TAREA 4_Auil_Cabezas

December 19, 2022

1 Tarea 4 - Mario Cabezas - Lucas Auil

2 Carga de bibliotecas

[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

%matplotlib inline
```

3 Carga y limpieza de datos

Lo primero que realizamos es cargar la data a trabajar que en este caso es la base de datos llamada junaeb2 , la cual viene en formato csv

[2]:
```
Xr = pd.read_csv('../TAREA FINAL/junaeb2.csv')
Xr
```

[2]:

	sexo	edad	imce	vive_padre	vive_madre	sk1	sk2	sk3	sk4	sk5	…	\
0	1	85	0.75	1	1	1	1	1	2	1	…	
1	0	76	0.71	0	1	1	1	1	1	1	…	
2	1	68	0.27	0	1	2	2	3	2	1	…	
3	1	84	2.05	1	1	1	1	1	1	1	…	
4	0	86	1.05	1	1	1	1	1	1	1	…	
…	…	…	…	…	…	…	…	…	…	…	…	…

```
59994     0    78  1.63           1           1    1    3    1    2    2  …
59995     1    79  2.57           1           1    1    1    2    2    1  …
59996     0    78  2.12           1           1    1    1    1    1    1  …
59997     1    78 -0.43           1           1    1    1    1    1    2  …
59998     0    78 -0.55           0           1    2    2    1    1    1  …

       sk9  sk10  sk11  sk12  sk13  act_fisica  area  educm  educp  madre_work
0        2     2     2     3     2         NaN     0   11.0     11          -1
1        1     1     1     1     1         5.0     0    8.0      8           1
2        2     3     2     1     3         NaN     1   13.0     13           1
3        1     1     1     1     1         2.0     1   16.0     12          -1
4        1     1     1     1     1         1.0     1   17.0     15           0
…    …   …     …     …   …   …           …   …     …    …         …
59994    2     2     2     1     1         2.0     1   13.0     13          -1
59995    1     3     2     1     4         3.0     1   18.0     19           0
59996    1     3     1     1     1         3.0     1   13.0      9           1
59997    1     2     1     1     2         2.0     1   17.0     15           1
59998    1     1     1     1     1         1.0     1   18.0     11           1

[59999 rows x 23 columns]
```

Analizando tanto los datos desplegados en python como la base de datos en Excel , nos dimos cuenta que en las variables act_fisica y educm existian datos vacios, por lo que procedimos a eliminar completamente dichas filas de nuestra data

[3]:
```
Xr.dropna(inplace=True)
Xr.reset_index(drop=True, inplace=True)
Xr
```

[3]:
```
       sexo  edad  imce  vive_padre  vive_madre  sk1  sk2  sk3  sk4  sk5  … \
0         0    76  0.71           0           1    1    1    1    1    1  …
1         1    84  2.05           1           1    1    1    1    1    1  …
2         0    86  1.05           1           1    1    1    1    1    1  …
3         0    74  1.39           1           1    1    2    1    1    1  …
4         1    91  2.75           1           1    1    1    1    2    2  …
…     …   …    …         …           …  …  …  …  …  …  …
57352     0    78  1.63           1           1    1    3    1    2    2  …
57353     1    79  2.57           1           1    1    1    2    2    1  …
57354     0    78  2.12           1           1    1    1    1    1    1  …
57355     1    78 -0.43           1           1    1    1    1    1    2  …
57356     0    78 -0.55           0           1    2    2    1    1    1  …

       sk9  sk10  sk11  sk12  sk13  act_fisica  area  educm  educp  madre_work
0        1     1     1     1     1         5.0     0    8.0      8           1
1        1     1     1     1     1         2.0     1   16.0     12          -1
2        1     1     1     1     1         1.0     1   17.0     15           0
3        1     1     1     1     1         4.0     0    8.0      8          -1
```

```
4        3     3     3     2     2          2.0     1   20.0     19           1
…    …   …   …   …   …   …   …   …   …   …   …   …   …   …   …   …   …   …
57352    2     2     2     1     1          2.0     1   13.0     13          -1
57353    1     3     2     1     4          3.0     1   18.0     19           0
57354    1     3     1     1     1          3.0     1   13.0      9           1
57355    1     2     1     1     2          2.0     1   17.0     15           1
57356    1     1     1     1     1          1.0     1   18.0     11           1

[57357 rows x 23 columns]
```

[4]:
```
Xr.dtypes
```

[4]:
```
sexo            int64
edad            int64
imce          float64
vive_padre      int64
vive_madre      int64
sk1             int64
sk2             int64
sk3             int64
sk4             int64
sk5             int64
sk6             int64
sk7             int64
sk8             int64
sk9             int64
sk10            int64
sk11            int64
sk12            int64
sk13            int64
act_fisica    float64
area            int64
educm         float64
educp           int64
madre_work      int64
dtype: object
```

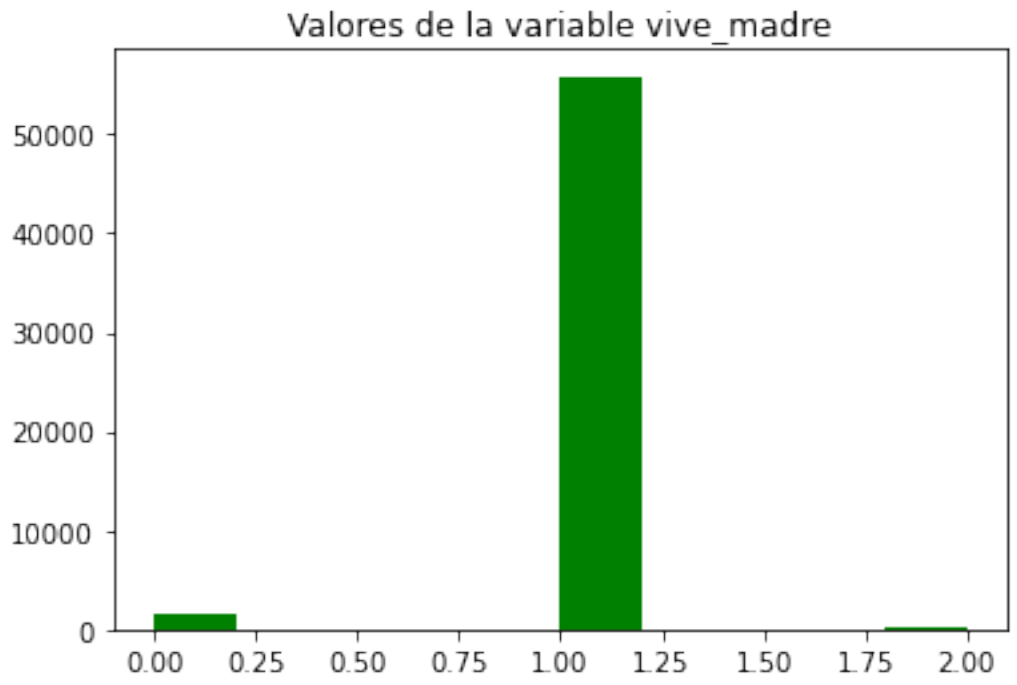
[5]:
```
Xr["act_fisica"] = Xr["act_fisica"].astype("int64")
Xr["educm"] = Xr["educm"].astype("int64")
Xr.dtypes
```

[5]:
```
sexo            int64
edad            int64
imce          float64
vive_padre      int64
vive_madre      int64
sk1             int64
```

```
sk2             int64
sk3             int64
sk4             int64
sk5             int64
sk6             int64
sk7             int64
sk8             int64
sk9             int64
sk10            int64
sk11            int64
sk12            int64
sk13            int64
act_fisica      int64
area            int64
educm           int64
educp           int64
madre_work      int64
dtype: object
```

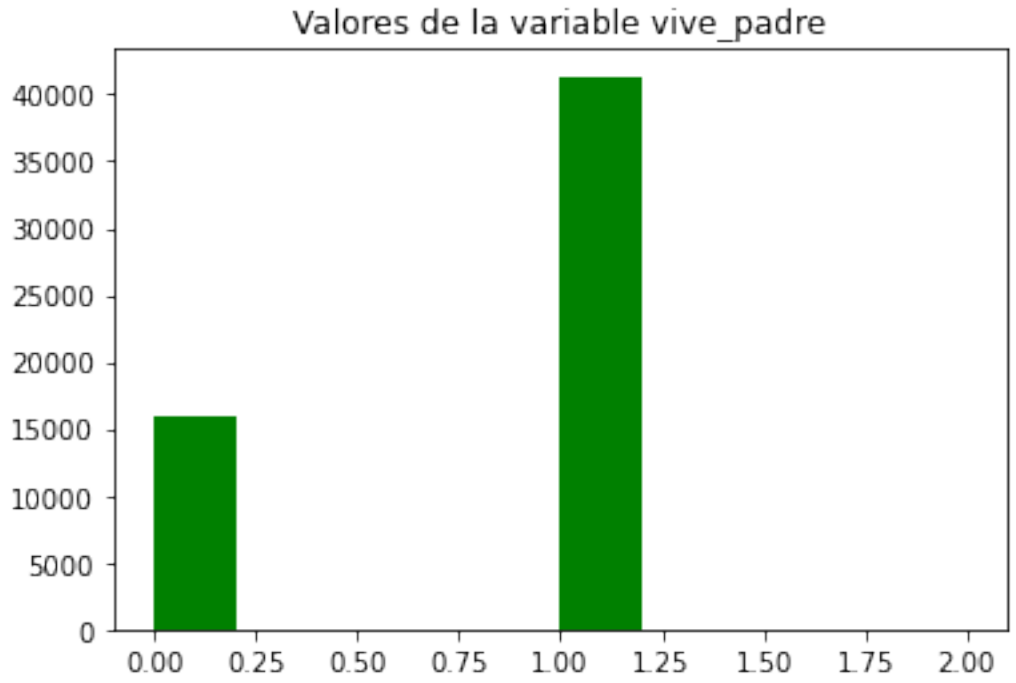
[6]:
```
plt.hist(Xr['vive_madre'],color="green")
plt.title("Valores de la variable vive_madre")
Xr.vive_madre.value_counts()
```

[6]:
```
1    55832
0     1437
2       88
Name: vive_madre, dtype: int64
```



[7]:
```
plt.hist(Xr['vive_padre'],color="green")
plt.title("Valores de la variable vive_padre")
Xr.vive_padre.value_counts()
```

[7]:
```
1    41337
0    15998
2       22
Name: vive_padre, dtype: int64
```



[8]:
```
Xr.drop(Xr[Xr['vive_madre']==2].index,inplace =True)
Xr.drop(Xr[Xr['vive_padre']==2].index,inplace =True)
Xr.reset_index(drop=True, inplace=True)
Xr
```

[8]:

	sexo	edad	imce	vive_padre	vive_madre	sk1	sk2	sk3	sk4	sk5	…	\
0	0	76	0.71	0	1	1	1	1	1	1	…	
1	1	84	2.05	1	1	1	1	1	1	1	…	
2	0	86	1.05	1	1	1	1	1	1	1	…	
3	0	74	1.39	1	1	1	2	1	1	1	…	
4	1	91	2.75	1	1	1	1	1	2	2	…	
…	…	…	…	…	…	…	…	…	…	…	…	…
57242	0	78	1.63	1	1	1	3	1	2	2	…	

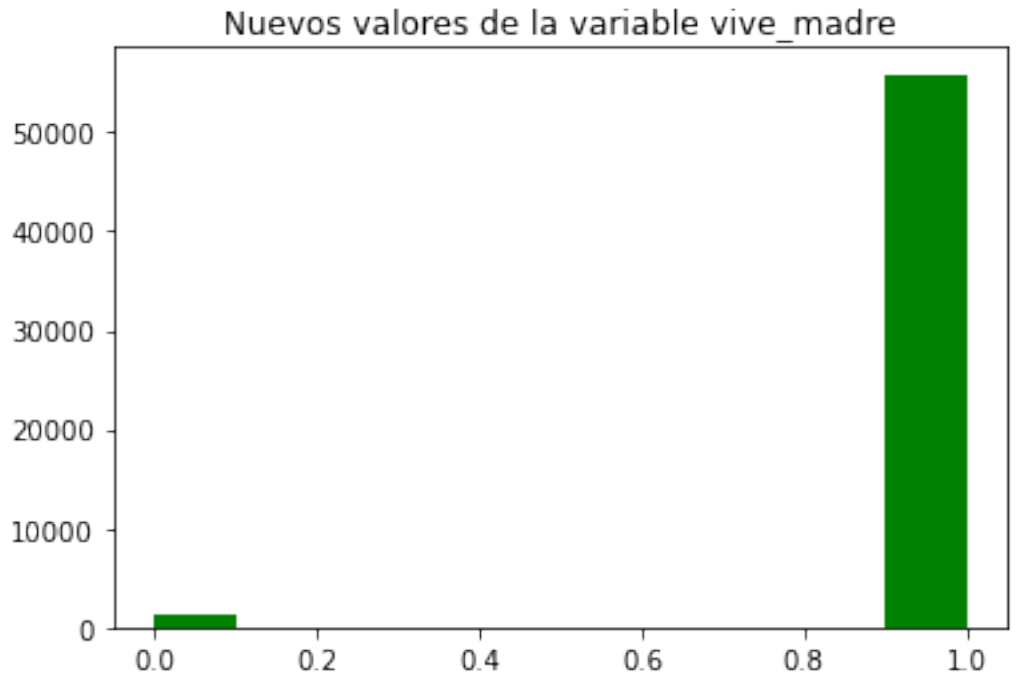
```
57243     1    79  2.57           1           1    1    1    2    2    1  …
57244     0    78  2.12           1           1    1    1    1    1    1  …
57245     1    78 -0.43           1           1    1    1    1    1    2  …
57246     0    78 -0.55           0           1    2    2    1    1    1  …
```

	sk9	sk10	sk11	sk12	sk13	act_fisica	area	educm	educp	madre_work
0	1	1	1	1	1	5	0	8	8	1
1	1	1	1	1	1	2	1	16	12	-1
2	1	1	1	1	1	1	1	17	15	0
3	1	1	1	1	1	4	0	8	8	-1
4	3	3	3	2	2	2	1	20	19	1
…	…	…	…	…	…	…	…	…	…	…
57242	2	2	2	1	1	2	1	13	13	-1
57243	1	3	2	1	4	3	1	18	19	0
57244	1	3	1	1	1	3	1	13	9	1
57245	1	2	1	1	2	2	1	17	15	1
57246	1	1	1	1	1	1	1	18	11	1

[57247 rows x 23 columns]

[9]:
```
plt.hist(Xr['vive_madre'],color="green")
plt.title("Nuevos valores de la variable vive_madre")
Xr.vive_madre.value_counts()
```

[9]:
```
1    55828
0     1419
Name: vive_madre, dtype: int64
```



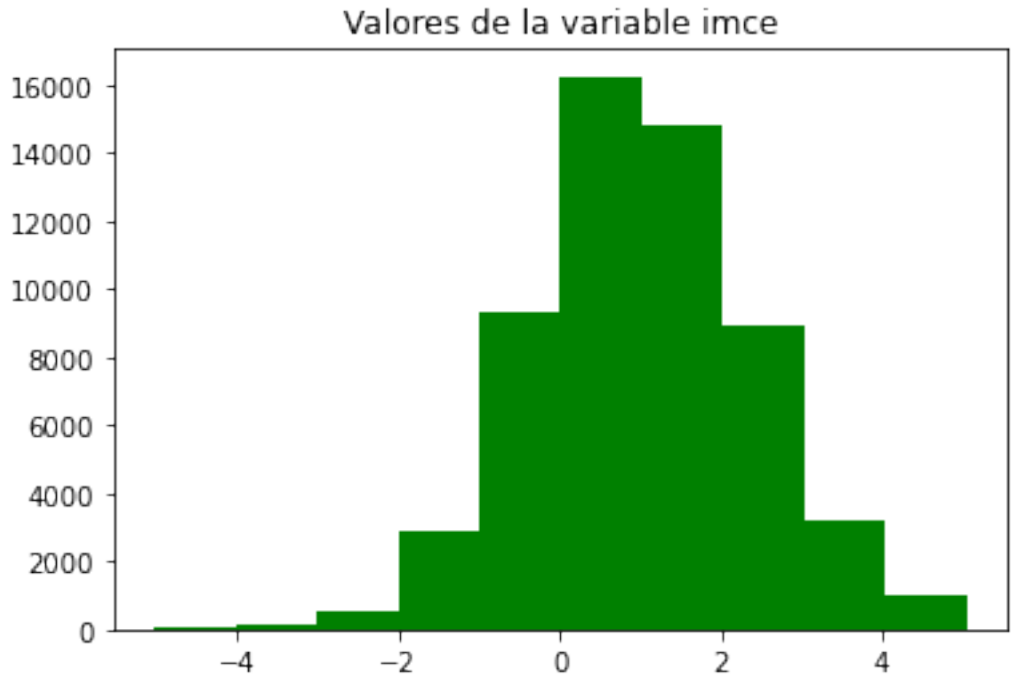
[10]:
```
plt.hist(Xr['vive_padre'],color="green")
plt.title("Nuevos valores de la variable vive_padre")
Xr.vive_padre.value_counts()
```

[10]:
```
1    41280
0    15967
Name: vive_padre, dtype: int64
```

[11]:
```
plt.hist(Xr['imce'],color="green")
plt.title("Valores de la variable imce")
Xr.imce.value_counts()
```

[11]:
```
 0.74    208
 1.07    197
 0.87    197
 0.73    195
 0.39    195
        ...
-3.35      1
-4.36      1
-4.66      1
-3.17      1
```

```
-4.88      1
Name: imce, Length: 928, dtype: int64
```



[12]:
```
Xr.drop(Xr[Xr['imce']<0].index,inplace =True)
Xr.reset_index(drop=True, inplace=True)
Xr
```

[12]:

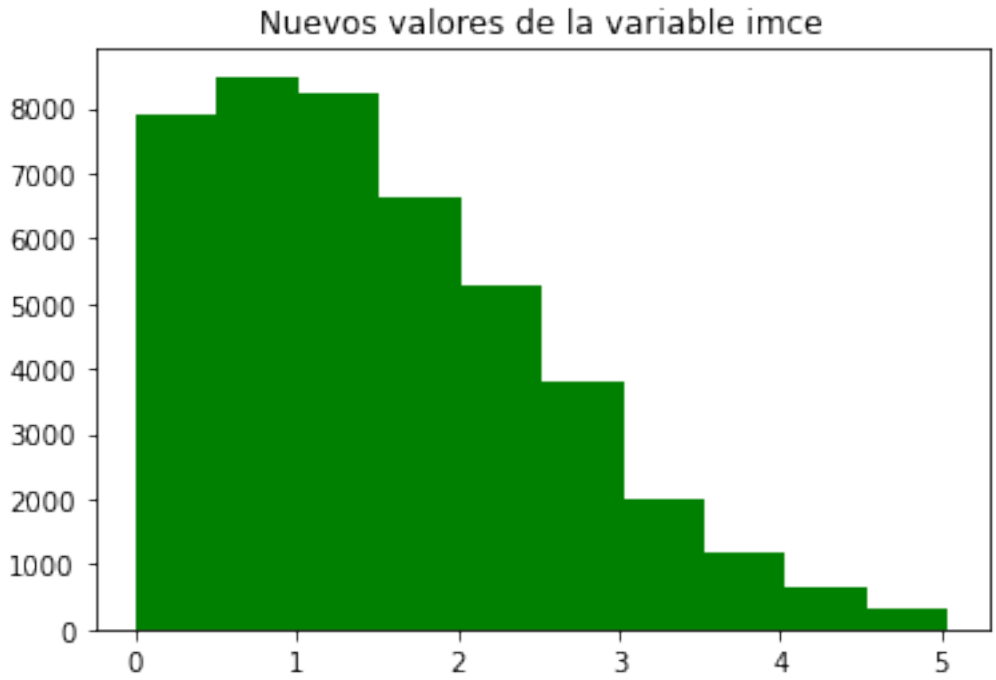
	sexo	edad	imce	vive_padre	vive_madre	sk1	sk2	sk3	sk4	sk5	…	\
0	0	76	0.71	0	1	1	1	1	1	1	…	
1	1	84	2.05	1	1	1	1	1	1	1	…	
2	0	86	1.05	1	1	1	1	1	1	1	…	
3	0	74	1.39	1	1	1	2	1	1	1	…	
4	1	91	2.75	1	1	1	1	1	2	2	…	
…	…	…	…	…	…	…	…	…	…	…	…	…
44497	1	78	1.58	1	1	1	1	1	1	1	…	
44498	1	79	1.73	1	0	1	1	2	1	1	…	
44499	0	78	1.63	1	1	1	3	1	2	2	…	
44500	1	79	2.57	1	1	1	1	2	2	1	…	
44501	0	78	2.12	1	1	1	1	1	1	1	…	

	sk9	sk10	sk11	sk12	sk13	act_fisica	area	educm	educp	madre_work
0	1	1	1	1	1	5	0	8	8	1
1	1	1	1	1	1	2	1	16	12	-1
2	1	1	1	1	1	1	1	17	15	0

```
3        1     1     1     1     1          4     0      8      8          -1
4        3     3     3     2     2          2     1     20     19           1
…     …    …    …   …    …        …   …     …     …        …
44497    1     4     3     3     5          1     0     13     12          -1
44498    1     1     2     1     2          2     1     17     17           1
44499    2     2     2     1     1          2     1     13     13          -1
44500    1     3     2     1     4          3     1     18     19           0
44501    1     3     1     1     1          3     1     13      9           1

[44502 rows x 23 columns]
```

[13]:
```
plt.hist(Xr['imce'],color="green")
plt.title("Nuevos valores de la variable imce")
Xr.imce.value_counts()
```

[13]:
```
0.74    208
1.07    197
0.87    197
0.39    195
0.73    195
       …
5.02      3
4.93      3
4.82      2
5.00      2
4.89      2
Name: imce, Length: 505, dtype: int64
```

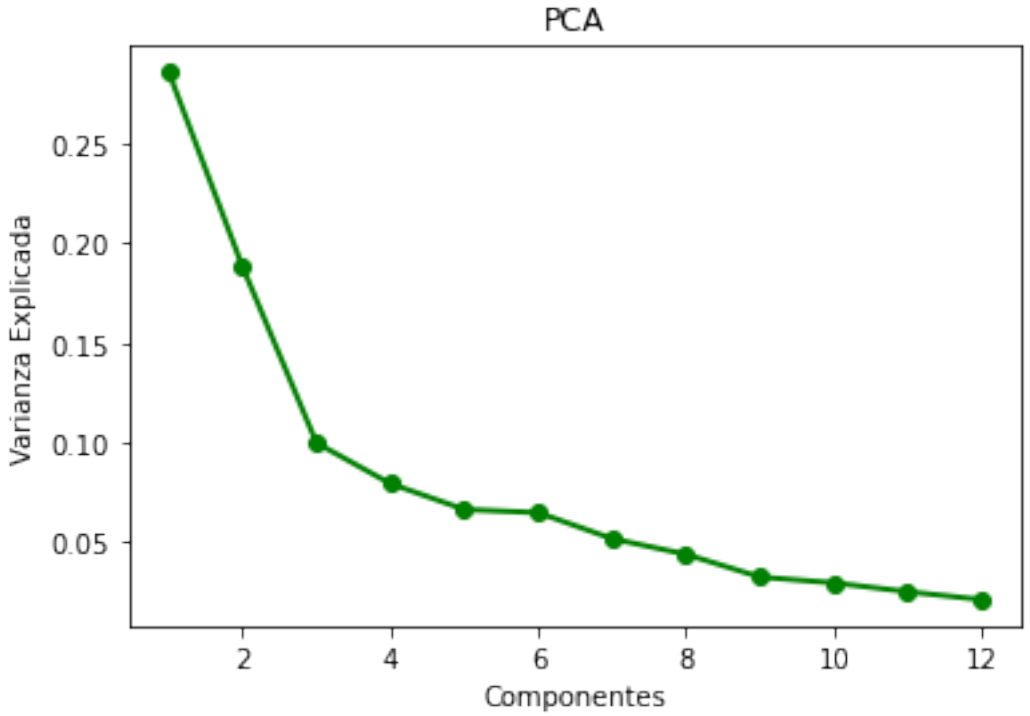


4 1 PCA

[14]:
```
vector_sks =␣
  ↪Xr[["sk1","sk2","sk3","sk4","sk5","sk6","sk7","sk8","sk9","sk10","sk11","sk12","sk13"]]
pca = PCA(n_components=12)
pca_features = pca.fit_transform(vector_sks)
print(pca.explained_variance_ratio_)
```

```
[0.28606755 0.18817514 0.09973579 0.07929705 0.06632621 0.06471883
 0.05164867 0.0437851  0.03238106 0.0294521  0.02499784 0.02101661]
```

[15]:
```
PC_values = np.arange(pca.n_components_) + 1
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2,␣
  ↪color='green')
plt.title('PCA')
plt.xlabel('Componentes')
plt.ylabel('Varianza Explicada')
plt.show()
```



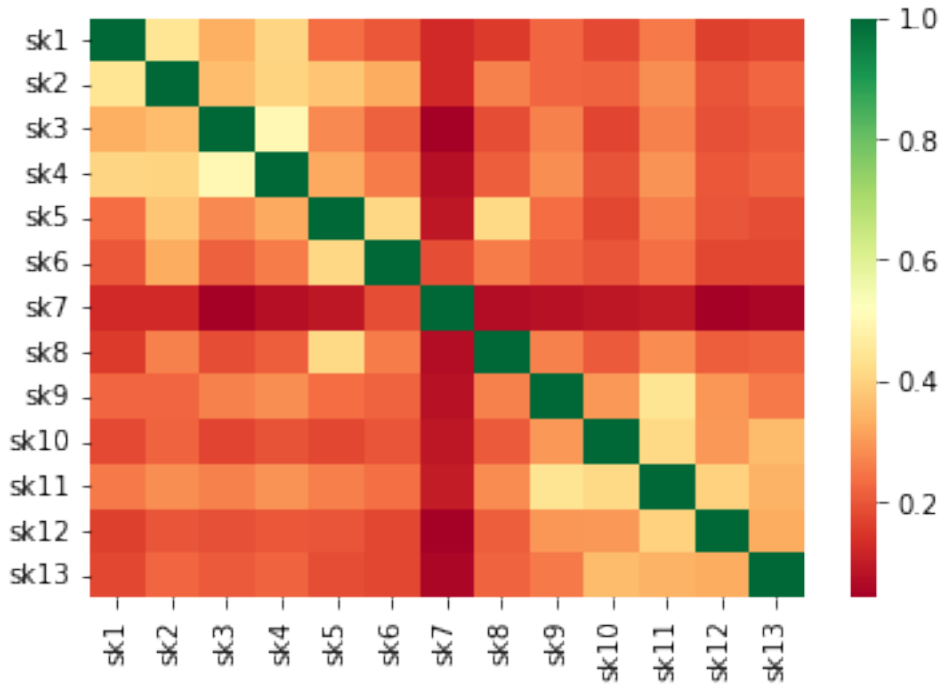
[16]:
```
pca = PCA(n_components='mle')
pca_features = pca.fit_transform(vector_sks)
print(pca.explained_variance_ratio_)
```

```
[0.28606755 0.18817514 0.09973579 0.07929705 0.06632621 0.06471883
 0.05164867 0.0437851  0.03238106 0.0294521  0.02499784 0.02101661]
```

Los valores propios de cada varaible son: [0.28606755 0.18817514 0.09973579 0.07929705 0.06632621 0.06471883 0.05164867 0.0437851 0.03238106 0.0294521 0.02499784 0.02101661]. Además, no fue posible disminuir las variables a un número óptimo, en función de la varianza de los datos, puesto que estas no tenían correlación entre sí.

[17]:
```
sns.heatmap(vector_sks.corr(), cmap='RdYlGn')
```

[17]:
```
<AxesSubplot:>
```



[18]:
```
pca_vectors = pd.DataFrame(data = pca.components_)
pca_vectors.head()
```

[18]:

	0	1	2	3	4	5	6	\
0	0.101837	0.228681	0.166704	0.180646	0.186154	0.249874	0.351651	
1	0.007632	0.034134	0.066008	0.055333	0.040743	-0.028742	-0.923058	
2	0.084242	0.266514	0.180453	0.191232	0.300941	0.341428	-0.115066	
3	-0.004360	-0.033451	-0.039866	-0.031587	-0.046156	-0.002758	-0.032316	
4	0.143585	0.268222	0.309358	0.295992	-0.021423	0.162823	-0.030117	

```
          7         8         9         10        11        12
0  0.291481  0.234854  0.406509  0.282787  0.293839  0.429516
1  0.099394  0.086327  0.168557  0.104073  0.155922  0.236538
2  0.477371  0.079394 -0.373427  0.000937 -0.089114 -0.500784
3 -0.095896  0.105624  0.707258  0.138852  0.022035 -0.672875
4 -0.746663  0.111445 -0.196699  0.105288  0.259159 -0.096871
```

[19]:
```
pca_df = pd.DataFrame(data=pca_features,columns=['PC1', 'PC2',␣
↪'PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11','P12'])
pca_df.describe().apply(lambda s: s.apply('{0:.3f}'.format))
```

[19]:

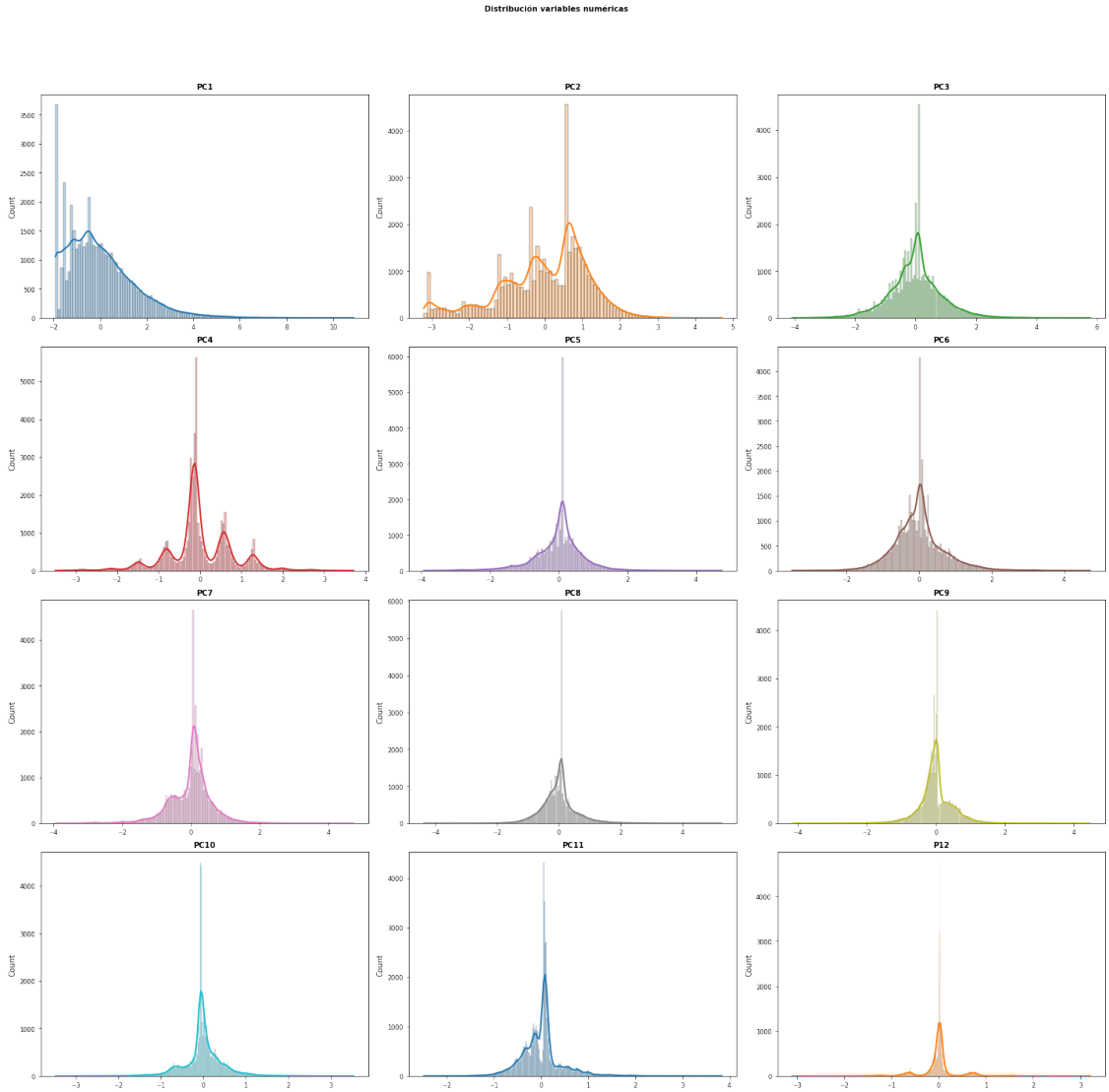
	PC1	PC2	PC3	PC4	PC5	PC6
count	44502.000	44502.000	44502.000	44502.000	44502.000	44502.000
mean	0.000	0.000	-0.000	-0.000	-0.000	-0.000
std	1.464	1.187	0.864	0.771	0.705	0.696
min	-1.910	-3.208	-4.086	-3.485	-3.926	-3.496
25%	-1.100	-0.714	-0.465	-0.255	-0.299	-0.400
50%	-0.260	0.185	-0.002	-0.097	0.094	0.018
75%	0.790	0.809	0.454	0.514	0.346	0.283
max	10.878	4.703	5.779	3.710	4.752	4.718

	PC7	PC8	PC9	PC10	PC11	P12
count	44502.000	44502.000	44502.000	44502.000	44502.000	44502.000
mean	0.000	0.000	-0.000	0.000	-0.000	-0.000
std	0.622	0.573	0.492	0.470	0.433	0.397
min	-3.943	-4.363	-4.163	-3.469	-2.495	-3.101
25%	-0.311	-0.289	-0.212	-0.116	-0.221	-0.062
50%	0.065	0.026	-0.016	-0.038	0.051	0.021
75%	0.315	0.191	0.202	0.189	0.113	0.047
max	4.730	5.275	4.471	3.543	3.841	3.208

En las siguientes gráficas se puede observar la distribución de cada componente

[20]:
```
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(20, 20))
axes = axes.flat
columnas_numeric = pca_df.select_dtypes(include=['float64', 'int']).columns



for i, colum in enumerate(columnas_numeric):
        sns.histplot(
            data    = pca_df,
            x       = colum,
            stat    = "count",
            kde     = True,
            color   = (list(plt.rcParams['axes.prop_cycle'])*2)[i]["color"],
```

```
            line_kws= {'linewidth': 2},
            alpha    = 0.3,
            ax       = axes[i]
        )
        axes[i].set_title(colum, fontsize = 10, fontweight = "bold")
        axes[i].tick_params(labelsize = 8)
        axes[i].set_xlabel("")




fig.tight_layout()
plt.subplots_adjust(top = 0.9)
fig.suptitle('Distribución variables numéricas', fontsize = 10, fontweight =␣
 ↪"bold");
```

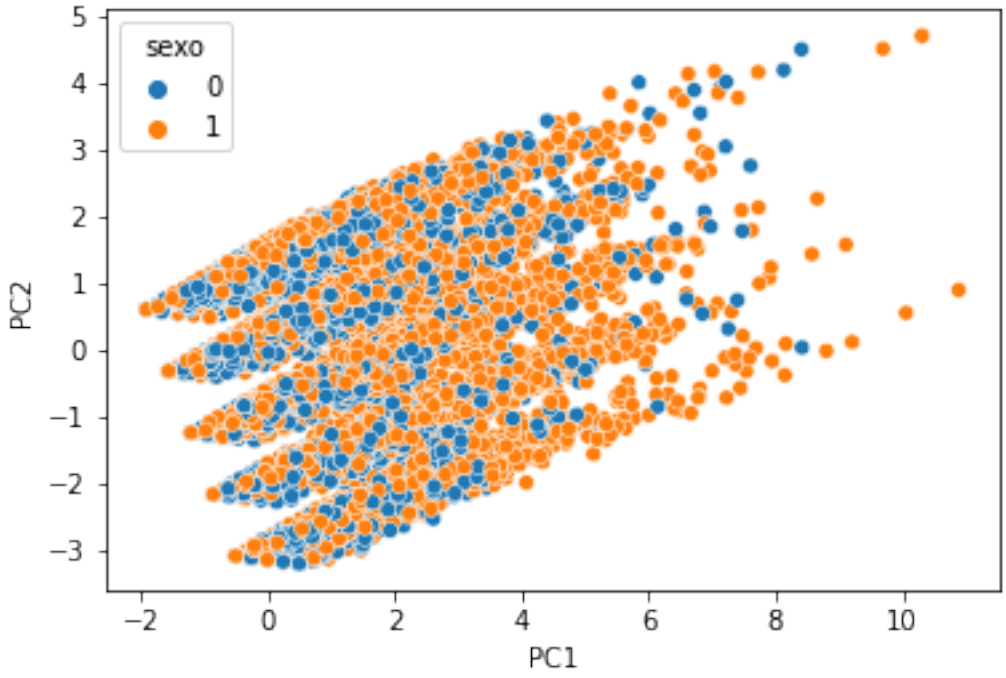


5 3

[35]:
```
pca_df['sexo'] = 0
pca_df['sexo'] = np.where(Xr["sexo"] > 0, 1, pca_df['sexo'])

sns.scatterplot('PC1', 'PC2', data=pca_df, hue='sexo')
```

```
C:\Users\Hpp\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(
```

[35]:
```
<AxesSubplot:xlabel='PC1', ylabel='PC2'>
```

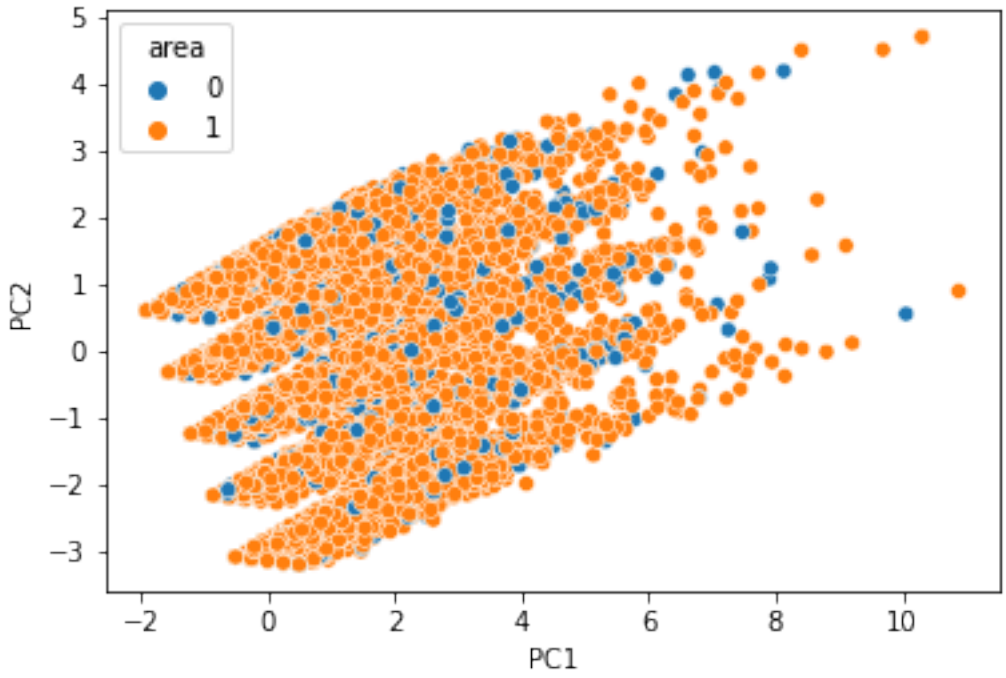


[36]:
```
pca_df['area'] = 0
pca_df['area'] = np.where(Xr["area"] > 0, 1, pca_df['area'])

sns.scatterplot('PC1', 'PC2', data=pca_df, hue='area')
```

```
C:\Users\Hpp\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(
```

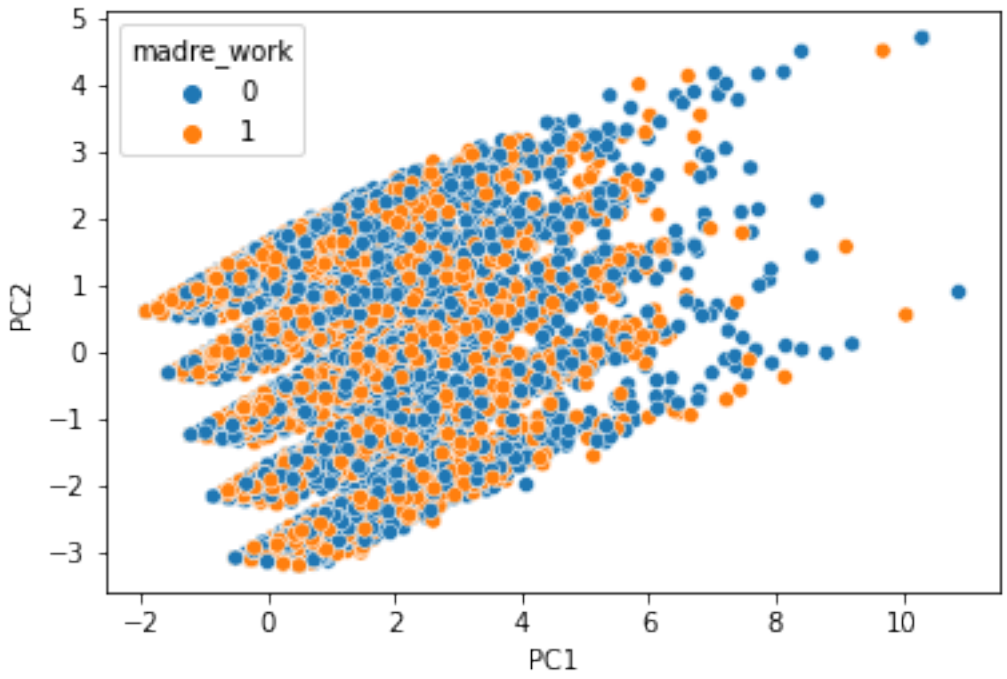
[36]: <AxesSubplot:xlabel='PC1', ylabel='PC2'>



[37]:
```
pca_df["madre_work"] = 0
pca_df["madre_work"] = np.where(Xr["madre_work"] > 0, 1, pca_df["madre_work"])

sns.scatterplot('PC1', 'PC2', data=pca_df, hue="madre_work")
```

```
C:\Users\Hpp\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(
```

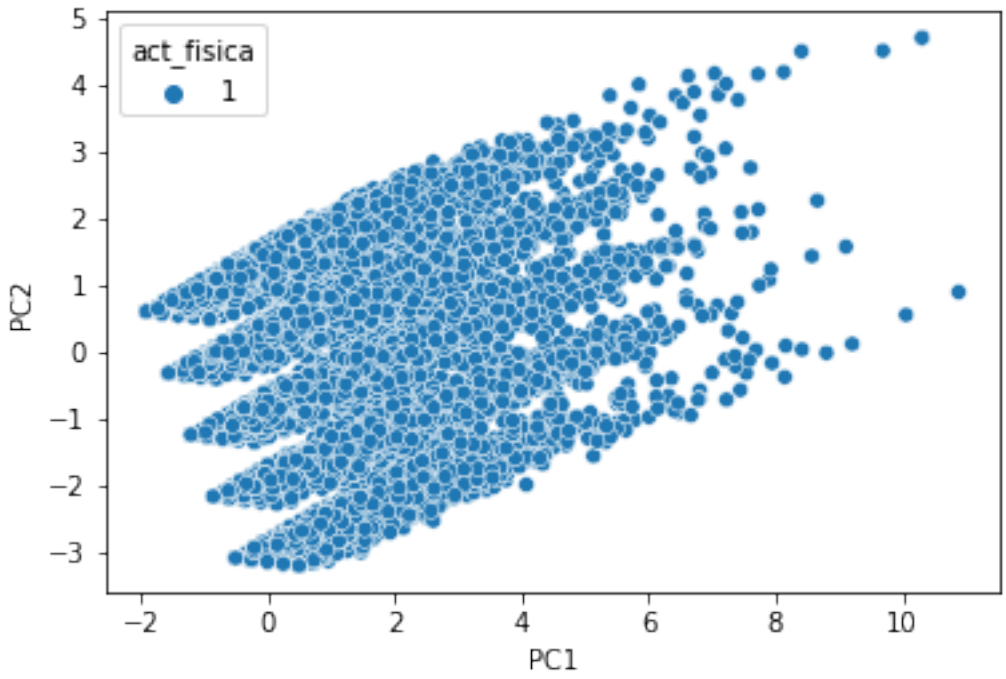
[37]: <AxesSubplot:xlabel='PC1', ylabel='PC2'>



[38]:
```
pca_df["act_fisica"] = 0
pca_df["act_fisica"] = np.where(Xr["act_fisica"] > 0, 1, pca_df["act_fisica"])

sns.scatterplot('PC1', 'PC2', data=pca_df, hue="act_fisica")
```

```
C:\Users\Hpp\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(
```

[38]:
```
<AxesSubplot:xlabel='PC1', ylabel='PC2'>
```



se puede observar que no existen diferencias significativas entre grupos , ya que no se observa una clara separación entre grupos con respecto a ambos ejes del gráfico

6 4

[ ]:
```
## Ser egoista modificamos sus valores
var_sk["sk7"].replace(4,2,inplace=True)
var_sk["sk7"].replace(5,1,inplace=True)
var_sk["sk7"].replace(2,4,inplace=True)
var_sk["sk7"].replace(1,5,inplace=True)
```

[40]:
```
fa = FactorAnalyzer(rotation='promax')
fa.fit(vector_sks)
```

[40]:
```
FactorAnalyzer(rotation_kwargs={})
```

[41]:
```
fa.loadings_
```

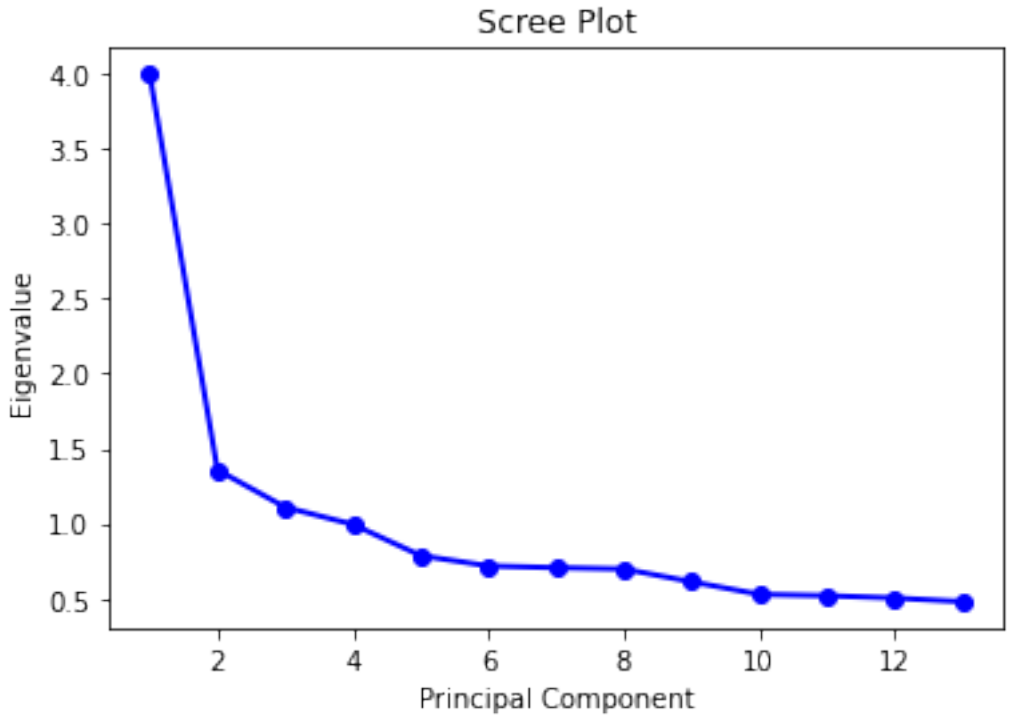
[41]:
```
array([[ 8.99735076e-03,  6.04089504e-01, -2.76161552e-02],
       [-3.08973511e-02,  4.81834611e-01,  2.44925078e-01],
       [ 3.18443234e-02,  6.38488215e-01, -4.60469009e-02],
       [-1.51671053e-04,  7.38733090e-01, -2.95482701e-02],
       [-1.42106910e-01, -1.64061245e-02,  8.46895012e-01],
       [-2.70462553e-04,  4.13009915e-02,  5.22560162e-01],
```

```
       [ 1.59537016e-02,  3.14375943e-02,  1.51335054e-01],
       [ 1.47858798e-01, -1.09890859e-01,  5.16256139e-01],
       [ 4.79056757e-01,  9.20273982e-02,  3.93230433e-02],
       [ 6.13861842e-01, -3.81052232e-02, -2.07578238e-02],
       [ 6.97308041e-01,  3.09136373e-02, -7.26688327e-04],
       [ 5.69219552e-01, -2.81948703e-02, -1.42155433e-03],
       [ 5.26432002e-01,  2.16588378e-02, -1.00089677e-02]])
```

[42]:
```
fa.get_eigenvalues()
```

[42]:
```
(array([3.99403324, 1.35709614, 1.10826633, 0.99384722, 0.78610007,
        0.71734716, 0.70564612, 0.69689018, 0.61153169, 0.52774476,
        0.51879723, 0.50343626, 0.47926359]),
 array([ 3.39429304,  0.75978216,  0.58015174,  0.2047792 ,  0.08742927,
         0.06254669,  0.03313104,  0.02348758, -0.02586072, -0.07677543,
        -0.09370486, -0.109146  , -0.17884126]))
```

[44]:
```
values = np.arange(1,14)
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()
```



[45]:
```
fa.get_factor_variance()
```

[45]:
```
(array([1.73806267, 1.57766252, 1.34554726]),
 array([0.13369713, 0.12135866, 0.10350364]),
 array([0.13369713, 0.25505578, 0.35855942]))
```

[48]:
```
print(semopy.efa.explore_cfa_model(vector_sks, pval=0.05))
```

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

[51]:
```
efa_vectors = pd.DataFrame(data = fa.loadings_,
 ↪index=["sk1","sk2","sk3","sk4","sk5","sk6","sk7","sk8","sk9","sk10","sk11","sk12","sk13"])
efa_vectors
```

[51]:

	0	1	2
sk1	0.008997	0.604090	-0.027616
sk2	-0.030897	0.481835	0.244925
sk3	0.031844	0.638488	-0.046047
sk4	-0.000152	0.738733	-0.029548
sk5	-0.142107	-0.016406	0.846895
sk6	-0.000270	0.041301	0.522560
sk7	0.015954	0.031438	0.151335
sk8	0.147859	-0.109891	0.516256
sk9	0.479057	0.092027	0.039323
sk10	0.613862	-0.038105	-0.020758
sk11	0.697308	0.030914	-0.000727
sk12	0.569220	-0.028195	-0.001422
sk13	0.526432	0.021659	-0.010009

- En el grupo 0 quedará sk9,sk10,sk11,sk12,sk13 (intelectuales-curiosos);
- En el grupo 1 quedará sk1,sk2,sk3,sk4 (cariñosos);
- En el grupo 2 quedará sk5,sk6,sk7,sk8(sociables)

7 5

[52]:
```
Xf=vector_sks

mod = """
# measurement model
```

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

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

```
Name of objective: MLW
Optimization method: SLSQP
Optimization successful.
Optimization terminated successfully
Objective value: 0.164
Number of iterations: 40
Params: 1.393 1.256 1.144 1.351 1.924 1.625 1.781 1.042 0.572 1.143 0.478 0.239
1.429 0.440 0.203 0.156 0.277 0.594 0.700 0.090 0.201 0.160 0.360 0.141 0.087
0.048 0.157 0.057 0.047
```

[53]:
```
model.inspect(mode='list', what="names", std_est=True)
```

[53]:

	lval	op	rval	Estimate	Est. Std	Std. Err	z-value	p-value
0	sk9	~	eta1	1.000000	0.580567	-	-	-
1	sk10	~	eta1	1.393283	0.561740	0.016007	87.039845	0.0
2	sk11	~	eta1	1.256019	0.722753	0.012557	100.023741	0.0
3	sk12	~	eta1	1.144320	0.543799	0.013443	85.121476	0.0
4	sk13	~	eta1	1.350601	0.518536	0.016411	82.29952	0.0
5	sk1	~	eta2	1.000000	0.586883	-	-	-
6	sk2	~	eta2	1.923701	0.650616	0.019852	96.902529	0.0
7	sk3	~	eta2	1.625467	0.619708	0.01727	94.118211	0.0
8	sk4	~	eta2	1.780563	0.695271	0.017746	100.334697	0.0
9	sk5	~	eta3	1.000000	0.708895	-	-	-
10	sk6	~	eta3	1.041920	0.566983	0.0118	88.300274	0.0
11	sk7	~	eta3	0.571564	0.186279	0.017336	32.969296	0.0
12	sk8	~	eta3	1.142588	0.548128	0.013232	86.352139	0.0
13	eta1	~~	eta1	0.141058	1.000000	0.00241	58.538392	0.0
14	eta1	~~	eta3	0.086894	0.583468	0.001321	65.772745	0.0
15	eta1	~~	eta2	0.048393	0.592172	0.000754	64.196223	0.0
16	eta3	~~	eta3	0.157234	1.000000	0.002309	68.099692	0.0
17	eta3	~~	eta2	0.056983	0.660446	0.000804	70.900865	0.0
18	eta2	~~	eta2	0.047344	1.000000	0.000794	59.602994	0.0
19	sk8	~~	sk8	0.477951	0.699556	0.003866	123.623766	0.0
20	sk2	~~	sk2	0.238696	0.576699	0.002049	116.507167	0.0
21	sk7	~~	sk7	1.428924	0.965300	0.009713	147.114692	0.0
22	sk12	~~	sk12	0.439910	0.704283	0.003382	130.062828	0.0
23	sk11	~~	sk11	0.203470	0.477629	0.002053	99.087724	0.0
24	sk5	~~	sk5	0.155650	0.497468	0.001763	88.276716	0.0

25	sk9	~~	sk9	0.277440	0.662942	0.002203	125.927537	0.0
26	sk10	~~	sk10	0.593947	0.684449	0.004635	128.145625	0.0
27	sk13	~~	sk13	0.699653	0.731120	0.005281	132.479658	0.0
28	sk1	~~	sk1	0.090112	0.655568	0.000715	125.973925	0.0
29	sk3	~~	sk3	0.200634	0.615962	0.001651	121.53224	0.0
30	sk4	~~	sk4	0.160409	0.516598	0.001493	107.443359	0.0
31	sk6	~~	sk6	0.360284	0.678531	0.002981	120.845663	0.0

[54]:
```
semopy.calc_stats(model)
```

[54]:

	DoF	DoF Baseline	chi2	chi2 p-value	chi2 Baseline	CFI
Value	62	78	7300.357081	0.0	120370.676623	0.939827

	GFI	AGFI	NFI	TLI	RMSEA	AIC	BIC
Value	0.939351	0.9237	0.939351	0.924299	0.05122	57.671909	310.067302

	LogLik
Value	0.164046

[ ]: