

Tarea_4_Franco_Díaz_Nicolás_Vallejos

December 19, 2022

1 Tarea 4 - Franco Díaz - Nicolás Vallejos

1.1 Variables

Las siguientes variables serán las estudiadas y analizadas en los métodos posteriores:

sexo: sexo del estudiante

edad: edad del estudiante (meses)

imce: índice de masa corporal estandarizado

vive_padre: si el padre vive en el hogar

vive_madre: si la madre vive en el hogar

area: urbana=1, rural=0

sk1: muestra afecto a padres (1: siempre - 5: nunca)

sk2: muestra afecto a sus pares (1: siempre - 5: nunca)

sk3: expresa sus sentimientos (1: siempre - 5: nunca)

sk4: usa gestos para mostrar sentimientos (1: siempre - 5: nunca)

sk5: juega con otros (1: siempre - 5: nunca)

sk6: comparte sus cosas con otros (1: siempre - 5: nunca)

sk7: es agresivo (1: siempre - 5: nunca)

sk8: participa en juegos grupales (1: siempre - 5: nunca)

sk9: hace preguntas a adultos (1: siempre - 5: nunca)

sk10: tiene interes por libros (1: siempre - 5: nunca)

sk11: tiene interes por su entorno (1: siempre - 5: nunca)

sk12: juega a armar y desarmar cosas (1: siempre - 5: nunca)

sk13: tiene expresiones artisticas (1: siempre - 5: nunca)

act_fisica: frecuencia actividad fisica (1: nunca - 5: 5 o mas veces a la semana)

educm: años de escolaridad de la madre

educp: años de escolaridad del padre

madre_work: si la madre trabaja (-1: labor domestica, 0: desempleada, 1: empleada)

1.2 Preguntas

1. Cargue la base de datos y realice los ajustes necesarios para su uso (missing values, recodificar variables, etcetera). Identifique los tipos de datos que se encuentran en la base, realice estadísticas descriptivas sobre las variables importantes (Hint: Revisar la distribuciones, datos faltantes, outliers, etc.) y limpie las variables cuando sea necesario.
2. Usando las variables sk1-sk13 realice un PCA. En particular, identifique los valores propios y determine el número óptimo de componentes. Luego estime y grafique la distribución de los componentes. Además discuta la importancia relativa de las variables sobre cada uno de los componentes estimados. ¿Que se puede concluir de este análisis?
3. Con los resultados de la Pregunta 2, mantenga los primeros 3 componentes principales. Gráficamente indique si existen diferencias significativas entre grupos usando las siguientes variables: sexo, area, madre_work y act_fisica. ¿Que puede concluir de los resultados?
4. A partir del mismo set de variables sk1-sk13 realice un EFA. En particular determine el número óptimo de factores y las variables que se asocian a cada factor. También discuta si existen variables que no son informativas (Hint: para realizar un EFA, todas las variables deben estar representadas en el mismo sentido lógico. Si una característica es negativa debe ser invertida en la escala, de tal forma que todas las variables representen aspectos positivos).
5. Con los resultados obtenidos en la Pregunta 4, proponga un CFA donde cada variable solo se asocia con un factor. Entregue un nombre a cada factor que representa el concepto común entre todas las variables. Reporte la importancia de cada medida (variable) a cada factor e indique la correlación entre factores.
6. Finalmente, implemente un SEM completo usando la estructura propuesta en la Pregunta 5. En particular, estime un modelo donde los factores explican el nivel de actividad física, junto con otras variables que existen en la base de datos. Además utilice otras variables relevantes de la base de datos para explicar los factores latentes. Las variables a incluir en el modelo final deben tener sustento teórico y el modelo final debe optimizar el ajuste a los datos, en base a los criterios vistos en clase. ¿Que puede concluir en base a sus resultados?

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
import sklearn
import scipy
from scipy.linalg import eigh, cholesky
from scipy.stats import norm
import linearmodels.panel as lmp
from pylab import plot, show, axis, subplot, xlabel, ylabel, grid
import semopy
import seaborn as sns
from factor_analyzer import FactorAnalyzer
```

```

from sklearn.decomposition import PCA
from pandas.plotting import scatter_matrix

%matplotlib inline

```

1.2.1 Pregunta 1

```

[2]: # Leemos la data a partir del archivo junaeb2.csv
junaeb2 = pd.read_csv('../junaeb2.csv')
junaeb2.dropna(inplace=True)
junaeb2.reset_index(drop=True, inplace=True)

# Se invierte la variable sk7 para que tenga el mismo sentido que las demás
↪ variables
for i in range(len(junaeb2["sk7"])):
    if junaeb2["sk7"][i]==1:
        junaeb2["sk7"][i]=5
    elif junaeb2["sk7"][i]==2:
        junaeb2["sk7"][i]=4
    elif junaeb2["sk7"][i]==4:
        junaeb2["sk7"][i]=2
    elif junaeb2["sk7"][i]==5:
        junaeb2["sk7"][i]=1
    else:
        junaeb2["sk7"][i]=3

# Se eliminan los valores 2 en las variables vive padre y vive madre, ya que
↪ deben ser binarias
delet_p = junaeb2[junaeb2["vive_padre"] == 2].index
junaeb2 = junaeb2.drop(delet_p)
delet_m = junaeb2[junaeb2["vive_madre"] == 2].index
junaeb2 = junaeb2.drop(delet_m)

```

<ipython-input-2-c5f1d8e82995>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
junaeb2["sk7"][i]=2
```

<ipython-input-2-c5f1d8e82995>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
junaeb2["sk7"][i]=4
```

<ipython-input-2-c5f1d8e82995>:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
junaeb2["sk7"][i]=3
```

<ipython-input-2-c5f1d8e82995>:9: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
junaeb2["sk7"][i]=5
```

<ipython-input-2-c5f1d8e82995>:15: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
junaeb2["sk7"][i]=1
```

```
[3]: print(junaeb2["sk7"].value_counts())
```

```
5    20608
```

```
4    15357
```

```
3    12827
```

```
2     4384
```

```
1     4071
```

```
Name: sk7, dtype: int64
```

```
[4]: datosNP= junaeb2.to_numpy()
```

```
[5]: # Se seleccionan las variables desde sk1 a sk13, construyendo el vector X.
```

```
X= datosNP[:,5:18]
```

```
from sklearn.preprocessing import StandardScaler
```

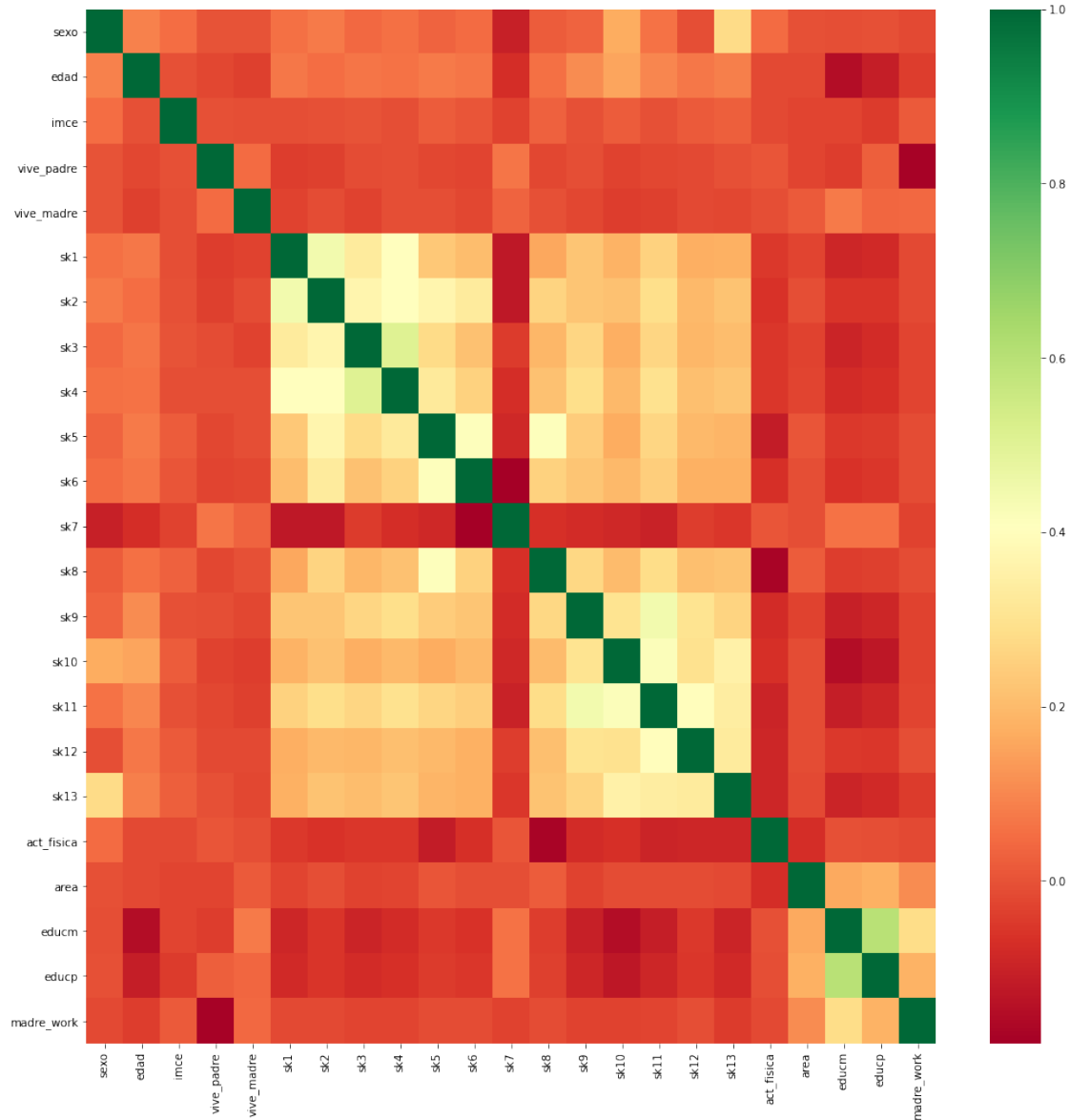
```
# Normalizamos la matriz de la data con media 0 y varianza 1
```

```
X = StandardScaler().fit_transform(X)
```

```
[6]: plt.figure(figsize = (17,17))
```

```
sns.heatmap(junaeb2.corr(), cmap='RdYlGn') #mapa de correlaciones entre las  
↪ variables
```

```
[6]: <AxesSubplot:>
```



En base al cuadro de calor es posible notar que existe una muy baja correlación entre variables en especial las variables sk1-sk13 y el resto de variables demográficas como sexo, edad y área. Del mismo modo se aprecia que las únicas correlaciones importantes se dan entre las variables sk3 y sk4, lo cual se genera dado que ambas variables capturan la expresión de los sentimientos, tanto en su expresión como en gestos. De igual forma los años de educación de la madre y los años de educación del padre tienen una correlación positiva, esto implica que al aumentar los años de educación de cada uno, es más probable que su pareja también evidencie un aumento en sus años de educación.

A simple vista, es posible observar que dada la baja correlación entre variables sk1-sk13, de modo que aplicar un PCA puede no reducir de manera drástica el número de variables a partir de los nuevos componentes.

1.2.2 Pregunta 2

1.3 PCA

Primero se utilizó la función de máxima verosimilitud “mle” para determinar el número óptimo de eigenvalores (valores propios) basados en la varianza de la data.

Luego, se realiza un Scree Plot que muestra el porcentaje de varianza que cada componente contribuye.

Finalmente, podemos usar los componentes estimados y transformarlos en un data frame y estudiar los pesos de cada una de las variables (sk1-sk13) para cada componente y analizar sus distribuciones. También se muestra que los componentes son ortogonales, es decir, poseen una covarianza cero.

```
[7]: # Ocupamos estimador de máxima verosimilitud, para estimar
      # solo el número de factores óptimos desde el punto de vista de la varianza
```

```
pca = PCA(n_components='mle')
pca_features = pca.fit_transform(X)
print(pca.explained_variance_ratio_)
```

```
[0.30878559 0.10430667 0.08490926 0.07684746 0.06018862 0.05493641
 0.05431338 0.05357955 0.04692229 0.04065751 0.03974286 0.03838525]
```

```
[8]: # Qué variables aparecen más fuertemente en cada componente
print(pca.components_)
```

```
[ [ 0.27646211  0.31849051  0.29182449  0.31950231  0.29652846  0.26712675
   -0.10121318  0.26091289  0.29307818  0.2621151  0.32976031  0.25945278
    0.25812207]
 [ 0.29585057  0.30991259  0.26885933  0.28695729  0.21301505  0.16873562
   -0.11066157 -0.02032742 -0.23508952 -0.39456247 -0.31768596 -0.3777505
   -0.34745368]
 [ 0.28871043  0.05283652  0.35682173  0.31234591 -0.37851236 -0.44698967
    0.41558586 -0.39475993  0.04357974  0.03013264  0.04389703  0.07316565
    0.08524766]
 [ 0.23631541  0.05216424 -0.06906239 -0.01757389 -0.33609258  0.01307991
   -0.79126978 -0.40014196 -0.02175799  0.17540581  0.06309294 -0.02126606
    0.0364846 ]
 [-0.13657751 -0.28296996  0.17167441  0.11090712 -0.0586397  -0.13457831
   -0.17403114  0.08774254  0.6393772  -0.23873933  0.23127778 -0.03651108
   -0.52853044]
 [ 0.4902273  0.27994285 -0.39761416 -0.23422759 -0.0502977  -0.38137456
    0.08439779  0.36360774  0.09296147  0.20063648  0.12559124 -0.25631559
   -0.2251043 ]
 [ 0.07991516  0.12153991 -0.2041359  -0.11990732  0.01316659  0.5445307
    0.35381636 -0.49704106  0.14023209  0.24417376  0.16591064 -0.03056198
   -0.38226074]
 [ 0.24378899  0.12123263 -0.21292324 -0.11825767  0.02681411 -0.03491715
   -0.02006871 -0.04026837 -0.10862524 -0.48094963  0.02225245  0.76738849]
```

```

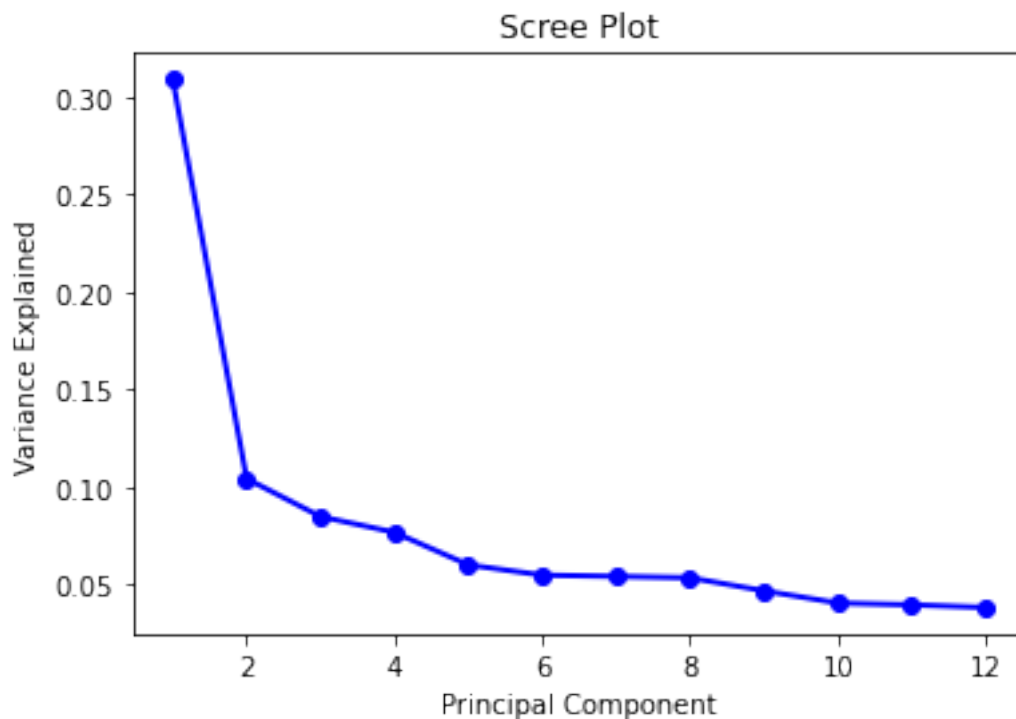
-0.17347519]
[ 0.20217762  0.01708776 -0.23655341 -0.10432365 -0.06877809  0.17773003
 0.08094735 -0.12241499  0.48506834 -0.48662128 -0.09943529 -0.25473262
 0.53597325]
[ 0.47807701 -0.70416249 -0.17130905  0.29040315  0.07450785  0.1572141
 0.03637968  0.10782611 -0.02728053  0.18048691 -0.28003779  0.07920765
-0.0261606 ]
[-0.01663711 -0.11246204 -0.22530159  0.14376253  0.70071536 -0.37853782
-0.06939313 -0.41584664 -0.0571706  -0.04751527  0.2823404  -0.11119439
 0.07748436]
[ 0.0952494  -0.21684317 -0.01577015  0.09563851 -0.23775311  0.19827265
 0.05469339  0.14883462 -0.40915007 -0.29003841  0.71838782 -0.20744217
 0.05375965]]

```

```

[9]: #scree plot para observar la varianza explicada
PC_values = np.arange(pca.n_components_) + 1
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2,
         color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()

```



Basado en los resultados, la data se puede resumir usando 12 componentes. Utilizando la función

de maxima verosimilitud, te tiene que el número de componentes (no correlacionados) a usar es 12, pudiendo reducir el número de variables solamente a n-1, de igual forma puede observarse a partir del Scree plot que a partir de 10 componentes no existe grandes variaciones en cuanto a la explicación de la varianza.

Estos resultados tienen mucho sentido con lo mostrado en el mapa de calor de la pregunta 1, en donde se observa que en general, entre las variables de estudio, existe una baja correlación.

Por otro lado, podemos observar que a partir del porcentaje de varianza el primer componente captura casi el 31% aproximado de la varianza.

```
[10]: #Se crea un dataframe con los componentes y las cada una de las variables para
      ↪ analizar los pesos quee
      #carga cada componente.
      pca_vectors = pd.DataFrame(data = pca.components_)
      pca_vectors.head(12)
```

```
[10]:
```

	0	1	2	3	4	5	6	\
0	0.276462	0.318491	0.291824	0.319502	0.296528	0.267127	-0.101213	
1	0.295851	0.309913	0.268859	0.286957	0.213015	0.168736	-0.110662	
2	0.288710	0.052837	0.356822	0.312346	-0.378512	-0.446990	0.415586	
3	0.236315	0.052164	-0.069062	-0.017574	-0.336093	0.013080	-0.791270	
4	-0.136578	-0.282970	0.171674	0.110907	-0.058640	-0.134578	-0.174031	
5	0.490227	0.279943	-0.397614	-0.234228	-0.050298	-0.381375	0.084398	
6	0.079915	0.121540	-0.204136	-0.119907	0.013167	0.544531	0.353816	
7	0.243789	0.121233	-0.212923	-0.118258	0.026814	-0.034917	-0.020069	
8	0.202178	0.017088	-0.236553	-0.104324	-0.068778	0.177730	0.080947	
9	0.478077	-0.704162	-0.171309	0.290403	0.074508	0.157214	0.036380	
10	-0.016637	-0.112462	-0.225302	0.143763	0.700715	-0.378538	-0.069393	
11	0.095249	-0.216843	-0.015770	0.095639	-0.237753	0.198273	0.054693	

	7	8	9	10	11	12
0	0.260913	0.293078	0.262115	0.329760	0.259453	0.258122
1	-0.020327	-0.235090	-0.394562	-0.317686	-0.377750	-0.347454
2	-0.394760	0.043580	0.030133	0.043897	0.073166	0.085248
3	-0.400142	-0.021758	0.175406	0.063093	-0.021266	0.036485
4	0.087743	0.639377	-0.238739	0.231278	-0.036511	-0.528530
5	0.363608	0.092961	0.200636	0.125591	-0.256316	-0.225104
6	-0.497041	0.140232	0.244174	0.165911	-0.030562	-0.382261
7	-0.040268	-0.108625	-0.480950	0.022252	0.767388	-0.173475
8	-0.122415	0.485068	-0.486621	-0.099435	-0.254733	0.535973
9	0.107826	-0.027281	0.180487	-0.280038	0.079208	-0.026161
10	-0.415847	-0.057171	-0.047515	0.282340	-0.111194	0.077484
11	0.148835	-0.409150	-0.290038	0.718388	-0.207442	0.053760

```
[11]: pca_df = pd.DataFrame(data=pca_features,columns=['PC1', 'PC2',
      ↪ 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC12'])
      pca_df.describe().apply(lambda s: s.apply('{0:.3f}'.format))
```



```
[11]:
```

	PC1	PC2	PC3	PC4	PC5	PC6	\
count	57247.000	57247.000	57247.000	57247.000	57247.000	57247.000	
mean	0.000	-0.000	-0.000	-0.000	0.000	-0.000	
std	2.004	1.164	1.051	1.000	0.885	0.845	
min	-2.042	-8.196	-6.845	-5.165	-5.330	-6.206	
25%	-1.474	-0.593	-0.590	-0.643	-0.461	-0.411	
50%	-0.539	0.149	0.124	-0.076	0.086	0.041	
75%	0.853	0.555	0.567	0.659	0.411	0.402	
max	19.331	9.662	9.615	5.327	8.109	8.396	

	PC7	PC8	PC9	PC10	PC11	PC12
count	57247.000	57247.000	57247.000	57247.000	57247.000	57247.000
mean	-0.000	-0.000	-0.000	0.000	0.000	-0.000
std	0.840	0.835	0.781	0.727	0.719	0.706
min	-5.623	-5.332	-5.295	-6.095	-5.278	-4.816
25%	-0.455	-0.451	-0.440	-0.276	-0.362	-0.370
50%	0.039	0.095	0.070	0.039	0.111	0.062
75%	0.458	0.321	0.351	0.366	0.314	0.294
max	5.706	5.708	5.453	6.213	6.427	4.930

```
[12]: pca_df.corr().apply(lambda s: s.apply('{0:.3f}'.format))
```

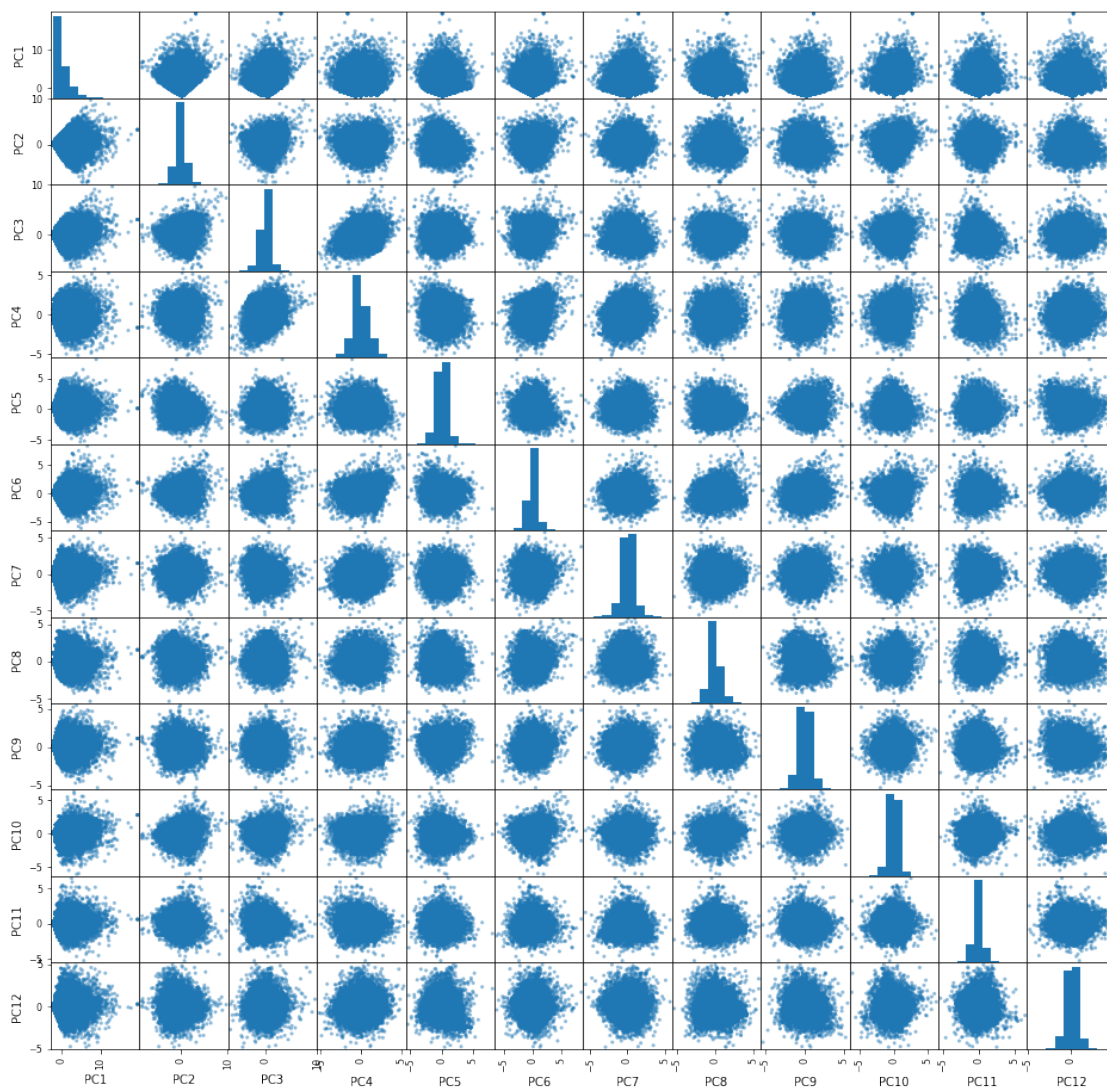
```
[12]:
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	\
PC1	1.000	0.000	0.000	0.000	-0.000	0.000	-0.000	0.000	0.000	
PC2	0.000	1.000	0.000	0.000	-0.000	0.000	0.000	0.000	0.000	
PC3	0.000	0.000	1.000	-0.000	-0.000	-0.000	0.000	0.000	0.000	
PC4	0.000	0.000	-0.000	1.000	0.000	0.000	-0.000	0.000	0.000	
PC5	-0.000	-0.000	-0.000	0.000	1.000	-0.000	-0.000	-0.000	-0.000	
PC6	0.000	0.000	-0.000	0.000	-0.000	1.000	0.000	0.000	0.000	
PC7	-0.000	0.000	0.000	-0.000	-0.000	0.000	1.000	0.000	0.000	
PC8	0.000	0.000	0.000	0.000	-0.000	0.000	0.000	1.000	0.000	
PC9	0.000	0.000	0.000	0.000	-0.000	0.000	0.000	0.000	1.000	
PC10	0.000	0.000	0.000	0.000	-0.000	0.000	0.000	0.000	0.000	
PC11	-0.000	-0.000	0.000	0.000	-0.000	-0.000	0.000	0.000	0.000	
PC12	0.000	-0.000	0.000	-0.000	-0.000	0.000	-0.000	0.000	-0.000	

	PC10	PC11	PC12
PC1	0.000	-0.000	0.000
PC2	0.000	-0.000	-0.000
PC3	0.000	0.000	0.000
PC4	0.000	0.000	-0.000
PC5	-0.000	-0.000	-0.000
PC6	0.000	-0.000	0.000
PC7	0.000	0.000	-0.000
PC8	0.000	0.000	0.000
PC9	0.000	0.000	-0.000
PC10	1.000	-0.000	0.000

```
PC11  -0.000    1.000    0.000
PC12   0.000    0.000    1.000
```

```
[13]: scatter_matrix(pca_df[['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10', 'PC11', 'PC12']], figsize = (16, 16))
plt.show()
```



Se observa a partir de la matriz de correlaciones de los componentes que estos son ortogonales. Además, luego de transformar la predicción de componente en Data, es posible observar que cada uno de estos está nomaralizado con media 0 y la desviación estándar de cada factor es proporcional a la proporción de varianza que contribuye cada factor.

En cuanto a la distribución de los componentes, es posible observar que todos siguen una distribu-

ción aproximadamente normal, a excepción del primer componente (PC1) el cual se ajusta mas a una distribución exponencial. De igual manera, se observa que los componentes resultantes no se encuentran relacionados.

1.3.1 Pregunta 3

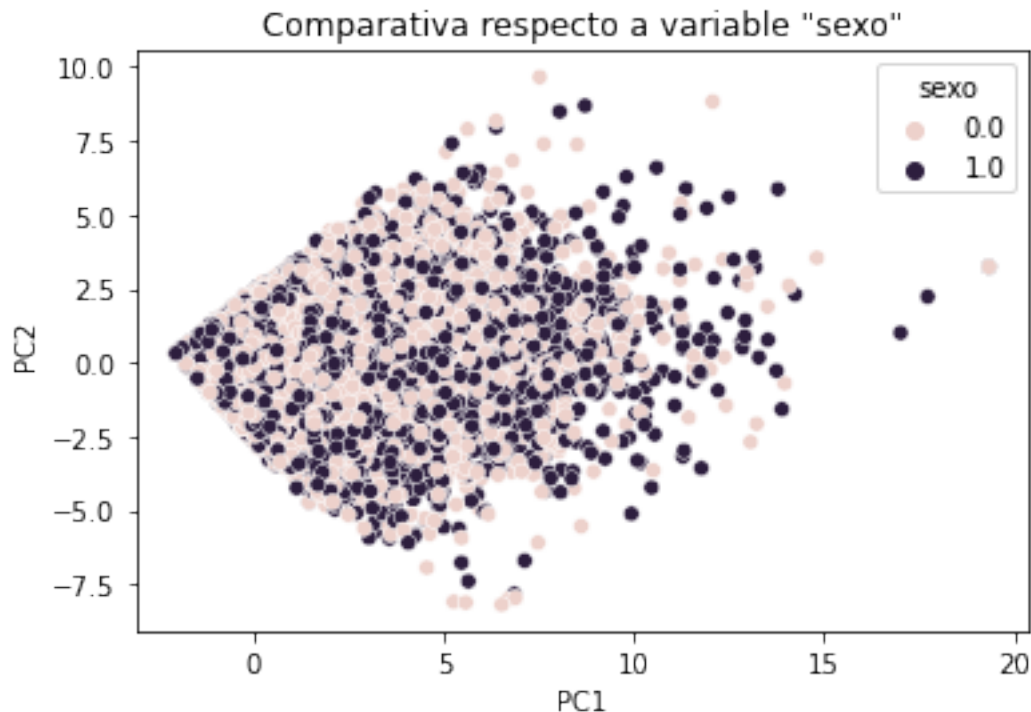
```
[14]: #Primero agregamos las columnas de la base de datos al data frame de los
      ↪componentes
pca_df['sexo'] = junaeb2['sexo']
pca_df['area'] = junaeb2['area']
pca_df['madre_work'] = junaeb2['madre_work']
pca_df['act_fisica'] = junaeb2['act_fisica']

[15]: # Se utiliza scatterplot para comparar grupos respecto a las siguientes
      ↪variables:
sns.scatterplot('PC1', 'PC2', data=pca_df, hue='sexo').set(title='Comparativa
      ↪respecto a variable "sexo"')
```

C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.

```
warnings.warn(
```

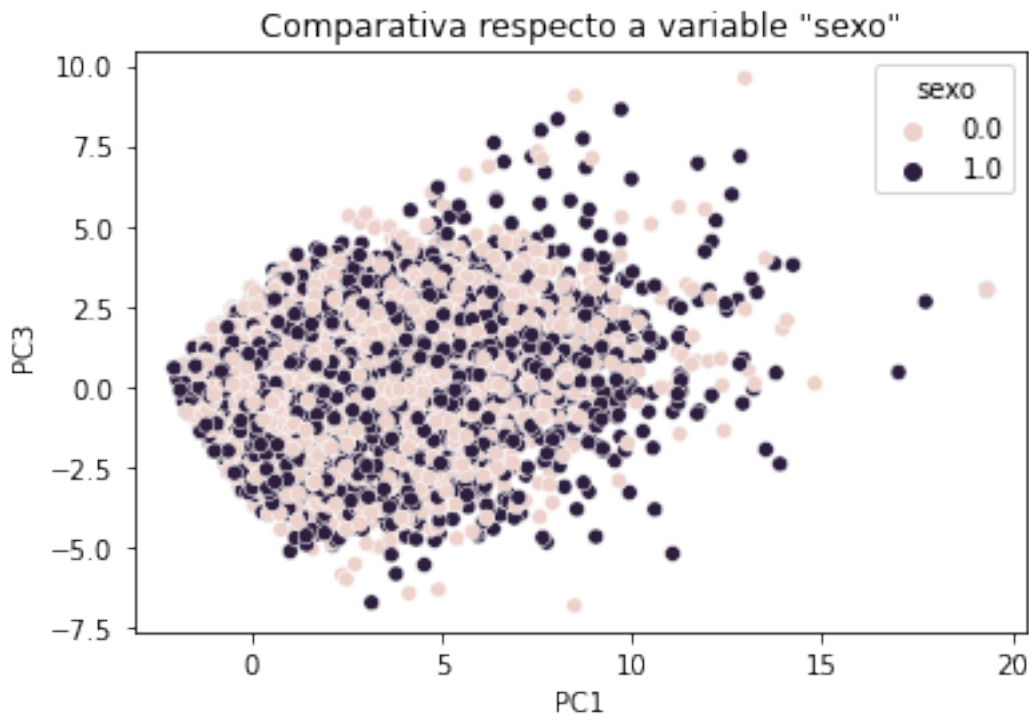
```
[15]: [Text(0.5, 1.0, 'Comparativa respecto a variable "sexo"')]
```



```
[16]: sns.scatterplot('PC1', 'PC3', data=pca_df, hue='sexo').set(title='Comparativa_
↳respecto a variable "sexo")
```

```
C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

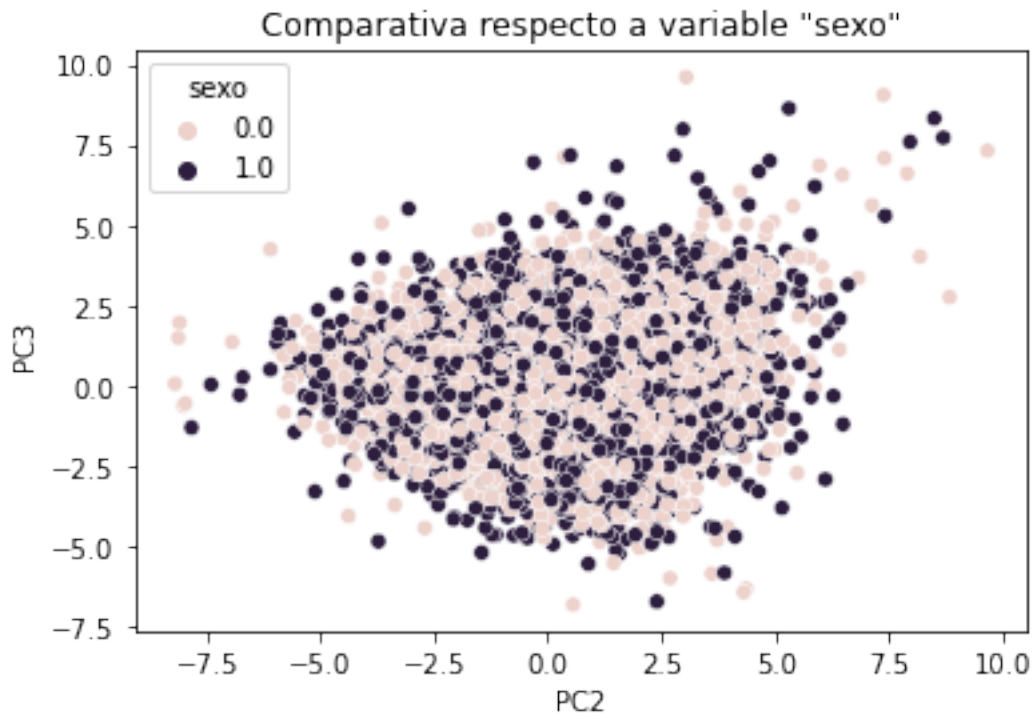
```
[16]: [Text(0.5, 1.0, 'Comparativa respecto a variable "sexo")]
```



```
[17]: sns.scatterplot('PC2', 'PC3', data=pca_df, hue='sexo').set(title='Comparativa_
↳respecto a variable "sexo")
```

```
C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

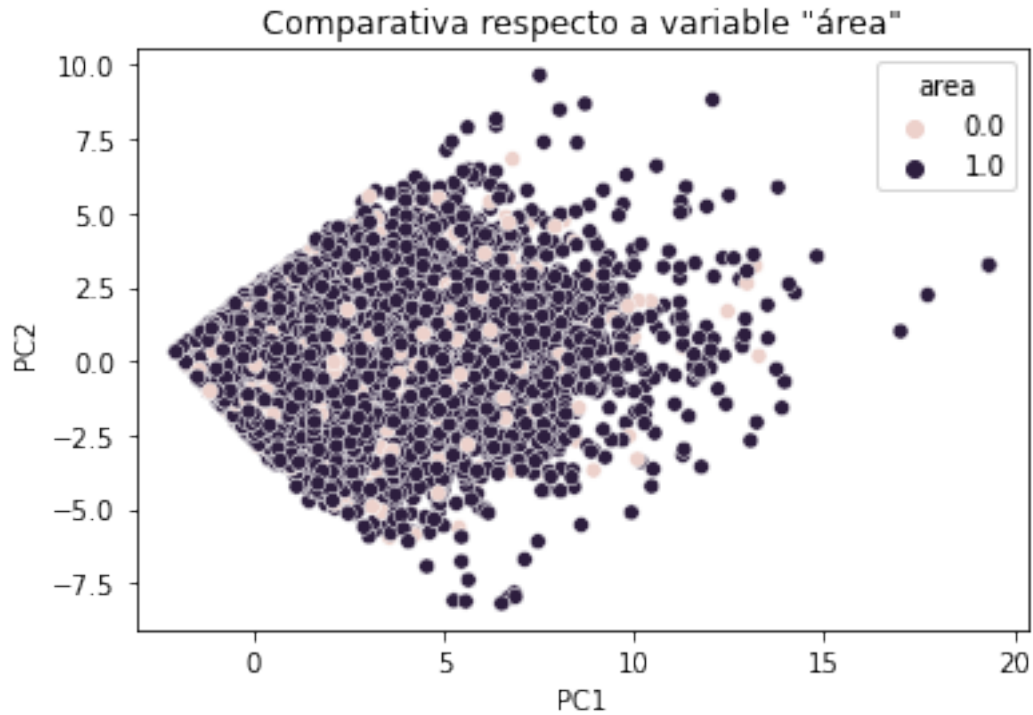
```
[17]: [Text(0.5, 1.0, 'Comparativa respecto a variable "sexo"')]
```



```
[18]: sns.scatterplot('PC1', 'PC2', data=pca_df, hue='area').set(title='Comparativa_
↳respecto a variable "área"')
```

```
C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

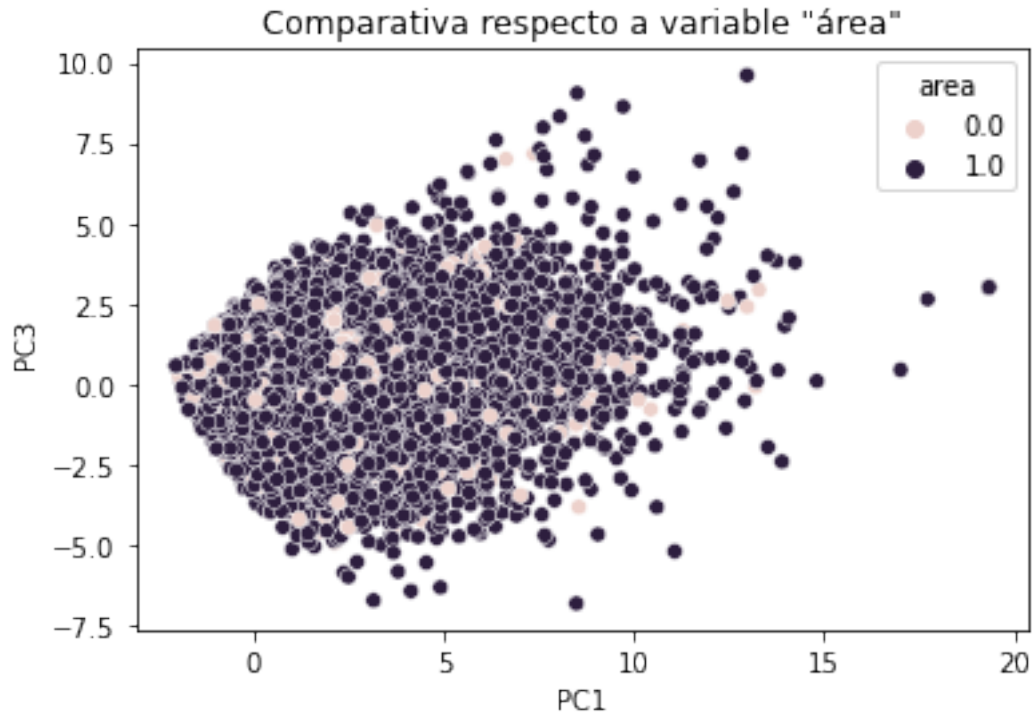
```
[18]: [Text(0.5, 1.0, 'Comparativa respecto a variable "área"')]
```



```
[19]: sns.scatterplot('PC1', 'PC3', data=pca_df, hue='area').set(title='Comparativa_
↳ respecto a variable "área")
```

C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
warnings.warn(

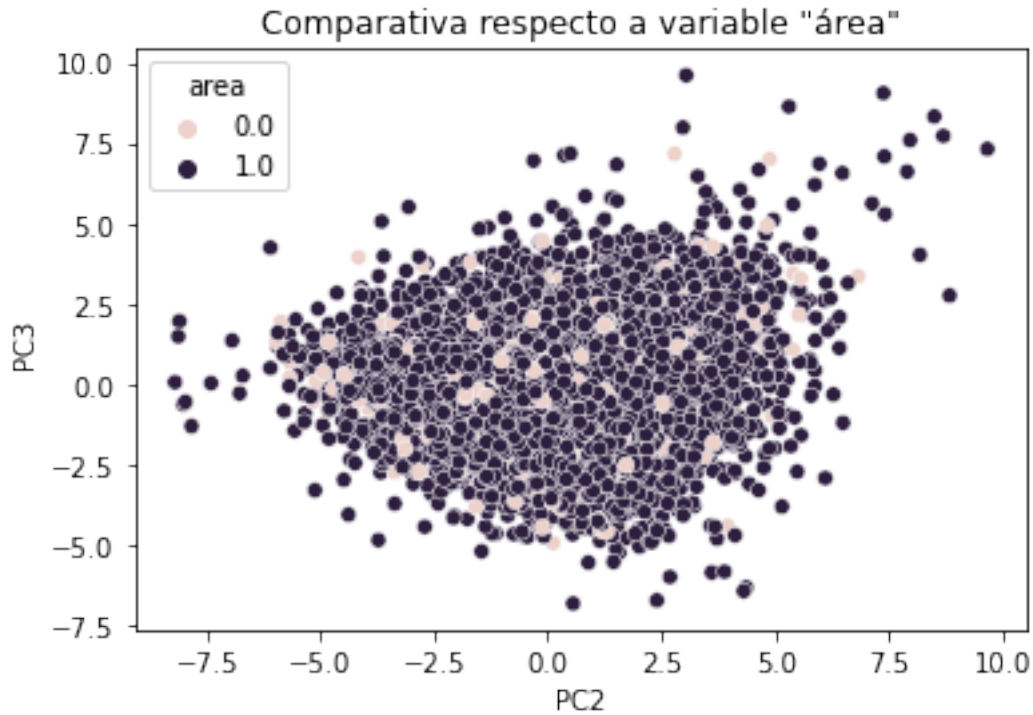
```
[19]: [Text(0.5, 1.0, 'Comparativa respecto a variable "área")]
```



```
[20]: sns.scatterplot('PC2', 'PC3', data=pca_df, hue='area').set(title='Comparativa_
↳ respecto a variable "área")
```

C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
warnings.warn(

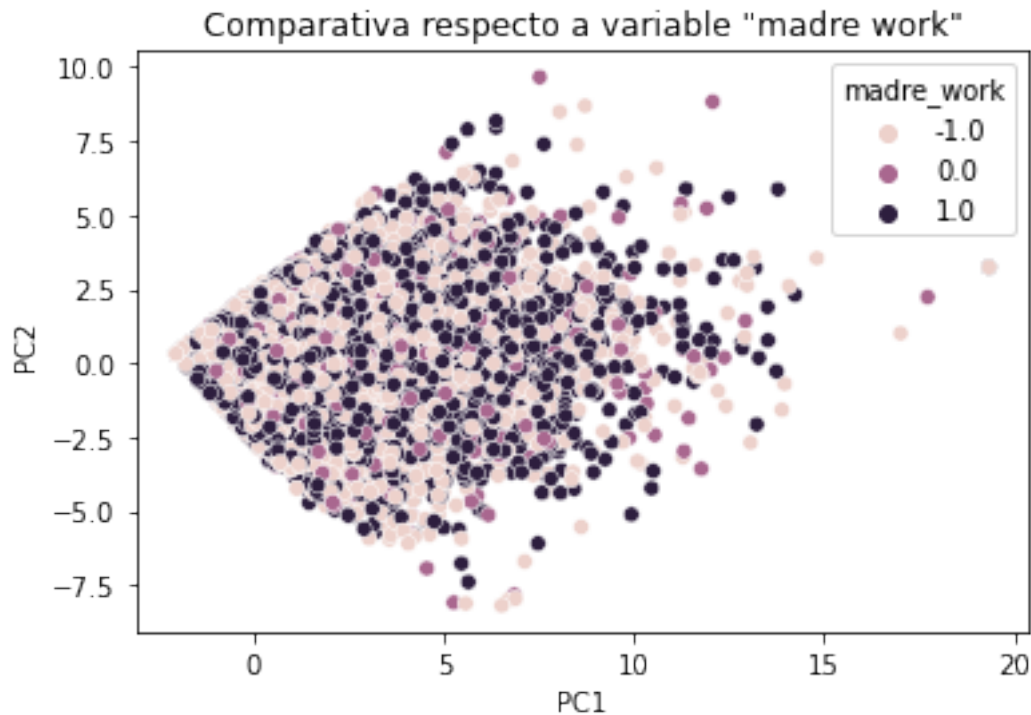
```
[20]: [Text(0.5, 1.0, 'Comparativa respecto a variable "área")]
```

```
[21]: sns.scatterplot('PC1', 'PC2', data=pca_df, hue='madre_work').
      ↪set(title='Comparativa respecto a variable "madre work"')
```

C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
warnings.warn(

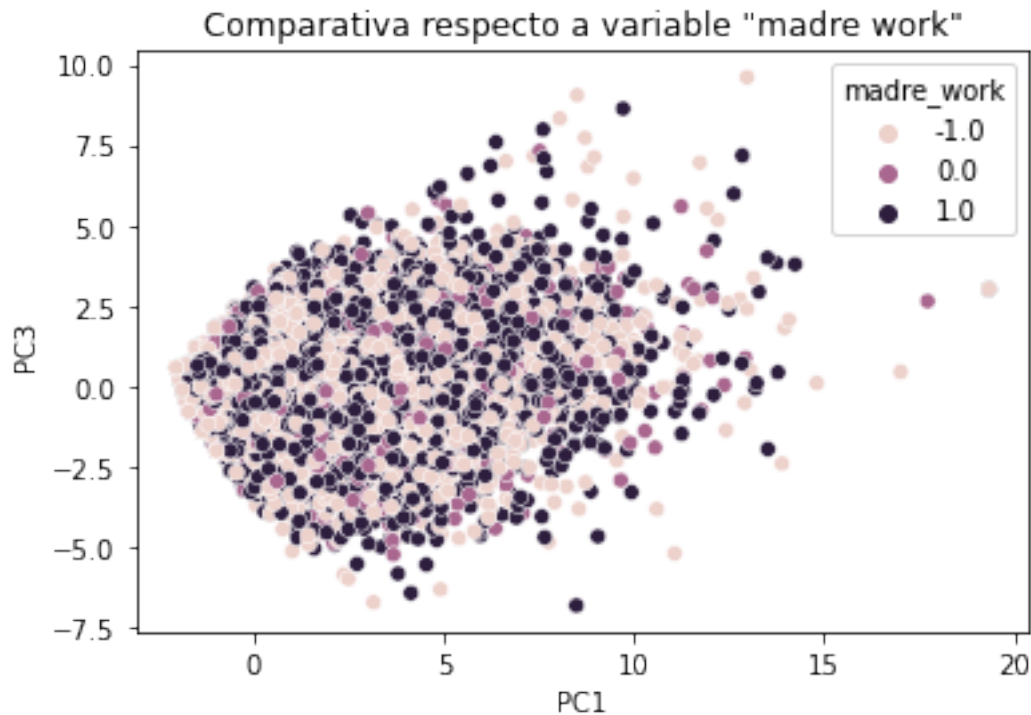
```
[21]: [Text(0.5, 1.0, 'Comparativa respecto a variable "madre work"')]
```

```
[22]: sns.scatterplot('PC1', 'PC3', data=pca_df, hue='madre_work').
      ↪set(title='Comparativa respecto a variable "madre work"')
```

C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
warnings.warn(

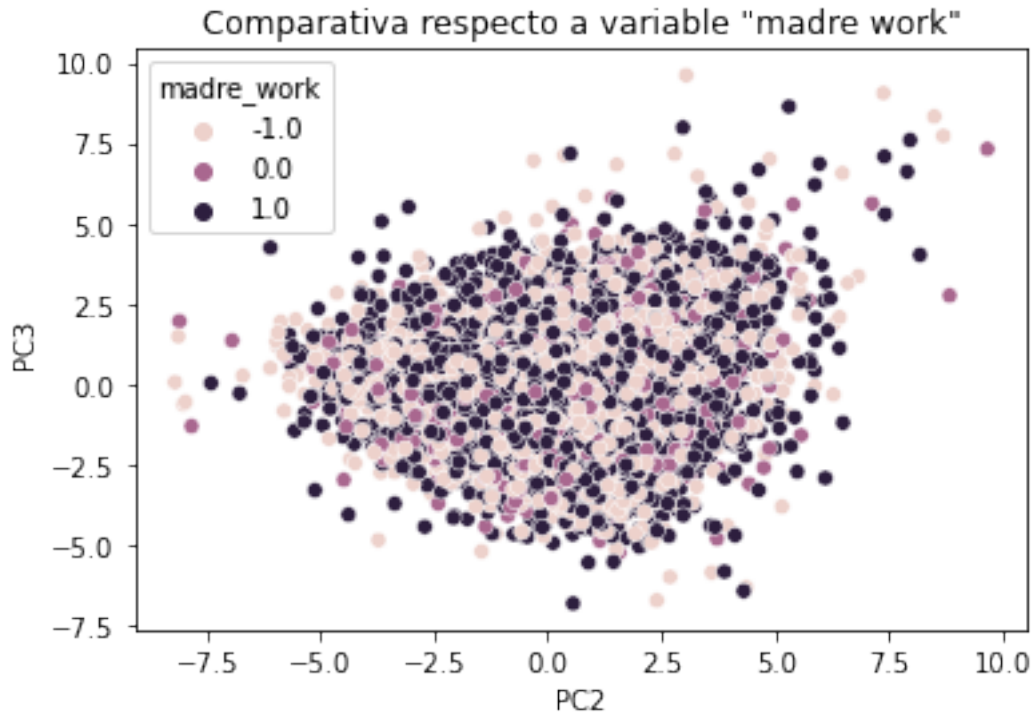
```
[22]: [Text(0.5, 1.0, 'Comparativa respecto a variable "madre work"')]
```



```
[23]: sns.scatterplot('PC2', 'PC3', data=pca_df, hue='madre_work').
      ↪set(title='Comparativa respecto a variable "madre work"')
```

C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
warnings.warn(

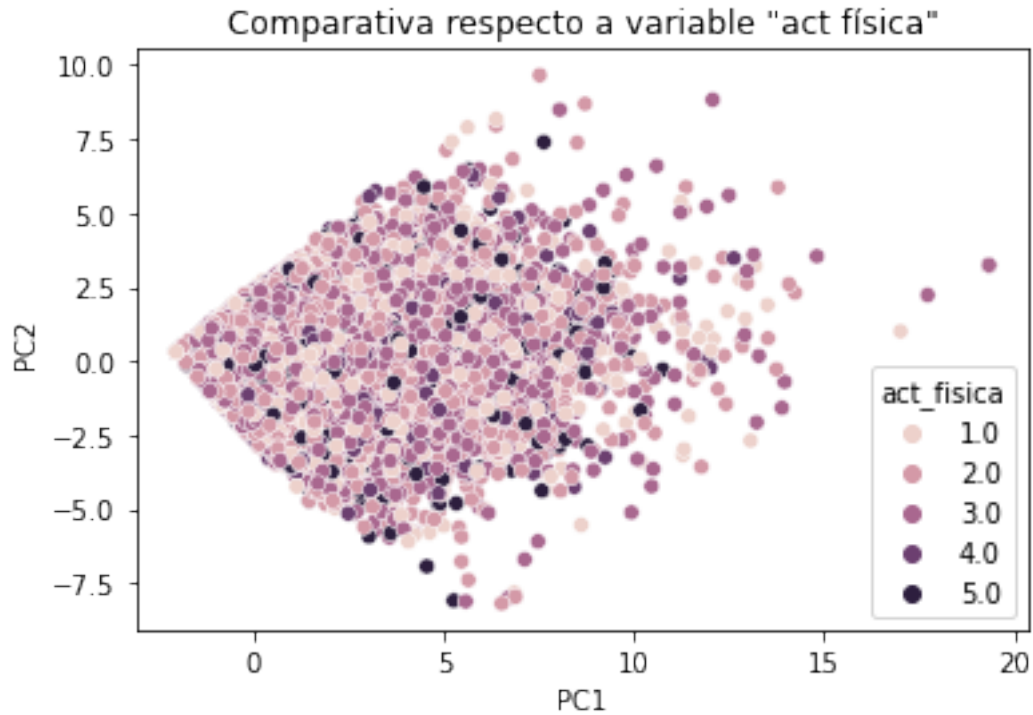
```
[23]: [Text(0.5, 1.0, 'Comparativa respecto a variable "madre work"')]
```



```
[24]: sns.scatterplot('PC1', 'PC2', data=pca_df, hue='act_fisica').
      ↪set(title='Comparativa respecto a variable "act física"')
```

C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
warnings.warn(

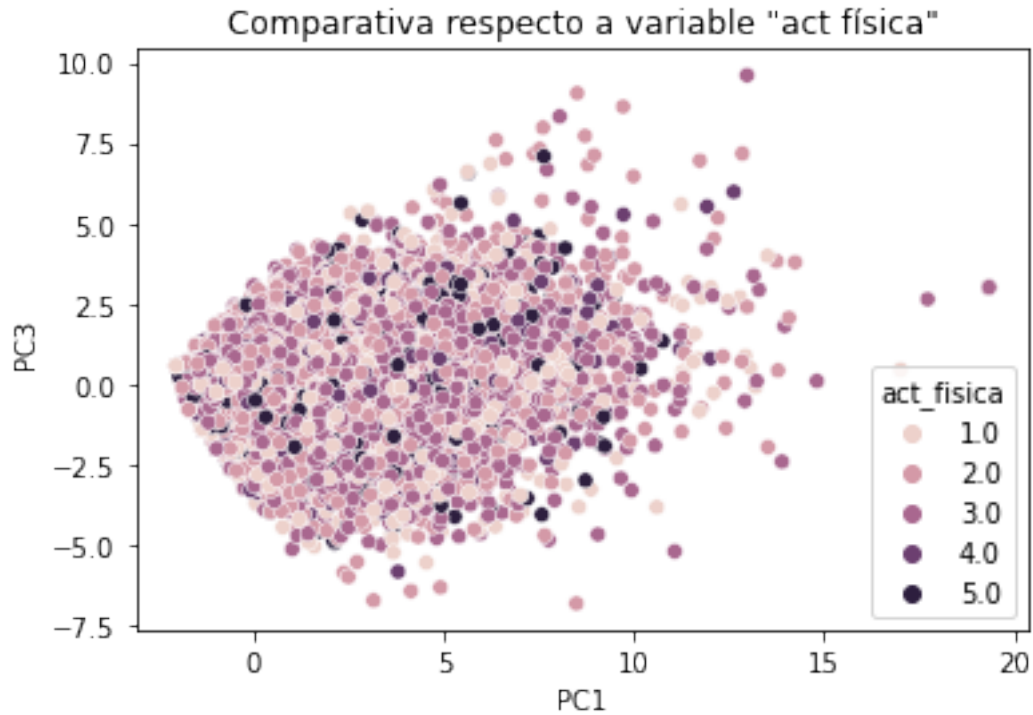
```
[24]: [Text(0.5, 1.0, 'Comparativa respecto a variable "act física"')]
```



```
[25]: sns.scatterplot('PC1', 'PC3', data=pca_df, hue='act_fisica').
      ↪set(title='Comparativa respecto a variable "act física"')
```

C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
warnings.warn(

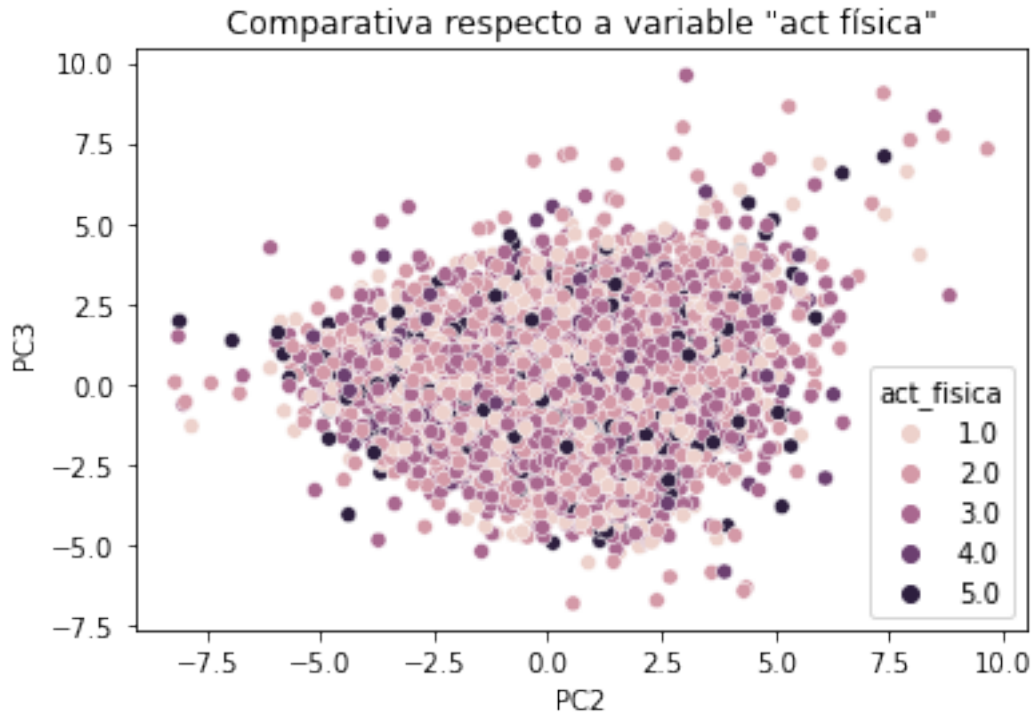
```
[25]: [Text(0.5, 1.0, 'Comparativa respecto a variable "act física"')]
```




```
[26]: sns.scatterplot('PC2', 'PC3', data=pca_df, hue='act_fisica').
      ↪set(title='Comparativa respecto a variable "act física"')
```

C:\Users\FDiazRiffo\anaconda3\lib\site-packages\seaborn_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
warnings.warn(

```
[26]: [Text(0.5, 1.0, 'Comparativa respecto a variable "act física"')]
```



Como resultado del análisis gráfico entre los primeros 3 componentes establecidos y variables “sexo”, “area”, “act_fisica” y “madre work” no es fácil establecer algún tipo de clasificación en base a estos componentes. En primera instancia esto parece ser lo mas lógico debido a que no se incluyen estas variables al momento de obtener estos componentes, así componentes principales no es capaz de capturar información respecto a las variables de estudio. 

1.3.2 Pregunta 4

1.4 EFA

```
[27]: # Se crea un objeto de análisis factorial y se realiza el análisis factorial
fa = FactorAnalyzer(rotation='promax')

# Creamos una variable auxiliar para ingresar las variables del modelo a
    ↪ utilizar
var_aux=
    ↪ junaeb2[["sk1", "sk2", "sk3", "sk4", "sk5", "sk6", "sk7", "sk8", "sk9", "sk10", "sk11", "sk12", "sk13"]]

fa.fit(var_aux)

[27]: FactorAnalyzer(rotation_kwargs={})

[28]: #Se obtienen los pesos relativos para cada variable y factor.
fa.loadings_
```

```
[28]: array([[ 0.01513617,  0.60642246, -0.03745664],
             [-0.03621966,  0.49631702,  0.23113835],
             [ 0.02425043,  0.64625691, -0.04354572],
             [ 0.00349246,  0.73969959, -0.03312391],
             [-0.14810408, -0.02621475,  0.86247599],
             [ 0.00590849,  0.04188805,  0.51846545],
             [-0.01588018, -0.03990937, -0.13987395],
             [ 0.14617103, -0.10754942,  0.51519416],
             [ 0.48366606,  0.08146774,  0.05186671],
             [ 0.61961759, -0.03468374, -0.03260351],
             [ 0.696416   ,  0.03005479,  0.00609085],
             [ 0.56966766, -0.02274258, -0.00423132],
             [ 0.52730608,  0.01921561, -0.01143569]])
```

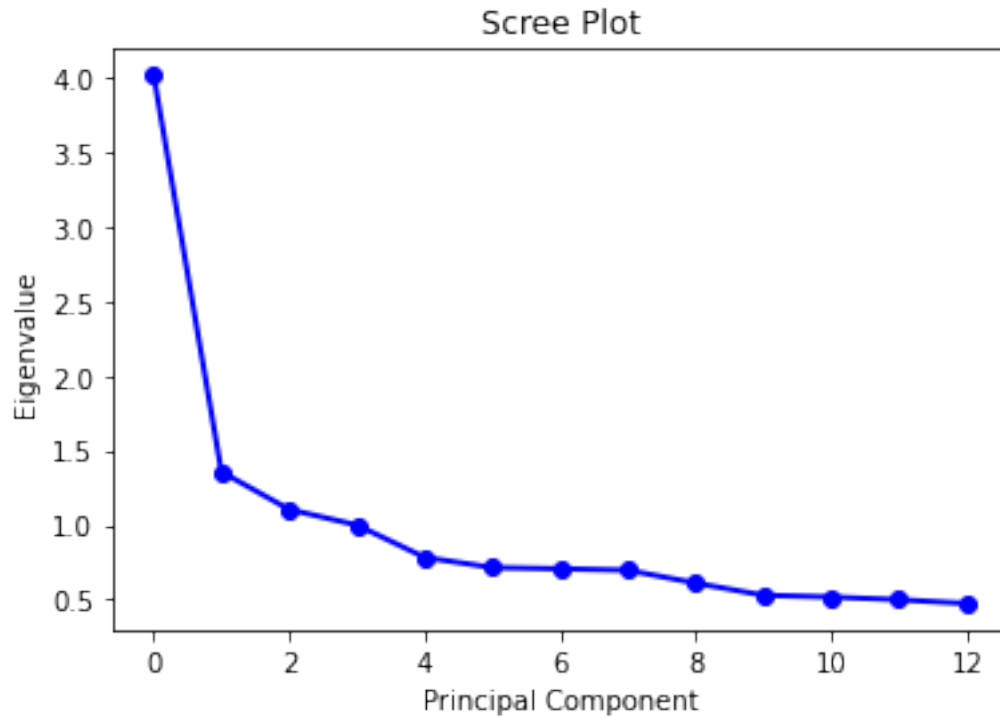
Primero se define la rotación con la que se llevará a cabo la normalización, para este caso se utiliza “promax”. Luego en base a la data y a partir del Análisis Factorial se estiman 3 factores y sus pesos relativos para cada variable.

Así, a cada factor en base a sus pesos relativos, se le asociarían las siguientes variables: - $F1$: $sk9$, $sk10$, $sk11$, $sk12$, $sk13$ - $F2$: $sk1, sk2, sk3, sk4$ - $F3$: $sk5$, $sk6$, $sk7$, $sk8$

```
[29]: fa.get_eigenvalues()
```

```
[29]: (array([4.01421272, 1.35598674, 1.10382041, 0.99901701, 0.78245204,
             0.71417333, 0.70607397, 0.69653413, 0.60998972, 0.52854767,
             0.51665716, 0.49900826, 0.47352684]),
      array([ 3.41985323,  0.7654264 ,  0.59258439,  0.20923866,  0.08734352,
             0.06289151,  0.03437131,  0.02794699, -0.02552849, -0.07612027,
             -0.09466116, -0.1133073 , -0.17534796]))
```

```
[30]: values = np.arange(0,13)
      eigenvalues = pd.DataFrame(data=fa.get_eigenvalues())
      plt.plot(values, eigenvalues.loc[0], 'o-', linewidth=2, color='blue')
      plt.title('Scree Plot')
      plt.xlabel('Principal Component')
      plt.ylabel('Eigenvalue')
      plt.show()
```



Como resultado del Scree Plot tenemos resultados similares a PCA, dado que es la misma data, de modo que a partir del 11vo componente, la contribución relativa es casi 0.

```
[31]: #Se obtiene 3 elemntos:
      #a.- varianza
      #b.- proporción explicada por cada factor
      #c.- acumulado
      fa.get_factor_variance()
```

```
[31]: (array([1.75115613, 1.60411343, 1.35942128]),
      array([0.13470432, 0.12339334, 0.10457087]),
      array([0.13470432, 0.25809766, 0.36266853]))
```

Finalmente mediante semopy se obtiene una aproximación a cual debería ser la representación del modelo, del cual se obtiene 4 factores latentes, considerando que la data entrega el mejor criterio de información y los criterios son significativos al 95% de confianza. El modelo estructural potencial obtenido se muestra a continuación.

```
[32]: print(semopy.efa.explore_cfa_model(var_aux, pval=0.05))
```

```
eta1 =~ sk11 + sk9 + sk10 + sk12
eta2 =~ sk7 + sk6
eta3 =~ sk4 + sk2 + sk11 + sk5 + sk3 + sk9 + sk1 + sk6 + sk8 + sk12
eta4 =~ sk11 + sk12 + sk13
```


Así, finalmente, a partir del Análisis factorial, los pesos relativos de cada variable y el modelo exploratorio (donde sk11, sk12, sk6 y sk9 son incorporados a mas de un factor), se tiene la siguiente propuesta de modelo estructural:

- $\eta_1 \sim \text{sk}_9 + \text{sk}_{10} + \text{sk}_{11} + \text{sk}_{12}$
- $\eta_2 \sim \text{sk}_7$
- $\eta_3 \sim \text{sk}_4 + \text{sk}_2 + \text{sk}_5 + \text{sk}_3 + \text{sk}_1 + \text{sk}_6 + \text{sk}_8$
- $\eta_4 \sim \text{sk}_{13}$

1.4.1 Pregunta 5

1.5 CFA

```
[33]: Xf=var_aux

mod = """
eta1 =~ sk11 + sk9 + sk10 + sk12
eta2 =~ sk7 + sk6
eta3 =~ sk4 + sk2 + sk11 + sk5 + sk3 + sk9 + sk1 + sk6 + sk8 + sk12
eta4 =~ sk11 + sk12 + sk13
      """

model = semopy.Model(mod)
out=model.fit(var_aux)
print(out)
```

Name of objective: MLW

Optimization method: SLSQP

Optimization successful.

Optimization terminated successfully

Objective value: 0.266

Number of iterations: 79

Params: 0.078 1.100 0.306 2.021 1.694 0.124 -0.295 -0.953 0.473 1.136 0.855
0.915 0.561 0.992 2.556 0.452 1.336 0.799 0.316 0.208 0.241 0.554 0.098 0.213
0.291 0.212 0.586 0.180 0.137 0.048 0.055 -0.067 0.024 0.057 -0.027 0.070 -0.038
0.141

```
[34]: model_aux= model.inspect(mode='list', what="names", std_est=True)
print(model_aux)
```

	lval	op	rval	Estimate	Est. Std	Std. Err	z-value	p-value
0	sk11	~	eta1	1.000000	0.401483	-	-	-
1	sk11	~	eta3	0.077847	0.043703	0.0141252	5.51126	3.56275e-08
2	sk11	~	eta4	1.000000	0.234795	-	-	-
3	sk9	~	eta1	1.099535	0.445673	0.21104	5.21009	1.88752e-07
4	sk9	~	eta3	0.306028	0.173449	0.0123151	24.8499	0
5	sk10	~	eta1	2.020841	0.572213	0.388749	5.19832	2.01099e-07
6	sk12	~	eta1	1.694323	0.569069	0.373701	4.5339	5.79045e-06
7	sk12	~	eta3	0.124235	0.058347	0.0223928	5.54799	2.88967e-08

8	sk12	~	eta4	-0.294804	-0.057906	0.0793506	-3.71521	0.000203037
9	sk7	~	eta2	1.000000	0.308713	-	-	-
10	sk6	~	eta2	-0.952861	-0.489244	0.151123	-6.30522	2.87783e-10
11	sk6	~	eta3	0.472682	0.239004	0.0780668	6.05484	1.40558e-09
12	sk4	~	eta3	1.000000	0.656761	-	-	-
13	sk2	~	eta3	1.136365	0.649615	0.00925991	122.719	0
14	sk5	~	eta3	0.855483	0.566234	0.00775419	110.325	0
15	sk3	~	eta3	0.914588	0.590811	0.0080117	114.157	0
16	sk1	~	eta3	0.561078	0.551823	0.00519422	108.02	0
17	sk8	~	eta3	0.992243	0.441774	0.0111238	89.1998	0
18	sk13	~	eta4	2.555955	0.404218	0.586168	4.36045	1.29798e-05
19	eta3	~~	eta3	0.136532	1.000000	0.00173079	78.8844	0
20	eta3	~~	eta4	0.048344	0.846618	0.0110784	4.36379	1.27827e-05
21	eta3	~~	eta1	0.055113	0.564445	0.010665	5.16765	2.3706e-07
22	eta3	~~	eta2	-0.066731	-0.481323	0.00219014	-30.469	0
23	eta4	~~	eta4	0.023882	1.000000	0.0165272	1.44501	0.148454
24	eta4	~~	eta1	0.057196	1.400599	0.00255966	22.345	0
25	eta4	~~	eta2	-0.027479	-0.473895	0.00680302	-4.03917	5.36408e-05
26	eta1	~~	eta1	0.069828	1.000000	0.0265914	2.62595	0.00864064
27	eta1	~~	eta2	-0.038046	-0.383726	0.00737785	-5.15683	2.51172e-07
28	eta2	~~	eta2	0.140784	1.000000	0.0176596	7.97208	1.55431e-15
29	sk12	~~	sk12	0.451837	0.729947	0.00451891	99.9881	0
30	sk7	~~	sk7	1.336427	0.904696	0.0189848	70.3947	0
31	sk13	~~	sk13	0.798852	0.836608	0.0370073	21.5863	0
32	sk6	~~	sk6	0.315585	0.590954	0.0154995	20.361	0
33	sk11	~~	sk11	0.208169	0.480532	0.00650741	31.9896	0
34	sk2	~~	sk2	0.241483	0.578001	0.00176293	136.979	0
35	sk8	~~	sk8	0.554344	0.804836	0.00349506	158.608	0
36	sk1	~~	sk1	0.098169	0.695492	0.000654508	149.989	0
37	sk3	~~	sk3	0.212976	0.650942	0.00146257	145.618	0
38	sk9	~~	sk9	0.290726	0.684025	0.00201903	143.993	0
39	sk5	~~	sk5	0.211727	0.679379	0.001426	148.476	0
40	sk10	~~	sk10	0.585755	0.672573	0.00488335	119.949	0
41	sk4	~~	sk4	0.180002	0.568665	0.00132633	135.715	0

En base a la información obtenida previamente y en cuanto a las variables observadas para cada factor, se procede a realizar una asignación de nombre a cada uno de estos, siguiendo un enfoque en el que el concepto englobe las características o la naturaleza de cada una de las variables agrupadas:

- eta1: Curiosidad
- eta2: Comportamiento
- eta3: Afectividad
- eta4: Creatividad

```
[39]: # Para medir el grado de relevancia de cada variable con respecto a cada factor
      relev = model_aux.iloc[:13]
      print(relev)
```

lval	op	rval	Estimate	Est. Std	Std. Err	z-value	p-value
------	----	------	----------	----------	----------	---------	---------

```

0  sk11 ~ eta1 1.000000 0.401483 - - -
1  sk11 ~ eta3 0.077847 0.043703 0.0141252 5.51126 3.56275e-08
2  sk11 ~ eta4 1.000000 0.234795 - - -
3  sk9 ~ eta1 1.099535 0.445673 0.21104 5.21009 1.88752e-07
4  sk9 ~ eta3 0.306028 0.173449 0.0123151 24.8499 0
5  sk10 ~ eta1 2.020841 0.572213 0.388749 5.19832 2.01099e-07
6  sk12 ~ eta1 1.694323 0.569069 0.373701 4.5339 5.79045e-06
7  sk12 ~ eta3 0.124235 0.058347 0.0223928 5.54799 2.88967e-08
8  sk12 ~ eta4 -0.294804 -0.057906 0.0793506 -3.71521 0.000203037
9  sk7 ~ eta2 1.000000 0.308713 - - -
10 sk6 ~ eta2 -0.952861 -0.489244 0.151123 -6.30522 2.87783e-10
11 sk6 ~ eta3 0.472682 0.239004 0.0780668 6.05484 1.40558e-09
12 sk4 ~ eta3 1.000000 0.656761 - - -

```

```
[40]: semopy.calc_stats(model)
```

```

[40]:      DoF  DoF Baseline      chi2  chi2 p-value  chi2 Baseline      CFI \
Value   53           78 15237.067227           0.0 156539.1804 0.902953

      GFI      AGFI      NFI      TLI      RMSEA      AIC \
Value 0.902663 0.856749 0.902663 0.857176 0.070743 75.467673

      BIC      LogLik
Value 415.762633 0.266164

```

1.5.1 Pregunta 6

1.6 SEM Complete

```

[38]: # Se incluyen las variables sexo, área y actividad física
var_aux["sexo"] = junaeb2["sexo"]
var_aux["area"] = junaeb2["area"]
var_aux["act_fisica"] = junaeb2["act_fisica"]

```

```

<ipython-input-38-7b31328a2fdd>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```

var_aux["sexo"] = junaeb2["sexo"]
<ipython-input-38-7b31328a2fdd>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
var_aux["area"] = junaeb2["area"]
```

<ipython-input-38-7b31328a2fdd>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
var_aux["act_fisica"] = junaeb2["act_fisica"]

```
[41]: import semopy
import pandas as pd

mod = """
eta1 =~ sk11 + sk9 + sk10 + sk12
eta2 =~ sk7 + sk6
eta3 =~ sk4 + sk2 + sk11 + sk5 + sk3 + sk9 + sk1 + sk6 + sk8 + sk12
eta4 =~ sk11 + sk12 + sk13
      """
desc = mod
print(desc)
```

```
eta1 =~ sk11 + sk9 + sk10 + sk12
eta2 =~ sk7 + sk6
eta3 =~ sk4 + sk2 + sk11 + sk5 + sk3 + sk9 + sk1 + sk6 + sk8 + sk12
eta4 =~ sk11 + sk12 + sk13
```

```
[42]: data = var_aux
mod = semopy.Model(desc)
res = mod.fit(data)
```

```
[43]: print(mod.inspect())
```

	lval	op	rval	Estimate	Std. Err	z-value	p-value
0	sk11	~	eta1	1.000000	-	-	-
1	sk11	~	eta3	0.077847	0.0141252	5.51126	3.56275e-08
2	sk11	~	eta4	1.000000	-	-	-
3	sk9	~	eta1	1.099535	0.21104	5.21009	1.88752e-07
4	sk9	~	eta3	0.306028	0.0123151	24.8499	0
5	sk10	~	eta1	2.020841	0.388749	5.19832	2.01099e-07
6	sk12	~	eta1	1.694323	0.373701	4.5339	5.79045e-06
7	sk12	~	eta3	0.124235	0.0223928	5.54799	2.88967e-08
8	sk12	~	eta4	-0.294804	0.0793506	-3.71521	0.000203037
9	sk7	~	eta2	1.000000	-	-	-
10	sk6	~	eta2	-0.952861	0.151123	-6.30522	2.87783e-10
11	sk6	~	eta3	0.472682	0.0780668	6.05484	1.40558e-09
12	sk4	~	eta3	1.000000	-	-	-
13	sk2	~	eta3	1.136365	0.00925991	122.719	0

14	sk5	~	eta3	0.855483	0.00775419	110.325	0
15	sk3	~	eta3	0.914588	0.0080117	114.157	0
16	sk1	~	eta3	0.561078	0.00519422	108.02	0
17	sk8	~	eta3	0.992243	0.0111238	89.1998	0
18	sk13	~	eta4	2.555955	0.586168	4.36045	1.29798e-05
19	eta3	~~	eta3	0.136532	0.00173079	78.8844	0
20	eta3	~~	eta4	0.048344	0.0110784	4.36379	1.27827e-05
21	eta3	~~	eta1	0.055113	0.010665	5.16765	2.3706e-07
22	eta3	~~	eta2	-0.066731	0.00219014	-30.469	0
23	eta4	~~	eta4	0.023882	0.0165272	1.44501	0.148454
24	eta4	~~	eta1	0.057196	0.00255966	22.345	0
25	eta4	~~	eta2	-0.027479	0.00680302	-4.03917	5.36408e-05
26	eta1	~~	eta1	0.069828	0.0265914	2.62595	0.00864064
27	eta1	~~	eta2	-0.038046	0.00737785	-5.15683	2.51172e-07
28	eta2	~~	eta2	0.140784	0.0176596	7.97208	1.55431e-15
29	sk12	~~	sk12	0.451837	0.00451891	99.9881	0
30	sk7	~~	sk7	1.336427	0.0189848	70.3947	0
31	sk13	~~	sk13	0.798852	0.0370073	21.5863	0
32	sk6	~~	sk6	0.315585	0.0154995	20.361	0
33	sk11	~~	sk11	0.208169	0.00650741	31.9896	0
34	sk2	~~	sk2	0.241483	0.00176293	136.979	0
35	sk8	~~	sk8	0.554344	0.00349506	158.608	0
36	sk1	~~	sk1	0.098169	0.000654508	149.989	0
37	sk3	~~	sk3	0.212976	0.00146257	145.618	0
38	sk9	~~	sk9	0.290726	0.00201903	143.993	0
39	sk5	~~	sk5	0.211727	0.001426	148.476	0
40	sk10	~~	sk10	0.585755	0.00488335	119.949	0
41	sk4	~~	sk4	0.180002	0.00132633	135.715	0

[]: