3-visualizacion

February 16, 2021

#

Ciencia de Datos

Víctor Muñiz Sánchez

Maestría en Cómputo Estadístico

Enero a junio 2021

1 Análisis de Componentes Principales (PCA)

1.1 PCA como un modelo interpretativo

[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
```

1.1.1 Nuestros datos: US air pollution

[5]:
```
f = open('data/usairpollution.txt', 'r')
print(f.read())
```

```
Air Pollution in US Cities

Description:

     Air pollution data of 41 US cities.

Format:

     A data frame with 41 observations on the following 7 variables.
```

```
    'SO2' SO2 content of air in micrograms per cubic metre.
    'temp' average annual temperature in Fahrenheit.
    'manu' number of manufacturing enterprises employing 20 or more workers.
    'popul' population size (1970 census); in thousands.
    'wind' average annual wind speed in miles per hour.
    'precip' average annual precipitation in inches.
    'predays' average number of days with precipitation per year.

Details:

    The annual mean concentration of sulphur dioxide, in micrograms per cubic
metre, is a measure of the air pollution of the city.
    The question of interest here is what aspects of climate and human ecology
as measured by the other six variables in the data determine pollution?

Source:

    R. R. Sokal and F. J. Rohlf (1981), _Biometry_, W. H. Freeman, San
    Francisco (2nd edition).
```

[4]:
```python
USairpollution = pd.read_csv('data/usairpollution.csv', index_col=0)
USairpollution
```

[4]:

City	SO2	temp	manu	popul	wind	precip	predays
Albany	46	47.6	44	116	8.8	33.36	135
Albuquerque	11	56.8	46	244	8.9	7.77	58
Atlanta	24	61.5	368	497	9.1	48.34	115
Baltimore	47	55.0	625	905	9.6	41.31	111
Buffalo	11	47.1	391	463	12.4	36.11	166
Charleston	31	55.2	35	71	6.5	40.75	148
Chicago	110	50.6	3344	3369	10.4	34.44	122
Cincinnati	23	54.0	462	453	7.1	39.04	132
Cleveland	65	49.7	1007	751	10.9	34.99	155
Columbus	26	51.5	266	540	8.6	37.01	134
Dallas	9	66.2	641	844	10.9	35.94	78
Denver	17	51.9	454	515	9.0	12.95	86
Des Moines	17	49.0	104	201	11.2	30.85	103
Detroit	35	49.9	1064	1513	10.1	30.96	129
Hartford	56	49.1	412	158	9.0	43.37	127
Houston	10	68.9	721	1233	10.8	48.19	103
Indianapolis	28	52.3	361	746	9.7	38.74	121
Jacksonville	14	68.4	136	529	8.8	54.47	116

Kansas City	14	54.5	381	507	10.0	37.00	99
Little Rock	13	61.0	91	132	8.2	48.52	100
Louisville	30	55.6	291	593	8.3	43.11	123
Memphis	10	61.6	337	624	9.2	49.10	105
Miami	10	75.5	207	335	9.0	59.80	128
Milwaukee	16	45.7	569	717	11.8	29.07	123
Minneapolis	29	43.5	699	744	10.6	25.94	137
Nashville	18	59.4	275	448	7.9	46.00	119
New Orleans	9	68.3	204	361	8.4	56.77	113
Norfolk	31	59.3	96	308	10.6	44.68	116
Omaha	14	51.5	181	347	10.9	30.18	98
Philadelphia	69	54.6	1692	1950	9.6	39.93	115
Phoenix	10	70.3	213	582	6.0	7.05	36
Pittsburgh	61	50.4	347	520	9.4	36.22	147
Providence	94	50.0	343	179	10.6	42.75	125
Richmond	26	57.8	197	299	7.6	42.59	115
Salt Lake City	28	51.0	137	176	8.7	15.17	89
San Francisco	12	56.7	453	716	8.7	20.66	67
Seattle	29	51.1	379	531	9.4	38.79	164
St. Louis	56	55.9	775	622	9.5	35.89	105
Washington	29	57.3	434	757	9.3	38.89	111
Wichita	8	56.6	125	277	12.7	30.58	82
Wilmington	36	54.0	80	80	9.0	40.25	114

Quitamos SO2 por el momento, y la variable `temp` se transforma en valores negativos negtemp, para que valores altos de todas las variables, representen un lugar con medio ambiente “poco atractivo”

[29]:
```
pollution = USairpollution.drop(['SO2'], axis = 1)
pollution['temp'] = pollution['temp']*-1
pollution.rename(columns={'temp':'negtemp'}, inplace=True)
pollution
```

[29]:

	negtemp	manu	popul	wind	precip	predays
City						
Albany	-47.6	44	116	8.8	33.36	135
Albuquerque	-56.8	46	244	8.9	7.77	58
Atlanta	-61.5	368	497	9.1	48.34	115
Baltimore	-55.0	625	905	9.6	41.31	111
Buffalo	-47.1	391	463	12.4	36.11	166
Charleston	-55.2	35	71	6.5	40.75	148
Chicago	-50.6	3344	3369	10.4	34.44	122
Cincinnati	-54.0	462	453	7.1	39.04	132
Cleveland	-49.7	1007	751	10.9	34.99	155
Columbus	-51.5	266	540	8.6	37.01	134
Dallas	-66.2	641	844	10.9	35.94	78
Denver	-51.9	454	515	9.0	12.95	86
Des Moines	-49.0	104	201	11.2	30.85	103

Detroit	-49.9	1064	1513	10.1	30.96	129
Hartford	-49.1	412	158	9.0	43.37	127
Houston	-68.9	721	1233	10.8	48.19	103
Indianapolis	-52.3	361	746	9.7	38.74	121
Jacksonville	-68.4	136	529	8.8	54.47	116
Kansas City	-54.5	381	507	10.0	37.00	99
Little Rock	-61.0	91	132	8.2	48.52	100
Louisville	-55.6	291	593	8.3	43.11	123
Memphis	-61.6	337	624	9.2	49.10	105
Miami	-75.5	207	335	9.0	59.80	128
Milwaukee	-45.7	569	717	11.8	29.07	123
Minneapolis	-43.5	699	744	10.6	25.94	137
Nashville	-59.4	275	448	7.9	46.00	119
New Orleans	-68.3	204	361	8.4	56.77	113
Norfolk	-59.3	96	308	10.6	44.68	116
Omaha	-51.5	181	347	10.9	30.18	98
Philadelphia	-54.6	1692	1950	9.6	39.93	115
Phoenix	-70.3	213	582	6.0	7.05	36
Pittsburgh	-50.4	347	520	9.4	36.22	147
Providence	-50.0	343	179	10.6	42.75	125
Richmond	-57.8	197	299	7.6	42.59	115
Salt Lake City	-51.0	137	176	8.7	15.17	89
San Francisco	-56.7	453	716	8.7	20.66	67
Seattle	-51.1	379	531	9.4	38.79	164
St. Louis	-55.9	775	622	9.5	35.89	105
Washington	-57.3	434	757	9.3	38.89	111
Wichita	-56.6	125	277	12.7	30.58	82
Wilmington	-54.0	80	80	9.0	40.25	114

PCA haciendo la descomposición espectral directamente

[70]:
```python
X = StandardScaler().fit_transform(pollution)
cov_x = np.cov(X.T)
vals, vecs = np.linalg.eig(cov_x)
# la descomposicion hecha con el modulo de algebra lineal de numpy no da los eigenvalores ordenados,
# entonces, los ordenamos de forma descendente
idx = vals.argsort()[::-1]
eigvals = vals[idx]
eigvecs = vecs[:,idx]
```

Varianza explicada (o desviación estándar)

[71]:
```python
dat = {'PC':range(1,7),'std':np.sqrt(eigvals), 'var_prop':eigvals/sum(eigvals),
       'cum_prop':np.cumsum(eigvals/sum(eigvals))}
stds = pd.DataFrame(data = dat)
stds
```

[71]:

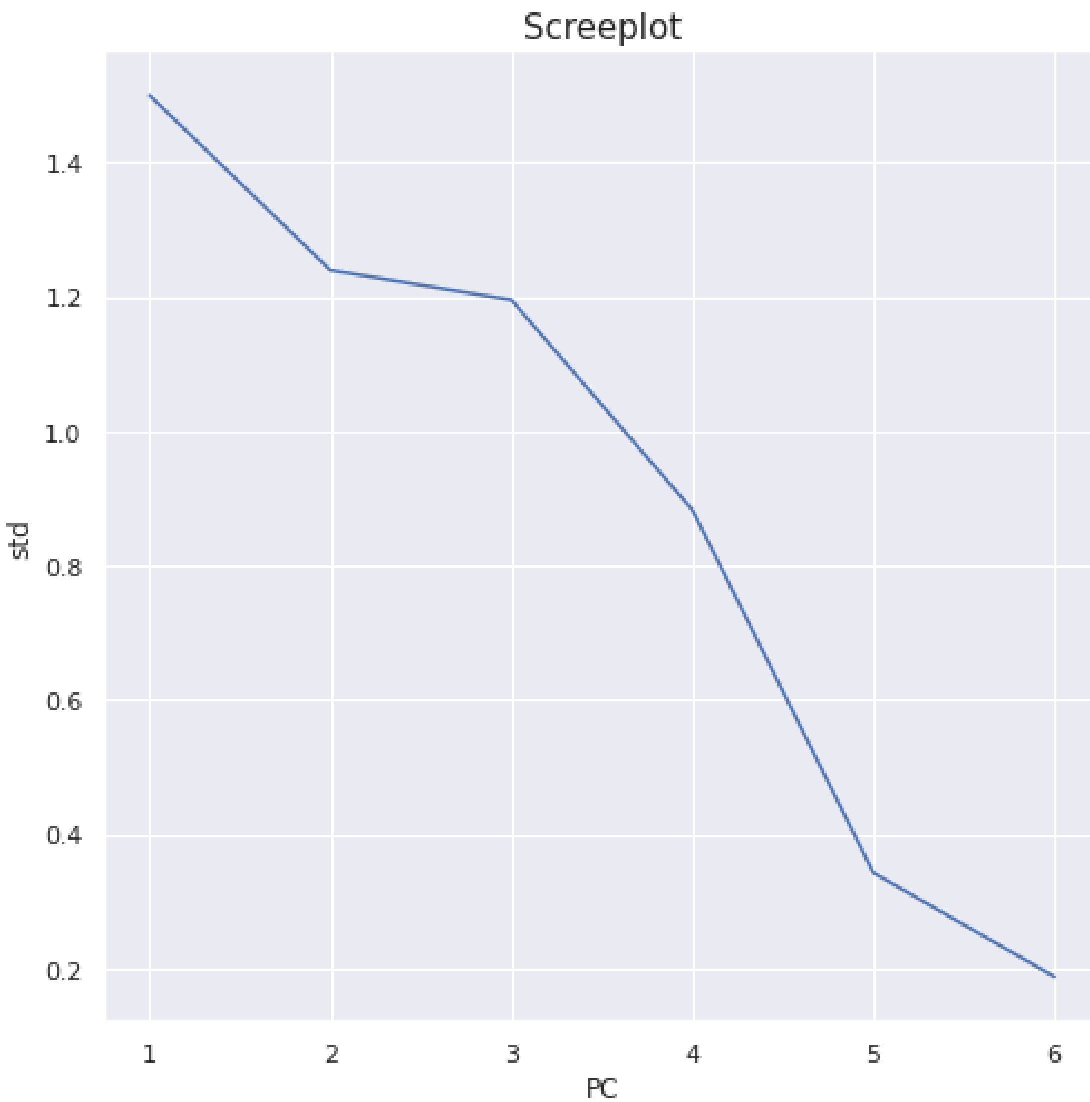
	PC	std	var_prop	cum_prop
0	1	1.500356	0.366027	0.366027
1	2	1.239936	0.249991	0.616018
2	3	1.195623	0.232442	0.848459
3	4	0.882742	0.126704	0.975164
4	5	0.342688	0.019095	0.994259
5	6	0.187905	0.005741	1.000000

[72]:

```
plt.figure(figsize=(8, 8))
sns.lineplot(x="PC", y="std", data=stds)
plt.title('Screeplot', fontsize=15)
```

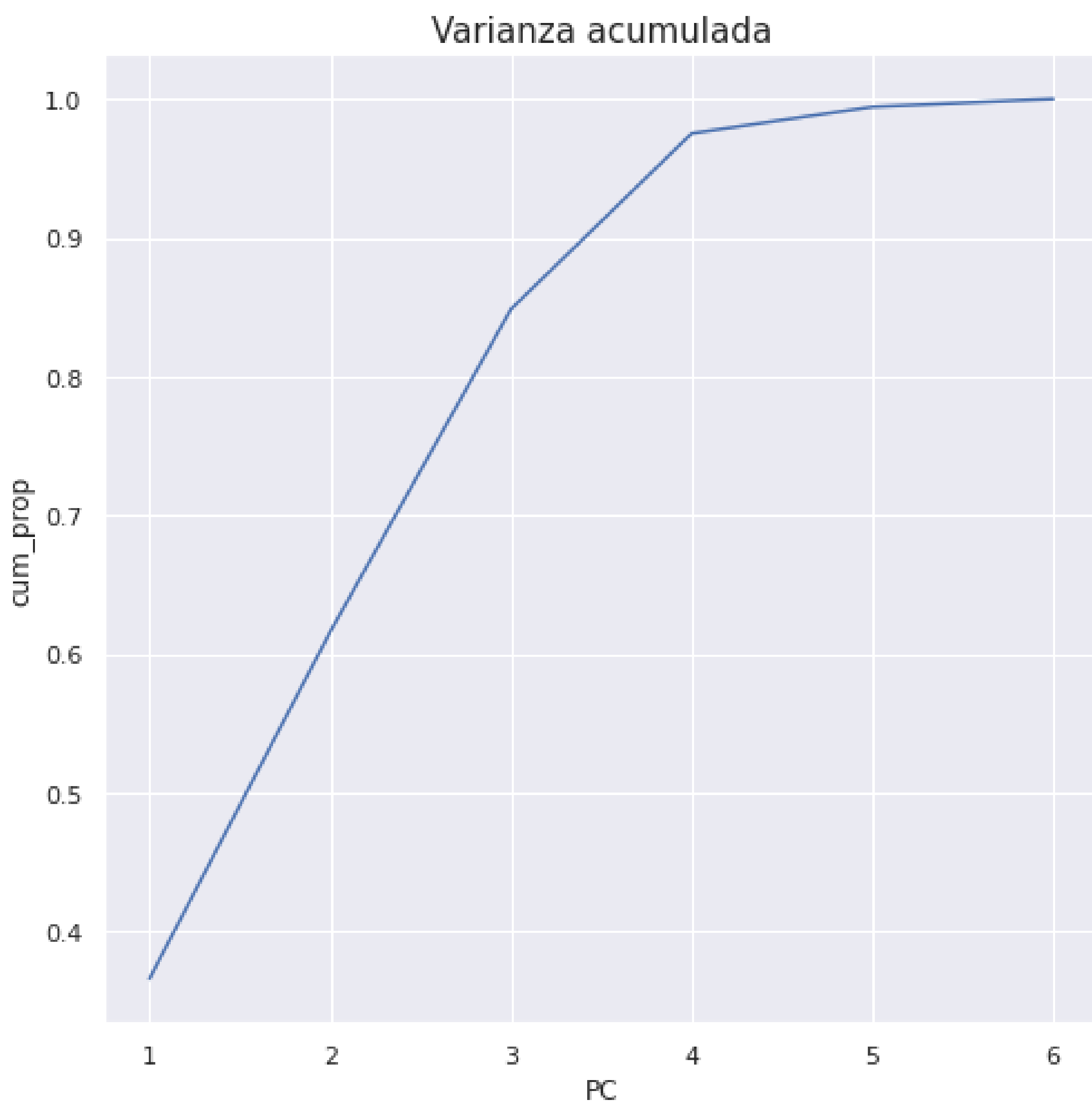
[72]:

```
Text(0.5, 1.0, 'Screeplot')
```



[13]:
```
plt.figure(figsize=(8, 8))
sns.lineplot(x="PC", y="cum_prop", data=stds)
plt.title('Varianza acumulada', fontsize=15)
```

[13]: Text(0.5, 1.0, 'Varianza acumulada')



[74]:
```
# componentes
comps = pd.DataFrame(data=eigvecs.T, columns=pollution.columns,
                     index=['pc1','pc2','pc3','pc4','pc5','pc6'])
print(comps.T)
```

```
              pc1       pc2       pc3       pc4       pc5       pc6
negtemp -0.329646  0.127597 -0.671686  0.306457 -0.558056 -0.136188
```

```
manu    -0.611542 -0.168058  0.272886  0.136841 -0.102042  0.702971
popul   -0.577822 -0.222453  0.350374  0.072481  0.078066 -0.694641
wind    -0.353839  0.130792 -0.297253 -0.869426  0.113267  0.024525
precip   0.040807  0.622858  0.504563 -0.171148 -0.568183 -0.060622
predays -0.237916  0.707765 -0.093089  0.311307  0.580004  0.021961
```

Con sklearn

[81]:
```
from sklearn.decomposition import PCA

pca = PCA()
# ajustar en los datos (estandarizados)
pca.fit(X)

dat = {'PC':range(1,7),'std':np.sqrt(pca.explained_variance_), 'var_prop':pca.
 ↪explained_variance_ratio_,
       'cum_prop':np.cumsum(pca.explained_variance_ratio_)}
stds = pd.DataFrame(data = dat)
stds
```

[81]:

	PC	std	var_prop	cum_prop
0	1	1.500356	0.366027	0.366027
1	2	1.239936	0.249991	0.616018
2	3	1.195623	0.232442	0.848459
3	4	0.882742	0.126704	0.975164
4	5	0.342688	0.019095	0.994259
5	6	0.187905	0.005741	1.000000

1.1.2 Loadings

Loadings (vectores propios de **S** o **R**)

- Magnitud
- Signos
- Contrastes

Observa que, en este ejemplo, dos variables se relacionan con "ecología humana" (popul y manu del primer PC) y cuatro relacionadas al clima (temp, wind, precip y predays).

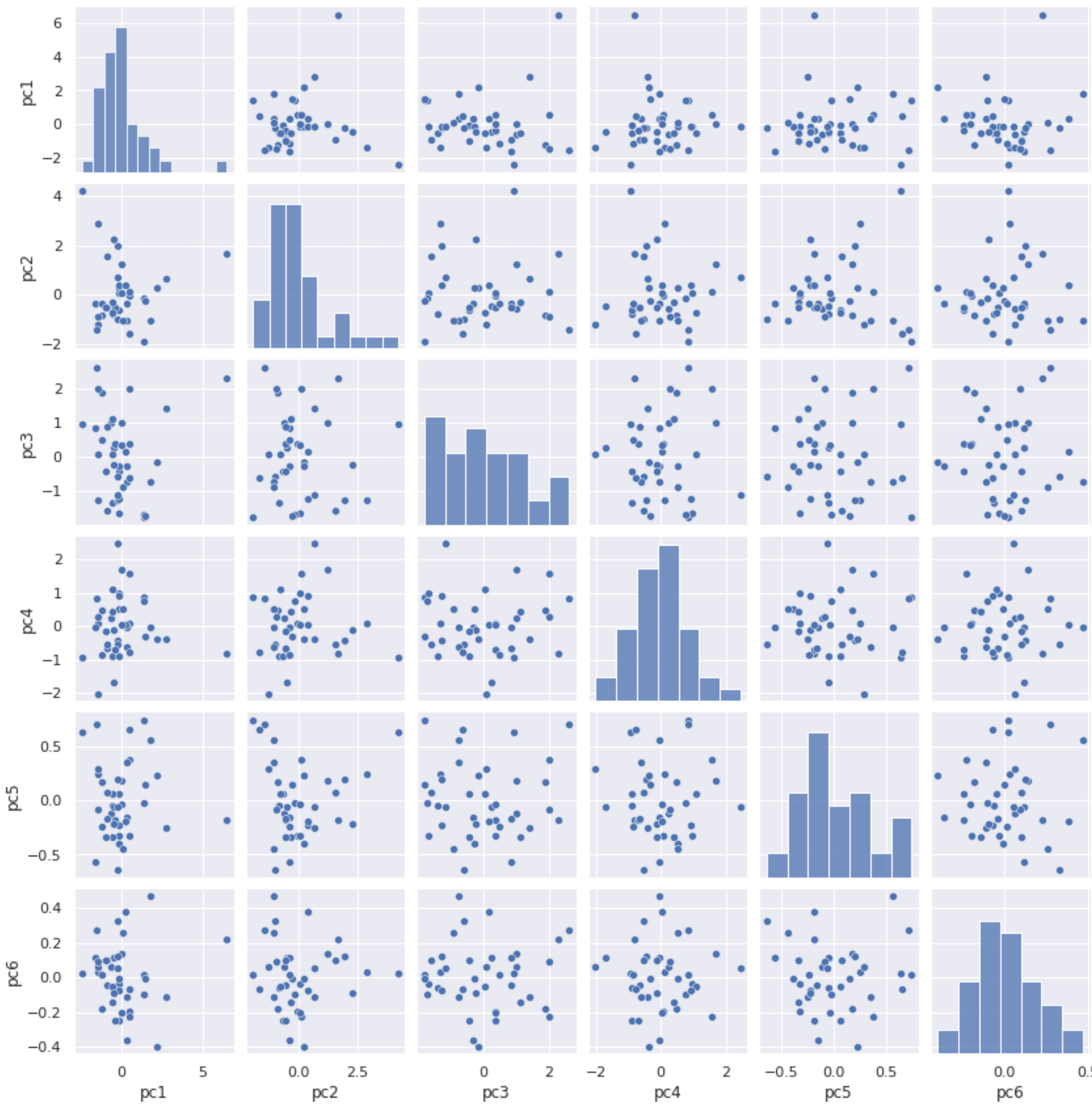
[78]:
```
comps = pd.DataFrame(data=pca.components_.T, columns=pollution.columns,
                     index=['pc1','pc2','pc3','pc4','pc5','pc6'])
print(comps)
```

```
      negtemp      manu     popul      wind    precip   predays
pc1  0.329646 -0.127597 -0.671686 -0.306457 -0.558056 -0.136188
pc2  0.611542  0.168058  0.272886 -0.136841 -0.102042  0.702971
pc3  0.577822  0.222453  0.350374 -0.072481  0.078066 -0.694641
pc4  0.353839 -0.130792 -0.297253  0.869426  0.113267  0.024525
pc5 -0.040807 -0.622858  0.504563  0.171148 -0.568183 -0.060622
pc6  0.237916 -0.707765 -0.093089 -0.311307  0.580004  0.021961
```

1.1.3 Scores

Proyección de los datos en los componentes principales

[19]:
```
# proyectar datos
proj = pd.DataFrame(pca.transform(X),columns =␣
 ↪['pc1','pc2','pc3','pc4','pc5','pc6'])
sns.set()
sns.pairplot(proj, height=2);
```

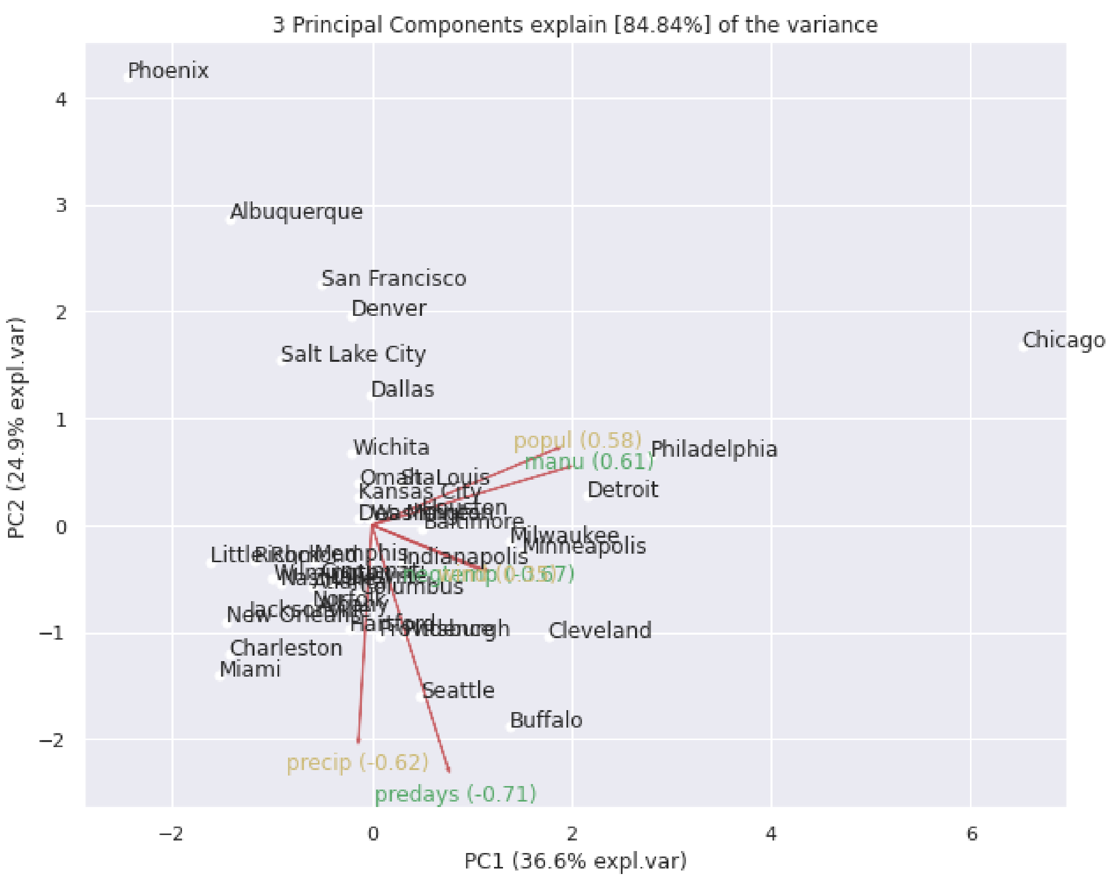


[20]:
```
import plotly.express as px

pca_dataset = pd.DataFrame({'pc1': proj['pc1'], 'pc2': proj['pc2'], 'city':␣
 ↪pollution.index})
```

```
fig = px.scatter(pca_dataset, x='pc1', y='pc2', hover_data=['city'])
fig.update_layout(
    autosize=False,
    width=600,
    height=600,
)
fig.show()
```

1.1.4 Biplot

[82]:
```
# usando el módulo pca (https://pypi.org/project/pca/)
from pca import pca
model = pca(n_components=3)

# Fit transform
results = model.fit_transform(X, row_labels=pollution.index,␣
 ↪col_labels=pollution.columns, verbose=False)
fig, ax = model.biplot(n_feat=6, legend=False, d3=False, cmap=None)
```



1.1.5 ¿Qué pasa en los últimos componentes?

Haremos una simulación de lo que vimos en clase: $\mathbf{x} \in \mathbb{R}^5$, donde $x_5 = \sum_{i=1}^4 x_i/4$.

[96]:
```
mean = (0,0,0,0)
cov = np.identity(4)
x = np.random.multivariate_normal(mean, cov, 100)
temp = np.sum(x,axis=1).reshape(x.shape[0],1)
x_data = np.append(x,temp/4,axis=1)
# nuestros datos
np.round(x_data[0:5,],3)
```

[96]:
```
array([[ 0.377,  0.499,  1.036,  1.101,  0.753],
       [-0.376, -0.129,  0.741,  0.26 ,  0.124],
       [ 0.857, -1.088,  0.524,  0.089,  0.095],
       [-0.158, -1.128,  0.706,  0.562, -0.004],
       [-1.788,  2.158, -0.512,  0.974,  0.208]])
```

Realizamos PCA. Observa el comportamiento del último PC, tanto en la varianza como en los loadings.

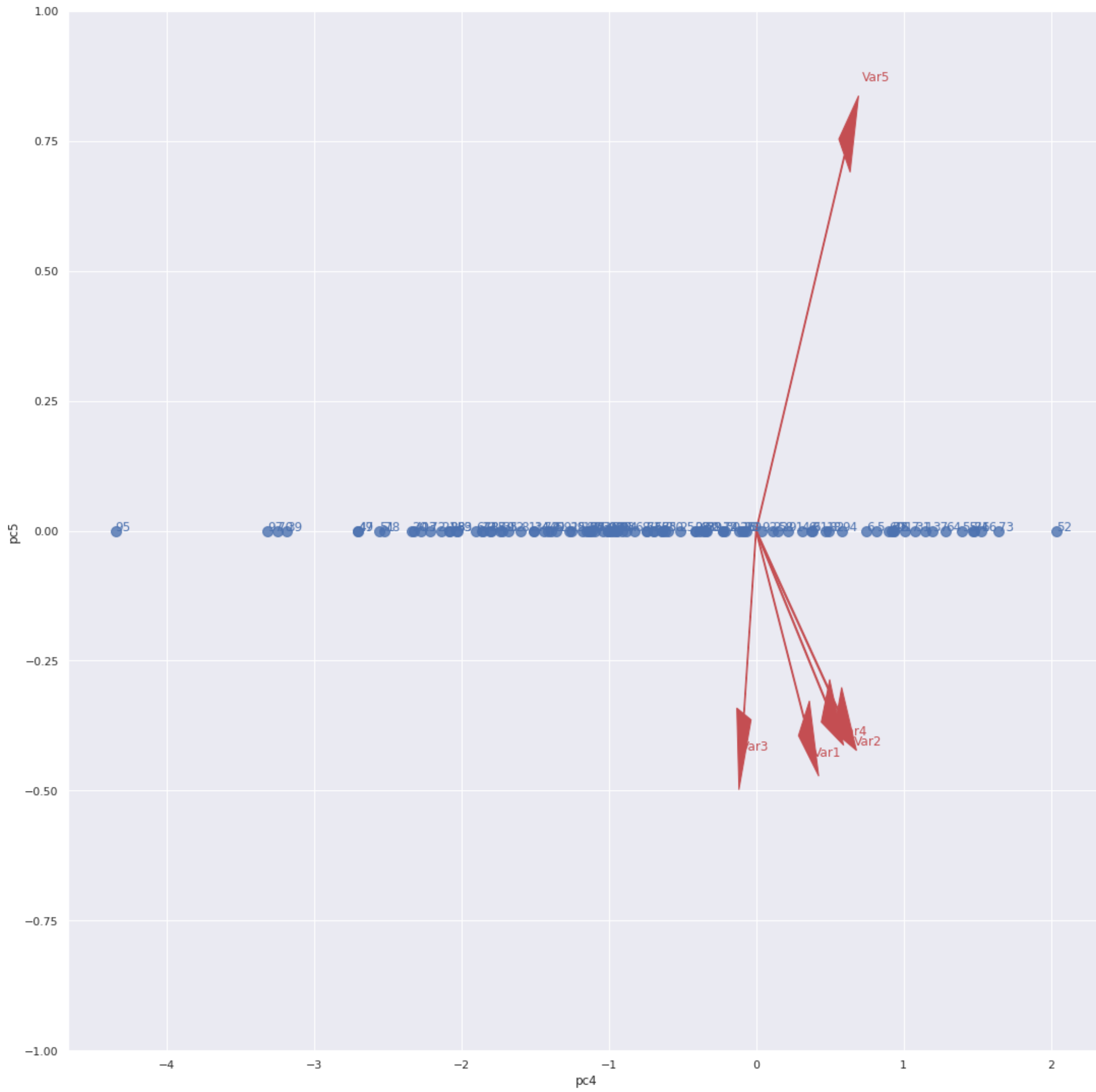
[97]:
```
pca2 = PCA()
x_std = StandardScaler().fit_transform(x_data)
pca2.fit(np.cov(x_std.T))
print('Std \n', np.round(np.sqrt(pca2.explained_variance_),3))
print('Prop. Var. \n', np.round(pca2.explained_variance_ratio_,3))
print('Loadings \n', np.round(pca2.components_.T,4))
```

```
Std
 [0.586 0.484 0.441 0.279 0.   ]
Prop. Var.
 [0.404 0.276 0.229 0.091 0.   ]
Loadings
 [[-0.3423  0.8036 -0.039   0.3246 -0.3607]
 [ 0.7555 -0.0546  0.0342  0.554  -0.3438]
 [-0.1518 -0.1456  0.9082 -0.0821 -0.3523]
 [-0.5234 -0.5727 -0.2636  0.4707 -0.3271]
 [-0.1226  0.045   0.3208  0.5995  0.7215]]
```

Una versión de biplot

[102]:
```
scores_x = pd.DataFrame(pca2.transform(x_std),columns =
 ↪['pc1','pc2','pc3','pc4','pc5'])
pca_datasetx = pd.DataFrame(data=scores_x)
xvector = pca2.components_[3]
yvector = pca2.components_[4]
# se visualizarán los últimos PC
xs = scores_x['pc4']
ys = scores_x['pc5']
```

[104]:
```
biplot = sns.lmplot('pc4', 'pc5', data=pca_datasetx, fit_reg = False,␣
 ↪sharex=False, sharey=False,
                    height = 15, scatter_kws={"s": 100})
biplot.set(ylim=(-1, 1))

# visualizacion de variables (espacio columna)
for i in range(len(xvector)):
    plt.arrow(0, 0, xvector[i], yvector[i],
              color='r', width=0.005, head_width=0.1)
    plt.text(xvector[i]*1.2, yvector[i]*1.2,
             "Var" + str(i+1), color='r')

# visualizacion de observaciones (espacio renglon)
for i in range(len(xs)):
    plt.text(xs[i], ys[i], i+1, color='b')

plt.show()
```


Var5
Var3
Var1
Var2
pc4
pc5