(ddata['icoord'], ddata['dcoord'],␣
↪ddata['color_list']):
            x = 0.5 * sum(i[1:3])
            y = d[1]
            if y > annotate_above:
```

```
                plt.plot(x, y, 'o', c=c)
                plt.annotate("%.3g" % y, (x, y), xytext=(0, -5),
                             textcoords='offset points',
                             va='top', ha='center')
        if max_d:
            plt.axhline(y=max_d, c='k')
    return ddata
```

[24]:
```
plt.figure(figsize=(20, 10))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('sample index')
plt.ylabel('distance')

max_d = 23
fancy_dendrogram(
    link_mat,
    #truncate_mode='lastp',
    #p=12,
    leaf_rotation=90.,
    leaf_font_size=12.,
    show_contracted=True,
    annotate_above=10,
    max_d=max_d,  # plot a horizontal cut-off line
)
plt.show()
```



Obtener la asignación de los objetos (datos) a cada clúster según cierto valor de distancia o especificación del número k de clústers

```
[58]: obj_clus = fcluster(link_mat, max_d, criterion='distance')
data_clus = pd.DataFrame(data).assign(Cluster = obj_clus)
#with pd.option_context('display.max_rows', None, 'display.max_columns', None):
#    print(data_clus)
print(data_clus.to_string())
```

```
    AD.BD  AD.CD  EA.CD  Dx.CD  SH.ACR  EAD  beta  Cluster
0   65.56  166.0  50.55  12.80    70.3  115    14        3
1   50.91   93.9  61.82  13.09    75.0  121    20        3
2   46.15   80.8  64.10  11.80    70.0  120    25        3
3   70.29  220.5  50.00  12.75    61.1  113    12        4
4   63.16  144.0  57.89  12.98    64.9  115    14        3
5   50.72  134.6  56.23  11.88    52.6  136    14        3
6   58.99  164.0  54.96  12.46    58.6  109    11        3
7   55.38  144.0  52.31  11.92    65.3  131    16        3
8   64.29  138.5  57.14  12.50    60.0  115    16        3
9   65.67  169.2  50.75  12.46    55.3  117    20        3
10  55.91  113.0  60.22  12.47    64.5  121    16        3
11  64.62  134.6  52.31  13.23    75.0  124    12        3
12  62.07  128.6  56.90  12.41    61.5  119    17        3
13  59.26  128.0  51.85  11.39    73.3  125    19        3
14  57.94  182.5  60.32  12.54    66.0  105    11        3
15  57.14  147.4  61.22  11.63    64.3  115    11        3
16  31.13   30.6  84.91  12.45    48.7  117    32        1
17  38.32   38.3  68.22  12.62    64.5  138    30        1
18  33.64   29.5  76.64  13.08    60.5  127    40        1
19  28.57   30.5  75.89  11.16    51.3  130    33        1
20  34.78   38.1  80.87  12.70    58.0  117    31        1
21  36.04   36.4  81.08  12.16    58.0  116    33        1
22  30.38   23.1  82.28  12.91    52.9  129    43        1
23  32.14   29.4  82.14  12.86    53.7  118    37        1
24  28.74   28.4  89.66  12.07    65.8  116    35        1
25  37.80   33.3  85.37  12.56    56.5  117    37        1
26  36.00   45.0  79.20  12.40    63.0  123    24        1
27  39.13   36.6  73.91  11.30    69.7  119    37        1
28  33.68   39.0  78.95  12.63    52.6  127    30        1
29  29.47   32.6  84.21  12.95    45.2  120    36        1
30  26.83   27.2  84.15  12.20    53.3  126    31        1
31  80.00  118.9  59.09  12.91    52.1   94    20        2
32  74.49  100.0  57.14  12.45    53.0  101    20        2
33  75.79   80.0  61.05  13.47    50.0   98    28        2
34  76.36  100.0  57.27  12.82    50.0   96    24        2
35  73.68   81.4  63.16  13.68    51.3  100    28        2
36  76.19  103.9  59.05  13.71    60.0   97    24        2
37  75.29   80.0  55.29  13.29    57.9  101    26        2
38  62.00   66.0  67.00  13.00    61.8  104    28        2
39  72.73  100.0  56.36  13.18    63.5  110    21        2
```

```
40  77.00   81.1  63.00  13.70    52.3   95    30        2
41  76.53   93.8  58.16  13.47    57.9  100    27        2
42  76.47   97.5  56.86  12.94    51.4  100    22        2
43  76.67   89.6  55.56  13.89    43.2  105    22        2
44  78.35   84.4  61.86  13.92    59.0   95    25        2
45  70.00   76.8  63.33  13.67    50.6  107    25        2
46  78.00  102.6  56.00  13.20    50.0  101    24        2
47  77.00  114.9  54.00  14.00    55.9  103    19        2
48  84.55  100.0  57.27  13.27    62.2   92    24        2
49  69.47   66.0  57.89  13.90    53.9  108    29        2
50  78.57   77.0  45.92  13.88    53.8  105    29        2
51  70.59  100.0  72.94  12.94    57.1   95    27        2
52  63.95   79.7  68.02  12.56    56.1  101    33        2
53  67.97   72.5  64.84  14.06    66.7  100    30        2
54  73.48   95.1  66.67  13.26    77.1   94    25        2
55  66.67   84.0  67.46  12.62    59.3   97    30        2
56  57.89   68.1  62.41  12.48    61.1  107    31        2
57  74.07  101.7  64.81  13.46    65.1   94    32        2
58  71.54   93.0  69.23  13.08    60.9   95    30        2
59  65.93   78.1  54.81  13.48    60.2  105    34        2
60  66.90   92.4  60.69  12.62    49.0  103    25        2
61  68.00   89.5  65.60  12.96    50.0  103    27        2
62  65.68   95.7  72.78  12.84    65.7   96    25        2
63  76.50  106.1  62.84  13.28    59.7   96    27        2
64  71.75   81.4  63.84  12.66    62.3   96    35        2
```

[28]:
```python
k = 4
obj_clus = fcluster(link_mat, k, criterion='maxclust')
data_clus = pd.DataFrame(data).assign(Cluster = obj_clus)
print(data_clus.to_string())
```

```
    AD.BD  AD.CD  EA.CD  Dx.CD  SH.ACR  EAD  beta  Cluster
0   65.56  166.0  50.55  12.80    70.3  115    14        3
1   50.91   93.9  61.82  13.09    75.0  121    20        3
2   46.15   80.8  64.10  11.80    70.0  120    25        3
3   70.29  220.5  50.00  12.75    61.1  113    12        4
4   63.16  144.0  57.89  12.98    64.9  115    14        3
5   50.72  134.6  56.23  11.88    52.6  136    14        3
6   58.99  164.0  54.96  12.46    58.6  109    11        3
7   55.38  144.0  52.31  11.92    65.3  131    16        3
8   64.29  138.5  57.14  12.50    60.0  115    16        3
9   65.67  169.2  50.75  12.46    55.3  117    20        3
10  55.91  113.0  60.22  12.47    64.5  121    16        3
11  64.62  134.6  52.31  13.23    75.0  124    12        3
12  62.07  128.6  56.90  12.41    61.5  119    17        3
13  59.26  128.0  51.85  11.39    73.3  125    19        3
14  57.94  182.5  60.32  12.54    66.0  105    11        3
15  57.14  147.4  61.22  11.63    64.3  115    11        3
```

16	31.13	30.6	84.91	12.45	48.7	117	32	1
17	38.32	38.3	68.22	12.62	64.5	138	30	1
18	33.64	29.5	76.64	13.08	60.5	127	40	1
19	28.57	30.5	75.89	11.16	51.3	130	33	1
20	34.78	38.1	80.87	12.70	58.0	117	31	1
21	36.04	36.4	81.08	12.16	58.0	116	33	1
22	30.38	23.1	82.28	12.91	52.9	129	43	1
23	32.14	29.4	82.14	12.86	53.7	118	37	1
24	28.74	28.4	89.66	12.07	65.8	116	35	1
25	37.80	33.3	85.37	12.56	56.5	117	37	1
26	36.00	45.0	79.20	12.40	63.0	123	24	1
27	39.13	36.6	73.91	11.30	69.7	119	37	1
28	33.68	39.0	78.95	12.63	52.6	127	30	1
29	29.47	32.6	84.21	12.95	45.2	120	36	1
30	26.83	27.2	84.15	12.20	53.3	126	31	1
31	80.00	118.9	59.09	12.91	52.1	94	20	2
32	74.49	100.0	57.14	12.45	53.0	101	20	2
33	75.79	80.0	61.05	13.47	50.0	98	28	2
34	76.36	100.0	57.27	12.82	50.0	96	24	2
35	73.68	81.4	63.16	13.68	51.3	100	28	2
36	76.19	103.9	59.05	13.71	60.0	97	24	2
37	75.29	80.0	55.29	13.29	57.9	101	26	2
38	62.00	66.0	67.00	13.00	61.8	104	28	2
39	72.73	100.0	56.36	13.18	63.5	110	21	2
40	77.00	81.1	63.00	13.70	52.3	95	30	2
41	76.53	93.8	58.16	13.47	57.9	100	27	2
42	76.47	97.5	56.86	12.94	51.4	100	22	2
43	76.67	89.6	55.56	13.89	43.2	105	22	2
44	78.35	84.4	61.86	13.92	59.0	95	25	2
45	70.00	76.8	63.33	13.67	50.6	107	25	2
46	78.00	102.6	56.00	13.20	50.0	101	24	2
47	77.00	114.9	54.00	14.00	55.9	103	19	2
48	84.55	100.0	57.27	13.27	62.2	92	24	2
49	69.47	66.0	57.89	13.90	53.9	108	29	2
50	78.57	77.0	45.92	13.88	53.8	105	29	2
51	70.59	100.0	72.94	12.94	57.1	95	27	2
52	63.95	79.7	68.02	12.56	56.1	101	33	2
53	67.97	72.5	64.84	14.06	66.7	100	30	2
54	73.48	95.1	66.67	13.26	77.1	94	25	2
55	66.67	84.0	67.46	12.62	59.3	97	30	2
56	57.89	68.1	62.41	12.48	61.1	107	31	2
57	74.07	101.7	64.81	13.46	65.1	94	32	2
58	71.54	93.0	69.23	13.08	60.9	95	30	2
59	65.93	78.1	54.81	13.48	60.2	105	34	2
60	66.90	92.4	60.69	12.62	49.0	103	25	2
61	68.00	89.5	65.60	12.96	50.0	103	27	2
62	65.68	95.7	72.78	12.84	65.7	96	25	2
63	76.50	106.1	62.84	13.28	59.7	96	27	2

```
64  71.75   81.4  63.84  12.66    62.3   96    35         2
```

Usando sklearn. Detalles, ve en la documentación: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering

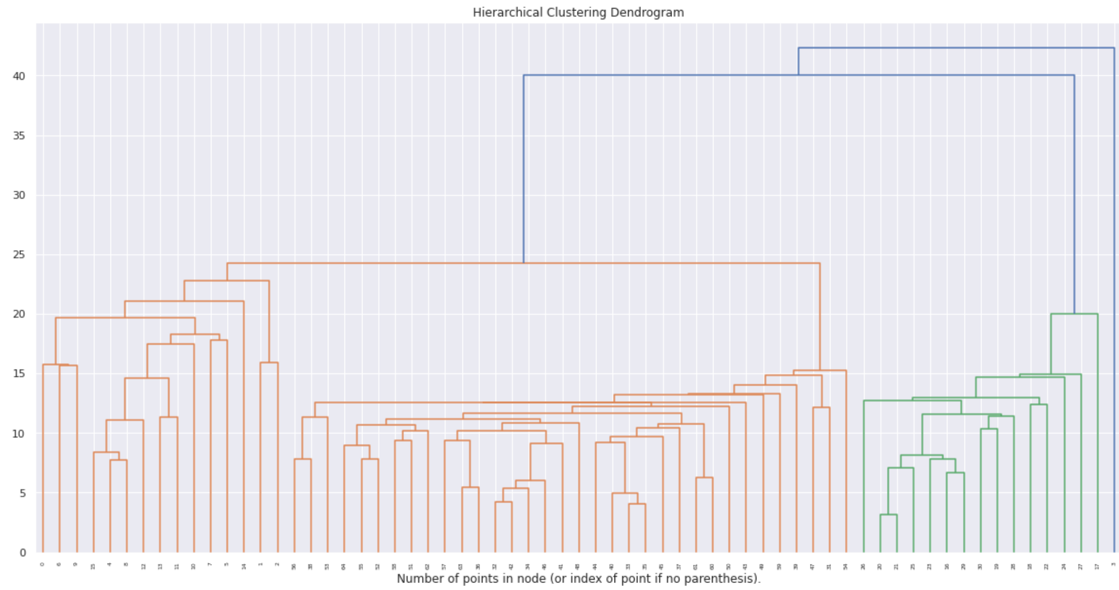
[59]:
```
from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n_clusters=None, distance_threshold=0,␣
 ↪affinity='euclidean', linkage='single')
cluster.fit_predict(data)

print('Clusters: ',cluster.n_clusters_)
```

```
Clusters:  65
```

Observa que, se puede definir un número de clusters (`n_clusters`) o un criterio basado en la distancia `distance_threshold`. En este caso, como la distancia se pone en 0, no da una asignación específica de clusters a los datos. De hecho, el número de clústers es n.

[31]:
```
def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,
                                      counts]).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
```

[60]:
```
plt.figure(figsize=(20, 10))
plt.title('Hierarchical Clustering Dendrogram')
plot_dendrogram(cluster)
#plot_dendrogram(cluster, truncate_mode='level', p=4)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```



[38]:
```python
cluster2 = AgglomerativeClustering(n_clusters=4, distance_threshold=None,␣
↪affinity='euclidean',
                                   linkage='single')
cluster2.fit_predict(data)

print('Clusters: ',cluster2.n_clusters_)
data_clus2 = pd.DataFrame(data).assign(Cluster = cluster2.labels_)
print(data_clus2.to_string())
```

```
Clusters:  4
    AD.BD  AD.CD  EA.CD  Dx.CD  SH.ACR  EAD  beta  Cluster
0   65.56  166.0  50.55  12.80    70.3  115    14        0
1   50.91   93.9  61.82  13.09    75.0  121    20        0
2   46.15   80.8  64.10  11.80    70.0  120    25        0
3   70.29  220.5  50.00  12.75    61.1  113    12        3
4   63.16  144.0  57.89  12.98    64.9  115    14        0
5   50.72  134.6  56.23  11.88    52.6  136    14        0
6   58.99  164.0  54.96  12.46    58.6  109    11        0
7   55.38  144.0  52.31  11.92    65.3  131    16        0
8   64.29  138.5  57.14  12.50    60.0  115    16        0
9   65.67  169.2  50.75  12.46    55.3  117    20        0
10  55.91  113.0  60.22  12.47    64.5  121    16        0
11  64.62  134.6  52.31  13.23    75.0  124    12        0
12  62.07  128.6  56.90  12.41    61.5  119    17        0
13  59.26  128.0  51.85  11.39    73.3  125    19        0
14  57.94  182.5  60.32  12.54    66.0  105    11        0
15  57.14  147.4  61.22  11.63    64.3  115    11        0
16  31.13   30.6  84.91  12.45    48.7  117    32        2
```

17	38.32	38.3	68.22	12.62	64.5	138	30	2
18	33.64	29.5	76.64	13.08	60.5	127	40	2
19	28.57	30.5	75.89	11.16	51.3	130	33	2
20	34.78	38.1	80.87	12.70	58.0	117	31	2
21	36.04	36.4	81.08	12.16	58.0	116	33	2
22	30.38	23.1	82.28	12.91	52.9	129	43	2
23	32.14	29.4	82.14	12.86	53.7	118	37	2
24	28.74	28.4	89.66	12.07	65.8	116	35	2
25	37.80	33.3	85.37	12.56	56.5	117	37	2
26	36.00	45.0	79.20	12.40	63.0	123	24	2
27	39.13	36.6	73.91	11.30	69.7	119	37	2
28	33.68	39.0	78.95	12.63	52.6	127	30	2
29	29.47	32.6	84.21	12.95	45.2	120	36	2
30	26.83	27.2	84.15	12.20	53.3	126	31	2
31	80.00	118.9	59.09	12.91	52.1	94	20	1
32	74.49	100.0	57.14	12.45	53.0	101	20	1
33	75.79	80.0	61.05	13.47	50.0	98	28	1
34	76.36	100.0	57.27	12.82	50.0	96	24	1
35	73.68	81.4	63.16	13.68	51.3	100	28	1
36	76.19	103.9	59.05	13.71	60.0	97	24	1
37	75.29	80.0	55.29	13.29	57.9	101	26	1
38	62.00	66.0	67.00	13.00	61.8	104	28	1
39	72.73	100.0	56.36	13.18	63.5	110	21	1
40	77.00	81.1	63.00	13.70	52.3	95	30	1
41	76.53	93.8	58.16	13.47	57.9	100	27	1
42	76.47	97.5	56.86	12.94	51.4	100	22	1
43	76.67	89.6	55.56	13.89	43.2	105	22	1
44	78.35	84.4	61.86	13.92	59.0	95	25	1
45	70.00	76.8	63.33	13.67	50.6	107	25	1
46	78.00	102.6	56.00	13.20	50.0	101	24	1
47	77.00	114.9	54.00	14.00	55.9	103	19	1
48	84.55	100.0	57.27	13.27	62.2	92	24	1
49	69.47	66.0	57.89	13.90	53.9	108	29	1
50	78.57	77.0	45.92	13.88	53.8	105	29	1
51	70.59	100.0	72.94	12.94	57.1	95	27	1
52	63.95	79.7	68.02	12.56	56.1	101	33	1
53	67.97	72.5	64.84	14.06	66.7	100	30	1
54	73.48	95.1	66.67	13.26	77.1	94	25	1
55	66.67	84.0	67.46	12.62	59.3	97	30	1
56	57.89	68.1	62.41	12.48	61.1	107	31	1
57	74.07	101.7	64.81	13.46	65.1	94	32	1
58	71.54	93.0	69.23	13.08	60.9	95	30	1
59	65.93	78.1	54.81	13.48	60.2	105	34	1
60	66.90	92.4	60.69	12.62	49.0	103	25	1
61	68.00	89.5	65.60	12.96	50.0	103	27	1
62	65.68	95.7	72.78	12.84	65.7	96	25	1
63	76.50	106.1	62.84	13.28	59.7	96	27	1
64	71.75	81.4	63.84	12.66	62.3	96	35	1

Ahora, especifico el número de Clústers en 4

1.1.1 Clustering de observaciones y variables. El Heatmap

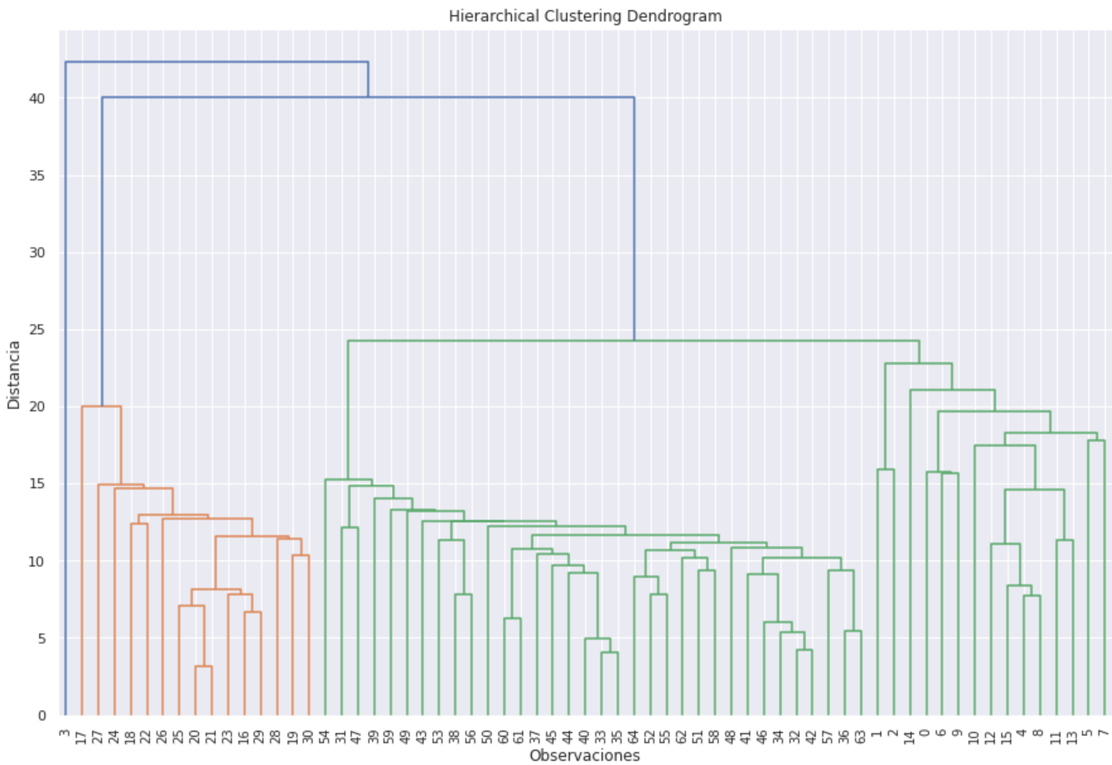
Como habíamos mencionado en clase, podemos hacer clústering de observaciones y variables eligiendo la medida de distancia (disimilaridad) apropiada. En este ejemplo, usaremos distancia euclideana para las observaciones y la distancia de correlación para las variables

[39]:
```
from scipy.spatial.distance import pdist, squareform

# distancia para observaciones
Y = data.transpose()
row_link = linkage(data, metric = 'euclidean', method='single')
col_link = linkage(Y, metric = 'correlation', method = 'single')
```

OJO: recuerda que la `correlación como distancia` no es lo mismo que la matriz de correlación. Checa tus notas de clase cuando vimos distancias… (espero que tomes nota).

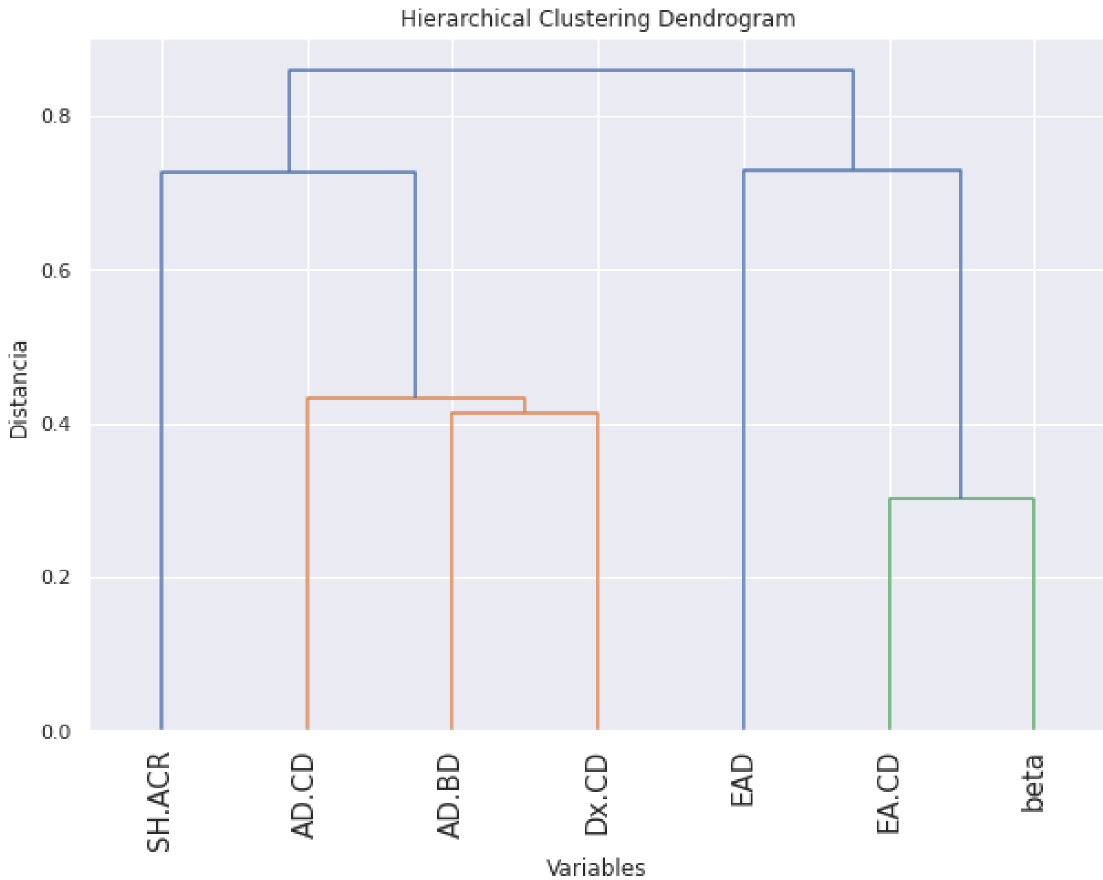
Clustering de observaciones (ya lo vimos pero lo repetimos)

[40]:
```
plt.figure(figsize=(15, 10))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Observaciones')
plt.ylabel('Distancia')
dendrogram(
    row_link,
    leaf_rotation=90.,  # rotates the x axis labels
    leaf_font_size=10.,  # font size for the x axis labels
)
plt.show()
```



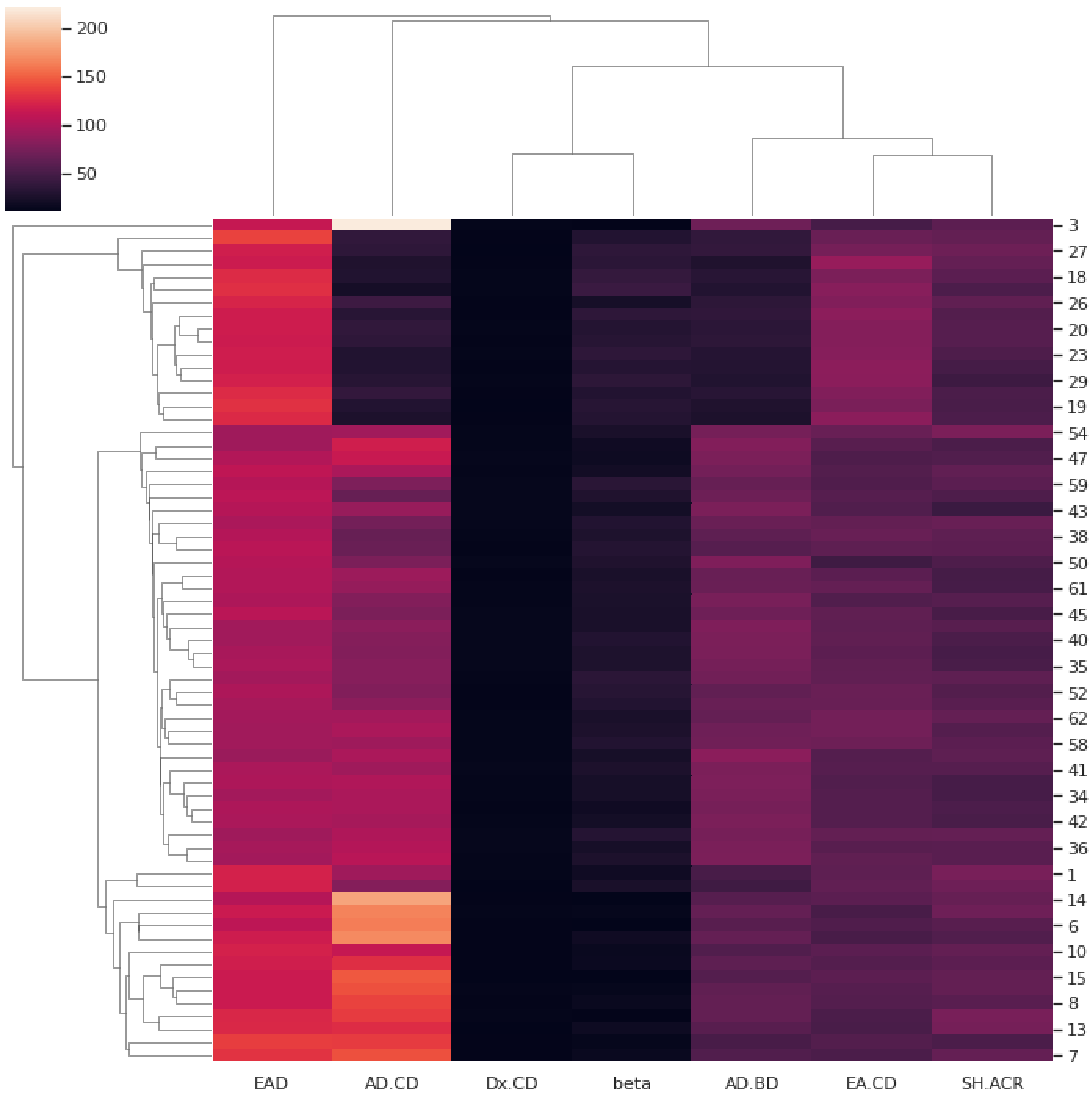
Clustering de variables

[41]:
```
plt.figure(figsize=(10, 7))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Variables')
plt.ylabel('Distancia')
dendrogram(
    col_link,
    labels = list(data.columns),
    leaf_rotation=90.,  # rotates the x axis labels
    leaf_font_size=15.,  # font size for the x axis labels
)
plt.show()
```



Todo junto en un heatmap que muestra las distancias (similaridades) entre las observaciones, ordenadas según los clústers jerárquicos para observaciones y variables

[42]:
```
import seaborn as sns; sns.set()

row_link = linkage(data, method='single')
col_link = linkage(Y, method = 'single')

g = sns.clustermap(data, row_linkage=row_link, col_linkage=col_link)
```



Para detalles sobre las opciones de configuración del heatmap, ver la documentación de `seaborn.clustermap`

1.1.2 Ejemplo Gower.

Tomado de https://github.com/wwwjk366/gower.

[45]:
```
import numpy as np
import pandas as pd

Xd=pd.DataFrame({'age':[21,21,19, 30,21,21,19,30],
'gender':['M','M','M','M','F','F','F','F'],
'civil_status':
 ↪['MARRIED','SINGLE','SINGLE','SINGLE','MARRIED','SINGLE','WIDOW','DIVORCED'],
```

```
'salary':[3000.0,1200.0 ,32000.0,1800.0 ,2900.0 ,1100.0 ,10000.0,1500.0],
'has_children':[1,0,1,1,1,0,0,1],
'available_credit':[2200,100,22000,1100,2000,100,6000,2200]})
Xd
```

[45]:

	age	gender	civil_status	salary	has_children	available_credit
0	21	M	MARRIED	3000.0	1	2200
1	21	M	SINGLE	1200.0	0	100
2	19	M	SINGLE	32000.0	1	22000
3	30	M	SINGLE	1800.0	1	1100
4	21	F	MARRIED	2900.0	1	2000
5	21	F	SINGLE	1100.0	0	100
6	19	F	WIDOW	10000.0	0	6000
7	30	F	DIVORCED	1500.0	1	2200

Datos sintéticos. Ver la documentación para las distintas formas de especificar el tipo de variables en un data frame de Pandas.

[48]:

```
# instalar con pip
import gower
from tabulate import tabulate

dm = gower.gower_matrix(Xd)
print('Matriz disimilaridad \n')
table = tabulate(dm, tablefmt="fancy_grid")
print(table)
```

```
Matriz disimilaridad



 0          0.359024   0.504073   0.317874   0.168728   0.52623    0.596979
0.477788


 0.359024   0          0.529764   0.313877   0.523629   0.167206   0.456002
0.653964


 0.504073   0.529764   0          0.488614   0.672801   0.69697    0.740428
0.815194


 0.317874   0.313877   0.488614   0          0.482479   0.481083   0.748186
0.343323


 0.168728   0.523629   0.672801   0.482479   0          0.357502   0.432373
```

0.312104

0.52623 0.167206 0.69697 0.481083 0.357502 0 0.289875
0.487836

0.596979 0.456002 0.740428 0.748186 0.432373 0.289875 0
0.574766

0.477788 0.653964 0.815194 0.343323 0.312104 0.487836 0.574766 0

Un detalle al usar `linkage` de `scipy`, es que el argumento y de los datos debe ser una matriz de datos $n \times d$ o una matriz de distancias condensada, es decir, un arreglo que contenga la diagonal superior. Ver los detalles en la documentación de la función.

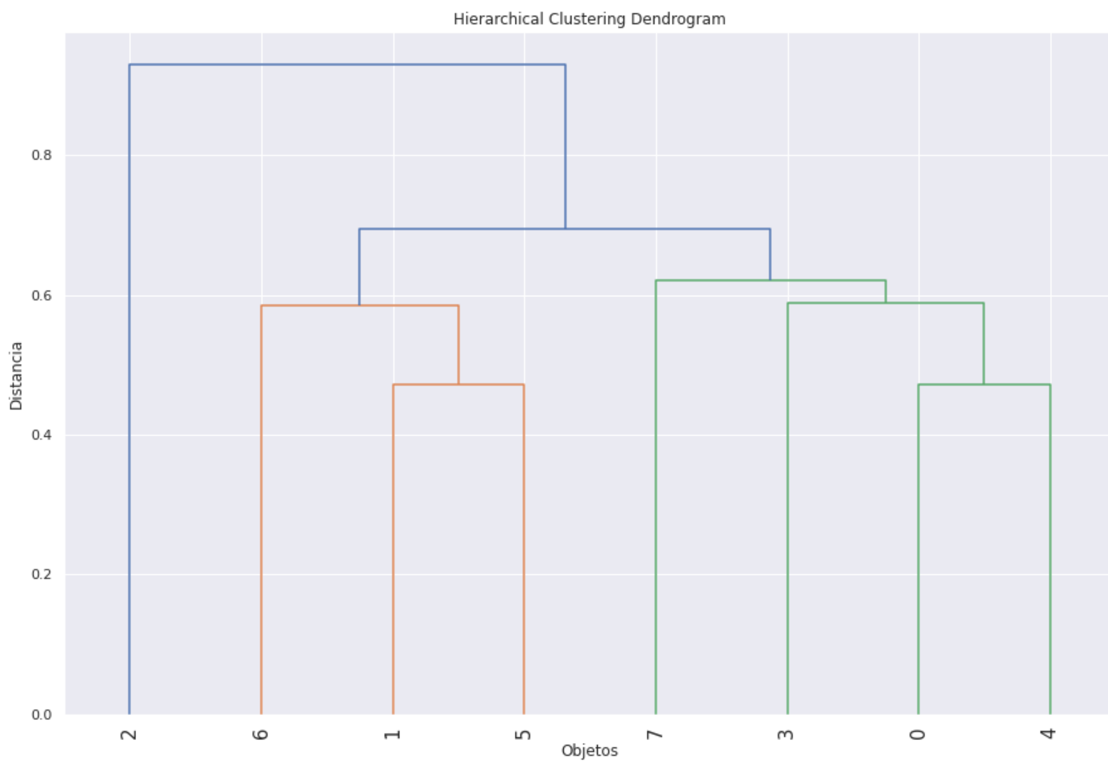
[49]:
```python
# clustering (3 grupos)
Zd = linkage(dm, method = 'single')
cld = fcluster(Zd, 3, criterion='maxclust')

data_clus3 = pd.DataFrame(Xd).assign(Cluster = cld)

print('Clustering \n')
data_clus3
```

```
Clustering


<ipython-input-49-21b6065830cf>:2: ClusterWarning: scipy.cluster: The symmetric
non-negative hollow observation matrix looks suspiciously like an uncondensed
distance matrix
  Zd = linkage(dm, method = 'single')
```

[49]:

	age	gender	civil_status	salary	has_children	available_credit	Cluster
0	21	M	MARRIED	3000.0	1	2200	2
1	21	M	SINGLE	1200.0	0	100	1
2	19	M	SINGLE	32000.0	1	22000	3
3	30	M	SINGLE	1800.0	1	1100	2
4	21	F	MARRIED	2900.0	1	2000	2
5	21	F	SINGLE	1100.0	0	100	1
6	19	F	WIDOW	10000.0	0	6000	1
7	30	F	DIVORCED	1500.0	1	2200	2

[57]:
```
plt.figure(figsize=(15, 10))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Objetos')
plt.ylabel('Distancia')
dendrogram(Zd, leaf_rotation=90., leaf_font_size=15.,)
plt.show()
```



[51]:
```
cluster_gower = AgglomerativeClustering(n_clusters=3, distance_threshold=None,␣
 ↪affinity='precomputed',
                                        linkage='single')
cld2 = cluster_gower.fit_predict(dm)
data_clus4 = pd.DataFrame(Xd).assign(Cluster = cld2)
print('Clustering \n')
data_clus4
```

Clustering

[51]:

	age	gender	civil_status	salary	has_children	available_credit	Cluster
0	21	M	MARRIED	3000.0	1	2200	2
1	21	M	SINGLE	1200.0	0	100	0
2	19	M	SINGLE	32000.0	1	22000	1
3	30	M	SINGLE	1800.0	1	1100	0

4	21	F	MARRIED	2900.0	1	2000	2
5	21	F	SINGLE	1100.0	0	100	0
6	19	F	WIDOW	10000.0	0	6000	0
7	30	F	DIVORCED	1500.0	1	2200	2

Alternativamente, con la función de `sklearn` puedes pasar directamente una matriz de distancias con la opción `affinity = precomputed`