Untitled10

April 24, 2020

1 Heiner Romero Leiva

2 Tarea 2

3 Programación ACP

[1]:
```
import pandas as pd
import numpy as np
import scipy.linalg as la
from sklearn import preprocessing
import matplotlib.pyplot as plt
from math import sqrt

class mynew_PCA:
    def __init__(self, datos = pd.DataFrame()):
        # Se enlistan en orden según requerimiento
        self.__columnas = datos.columns
        self.__filas = datos.index
        self.__datos = self.__transformar(datos)
        self.__correlaciones = self.__correlaciones(self.datos)
        self.__varvec_propios = self.__varvec_propios(self.correlaciones)
        self.__componentes = self.__componentes(self.datos, self.varvec_propios)
        self.__calidades_ind = self.__calidades_ind(self.datos, self.
↪componentes) #Calidades individuos
        self.__coordenadas_var = self.__coordenadas_var(self.varvec_propios)
        self.__calidades_var = self.__calidades_var(self.varvec_propios)
        self.__inercias = self.__inercias(self.varvec_propios, self.datos.
↪shape[1])
    @property
    def datos(self):
        return self.__datos
    @property
    def correlaciones(self):
        return self.__correlaciones
    @property
    def varvec_propios(self):
        return self.__varvec_propios
```

```python
    @property
    def componentes(self):
        return self.__componentes
    @property
    def calidades_ind(self):
        return self.__calidades_ind
    @property
    def coordenadas_var(self):
        return self.__coordenadas_var
    @property
    def calidades_var(self):
        return self.__calidades_var
    @property
    def inercias(self):
        return self.__inercias
    def __transformar(self, datos):
        return preprocessing.StandardScaler().fit_transform(datos)
    def __correlaciones(self, datos):
        return pd.DataFrame((1/datos.shape[0])*np.mat(datos.T)*np.mat(datos),␣
↪index=self.__columnas) #Se utiliza
    # formula propuesta por el profesor, no da igual que corr y cov.
    def __varvec_propios(self, correlaciones):
        valores_propios, vectores_propios = la.eig(correlaciones)
        valor_vector = [(np.abs(valores_propios[i]).real, vectores_propios[:
↪,i]) for i in range(len(valores_propios))]
        valor_vector.sort(key=lambda x: x[0], reverse=True)
        return valor_vector
    def __componentes(self, datos, varvec_propios):
        df = pd.DataFrame()
        for x in range(len(varvec_propios)):
            df[x] = varvec_propios[x][1]
        return pd.DataFrame(np.dot(datos, df), index=self.__filas)
    def __calidades_ind(self, datos, componentes):
        datos_2 = datos ** 2
        componentes_2 = componentes ** 2
        df = pd.DataFrame(datos)
        for i in range(componentes_2.shape[0]):
            for j in range(componentes_2.shape[1]):
                fila_suma = sum(datos_2[i, :])
                df.iloc[i,j] = componentes_2.iloc[i,j] / fila_suma
        return df
    def __coordenadas_var(self, varvec_propios):
        df = pd.DataFrame()
        for x in range(len(varvec_propios)):
            df[x] = (varvec_propios[x][0] ** (0.5)) * varvec_propios[x][1]
        return df
    def __calidades_var(self, varvec_propios):
```

```
        df = pd.DataFrame()
        for x in range(len(varvec_propios)):
            df[x] = varvec_propios[x][0] * (varvec_propios[x][1] ** 2)
        return df
    def __inercias(self, varvec_propios, m):
        arreglo = []
        for x in range(len(varvec_propios)):
            arreglo.append(100 * (varvec_propios[x][0] / m))
        return pd.DataFrame(np.matrix(arreglo))
    def plot_plano_principal(self, ejes = [0, 1], ind_labels = True, titulo =␣
↪'Plano Principal'):
        x = self.componentes[ejes[0]].values
        y = self.componentes[ejes[1]].values
        plt.style.use('seaborn-whitegrid')
        plt.scatter(x, y, color = 'gray')
        plt.title(titulo)
        plt.axhline(y = 0, color = 'dimgrey', linestyle = '--')
        plt.axvline(x = 0, color = 'dimgrey', linestyle = '--')
        inercia_x = round(self.inercias[ejes[0]], 2)
        inercia_y = round(self.inercias[ejes[1]], 2)
        plt.xlabel('Componente ' + str(ejes[0]) + ' (' + str(inercia_x) + '%)')
        plt.ylabel('Componente ' + str(ejes[1]) + ' (' + str(inercia_y) + '%)')
        if ind_labels:
            for i, txt in enumerate(self.componentes.index):
                plt.annotate(txt, (x[i], y[i]))
    def plot_circulo(self, ejes = [0, 1], var_labels = True, titulo = 'Círculo␣
↪de Correlación'):
        cor = self.coordenadas_var.iloc[:, ejes].values
        plt.style.use('seaborn-whitegrid')
        c = plt.Circle((0, 0), radius = 1, color = 'steelblue', fill = False)
        plt.gca().add_patch(c)
        plt.axis('scaled')
        plt.title(titulo)
        plt.axhline(y = 0, color = 'dimgrey', linestyle = '--')
        plt.axvline(x = 0, color = 'dimgrey', linestyle = '--')
        inercia_x = round(self.inercias[ejes[0]], 2)
        inercia_y = round(self.inercias[ejes[1]], 2)
        plt.xlabel('Componente ' + str(ejes[0]) + ' (' + str(inercia_x) + '%)')
        plt.ylabel('Componente ' + str(ejes[1]) + ' (' + str(inercia_y) + '%)')
        for i in range(cor.shape[0]):
            plt.arrow(0, 0, cor[i, 0] * 0.95, cor[i, 1] * 0.95, color =␣
↪'steelblue',
                      alpha = 0.5, head_width = 0.05, head_length = 0.05)
            if var_labels:
                plt.text(cor[i, 0] * 1.05, cor[i, 1] * 1.05, self.correlaciones.
↪index[i],
                         color = 'steelblue', ha = 'center', va = 'center')
```

```
    def plot_sobreposicion(self, ejes = [0, 1], ind_labels = True,
                          var_labels = True, titulo = 'Sobreposición␣
↪Plano-Círculo'):
        x = self.componentes[ejes[0]].values
        y = self.componentes[ejes[1]].values
        cor = self.correlaciones.iloc[:, ejes]
        scale = min((max(x) - min(x)/(max(cor[ejes[0]]) - min(cor[ejes[0]]))),
                    (max(y) - min(y)/(max(cor[ejes[1]]) - min(cor[ejes[1]])))))␣
↪* 0.7
        cor = self.coordenadas_var.iloc[:, ejes].values
        plt.style.use('seaborn-whitegrid')
        plt.axhline(y = 0, color = 'dimgrey', linestyle = '--')
        plt.axvline(x = 0, color = 'dimgrey', linestyle = '--')
        inercia_x = round(self.inercias[ejes[0]], 2)
        inercia_y = round(self.inercias[ejes[1]], 2)
        plt.xlabel('Componente ' + str(ejes[0]) + ' (' + str(inercia_x) + '%)')
        plt.ylabel('Componente ' + str(ejes[1]) + ' (' + str(inercia_y) + '%)')
        plt.scatter(x, y, color = 'gray')
        if ind_labels:
            for i, txt in enumerate(self.componentes.index):
                plt.annotate(txt, (x[i], y[i]))
        for i in range(cor.shape[0]):
            plt.arrow(0, 0, cor[i, 0] * scale, cor[i, 1] * scale, color =␣
↪'steelblue',
                      alpha = 0.5, head_width = 0.05, head_length = 0.05)
            if var_labels:
                plt.text(cor[i, 0] * scale * 1.15, cor[i, 1] * scale * 1.15,
                         self.correlaciones.index[i],
                         color = 'steelblue', ha = 'center', va = 'center')
```

4 Prueba Iris con ACP propia

[2]:
```
iris = pd.read_csv('iris.csv', delimiter=';', decimal='.', header=0)
acp_iris = mynew_PCA(iris.iloc[:,0:4])
```

5 Matriz de Correlaciones

[3]:
```
acp_iris.correlaciones
```

[3]:

	0	1	2	3
s.largo	1.000000	-0.109369	0.871754	0.817954
s.ancho	-0.109369	1.000000	-0.420516	-0.356544
p.largo	0.871754	-0.420516	1.000000	0.962757
p.ancho	0.817954	-0.356544	0.962757	1.000000

6 Valores y Vectores Propios

[5]:
```
acp_iris.varvec_propios
```

[5]:
```
[(2.910818083752052,
  array([ 0.52237162, -0.26335492,  0.58125401,  0.56561105])),
 (0.9212209307072244,
  array([-0.37231836, -0.92555649, -0.02109478, -0.06541577])),
 (0.1473532783050957,
  array([-0.72101681,  0.24203288,  0.14089226,  0.6338014 ])),
 (0.020607707235625165,
  array([ 0.26199559, -0.12413481, -0.80115427,  0.52354627]))]
```

7 Matriz de Componentes

[6]:
```
acp_iris.componentes
```

[6]:

	0	1	2	3
0	-2.264542	-0.505704	-0.121943	0.023073
1	-2.086426	0.655405	-0.227251	0.103208
2	-2.367950	0.318477	0.051480	0.027825
3	-2.304197	0.575368	0.098860	-0.066311
4	-2.388777	-0.674767	0.021428	-0.037397
..	…	…	…	…
145	1.870522	-0.382822	0.254532	0.388890
146	1.558492	0.905314	-0.025382	0.221322
147	1.520845	-0.266795	0.179277	0.118903
148	1.376391	-1.016362	0.931405	0.024146
149	0.959299	0.022284	0.528794	-0.163676

[150 rows x 4 columns]

8 Calidades de Individuos

[7]:
```
acp_iris.calidades_ind
```

[7]:

	0	1	2	3
0	0.949782	0.047365	0.002754	0.000099
1	0.898483	0.088659	0.010659	0.002199
2	0.981644	0.017757	0.000464	0.000136
3	0.938948	0.058545	0.001728	0.000778
4	0.925826	0.073873	0.000074	0.000227
..	…	…	…	…
145	0.906103	0.037953	0.016778	0.039166
146	0.736450	0.248503	0.000195	0.014852

```
147  0.951672  0.029287  0.013224  0.005817
148  0.499126  0.272159  0.228562  0.000154
149  0.749903  0.000405  0.227862  0.021831

[150 rows x 4 columns]
```

9 Coordenadas de las variables

[9]:
```
acp_iris.coordenadas_var
```

[9]:
```
          0         1         2         3
0  0.891224 -0.357352 -0.276774  0.037610
1 -0.449313 -0.888351  0.092908 -0.017820
2  0.991684 -0.020247  0.054084 -0.115009
3  0.964996 -0.062786  0.243295  0.075157
```

10 Calidad de Variables

[10]:
```
acp_iris.calidades_var
```

[10]:
```
          0         1         2         3
0  0.794281  0.127701  0.076604  0.001415
1  0.201882  0.789168  0.008632  0.000318
2  0.983438  0.000410  0.002925  0.013227
3  0.931217  0.003942  0.059192  0.005649
```

11 Inercias

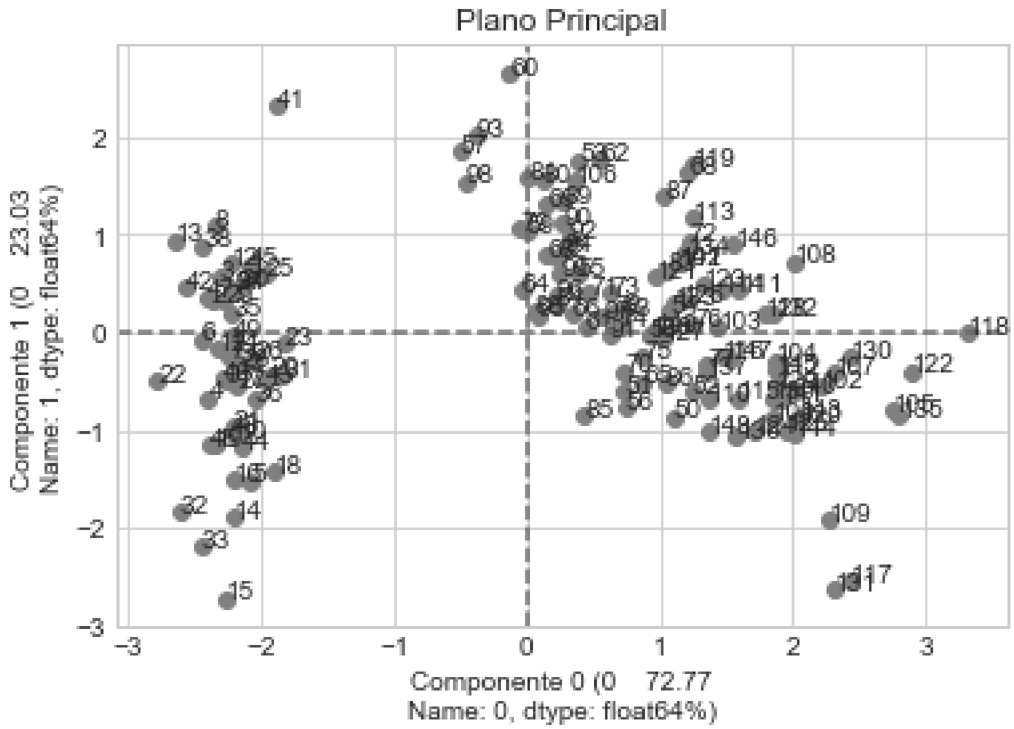
[11]:
```
acp_iris.inercias
```

[11]:
```
           0          1         2         3
0  72.770452  23.030523  3.683832  0.515193
```

12 Plots

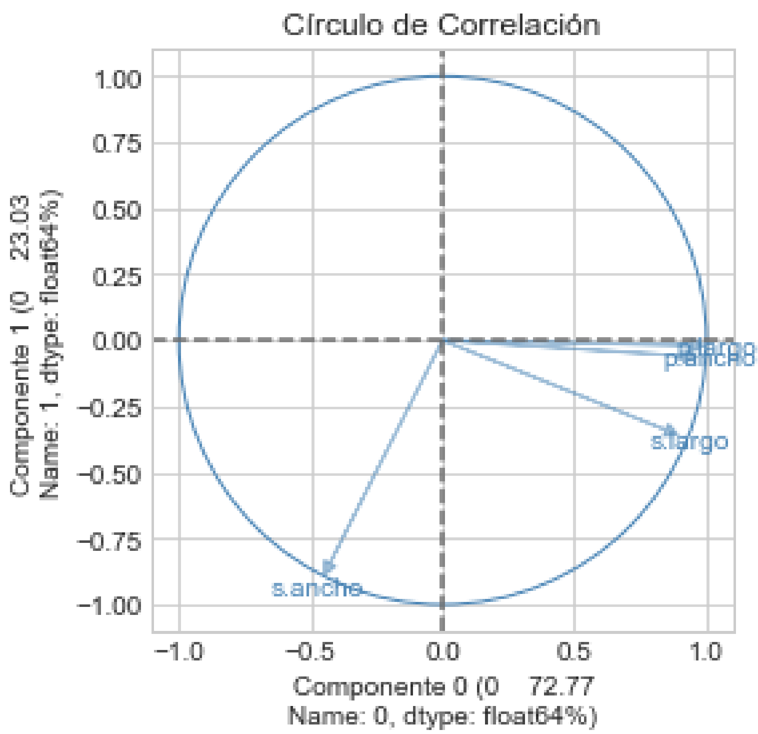
13 Plano principal

[12]:
```
acp_iris.plot_plano_principal()
```



14 Circulo

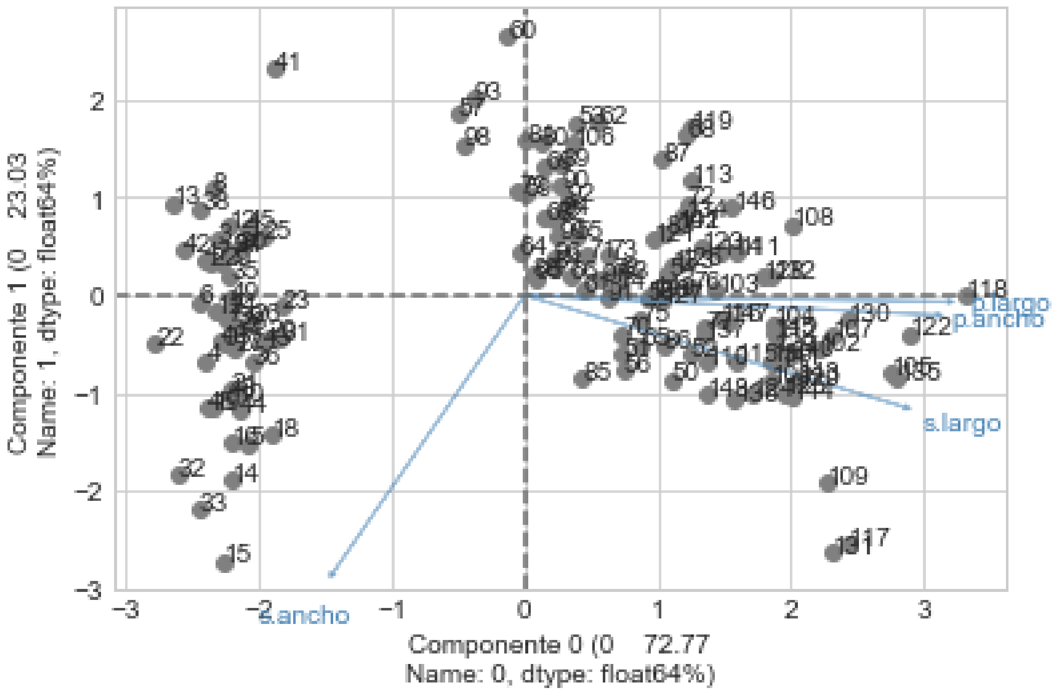
[39]:
```python
acp_iris.plot_circulo() #Grafico invertido, segun el profesor es normal
```



15 Sobreposicion

[15]:

```
acp_iris.plot_sobreposicion()
```



16 Prueba dataset Estudiantes con ACP propia

[16]:
```
estudiantes = pd.read_csv('EjemploEstudiantes.csv', delimiter=';', decimal=',',␣
↪header=0, index_col=0)
acp_estudiantes = mynew_PCA(estudiantes)
```

17 Matriz de Correlaciones

[17]:
```
acp_estudiantes.correlaciones
```

[17]:

	0	1	2	3	4
Matematicas	1.000000	0.854079	0.384574	0.207194	-0.787163
Ciencias	0.854079	1.000000	-0.020052	-0.021539	-0.687721
Espanol	0.384574	-0.020052	1.000000	0.820916	-0.365543
Historia	0.207194	-0.021539	0.820916	1.000000	-0.508001
EdFisica	-0.787163	-0.687721	-0.365543	-0.508001	1.000000

18 Valores y Vectores Propios

[18]:
```
acp_estudiantes.varvec_propios
```

[18]:
```
[(2.893249673417943,
  array([-0.52664397, -0.42493622, -0.35914704, -0.35269747,  0.53730181])),
 (1.6286504249773153,
  array([-0.2704963 , -0.50807221,  0.56208159,  0.58648985,  0.09374599])),
 (0.3465960485145294,
  array([-0.43820071, -0.04049491, -0.56227583,  0.39418032, -0.57862603])),
 (0.12261245959725253,
  array([-0.26121779,  0.67362724, -0.07008647,  0.44664495,  0.52305619])),
 (0.008891393492955155,
  array([-0.62387762,  0.32538951,  0.48374732, -0.42043348, -0.30679407]))]
```

19 Matriz de Componentes

[20]:
```
acp_estudiantes.componentes
```

[20]:

	0	1	2	3	4
Lucia	-0.323063	1.772525	-1.198801	-0.055015	-0.003633
Pedro	-0.665441	-1.638702	-0.145476	-0.023065	0.123377
Ines	-1.002547	-0.515692	-0.628888	0.516444	-0.142876
Luis	3.172095	-0.262782	0.381960	0.677777	0.062504
Andres	0.488868	1.365402	0.835236	-0.155792	-0.123367
Ana	-1.708633	-1.021700	0.127077	0.066833	-0.025292
Carlos	-0.067586	1.462336	0.506240	-0.117928	-0.013124
Jose	-2.011855	-1.275865	0.542150	-0.197787	-0.017434
Sonia	3.042030	-1.254881	-0.448829	-0.639999	-0.037885
Maria	-0.923869	1.369359	0.029330	-0.071467	0.177730

20 Calidades individuos

[21]:
```
acp_estudiantes.calidades_ind
```

[21]:

	0	1	2	3	4
0	0.022271	0.670421	0.306660	0.000646	0.000003
1	0.139906	0.848431	0.006687	0.000168	0.004809
2	0.514469	0.136123	0.202440	0.136520	0.010449
3	0.936852	0.006429	0.013584	0.042771	0.000364
4	0.084140	0.656354	0.245604	0.008545	0.005358
5	0.732686	0.261980	0.004053	0.001121	0.000161
6	0.001893	0.886081	0.106192	0.005763	0.000071
7	0.673612	0.270910	0.048917	0.006510	0.000051
8	0.808830	0.137637	0.017607	0.035800	0.000125
9	0.308554	0.677869	0.000311	0.001846	0.011419

21 Coordenadas variables

[22]:
```
acp_estudiantes.coordenadas_var
```

[22]:

	0	1	2	3	4
0	-0.895798	-0.345204	-0.257979	-0.091468	-0.058828
1	-0.722798	-0.648395	-0.023840	0.235878	0.030682
2	-0.610893	0.717321	-0.331025	-0.024542	0.045615
3	-0.599923	0.748470	0.232063	0.156397	-0.039644
4	0.913926	0.119637	-0.340651	0.183154	-0.028929

22 Calidades variables

[24]:
```
acp_estudiantes.calidades_var
```

[24]:

	0	1	2	3	4
0	0.802454	0.119166	0.066553	0.008366	0.003461
1	0.522436	0.420416	0.000568	0.055638	0.000941
2	0.373190	0.514549	0.109578	0.000602	0.002081
3	0.359907	0.560207	0.053853	0.024460	0.001572
4	0.835262	0.014313	0.116043	0.033545	0.000837

23 Inercias

[25]:
```
acp_estudiantes.inercias
```

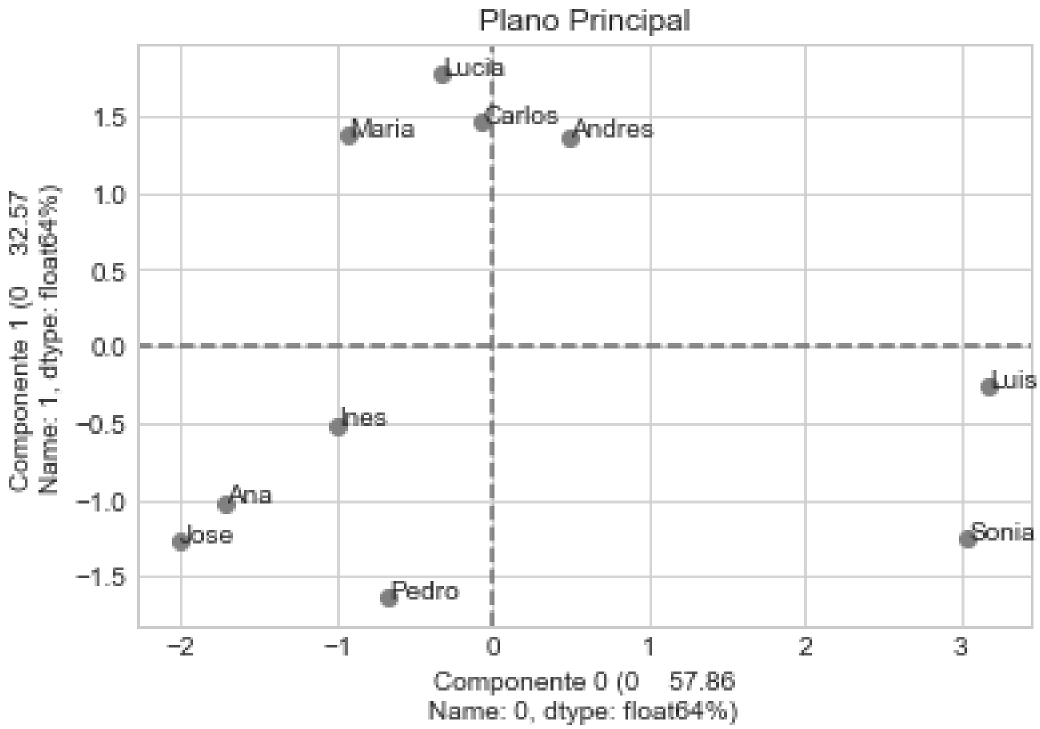
[25]:

	0	1	2	3	4
0	57.864993	32.573008	6.931921	2.452249	0.177828

24 Plots

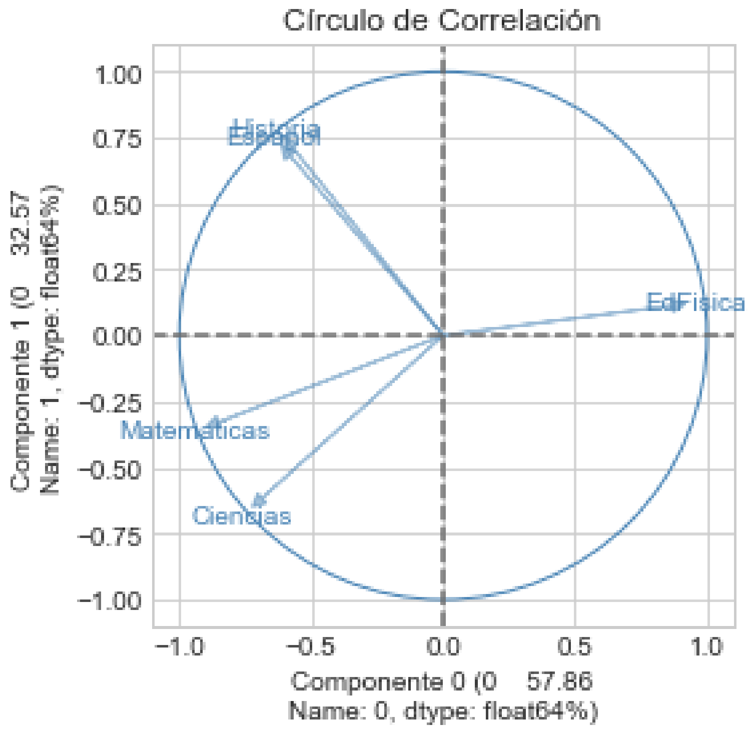
25 Plano Principal

[28]:
```
acp_estudiantes.plot_plano_principal()
```



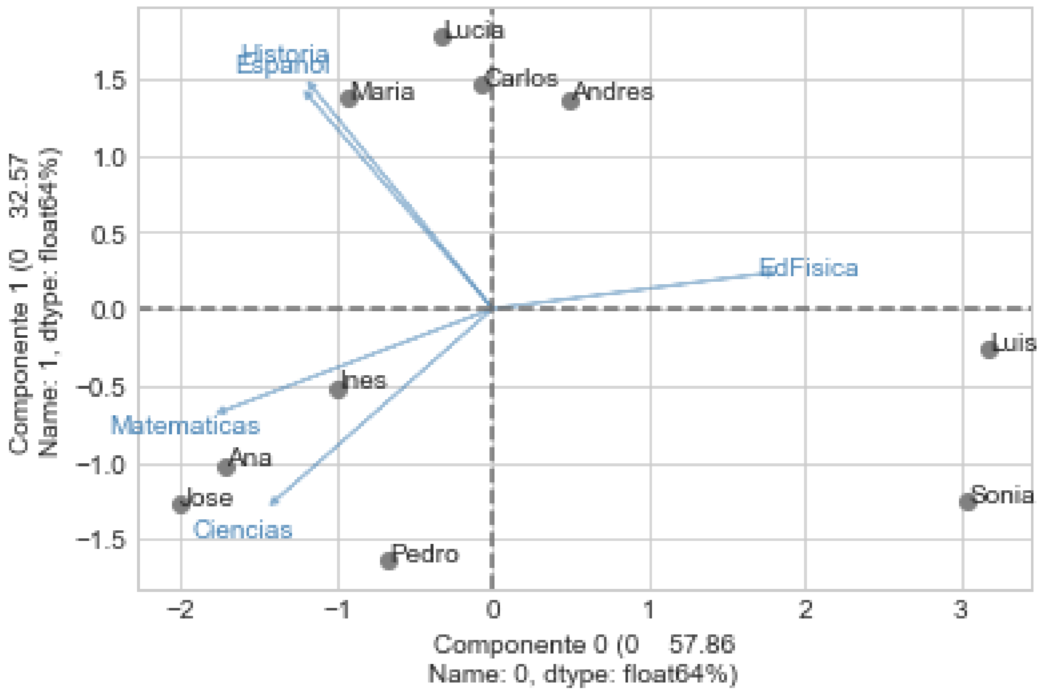
26 Circulo

[29]:
```
acp_estudiantes.plot_circulo()
```



27 Sobreposicion

[31]:
```
acp_estudiantes.plot_sobreposicion()
```



28 Carga Modelo PCA con Prince

[34]:
```
from prince import PCA

class ACP:
    def __init__(self, datos, n_componentes = 5):
        self.__datos = datos
        self.__modelo = PCA(n_components = n_componentes).fit(self.__datos)
        self.__correlacion_var = self.__modelo.column_correlations(datos)
        self.__coordenadas_ind = self.__modelo.row_coordinates(datos)
        self.__contribucion_ind = self.__modelo.row_contributions(datos)
        self.__cos2_ind = self.__modelo.row_cosine_similarities(datos)
        self.__var_explicada = [x * 100 for x in self.__modelo.
 ↪explained_inertia_]
    @property
    def datos(self):
        return self.__datos
    @datos.setter
    def datos(self, datos):
        self.__datos = datos
    @property
    def modelo(self):
        return self.__modelo
```

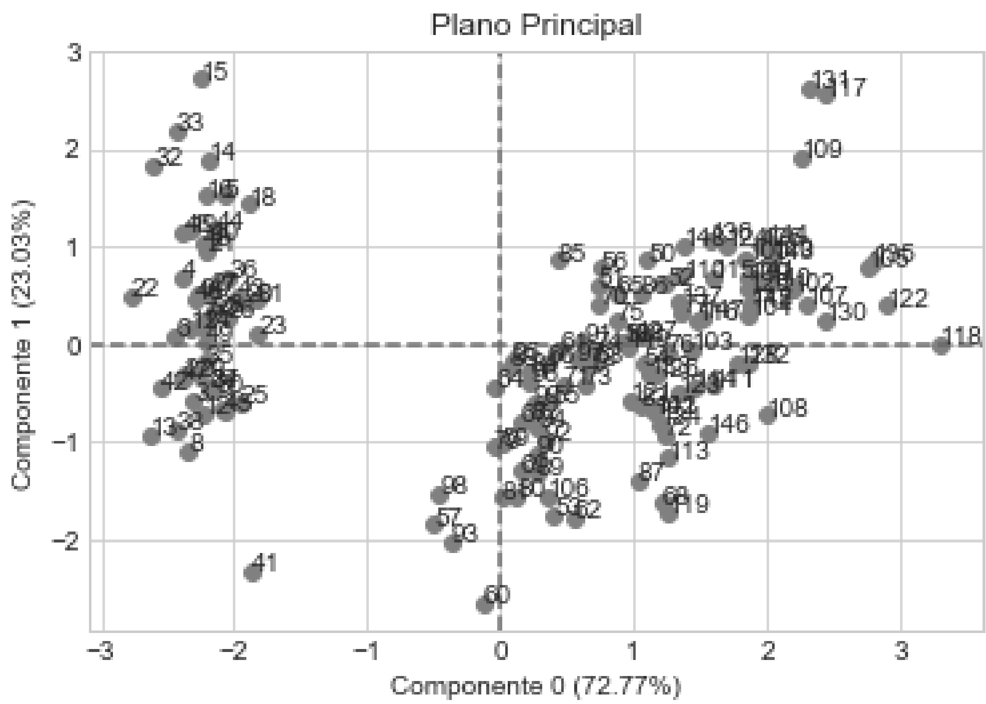
```
    @property
    def correlacion_var(self):
        return self.__correlacion_var
    @property
    def coordenadas_ind(self):
        return self.__coordenadas_ind
    @property
    def contribucion_ind(self):
        return self.__contribucion_ind
    @property
    def cos2_ind(self):
        return self.__cos2_ind
    @property
    def var_explicada(self):
        return self.__var_explicada
        self.__var_explicada = var_explicada
    def plot_plano_principal(self, ejes = [0, 1], ind_labels = True, titulo = 'Plano Principal'):
        x = self.coordenadas_ind[ejes[0]].values
        y = self.coordenadas_ind[ejes[1]].values
        plt.style.use('seaborn-whitegrid')
        plt.scatter(x, y, color = 'gray')
        plt.title(titulo)
        plt.axhline(y = 0, color = 'dimgrey', linestyle = '--')
        plt.axvline(x = 0, color = 'dimgrey', linestyle = '--')
        inercia_x = round(self.var_explicada[ejes[0]], 2)
        inercia_y = round(self.var_explicada[ejes[1]], 2)
        plt.xlabel('Componente ' + str(ejes[0]) + ' (' + str(inercia_x) + '%)')
        plt.ylabel('Componente ' + str(ejes[1]) + ' (' + str(inercia_y) + '%)')
        if ind_labels:
            for i, txt in enumerate(self.coordenadas_ind.index):
                plt.annotate(txt, (x[i], y[i]))
    def plot_circulo(self, ejes = [0, 1], var_labels = True, titulo = 'Círculo de Correlación'):
        cor = self.correlacion_var.iloc[:, ejes].values
        plt.style.use('seaborn-whitegrid')
        c = plt.Circle((0, 0), radius = 1, color = 'steelblue', fill = False)
        plt.gca().add_patch(c)
        plt.axis('scaled')
        plt.title(titulo)
        plt.axhline(y = 0, color = 'dimgrey', linestyle = '--')
        plt.axvline(x = 0, color = 'dimgrey', linestyle = '--')
        inercia_x = round(self.var_explicada[ejes[0]], 2)
        inercia_y = round(self.var_explicada[ejes[1]], 2)
        plt.xlabel('Componente ' + str(ejes[0]) + ' (' + str(inercia_x) + '%)')
        plt.ylabel('Componente ' + str(ejes[1]) + ' (' + str(inercia_y) + '%)')
        for i in range(cor.shape[0]):
```

```
            plt.arrow(0, 0, cor[i, 0] * 0.95, cor[i, 1] * 0.95, color =␣
↪'steelblue',
                      alpha = 0.5, head_width = 0.05, head_length = 0.05)
            if var_labels:
                plt.text(cor[i, 0] * 1.05, cor[i, 1] * 1.05, self.
↪correlacion_var.index[i],
                         color = 'steelblue', ha = 'center', va = 'center')
    def plot_sobreposicion(self, ejes = [0, 1], ind_labels = True,
                      var_labels = True, titulo = 'Sobreposición␣
↪Plano-Círculo'):
        x = self.coordenadas_ind[ejes[0]].values
        y = self.coordenadas_ind[ejes[1]].values
        cor = self.correlacion_var.iloc[:, ejes]
        scale = min((max(x) - min(x)/(max(cor[ejes[0]]) - min(cor[ejes[0]]))),
                    (max(y) - min(y)/(max(cor[ejes[1]]) - min(cor[ejes[1]])))) ␣
↪* 0.7
        cor = self.correlacion_var.iloc[:, ejes].values
        plt.style.use('seaborn-whitegrid')
        plt.axhline(y = 0, color = 'dimgrey', linestyle = '--')
        plt.axvline(x = 0, color = 'dimgrey', linestyle = '--')
        inercia_x = round(self.var_explicada[ejes[0]], 2)
        inercia_y = round(self.var_explicada[ejes[1]], 2)
        plt.xlabel('Componente ' + str(ejes[0]) + ' (' + str(inercia_x) + '%)')
        plt.ylabel('Componente ' + str(ejes[1]) + ' (' + str(inercia_y) + '%)')
        plt.scatter(x, y, color = 'gray')
        if ind_labels:
            for i, txt in enumerate(self.coordenadas_ind.index):
                plt.annotate(txt, (x[i], y[i]))
        for i in range(cor.shape[0]):
            plt.arrow(0, 0, cor[i, 0] * scale, cor[i, 1] * scale, color =␣
↪'steelblue',
                      alpha = 0.5, head_width = 0.05, head_length = 0.05)
            if var_labels:
                plt.text(cor[i, 0] * scale * 1.15, cor[i, 1] * scale * 1.15,
                         self.correlacion_var.index[i],
                         color = 'steelblue', ha = 'center', va = 'center')
```

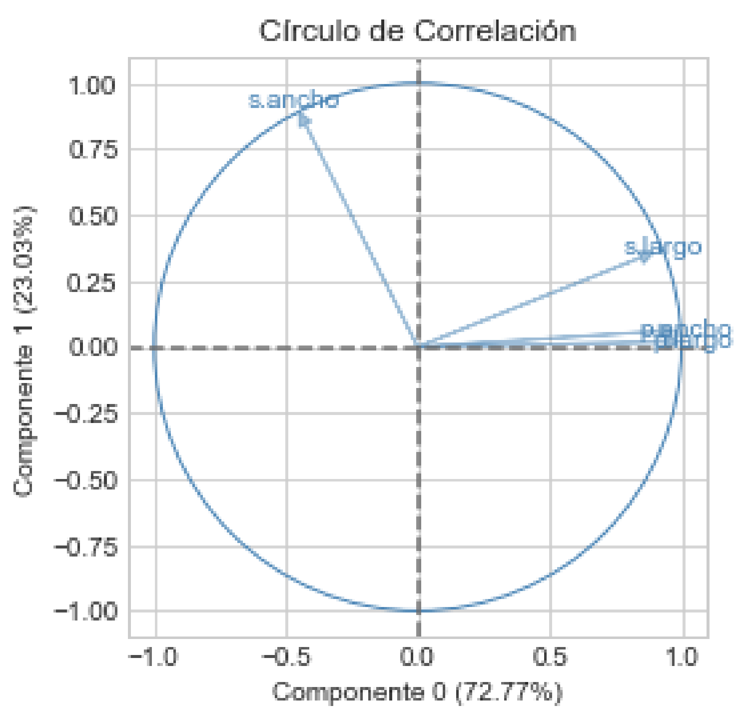
29 Plots de comparacion

30 Iris

[36]:
```
acp_prince_iris = ACP(iris.iloc[:, 0:4])
acp_prince_iris.plot_plano_principal()
```

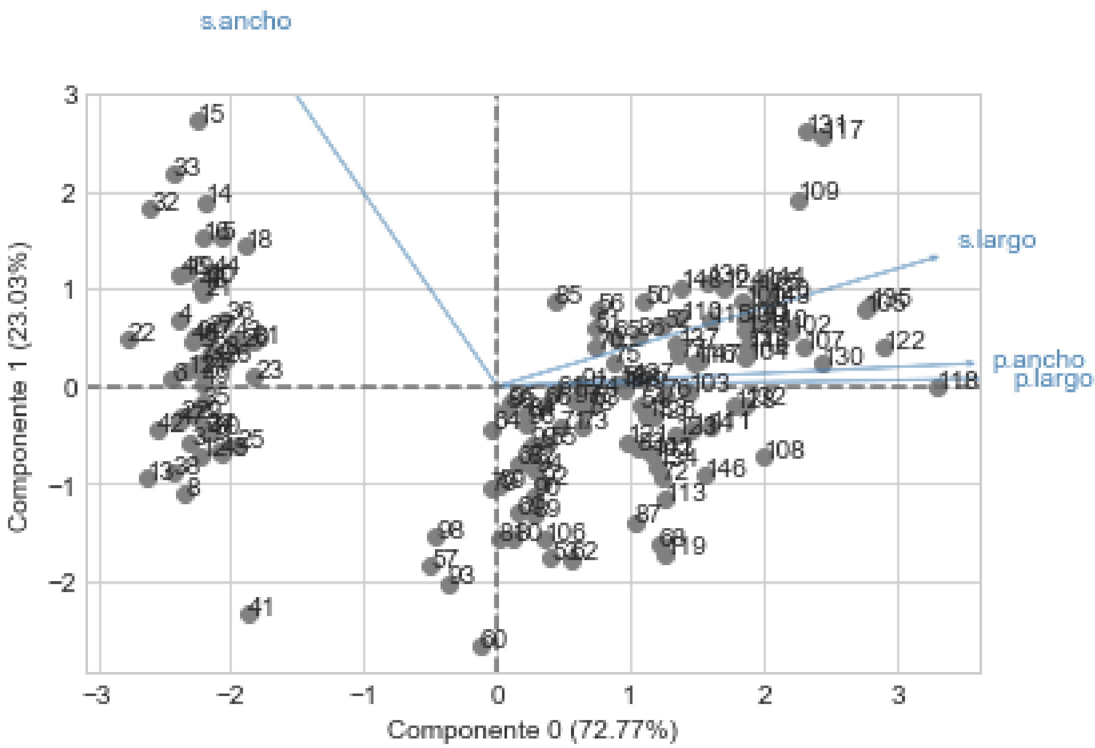


[37]:
```
acp_prince_iris.plot_circulo()
```



[38]:

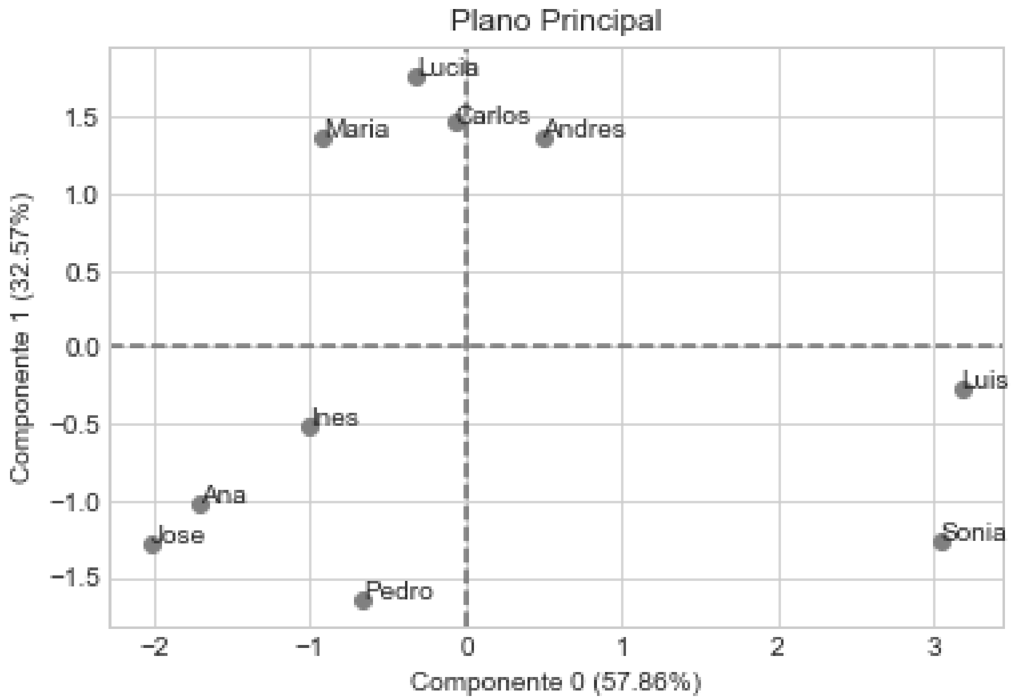
```
acp_prince_iris.plot_sobreposicion()
```



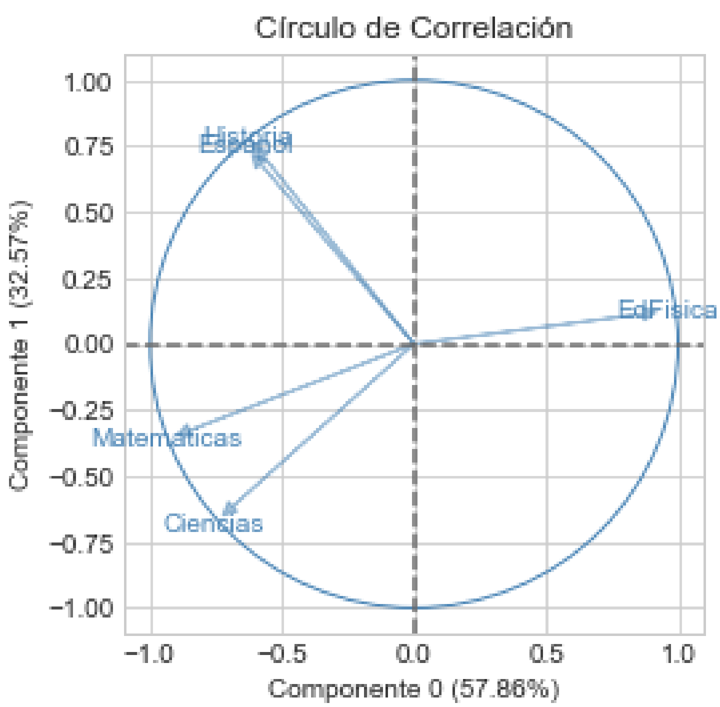
31 Plots de Estudiantes

[40]:

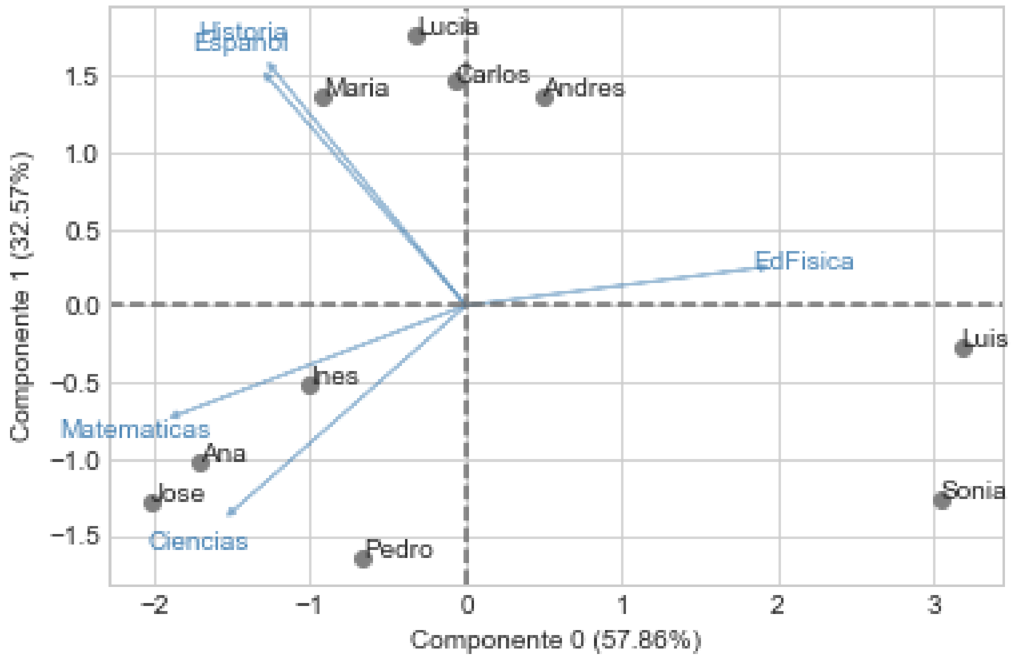
```
acp_prince_estudiantes = ACP(estudiantes)
acp_prince_estudiantes.plot_plano_principal()
```



[41]:
```
acp_prince_estudiantes.plot_circulo()
```



[42]:
```
acp_prince_estudiantes.plot_sobreposicion()
```



32 — FINAL —

[ ]:
```
```