IndiceDeMarginacionGtoConPCA

March 24, 2026

1 Análisis de marginación con PCA

En este notebook se realiza un análisis exploratorio del Índice de Marginación 2020: primero se cargan y limpian los datos, después se aplica PCA para estudiar la estructura de las variables socioeconómicas tanto en Guanajuato como a nivel nacional, y finalmente se propone y compara un índice alternativo de marginación basado en componentes principales con el índice oficial (IMN 2020).

[390]:
```
from pathlib import Path
import numpy as np
import matplotlib.pylab as plt
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import pandas as pd
```

[394]:
```
! pip install pca --quiet
```

```
137613.85s - pydevd: Sending message related to process being replaced timed-out
after 5 seconds
```

[335]:
```
data_path = Path.cwd().parent / "data" / "IMM_2020.csv"
images_path = Path.cwd().parent / "reporte" / "images"
df_ = pd.read_csv(data_path)
```

Descripción de columnas: - CVE_ENT Clave de entidad federativa - NOM_ENT Nombre de entidad federativa - CVE_MUN Clave del municipio - NOM_MUN Nombre del municipio - POB_TOT Población total - ANALF Porcentaje de población analfabeta de 15 años o más - SBASC Porcentaje de población de 15 años o más sin educación básica - OVSDE Porcentaje de ocupantes en viviendas particulares habitadas sin drenaje ni excusado - OVSEE Porcentaje de ocupantes en viviendas particulares habitadas sin energía eléctrica - OVSAE Porcentaje de ocupantes en viviendas particulares habitadas sin agua entubada - OVPT Porcentaje de ocupantes en viviendas particulares habitadas con piso de tierra - VHAC Porcentaje de viviendas particulares con hacinamiento - PL.5000 Porcentaje de población que vive en localidades menores a 5 000 habitantes - PO2SM Porcentaje de población ocupada con ingresos de hasta 2 salarios mínimos - IM_2020 Índice de marginación, 2020 - GM_2020 Grado de marginación, 2020 - IMN_2020 Índice de marginación normalizado, 2020

2 Limpieza

[355]:
```
df = clean_columns(df_)
print(df.isna().sum())
df.columns.to_list()
df_gto = df[df['nom_ent'] == 'Guanajuato']

predictors = ['analf', 'sbasc',
       'ovsde', 'ovsee', 'ovsae', 'ovpt', 'vhac', 'pl_5000', 'po_2_sm']
```

```
cve_ent     0
nom_ent     0
cve_mun     0
nom_mun     0
pob_tot     0
analf       0
sbasc       0
ovsde       0
ovsee       0
ovsae       0
ovpt        0
vhac        0
pl_5000     0
po_2_sm     0
im_2020     0
gm_2020     0
imn_2020    0
dtype: int64
```

[356]:
```
df = df.drop(columns=['cve_ent', 'cve_mun', 'nom_mun'])

# imprime cuales son los intervalos de imn_2020 para cada gm_2020 el cual esta
df.groupby('gm_2020')['imn_2020'].apply(lambda x: (x.min(), x.max()))
```

[356]:
```
gm_2020
Alto         (0.7631353916, 0.8242210741)
Bajo          (0.854816289, 0.8853437636)
Medio        (0.8242451973, 0.8547600364)
Muy alto      (0.3351981072, 0.762990246)
Muy bajo      (0.885398498, 0.9770524586)
Name: imn_2020, dtype: object
```

3 PCA Guanajuato

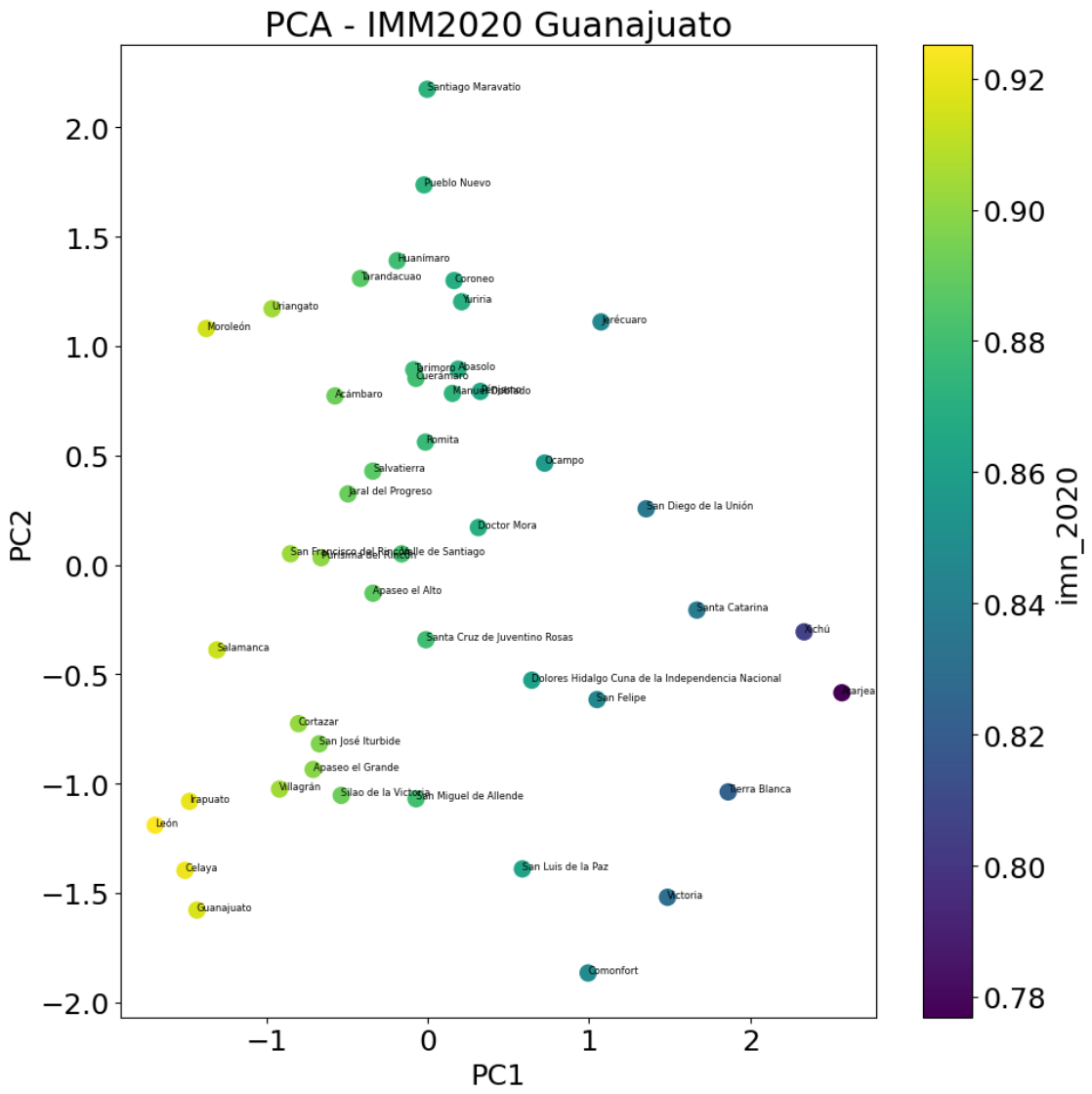
[380]:
```
df_tmp = df_gto
ncomp = 9
comp_columns = [f'pc{i+1}' for i in range(ncomp)]
```

[381]:
```
data = df_tmp[predictors]
X = StandardScaler().fit_transform(data)

# Sklearn PCA
pca = PCA(n_components=ncomp)
X_pca = pca.fit_transform(X)
print(pca.explained_variance_ratio_)

X_pca[:, [0, 1]] = StandardScaler().fit_transform(X_pca[:, [0, 1]])
```

```
[0.51107849 0.18027147 0.09163342 0.07342621 0.05367249 0.03462024
 0.02415163 0.02020293 0.01094312]
```

3.1 1 vs 2 Componente

[365]:
```
plt.figure(figsize=(10, 10))

color_col = 'imn_2020'
sc = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df_tmp[color_col], cmap='viridis',␣
 ↪s=100)

label_col = 'nom_mun'
for i, txt in enumerate(df_tmp['nom_mun']):
    plt.text(X_pca[i, 0], X_pca[i, 1], txt, fontsize=6)

cbar = plt.colorbar(sc)
cbar.set_label(color_col)

plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('PCA - IMM2020 Guanajuato')
plt.savefig(images_path / "pca_guanajuato.png", dpi=300)
plt.tight_layout()
plt.show()
plt.close()
```



3.2 Loadings

[382]:
```
# Componentes
comps = pd.DataFrame(data=pca.components_.T, columns=comp_columns,
                     index=data.columns)
print(comps)
```

```
             pc1       pc2       pc3       pc4       pc5       pc6       pc7  \
analf   0.422409  0.206124  0.029418 -0.140263 -0.076520 -0.290414  0.022235
sbasc   0.247247  0.518218 -0.043423  0.065118  0.731607 -0.136382 -0.046526
ovsde   0.390671 -0.216153 -0.069867  0.254967 -0.011641  0.505404  0.507492
ovsee   0.393634 -0.214151  0.205794 -0.132959 -0.020170 -0.480500  0.518954
ovsae   0.280668 -0.241444  0.690184 -0.332094  0.170503  0.320215 -0.379307
```

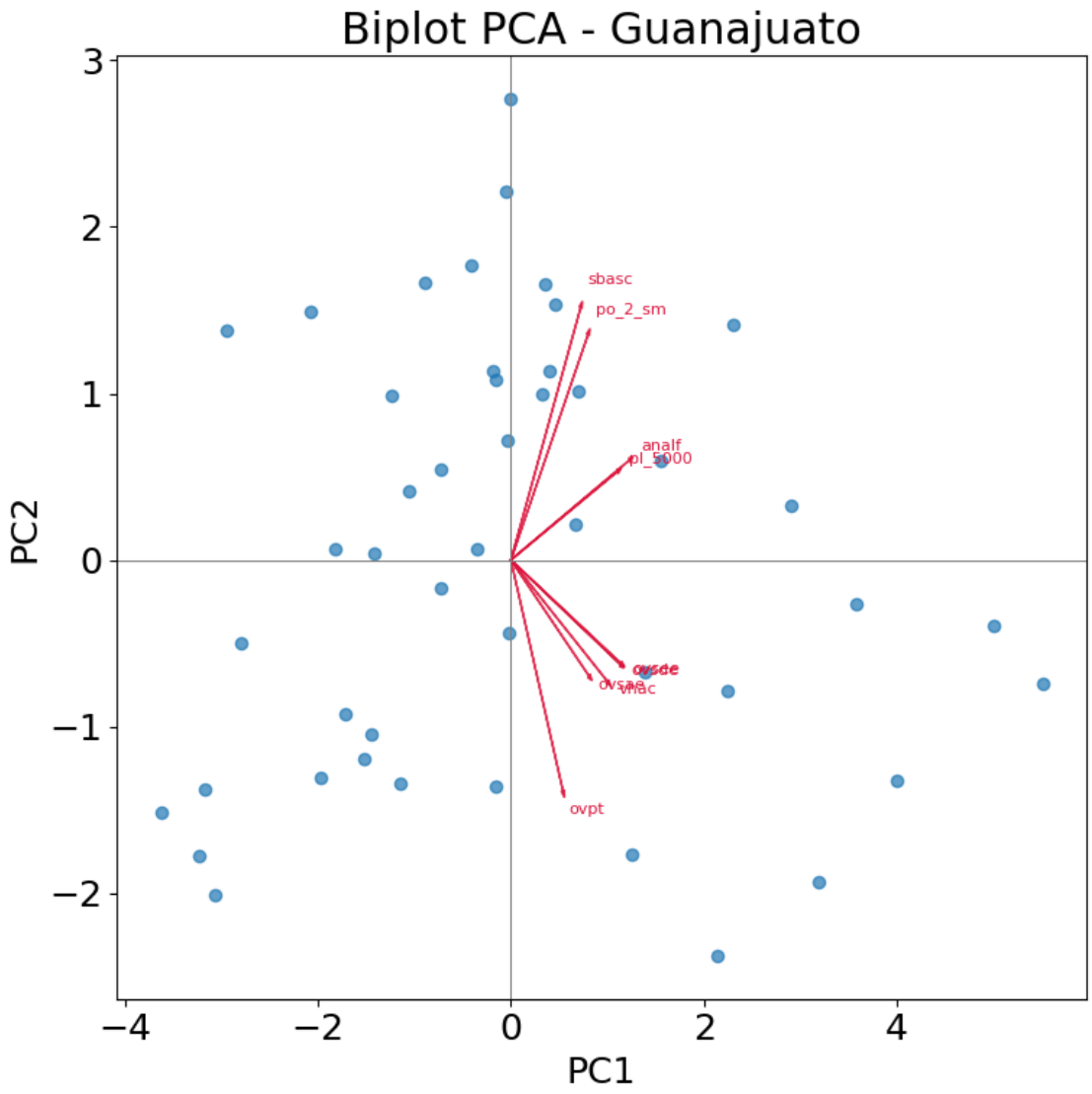
```
ovpt      0.185129 -0.475730 -0.585770 -0.479991  0.139660 -0.142391 -0.251240
vhac      0.345930 -0.252097 -0.224825  0.559823  0.198630  0.120917 -0.312106
pl_5000   0.381213  0.183256  0.036276  0.257331 -0.560493 -0.226111 -0.404043
po_2_sm   0.273550  0.463283 -0.280488 -0.416054 -0.237004  0.471366  0.053136

               pc8       pc9
analf    -0.140269  0.805096
sbasc    -0.198315 -0.262395
ovsde    -0.466227 -0.016701
ovsee     0.381035 -0.305573
ovsae    -0.006945 -0.027540
ovpt     -0.226473 -0.108161
vhac      0.527539  0.151818
pl_5000  -0.287645 -0.377213
po_2_sm   0.407511 -0.107334
```

El primer componente está capturando fuertemente la escolaridad, el drenaje, la energía eléctrica y el número de habitantes en la comunidad.

3.3 Biplot

[383]:
```python
from pca import pca as pca_package

# datos estandarizados (copia escribible)
X_biplot = StandardScaler().fit_transform(data).copy()

model = pca_package(n_components=2, normalize=False)
results = model.fit_transform(
    X_biplot,
    row_labels=data.index.astype(str),
    col_labels=data.columns,
    verbose=0
)

scores = results["PC"].to_numpy(copy=True)

# En pca package: loadings viene como (PCs, variables) => transponer a␣
 ↪(variables, PCs)
loadings = results["loadings"].to_numpy(copy=True).T

plt.figure(figsize=(8, 8))
plt.scatter(scores[:, 0], scores[:, 1], s=35, alpha=0.7)

scale = 3.0
for i, var_name in enumerate(data.columns):
    plt.arrow(
        0, 0,
        loadings[i, 0] * scale, loadings[i, 1] * scale,
```

```
        color="crimson", alpha=0.85, head_width=0.03, length_includes_head=True
    )
    plt.text(
        loadings[i, 0] * (scale + 0.2),
        loadings[i, 1] * (scale + 0.2),
        var_name, color="crimson", fontsize=8
    )

plt.axhline(0, color="gray", lw=0.8)
plt.axvline(0, color="gray", lw=0.8)
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.title("Biplot PCA - Guanajuato")
plt.tight_layout()
plt.savefig(images_path / "biplot_guanajuato.png", dpi=300)
plt.show()
plt.close()
```

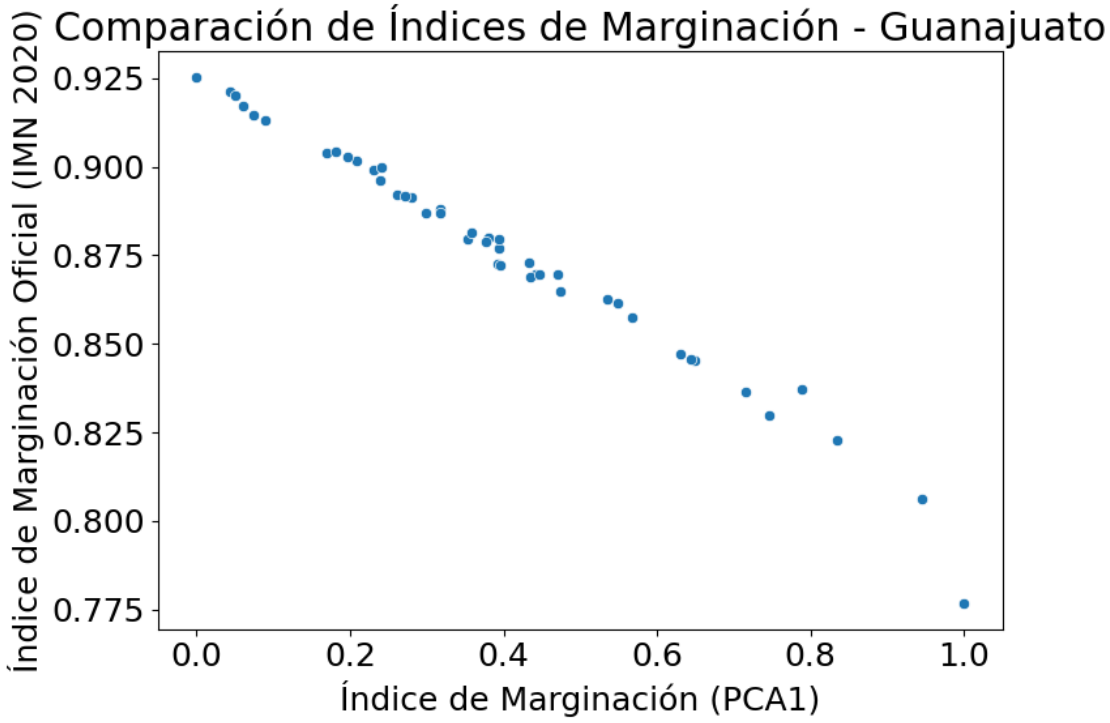


4 Propuesta de Índice de Marginación

4.1 PCA1

[368]:
```
# Indice de marginación
indice = X_pca[:,0]
indice_norm = (indice - indice.min()) / (indice.max() - indice.min())
plt.figure(figsize=(8, 6))
sns.scatterplot(x=indice_norm, y=df_tmp['imn_2020'])
plt.xlabel('Índice de Marginación (PCA1)')
plt.ylabel('Índice de Marginación Oficial (IMN 2020)')
plt.title('Comparación de Índices de Marginación - Guanajuato')
```

```
plt.tight_layout()
plt.savefig(images_path / "marginacion_comparacion_guanajuato_pc1.png", dpi=300)
plt.show()
plt.close()
```



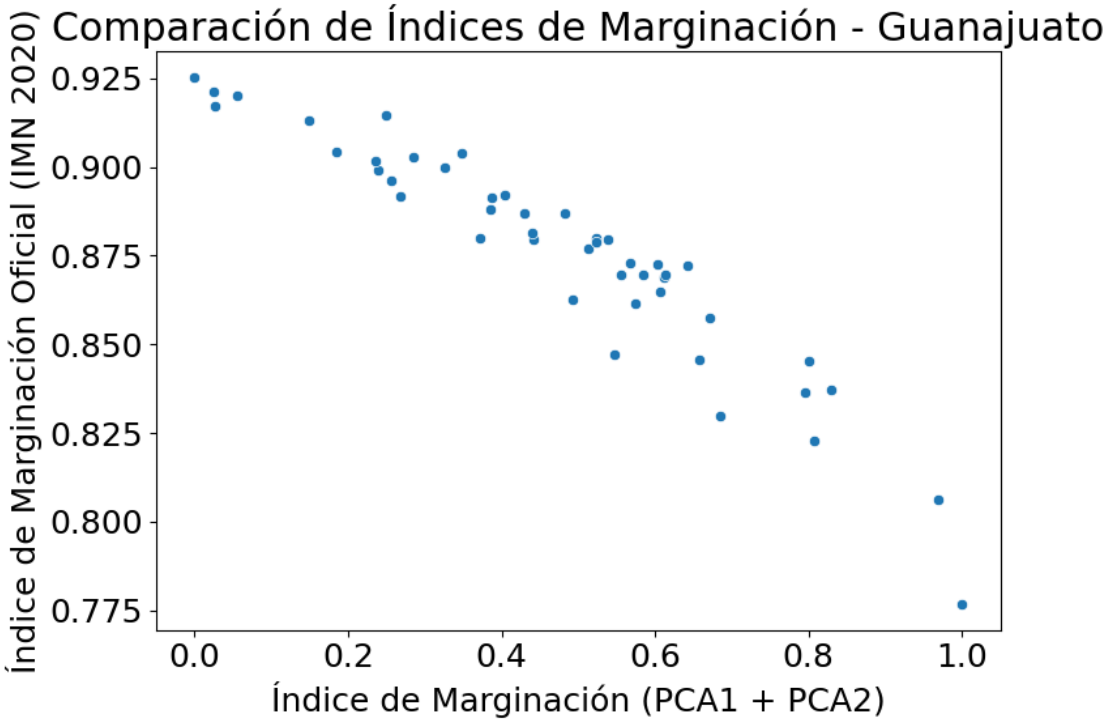
[387]:
```
# calcular la distancia al cuadrado de PC1 vs imn_2020
residual = df_tmp['imn_2020'] - indice_norm
residual_squared = residual ** 2
mse = np.mean(residual_squared)
print(f"Error cuadrático medio (MSE): {mse}")
```

```
Error cuadrático medio (MSE): 0.29424325571646465
```

4.2 Índice de Marginación PCA 1 + 2

[373]:
```
# Índice de marginación combinado (PC1 + PC2)
weights = pca.explained_variance_ratio_[:2]
weights = weights / weights.sum()

indice = X_pca[:, :2] @ weights
indice_norm = (indice - indice.min()) / (indice.max() - indice.min())

plt.figure(figsize=(8, 6))
```

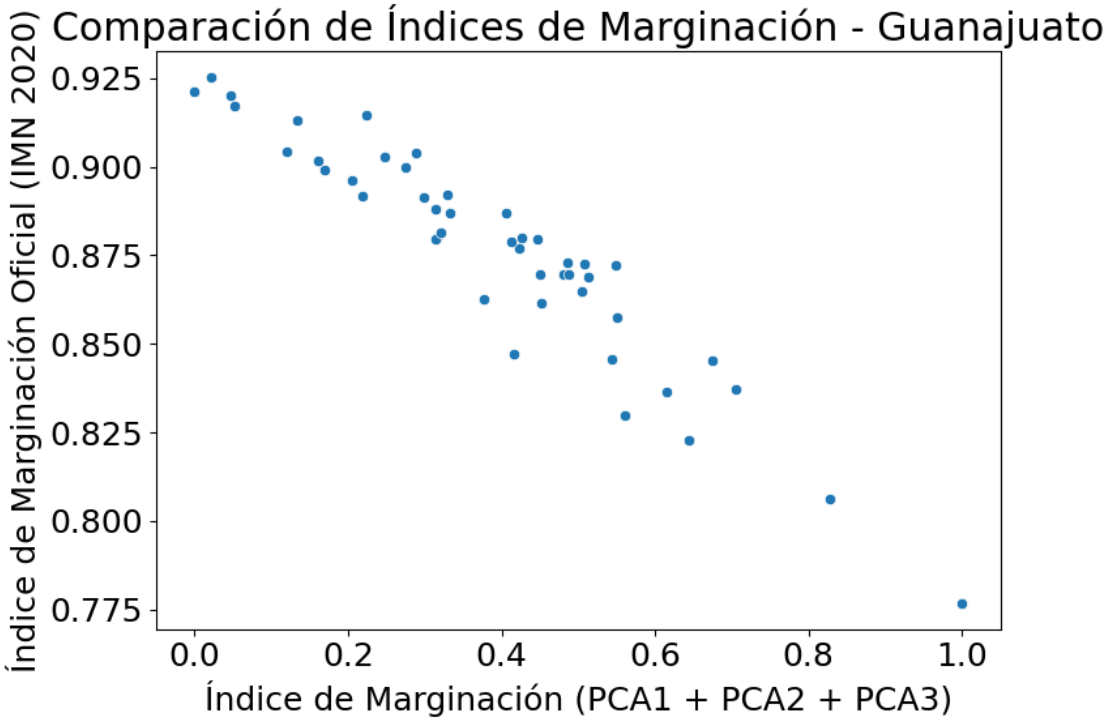
```
sns.scatterplot(x=indice_norm, y=df_tmp['imn_2020'])
plt.xlabel('Índice de Marginación (PCA1 + PCA2)')
plt.ylabel('Índice de Marginación Oficial (IMN 2020)')
plt.title('Comparación de Índices de Marginación - Guanajuato')
plt.tight_layout()
plt.savefig(images_path / "marginacion_comparacion_guanajuato_pc1_pc2.png",␣
 ↪dpi=300)
plt.show()
plt.close()
```



4.3 Índice de Marginación PCA 1 + 2 + 3

[376]:
```
# Índice de marginación combinado (PC1 + PC2 + PC3)
weights = pca.explained_variance_ratio_[:3]
weights = weights / weights.sum()

indice = X_pca[:, :3] @ weights
indice_norm = (indice - indice.min()) / (indice.max() - indice.min())

plt.figure(figsize=(8, 6))
sns.scatterplot(x=indice_norm, y=df_tmp['imn_2020'])
plt.xlabel('Índice de Marginación (PCA1 + PCA2 + PCA3)')
plt.ylabel('Índice de Marginación Oficial (IMN 2020)')
```

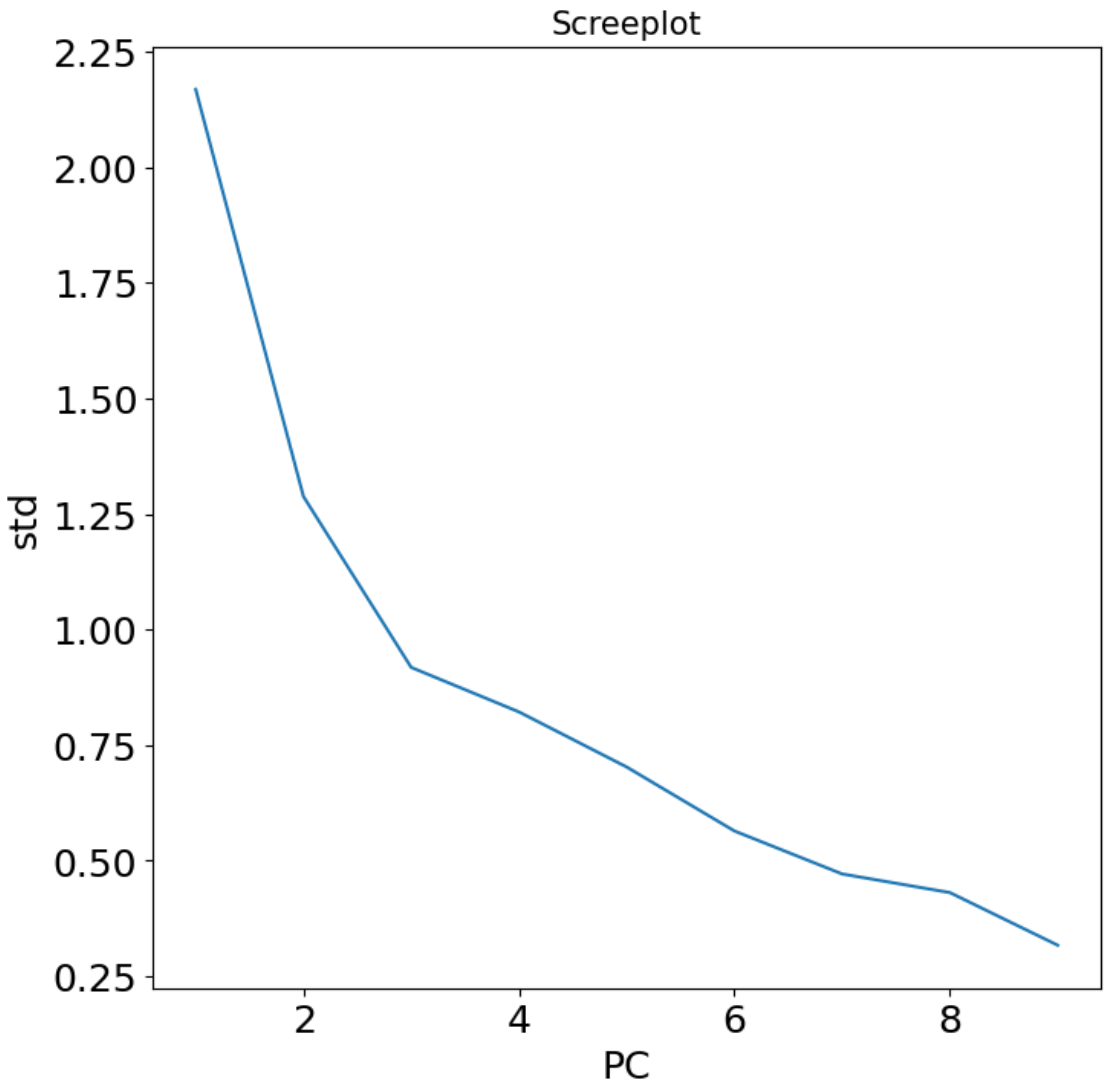
```
plt.title('Comparación de Índices de Marginación - Guanajuato')
plt.tight_layout()
plt.savefig(images_path / "marginacion_comparacion_guanajuato_pc1_pc2_pc3.png",␣
 ↪dpi=300)
plt.show()
plt.close()
```



4.4 Screenplot

[388]:
```
dat = {
       "PC": range(1, ncomp + 1),
       "std": np.sqrt(pca.explained_variance_),
       "var_prop": pca.explained_variance_ratio_,
       "cum_prop": np.cumsum(pca.explained_variance_ratio_),
}
stds = pd.DataFrame(dat)
stds

plt.figure(figsize=(8, 8))
sns.lineplot(x="PC", y="std", data=stds)
plt.title('Screeplot', fontsize=15)
plt.savefig(images_path / "screeplot_guanajuato.png", dpi=300)
plt.show()
```

```
plt.close()
```



[325]:
```
plt.figure(figsize=(8, 8))
sns.lineplot(x="PC", y="cum_prop", data=stds)
plt.title('Varianza acumulada', fontsize=15)
plt.show()
plt.close()
```

5 PCA Nacional

[326]:
```
df_tmp = df
ncomp = 2
comp_columns = [f'pc{i+1}' for i in range(ncomp)]
```

[327]:
```
data = df_tmp[predictors]
X = StandardScaler().fit_transform(data)

# Sklearn PCA
pca = PCA(n_components=ncomp)
X_pca = pca.fit_transform(X)
# X_pca[:, [0, 1]] = StandardScaler().fit_transform(X_pca[:, [0, 1]])
```

5.1 1 vs 2 Componente

[328]:
```
plt.figure(figsize=(10, 10))

color_col = 'imn_2020'
sc = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df_tmp[color_col], cmap='viridis',␣
 ↪s=100)

label_col = 'nom_ent'
for i, txt in enumerate(df_tmp[label_col]):
    plt.text(X_pca[i, 0], X_pca[i, 1], txt, fontsize=3)

cbar = plt.colorbar(sc)
cbar.set_label(color_col)

plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('PCA - IMM2020 Nacional')
plt.savefig(images_path / "pca_nacional.png", dpi=300)
plt.tight_layout()
plt.show()
plt.close()
```

5.2 Loadings

[329]:
```
# Componentes
comps = pd.DataFrame(data=pca.components_.T, columns=comp_columns,
                     index=data.columns)
print(comps)
```

```
             pc1       pc2
analf   0.411104 -0.108659
sbasc   0.394736 -0.291060
ovsde   0.237374  0.509972
```

```
ovsee    0.265188  0.561235
ovsae    0.238413  0.277631
ovpt     0.371528  0.118576
vhac     0.363212  0.021881
pl_5000  0.295071 -0.331886
po_2_sm  0.366936 -0.355886
```

5.3 Biplot

[330]:
```python
from pca import pca

# datos estandarizados (copia escribible)
X_biplot = StandardScaler().fit_transform(data).copy()

model = pca(n_components=2, normalize=False)
results = model.fit_transform(
    X_biplot,
    row_labels=data.index.astype(str),
    col_labels=data.columns,
    verbose=0
)

scores = results["PC"].to_numpy(copy=True)

# En pca package: loadings viene como (PCs, variables) => transponer a␣
 ↪(variables, PCs)
loadings = results["loadings"].to_numpy(copy=True).T

plt.figure(figsize=(8, 8))
plt.scatter(scores[:, 0], scores[:, 1], s=35, alpha=0.7)

scale = 3.0
for i, var_name in enumerate(data.columns):
    plt.arrow(
        0, 0,
        loadings[i, 0] * scale, loadings[i, 1] * scale,
        color="crimson", alpha=0.85, head_width=0.03, length_includes_head=True
    )
    plt.text(
        loadings[i, 0] * (scale + 0.2),
        loadings[i, 1] * (scale + 0.2),
        var_name, color="crimson", fontsize=8
    )

plt.axhline(0, color="gray", lw=0.8)
plt.axvline(0, color="gray", lw=0.8)
plt.xlabel("PC1")
```

```
plt.ylabel("PC2")
plt.title("Biplot PCA - Nacional")
plt.tight_layout()
plt.savefig(images_path / "biplot_nacional.png", dpi=300)
plt.show()
plt.close()
```

6 Propuesta de Índice de Marginación

6.1 PCA1

[331]:
```
# Indice de marginación
indice = X_pca[:,0]
indice_norm = (indice - indice.min()) / (indice.max() - indice.min())
```

Comparación con actual

[ ]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(x=indice_norm, y=df_tmp['imn_2020'])
plt.xlabel('Índice de Marginación (PCA1)')
plt.ylabel('Índice de Marginación Oficial (IMN 2020)')
plt.title('Comparación de Índices de Marginación - Nacional')
plt.tight_layout()
plt.savefig(images_path / "marginacion_comparacion_nacional_pc1.png", dpi=300)
plt.show()
plt.close()
```

[398]:
```
# Transformar para agregar al reporte
! jupyter nbconvert 3.ipynb \
  --to latex \
  --TemplateExporter.exclude_preamble=True \
  --output-dir=converted_scripts/
# \pagestyle{empty}
# \AtBeginDocument{\let\maketitle\relax}
```

```
138060.94s - pydevd: Sending message related to process being replaced timed-out
after 5 seconds

[NbConvertApp] WARNING | Config option `exclude_preamble` not recognized by
`LatexExporter`.  Did you mean one of: `exclude_input, exclude_input_prompt,
exclude_raw`?
[NbConvertApp] WARNING | Config option `exclude_preamble` not recognized by
`LatexExporter`.  Did you mean one of: `exclude_input, exclude_input_prompt,
exclude_raw`?
[NbConvertApp] Converting notebook 3.ipynb to latex
[NbConvertApp] Support files will be in 3_files/
[NbConvertApp] Writing 63332 bytes to converted_scripts/3.tex
```

[ ]: