PCA

Agregaremos los componentes principales al panel principal para cada posición. Sin embargo, primero obtendremos el número óptimo de componentes a usar por periodo mediante el método del codo y el método de permutaciones.

Importemos los modulos necesarios así como especificar la configuración deseada.

In [1]:
```
import pandas as pd
import numpy as np
import math
import os
import warnings
import statsmodels.api as sm
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from sklearn.decomposition import PCA
from matplotlib.colors import ListedColormap
from termcolor import colored

print('Modulos importados')
```

```
Modulos importados
```

In [2]:
```
# Configuraciones
warnings.filterwarnings('ignore')
```

In [3]:
```
# Directorio de trabajo
print("Directorio de trabajo previo: " + str(os.getcwd()))
# Cambiemoslo
os.chdir('/home/usuario/Documentos/Github/Proyectos/MLB_HN/')
```

```
Directorio de trabajo previo: /home/usuario/Documentos/Github/Proyectos/MLB_H
N/ETL_Scripts/Hand_Made_Scripts/Panel_merge
```

In [4]:
```
# Veamos el directorio actual de trabajo
print(os.getcwd())
# El directorio anterior es el correcto, pero si no lo fuese, hacemos lo sigu
path = '/home/usuario/Documentos/Github/Proyectos/MLB_HN'
print("Nuevo directorio de trabajo: " + str(os.chdir(path)))
```

```
/home/usuario/Documentos/Github/Proyectos/MLB_HN
Nuevo directorio de trabajo: None
```

Importación de las bases de datos

Importemos los páneles de ambas tipos de bases de datos: Anuales y acumuladas. Las bases de datos por juego no cuentan con más de 10 observaciones, razón por la que no se usarán

In [5]:
```
# Paths
cum_path = 'ETL_Data/Panel/Cumulative/Dynamic_model/'
csv = '.csv'

# Cumulative:
hitter_cum = pd.read_csv(cum_path + 'panel_hitters_cum_t_1' + csv)
fielder_cum = pd.read_csv(cum_path + 'panel_fielders_cum_t_1' + csv)
```

Veamos las dimensaiones de los páneles

In [6]:
```
# Dimentions
print("Acumulada \nBateadores:")
print(hitter_cum.shape)
print("Fildeadores:")
print(fielder_cum.shape)
```

```
Acumulada
Bateadores:
(570, 203)
Fildeadores:
(542, 217)
```

Se aprecia que son de las mismas dimensiones. Lo que haremos ahora es crear una lista de las variables a las que le agregaremos un sufijo que indique de qué panel parten

In [7]:

```
hitter_varlist = [
                  'At-bats_2_t',
                  'At-bats_t',
                  'At_bats_2_t',
                  'At_bats_t',
                  'Bateos_2_t',
                  'Bateos_promedio_2_t',
                  'Bateos_promedio_t',
                  'Bateos_t',
                  'Dobles_2_t',
                  'Dobles_t',
                  'Home-runs_t',
                  'Home_runs_2_t',
                  'Home_runs_t',
                  'Juegos totales_t',
                  'Juegos_iniciados_2_t',
                  'Juegos_iniciados_t',
                  'Juegos_t',
                  'Juegos_totales_t',
                  'Porcentaje_On-base-plus-slugging_2_t',
                  'Porcentaje_On-base-plus-slugging_t',
                  'Porcentaje_On_base_plus_slugging_2_t',
                  'Porcentaje_On_base_plus_slugging_t',
                  'Porcentaje_juegos_iniciados_2_t',
                  'Porcentaje_juegos_iniciados_t',
                  'Porcentaje_juegos_t',
                  'Porcentaje_on-base_2_t',
                  'Porcentaje_on-base_t',
                  'Porcentaje_on_base_2_t',
                  'Porcentaje_on_base_t',
                  'Porcentaje_slugging_2_t',
                  'Porcentaje_slugging_t',
                  'Runs-batted-in_2_t',
                  'Runs-batted-in_t',
                  'Runs_batted_in_2_t',
                  'Runs_batted_in_t',
                  'Triples_2_t',
                  'Triples_t',
                  'WAR_2_t',
                  'WAR_t',
                  'X_At_bats_2_t_1',
                  'X_At_bats_t_1',
                  'X_Bateos_2_t_1',
                  'X_Bateos_promedio_2_t_1',
                  'X_Bateos_promedio_t_1',
                  'X_Bateos_t_1',
                  'X_Dobles_2_t_1',
                  'X_Dobles_t_1',
                  'X_Home_runs_2_t_1',
                  'X_Home_runs_t_1',
                  'X_Juegos_iniciados_2_t_1',
                  'X_Juegos_iniciados_t_1',
                  'X_Porcentaje_On_base_plus_slugging_2_t_1',
                  'X_Porcentaje_On_base_plus_slugging_t_1',
                  'X_Porcentaje_on_base_2_t_1',
                  'X_Porcentaje_on_base_t_1',
                  'X_Porcentaje_slugging_2_t_1',
                  'X_Porcentaje_slugging_t_1',
```

```
                    'X_Runs_batted_in_2_t_1',
                    'X_Runs_batted_in_t_1',
                    'X_Triples_2_t_1',
                    'X_Triples_t_1',
                    'X_WAR_2_t_1',
                    'X_WAR_t_1',
                    'X_At_bats_2_t',
                    'X_At_bats_t',
                    'X_Bateos_2_t',
                    'X_Bateos_promedio_2_t',
                    'X_Bateos_promedio_t',
                    'X_Bateos_t',
                    'X_Dobles_2_t',
                    'X_Dobles_t',
                    'X_Home_runs_2_t',
                    'X_Home_runs_t',
                    'X_Juegos_iniciados_2_t',
                    'X_Juegos_iniciados_t',
                    'X_Porcentaje_On_base_plus_slugging_2_t',
                    'X_Porcentaje_On_base_plus_slugging_t',
                    'X_Porcentaje_on_base_2_t',
                    'X_Porcentaje_on_base_t',
                    'X_Porcentaje_slugging_2_t',
                    'X_Porcentaje_slugging_t',
                    'X_Runs_batted_in_2_t',
                    'X_Runs_batted_in_t',
                    'X_Triples_2_t',
                    'X_Triples_t',
                    'X_WAR_2_t',
                    'X_WAR_t'
                    ]
```

In [8]:

```
fielder_varlist = [
                    'Bateos_2_t',
                    'Bateos_t',
                    'Carreras_2_t',
                    'Carreras_ganadas_2_t',
                    'Carreras_ganadas_t',
                    'Carreras_t',
                    'Comando_2_t',
                    'Comando_t',
                    'Control_2_t',
                    'Control_t',
                    'Control_t_1',
                    'Dominio_2_t',
                    'Dominio_t',
                    'ERA_2_t',
                    'ERA_t',
                    'Inning_pitched_2_t',
                    'Inning_pitched_t',
                    'Juegos_totales_t',
                    'Juegos_iniciados_t',
                    'Juegos_t',
                    'Losses_2_t',
                    'Losses_t',
                    'Porcentaje_juegos_t',
                    'Promedio_victorias_t',
                    'Saves_2_t',
```

```
                    'Saves_t',
                    'Strike-outs_2_t',
                    'Strike-outs_t',
                    'Strike_outs_2_t',
                    'Strike_outs_t',
                    'WAR_2_t',
                    'WAR_t',
                    'WHIP_2_t',
                    'WHIP_t',
                    'Walks_2_t',
                    'Walks_t',
                    'Wins_2_t',
                    'Wins_t',
                    'X_Bateos_2_t_1',
                    'X_Bateos_t_1',
                    'X_Carreras_2_t_1',
                    'X_Carreras_ganadas_2_t_1',
                    'X_Carreras_ganadas_t_1',
                    'X_Carreras_t_1',
                    'X_Comando_2_t_1',
                    'X_Comando_t_1',
                    'X_Control_2_t_1',
                    'X_Control_t_1',
                    'X_Dominio_2_t_1',
                    'X_Dominio_t_1',
                    'X_ERA_2_t_1',
                    'X_ERA_t_1',
                    'X_Inning_pitched_2_t_1',
                    'X_Inning_pitched_t_1',
                    'X_Losses_2_t_1',
                    'X_Losses_t_1',
                    'X_Saves_2_t_1',
                    'X_Saves_t_1',
                    'X_Strike_outs_2_t_1',
                    'X_Strike_outs_t_1',
                    'X_WAR_2_t_1',
                    'X_WAR_t_1',
                    'X_WHIP_2_t_1',
                    'X_WHIP_t_1',
                    'X_Walks_2_t_1',
                    'X_Walks_t_1',
                    'X_Wins_2_t_1',
                    'X_Wins_t_1',
                    'X_Bateos_2_t',
```

In [9]:
```
hitter_dynamic_panel = hitter_cum.copy()
fielder_dynamic_panel = fielder_cum.copy()
```

```
                    'X_Carreras_ganadas_2_t',
                    'X_Carreras_ganadas_t',
                    'X_Carreras_t',
                    'X_Comando_2_t',
                    'X_Comando_t',
                    'X_Control_2_t',
```

In [10]:
```
# Get the total number of NaN values in the dataframe
print("Bateadores:")
print(hitter_dynamic_panel.isna().sum().sum())
print("Fildeadores:")
print(fielder_dynamic_panel.isna().sum().sum())
```

```
                    'X_Inning_pitched_2_t',
                    'X_Inning_pitched_t',
```

```
                    'X_Losses_2_t',
                    'X_Losses_t',
                    'X_Saves_2_t',
                    'X_Saves_t',
F                   'X_Strike_outs_2_t',
                    'X_Strike_outs_t',
                    'X_WAR_2_t',
```

In [11]:
```
hitter_dynamic_panel = hitter_dynamic_panel.drop_duplicates()
fielder_dynamic_panel = fielder_dynamic_panel.drop_duplicates()
```

```
                    'X_WHIP_t',
                    'X_Walks_2_t',
```

In [12]:
```
print("Bateadores:")
print(hitter_dynamic_panel.shape)
print("Fildeadores:")
print(fielder_dynamic_panel.shape)
```

```
Bateadores:
(570, 203)
Fildeadores:
(542, 217)
```

PCA

Ahora, filtraremos aquellas columnas que inicien con el prefijo **X** y luego de acuerdo al periodo al que pertenecen con el propósito de hacer un análisis de componentes principales.

In [13]:
```
# t-1:
hitter_dynamic_panel_aux = hitter_dynamic_panel.filter(regex = '^X_')
X_hitter_t_1 = hitter_dynamic_panel_aux.filter(like = '_t_1')

# filter columns with suffix '_t' and not '_t_1'
filtered_cols = hitter_dynamic_panel_aux.filter(regex = r'_t$').columns.diffe

# select the filtered columns from the dataframe
X_hitter_t = hitter_dynamic_panel_aux[filtered_cols]
```

In [14]:
```
# t-1:
fielder_dynamic_panel_aux = fielder_dynamic_panel.filter(regex = '^X_')
X_fielder_t_1 = fielder_dynamic_panel_aux.filter(like = '_t_1')

# filter columns with suffix '_t' and not '_t_1'
filtered_cols = fielder_dynamic_panel_aux.filter(regex = r'_t$').columns.diff

# select the filtered columns from the dataframe
X_fielder_t = fielder_dynamic_panel_aux[filtered_cols]
```

Para corroborar el método del *codo*, se usarán repeticiones para determinar el número de componentes que solo captura señales y no ruidos.

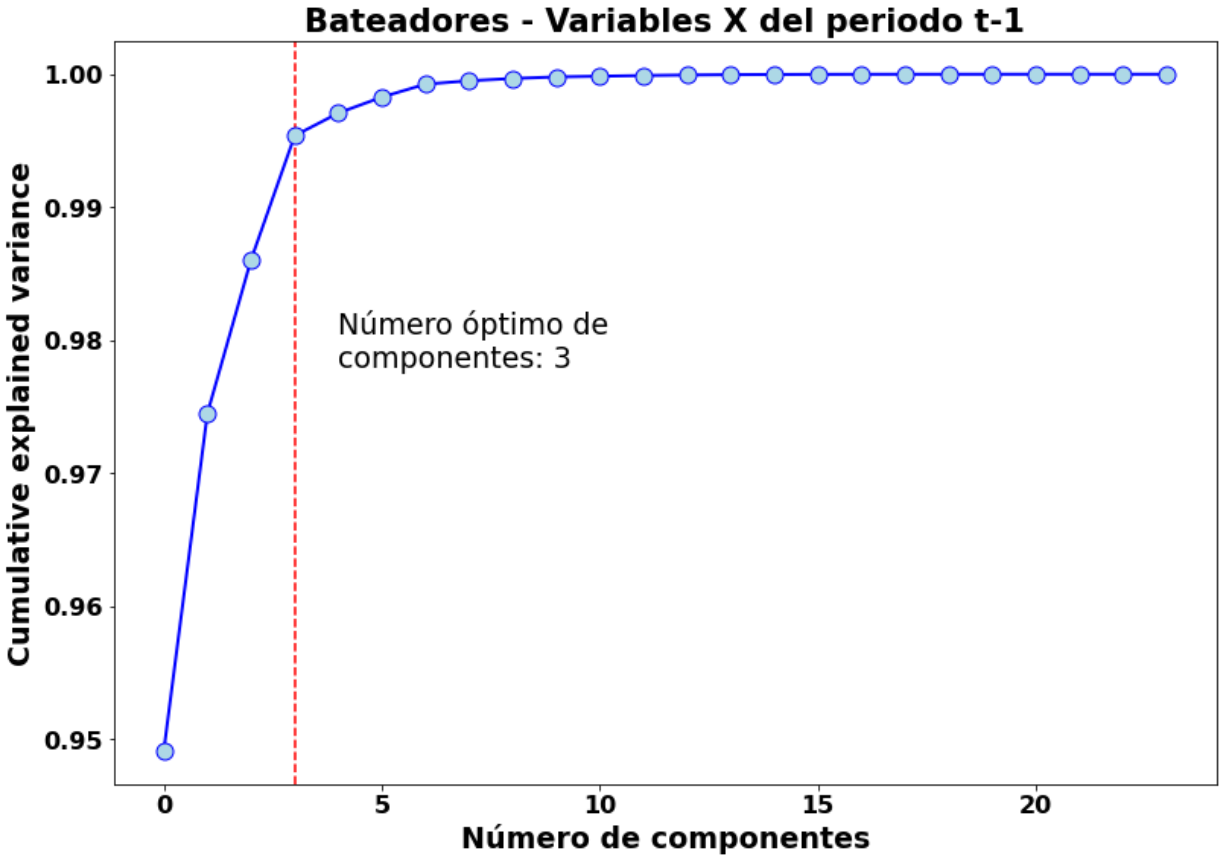
In [15]:

```
def de_correlate_df(df):
    X_aux = df.copy()
    for col in df.columns:
        X_aux[col] = df[col].sample(len(df)).values

    return X_aux
```

Luego, apliquemos PCA y hallemos el óptimo de componentes mediante la gráfica de la varianza que explican los componentes

In [16]:

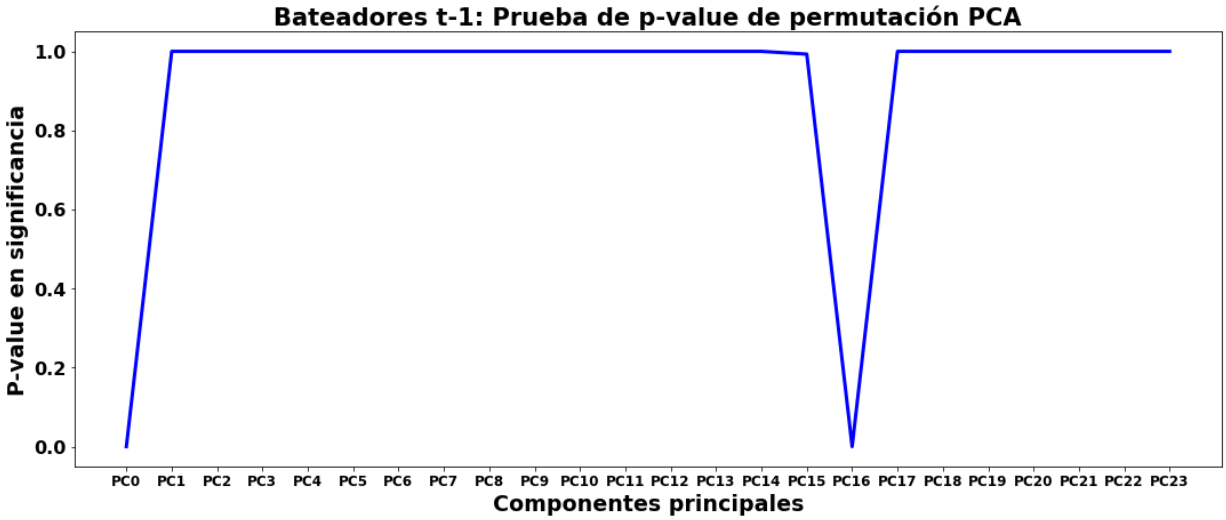
```
# Initialize PCA object
pca_hitter_t_1 = PCA()

# Fit PCA to data
pca_hitter_t_1.fit(X_hitter_t_1)

# Plot cumulative explained variance as a function of number of components
plt.subplots(figsize = (13,9))
cumulative_variance = np.cumsum(pca_hitter_t_1.explained_variance_ratio_)
# Find number of components that capture 80% of variance
optimal_n_components = np.argmax(cumulative_variance >= 0.98) + 1

plt.axvline(x = optimal_n_components,
            color='r',
            linestyle = 'dashed')
plt.plot(cumulative_variance,
         ls = '-' ,
         markerfacecolor = 'lightblue',
         marker = 'o',
         ms = 11,
         color = 'blue',
         linewidth = 2)
plt.xlabel('Número de componentes',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.ylabel('Cumulative explained variance',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.yticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.xticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.text(optimal_n_components + 1, 0.978,
         f'Número óptimo de\ncomponentes: {optimal_n_components}',
         fontsize = 19)
plt.title('Bateadores - Variables X del periodo t-1',
           fontsize = 21,
           fontweight = 'bold',
           color = 'black')
plt.show()
plt.savefig(path + "/Visualizations/Analysis/PCA/codo_pca_hitter_t_1.pdf",
            format = "pdf")
```




```
<Figure size 432x288 with 0 Axes>
```

In [32]:

```
pca = PCA()
pca.fit(X_hitter_t_1)
original_variance = pca.explained_variance_ratio_

N_permutations = 1000
variance = np.zeros((N_permutations, len(X_hitter_t_1.columns)))

for i in range(N_permutations):
    X_aux = de_correlate_df(X_hitter_t_1)

    pca.fit(X_aux)
    variance[i, :] = pca.explained_variance_ratio_

p_val = np.sum(variance > original_variance, axis=0) / N_permutations

plt.figure(figsize=(18, 7))
plt.plot([f'PC{i}' for i in range(len(X_hitter_t_1.columns))],
         p_val, label='p-value on significance',
         color = 'blue',
         linewidth = 3)
plt.xlabel('Componentes principales',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.ylabel('P-value en significancia',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.yticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.xticks(fontsize = 12,
           fontweight = 'bold',
           color = 'black')
plt.title("Bateadores t-1: Prueba de p-value de permutación PCA",
           fontsize = 21,
           fontweight = 'bold',
           color = 'black')
plt.show()
plt.savefig(path + "/Visualizations/Analysis/PCA/test_pca_hitter_t_1.pdf",
            format = "pdf")
```




```
<Figure size 432x288 with 0 Axes>
```

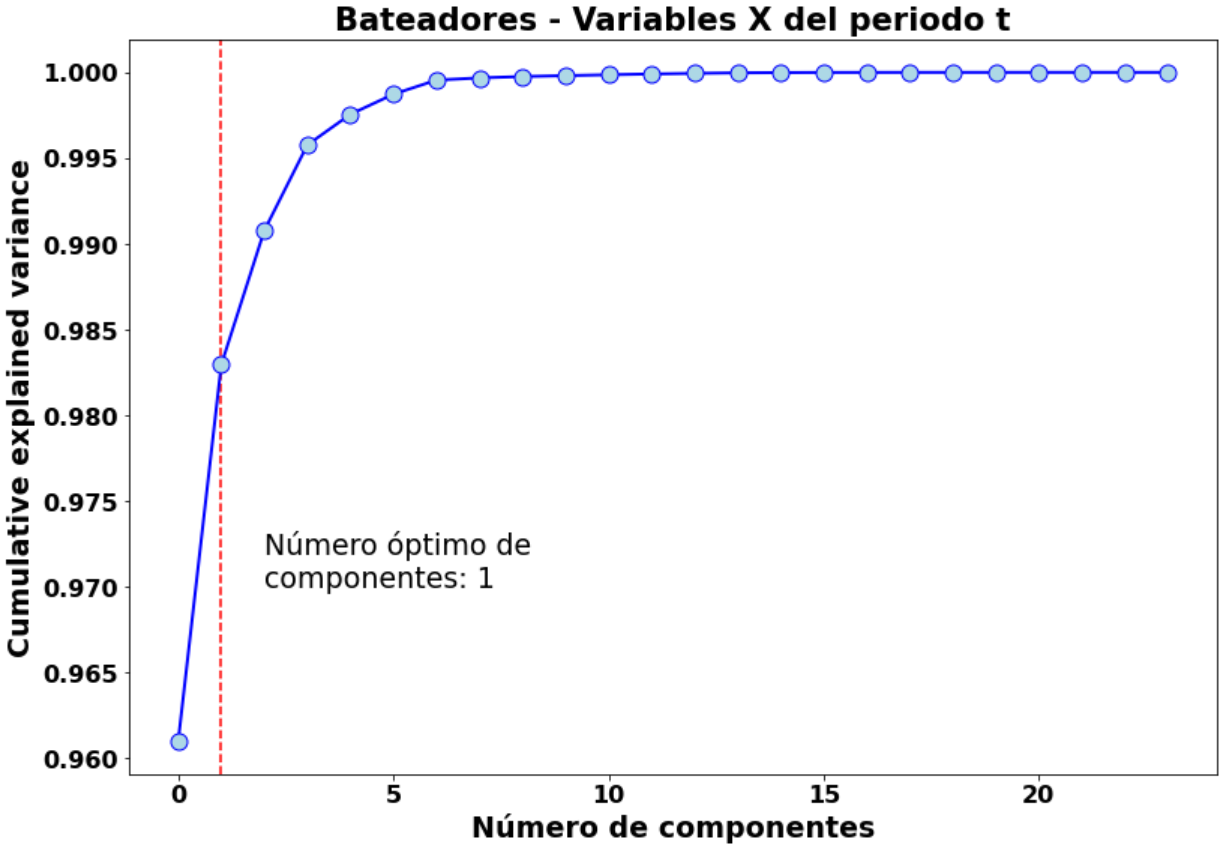
Vemos que el número correcto de componentes es 1

In [23]:

```
# instantiate a PCA object with n_components=2
pca_hitter_t_1 = PCA(n_components = 1)

# fit and transform the input data
pca_components = pca_hitter_t_1.fit_transform(X_hitter_t_1)

# create a DataFrame with the PCA components
pca_df = pd.DataFrame(data = pca_components,
                      columns = ['pca1_t_1'])

# concatenate the PCA DataFrame with the original DataFrame
hitter_dynamic_panel = pd.concat([hitter_dynamic_panel, pca_df], axis = 1)
```

In [24]:

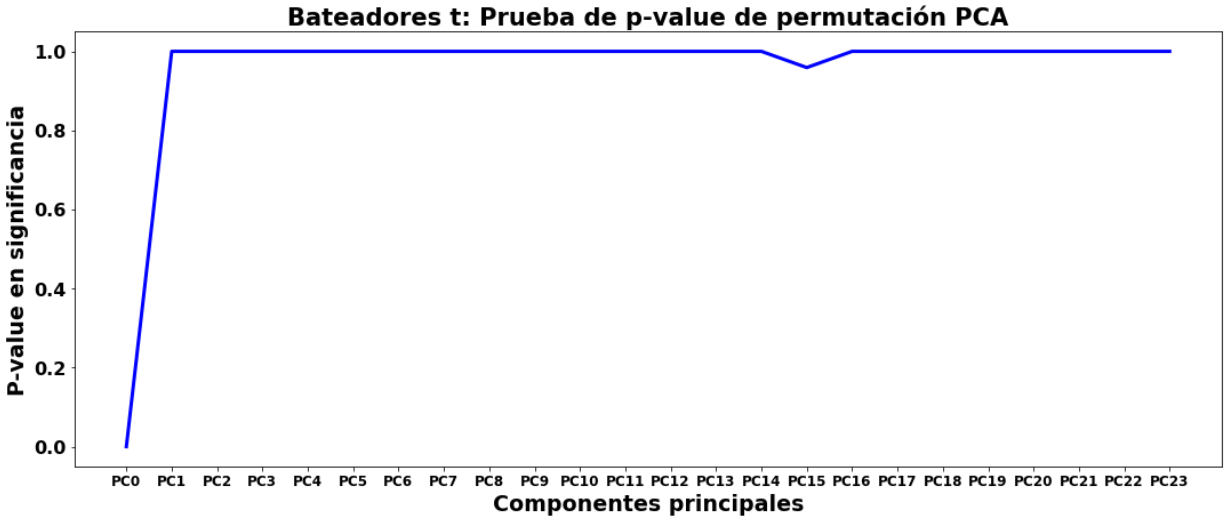
```
# Initialize PCA object
pca_hitter_t = PCA()

# Fit PCA to data
pca_hitter_t.fit(X_hitter_t)

# Plot cumulative explained variance as a function of number of components
plt.subplots(figsize = (13,9))
cumulative_variance = np.cumsum(pca_hitter_t.explained_variance_ratio_)
# Find number of components that capture 80% of variance
optimal_n_components = np.argmax(cumulative_variance >= 0.8) + 1

plt.axvline(x=optimal_n_components,
            color='r',
            linestyle = 'dashed')
plt.plot(cumulative_variance,
         ls = '-' ,
         markerfacecolor = 'lightblue',
         marker = 'o',
         ms = 11,
         color = 'blue',
         linewidth = 2)
plt.xlabel('Número de componentes',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.ylabel('Cumulative explained variance',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.yticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.xticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.text(optimal_n_components + 1, 0.97,
        f'Número óptimo de\ncomponentes: {optimal_n_components}',
        fontsize = 19)
plt.title('Bateadores - Variables X del periodo t',
           fontsize = 21,
           fontweight = 'bold',
           color = 'black')
plt.show()
plt.savefig(path + "/Visualizations/Analysis/PCA/codo_pca_hitter_t.pdf",
            format = "pdf")
```




```
<Figure size 432x288 with 0 Axes>
```

In [33]:

```
pca = PCA()
pca.fit(X_hitter_t)
original_variance = pca.explained_variance_ratio_

N_permutations = 1000
variance = np.zeros((N_permutations, len(X_hitter_t.columns)))

for i in range(N_permutations):
    X_aux = de_correlate_df(X_hitter_t)

    pca.fit(X_aux)
    variance[i, :] = pca.explained_variance_ratio_

p_val = np.sum(variance > original_variance, axis = 0) / N_permutations

plt.figure(figsize=(18, 7))
plt.plot([f'PC{i}' for i in range(len(X_hitter_t.columns))],
         p_val, label='p-value on significance',
         color = 'blue',
         linewidth = 3)
plt.xlabel('Componentes principales',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.ylabel('P-value en significancia',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.yticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.xticks(fontsize = 12,
           fontweight = 'bold',
           color = 'black')
plt.title("Bateadores t: Prueba de p-value de permutación PCA",
           fontsize = 21,
           fontweight = 'bold',
           color = 'black')
plt.show()
plt.savefig(path + "/Visualizations/Analysis/PCA/test_pca_hitter_t.pdf",
            format = "pdf")
```



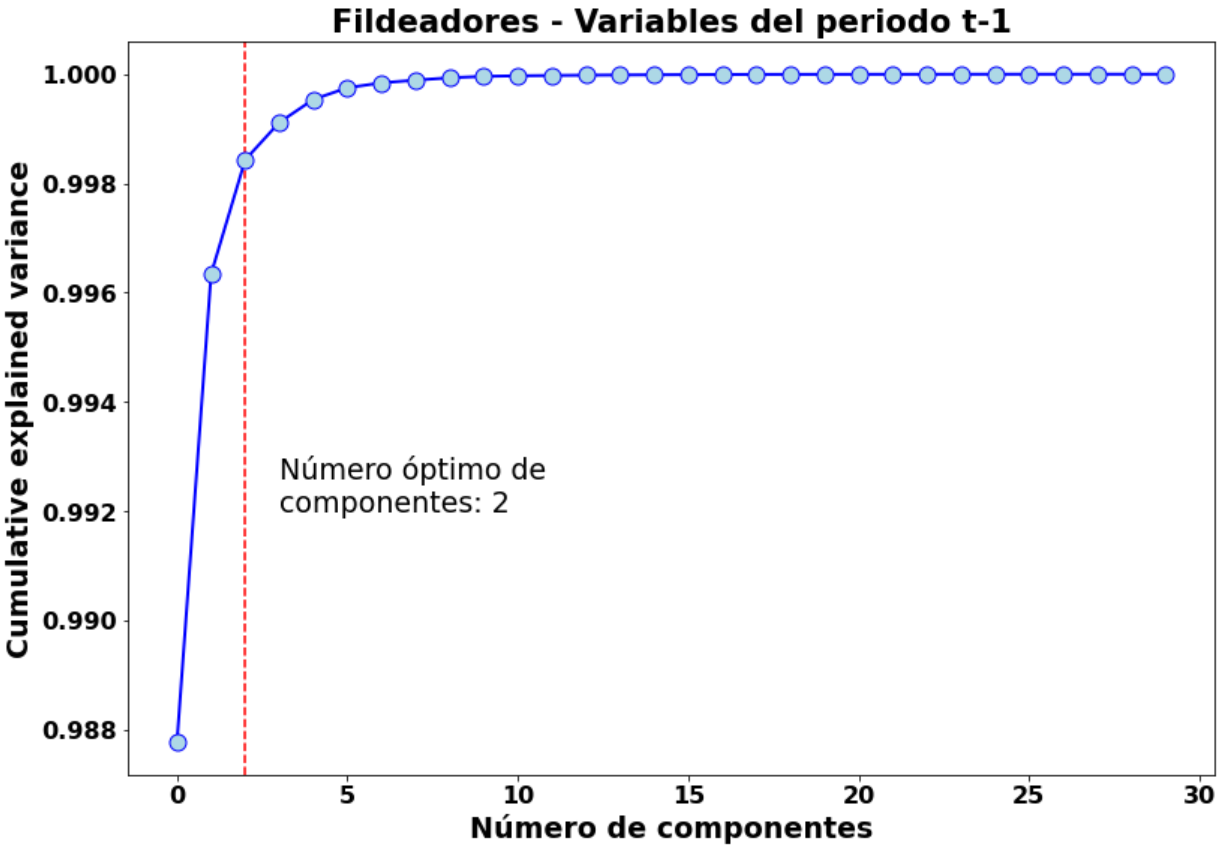

```
<Figure size 432x288 with 0 Axes>
```

In [26]:

```
# instantiate a PCA object with n_components=2
pca_hitter_t = PCA(n_components = 1)

# fit and transform the input data
pca_components = pca_hitter_t.fit_transform(X_hitter_t)

# create a DataFrame with the PCA components
pca_df = pd.DataFrame(data = pca_components,
                      columns = ['pca1_t'])

# concatenate the PCA DataFrame with the original DataFrame
hitter_dynamic_panel = pd.concat([hitter_dynamic_panel, pca_df], axis = 1)
```

In [27]:

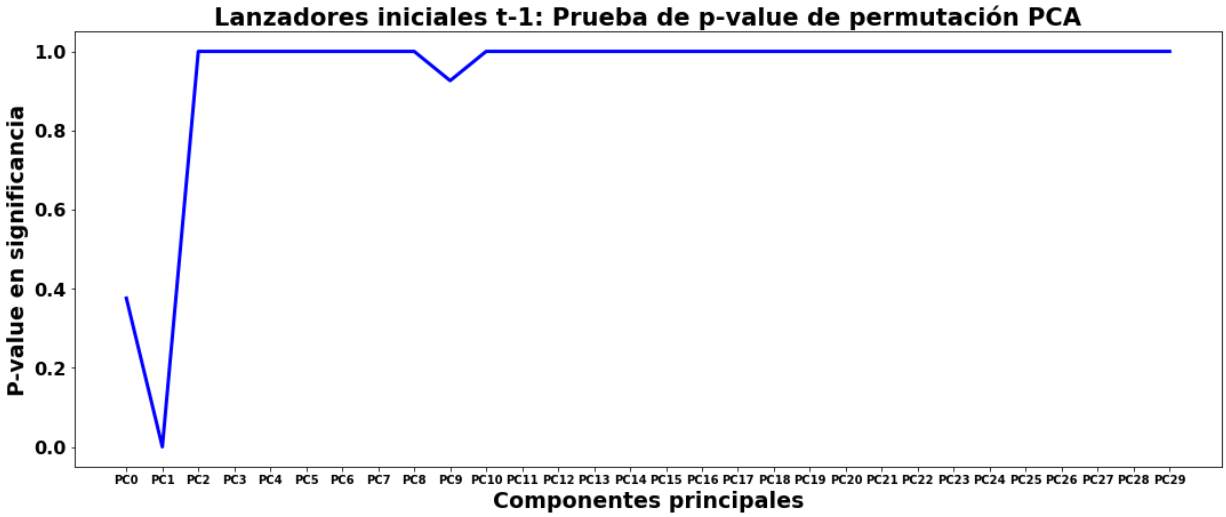
```
# Initialize PCA object
pca_fielder_t_1 = PCA()

# Fit PCA to data
pca_fielder_t_1.fit(X_fielder_t_1)

# Plot cumulative explained variance as a function of number of components
plt.subplots(figsize = (13,9))
cumulative_variance = np.cumsum(pca_fielder_t_1.explained_variance_ratio_)
# Find number of components that capture 80% of variance
optimal_n_components = np.argmax(cumulative_variance >= 0.99) + 1

plt.axvline(x = optimal_n_components,
            color='r',
            linestyle = 'dashed')
plt.plot(cumulative_variance,
         ls = '-' ,
         markerfacecolor = 'lightblue',
         marker = 'o',
         ms = 11,
         color = 'blue',
         linewidth = 2)
plt.xlabel('Número de componentes',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.ylabel('Cumulative explained variance',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.yticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.xticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.text(optimal_n_components + 1, 0.992,
         f'Número óptimo de\ncomponentes: {optimal_n_components}',
         fontsize = 19)
plt.title('Fildeadores - Variables del periodo t-1',
           fontsize = 21,
           fontweight = 'bold',
           color = 'black')
plt.show()
plt.savefig(path + "/Visualizations/Analysis/PCA/codo_pca_fielder_t_1.pdf",
            format = "pdf")
```




```
<Figure size 432x288 with 0 Axes>
```

In [34]:

```
pca = PCA()
pca.fit(X_fielder_t_1)
original_variance = pca.explained_variance_ratio_

N_permutations = 1000
variance = np.zeros((N_permutations, len(X_fielder_t_1.columns)))

for i in range(N_permutations):
    X_aux = de_correlate_df(X_fielder_t_1)

    pca.fit(X_aux)
    variance[i, :] = pca.explained_variance_ratio_

p_val = np.sum(variance > original_variance, axis=0) / N_permutations

plt.figure(figsize=(18, 7))
plt.plot([f'PC{i}' for i in range(len(X_fielder_t_1.columns))],
         p_val, label='p-value on significance',
         color = 'blue',
         linewidth = 3)
plt.xlabel('Componentes principales',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.ylabel('P-value en significancia',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.yticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.xticks(fontsize = 10,
           fontweight = 'bold',
           color = 'black')
plt.title("Lanzadores iniciales t-1: Prueba de p-value de permutación PCA",
           fontsize = 21,
           fontweight = 'bold',
           color = 'black')
plt.show()
plt.savefig(path + "/Visualizations/Analysis/PCA/test_pca_fielder_t_1.pdf",
            format = "pdf")
```



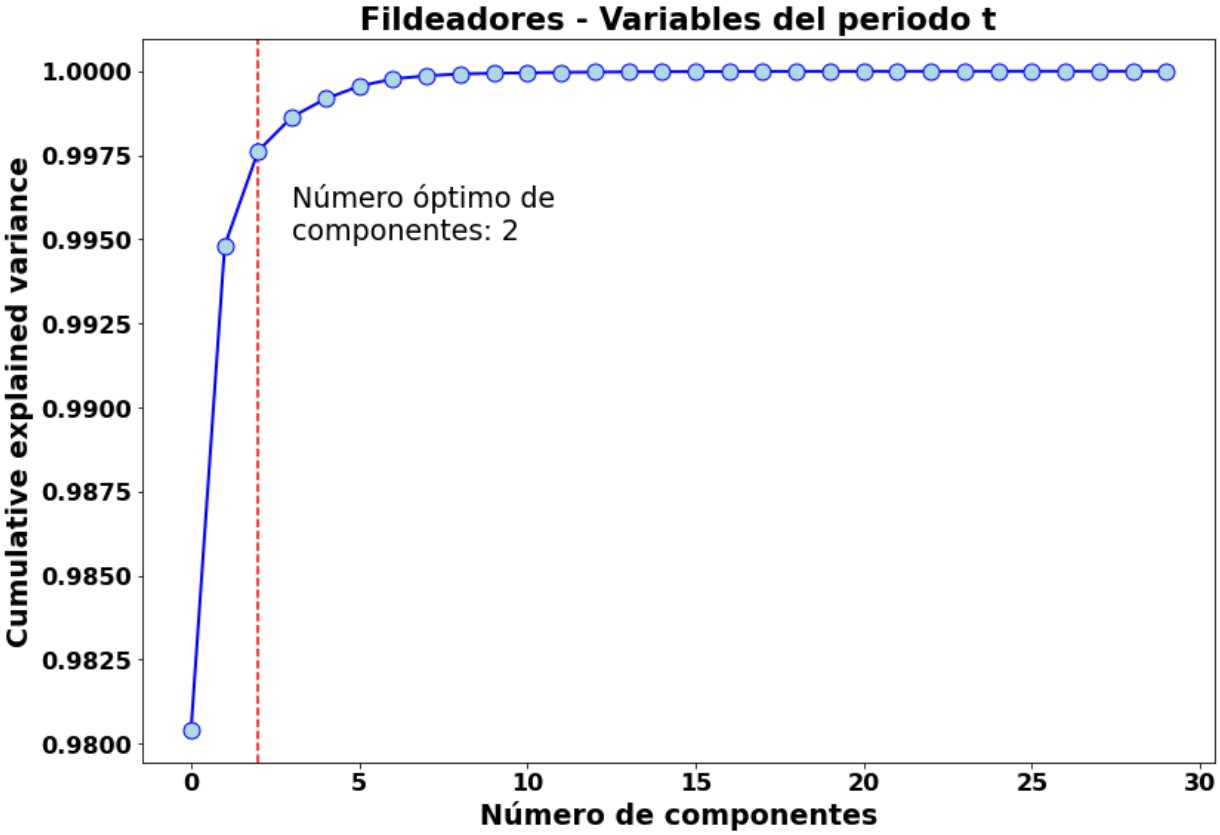

```
<Figure size 432x288 with 0 Axes>
```

In [35]:

```
# instantiate a PCA object with n_components=2
pca_fielder_t_1 = PCA(n_components = 2)

# fit and transform the input data
pca_components = pca_fielder_t_1.fit_transform(X_fielder_t_1)

# create a DataFrame with the PCA components
pca_df = pd.DataFrame(data = pca_components,
                      columns = ['pca1_t_1','pca2_t_1'])

# concatenate the PCA DataFrame with the original DataFrame
fielder_dynamic_panel = pd.concat([fielder_dynamic_panel, pca_df], axis = 1)
```

In [36]:

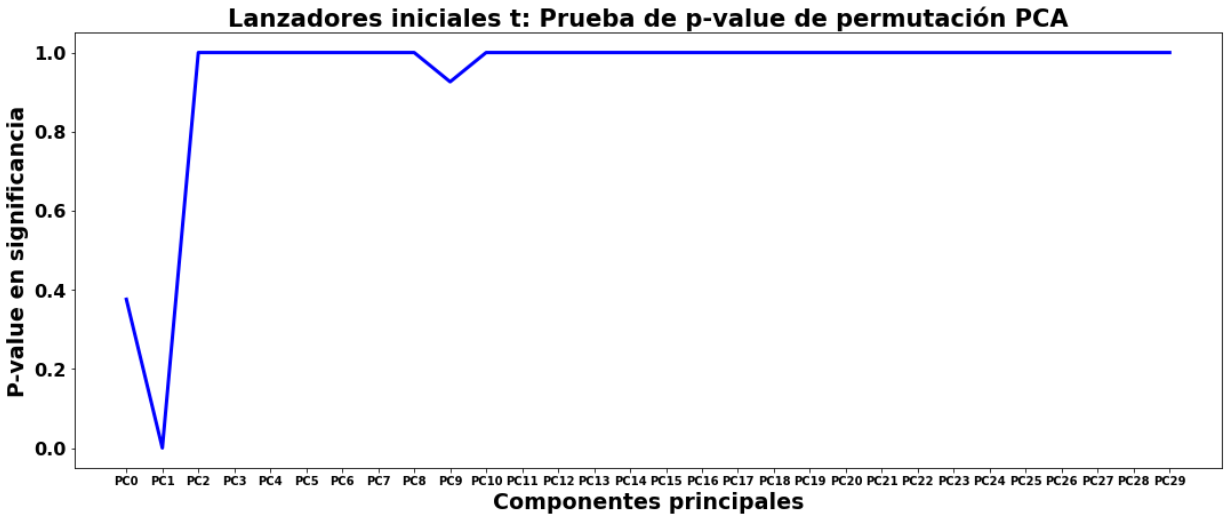
```
# Initialize PCA object
pca_fielder_t = PCA()

# Fit PCA to data
pca_fielder_t.fit(X_fielder_t)

# Plot cumulative explained variance as a function of number of components
plt.subplots(figsize = (13,9))
cumulative_variance = np.cumsum(pca_fielder_t.explained_variance_ratio_)
# Find number of components that capture 80% of variance
optimal_n_components = np.argmax(cumulative_variance >= 0.994) + 1


plt.axvline(x = optimal_n_components,
            color='r',
            linestyle = 'dashed')
plt.plot(cumulative_variance,
         ls = '-' ,
         markerfacecolor = 'lightblue',
         marker = 'o',
         ms = 11,
         color = 'blue',
         linewidth = 2)
plt.xlabel('Número de componentes',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.ylabel('Cumulative explained variance',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.yticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.xticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.text(optimal_n_components + 1, 0.995,
         f'Número óptimo de\ncomponentes: {optimal_n_components}',
         fontsize = 19)
plt.title('Fildeadores - Variables del periodo t',
           fontsize = 21,
           fontweight = 'bold',
           color = 'black')
plt.show()
plt.savefig(path + "/Visualizations/Analysis/PCA/codo_pca_fielder_t.pdf",
            format = "pdf")
```




```
<Figure size 432x288 with 0 Axes>
```

In [37]:

```
pca = PCA()
pca.fit(X_fielder_t)
original_variance = pca.explained_variance_ratio_

N_permutations = 1000
variance = np.zeros((N_permutations, len(X_fielder_t.columns)))

for i in range(N_permutations):
    X_aux = de_correlate_df(X_fielder_t)

    pca.fit(X_aux)
    variance[i, :] = pca.explained_variance_ratio_

plt.figure(figsize=(18, 7))
plt.plot([f'PC{i}' for i in range(len(X_fielder_t.columns))],
         p_val, label='p-value on significance',
         color = 'blue',
         linewidth = 3)
plt.xlabel('Componentes principales',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.ylabel('P-value en significancia',
           fontsize = 19,
           fontweight = 'bold',
           color = 'black')
plt.yticks(fontsize = 16,
           fontweight = 'bold',
           color = 'black')
plt.xticks(fontsize = 10,
           fontweight = 'bold',
           color = 'black')
plt.title("Lanzadores iniciales t: Prueba de p-value de permutación PCA",
           fontsize = 21,
           fontweight = 'bold',
           color = 'black')
plt.show()
plt.savefig(path + "/Visualizations/Analysis/PCA/test_pca_fielder_t.pdf",
            format = "pdf")
```



<Figure size 432x288 with 0 Axes>

In [38]:
```
# instantiate a PCA object with n_components=2
pca_fielder_t = PCA(n_components = 2)

# fit and transform the input data
pca_components = pca_fielder_t.fit_transform(X_fielder_t)

# create a DataFrame with the PCA components
pca_df = pd.DataFrame(data = pca_components,
                      columns = ['pca1_t', 'pca2_t'])

# concatenate the PCA DataFrame with the original DataFrame
fielder_dynamic_panel = pd.concat([fielder_dynamic_panel, pca_df], axis = 1)
```

Exportemos las bases de datos

In [39]:
```
path = 'ETL_Data/Panel/General/Dynamic_model/'
hitter_dynamic_panel.to_csv(path + 'dynamic_model_hitter_pca' + '.csv',
                            index = False)
fielder_dynamic_panel.to_csv(path + 'dynamic_model_fielder_pca' + '.csv',
                             index = False)
```