CÓDIGOS PYTHON

Anexo I. Preparación de datos: normalización y retardos

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
!pip install statsmodels
!pip install openpyxl
from functools import reduce
```

```
#Meto la informacion de la excel en un dataframe
df_merged2 = pd.read_excel('df_panel.xlsx')
#Quitamos índice y primera columna para empezar con fecha
df_merged2 = df_merged2.drop(df_merged2.columns[0], axis=1)
df_merged2.tail(10)
```

```
#Acotamos horizonte temporal datos
# Asegurarse de que 'fecha' esté en formato datetime
df_merged2['fecha'] = pd.to_datetime(df_merged2['fecha'], errors='coerce')

# Definir fechas límite
fecha_inicio = pd.to_datetime("2019-12-31")
fecha_fin = pd.to_datetime("2024-12-31")

# Filtrar DataFrame
df_merged2 = df_merged2[(df_merged2['fecha'] >= fecha_inicio) & (df_merged2['fecha'] <=
fecha_fin)]

# Verificar resultado
print(df_merged2['fecha'].min(), df_merged2['fecha'].max())
print(f"Número de observaciones: {len(df_merged2)}")
```

```
# Definimos una función para laggear y normalizar variables

def create_lag_and_pit_z(
    df,
    variables_indep,
    variable_dep,
    lag_periods=12,
    date_col='fecha',
    entity_col='ticker',
    variables_dummies=None,
    return_model_df=False
):
```

```
    """
    Calcula:
    - Lag + z-score PIT para variables independientes numéricas
    - Lag (sin z-score) para variables dummy binarias
    - z-score PIT para variable dependiente (sin lag)

    return_model_df: si True, devuelve solo columnas necesarias para regresión (X e y)
    """
    df_out = df.copy()
    model_cols = ['fecha', 'ticker']

    # 1. Lags y z-score de variables numéricas
    for var in variables_indep:
        lag_col = f"{var}_lag{lag_periods}"
        z_col = f"{lag_col}_z"
        df_out[lag_col] = df_out.groupby(entity_col)[var].shift(lag_periods)
        df_out[z_col] = df_out.groupby(date_col)[lag_col].transform(
            lambda x: (x - x.mean()) / x.std(ddof=0) if x.std(ddof=0) else np.nan
        )
        model_cols.append(z_col)

    # 2. Lag de dummies (sin z-score)
    if variables_dummies:
        for dvar in variables_dummies:
            lag_col = f"{dvar}_lag{lag_periods}"
            df_out[lag_col] = df_out.groupby(entity_col)[dvar].shift(lag_periods)
            model_cols.append(lag_col)

    # 3. z-score de la variable dependiente
    target_z = f"{variable_dep}_z"
    df_out[target_z] = df_out.groupby(date_col)[variable_dep].transform(
        lambda x: (x - x.mean()) / x.std(ddof=0) if x.std(ddof=0) else np.nan
    )
    model_cols.append(target_z)

    # 4. Devolver solo modelo limpio si se desea
    if return_model_df:
        return df_out[model_cols].dropna()
    else:
        return df_out
```

```
vars_indep = ['ND/Ebitda', 'PTBV', 'beta', 'ASSETS_pct_change_5yr', 'ROE_avg_last_60',
'ESG']
#variables dummies todas las del dataframe
vars_dummies = ['dummy_big_cap']+ [col for col in df_merged2.columns if
col.startswith('sector_')]
dep_var = 'ROE'
```

```
# Devuelve directamente un df listo para regresión con X e y
df_panel_z = create_lag_and_pit_z(
    df=df_merged2,
    variables_indep=vars_indep,
    variable_dep=dep_var,
    lag_periods=12,
    variables_dummies=vars_dummies,
    return_model_df=True
)

df_panel_z
```

Anexo II: Modificación del ROE para que no sea estático

```
#corregimos estaticidad de variable dependiente (ROE)
df_changes =
df_panel_z.sort_values(['ticker','fecha']).groupby(['ticker','ROE']).first().reset_index()
df_changes
```

```
print(len(df_merged2),len(df_changes))
```
17812 1131

Anexo III: Regresión OLS con valores normalizados

```
# Variables numéricas lag12 z-score
vars_z = [col for col in df_changes.columns if col.endswith('_lag12_z')]

# Dummies (binary), no z-score
vars_dummies = [col for col in df_changes.columns if '_lag12' in col and not col.endswith('_z')]

# Variables independientes totales
vars_indep_finales = vars_z + vars_dummies
```

```
# Forzar conversión a float en todas las columnas de X e y
X = df_changes[vars_indep_finales].apply(pd.to_numeric, errors='coerce').astype(float)
y = pd.to_numeric(df_changes['ROE_z'], errors='coerce').astype(float)

# Eliminar filas con NaNs
Xy = pd.concat([X, y], axis=1).dropna()
X = Xy[X.columns]
y = Xy['ROE_z']

# Añadir constante
X = sm.add_constant(X)

# Ajustar modelo
model = sm.OLS(y, X).fit()
```

```
print(model.summary())
```

Anexo IV: Regresión OLS corregida por variables no significativas

```
Corregir por variables no significativas, entre ellas ESG.

# 1. Extraer los p-valores del modelo completo
pvalues = model.pvalues

# 2. Seleccionar variables con p < 0.1
significant_vars = pvalues[pvalues < 0.1].index.tolist()

# 3. Mantener la constante si está incluida
if 'const' in pvalues.index and 'const' not in significant_vars:
    significant_vars = ['const'] + significant_vars

# 4. Crear el subconjunto reducido de X (variables explicativas)
X_reduced = X[significant_vars]

# 5. Ajustar el nuevo modelo con solo variables significativas
model_reduced = sm.OLS(y, X_reduced).fit()

# 6. Ver resumen del nuevo modelo
print(model_reduced.summary())
```

Anexo V: Regresión de residuales

```
#Etapa 1: regresión OLS sin ASG

# 1. Reconstruir la lista original de variables
vars_z = [col for col in df_changes.columns if col.endswith('_lag12_z')]
vars_dummies = [col for col in df_changes.columns if '_lag12' in col and not col.endswith('_z')]
vars_indep_finales = vars_z + vars_dummies

# 2. Eliminar ESG_lag12_z si está
vars_indep_finales = [col for col in vars_indep_finales if col != 'ESG_lag12_z']

# 3. Construir X e y
X = df_changes[vars_indep_finales].apply(pd.to_numeric, errors='coerce').astype(float)
y = pd.to_numeric(df_changes['ROE_z'], errors='coerce').astype(float)

# 4. Eliminar filas con NaNs
Xy = pd.concat([X, y], axis=1).dropna()
X = Xy[X.columns]
y = Xy['ROE_z']

# 5. Añadir constante
X = sm.add_constant(X)
```

```
# 6. Ajustar el modelo OLS
model_no_esg = sm.OLS(y, X).fit()

# 7. Mostrar resumen
print(model_no_esg.summary())
```

```
#Etapa 2: Mismo paso que en Etapa 1, pero aplicando significancia p value <0.1

# 2. Definir variables independientes
vars_z = [col for col in df_changes.columns if col.endswith('_lag12_z')]
vars_dummies = [col for col in df_changes.columns if '_lag12' in col and not col.endswith('_z')]
vars_indep_finales = vars_z + vars_dummies

# 3. Eliminar ESG_lag12_z
vars_indep_finales = [col for col in vars_indep_finales if col != 'ESG_lag12_z']

# 4. Preparar X e y
X = df_changes[vars_indep_finales].apply(pd.to_numeric, errors='coerce').astype(float)
y = pd.to_numeric(df_changes['ROE_z'], errors='coerce').astype(float)

# 5. Eliminar filas con NaNs
Xy = pd.concat([X, y], axis=1).dropna()
X = Xy[X.columns]
y = Xy['ROE_z']

# 6. Ajustar modelo completo sin ESG
X_full = sm.add_constant(X)
model_full = sm.OLS(y, X_full).fit()

# 7. Filtrar por p-value < 0.1
pvalues = model_full.pvalues
significant_vars = pvalues[pvalues < 0.1].index.tolist()

# Incluir constante si fue excluida
if 'const' in pvalues.index and 'const' not in significant_vars:
    significant_vars = ['const'] + significant_vars

# 8. Ajustar modelo reducido
X_reduced = X_full[significant_vars]
model_reduced = sm.OLS(y, X_reduced).fit()

# 9. Mostrar resumen
print(model_reduced.summary())
```

```
#AHORA HACEMOS LA REGRESIÓN DE LOS RESIDUALES DEL MODELO
REDUCIDO (P VALU <0.1) CON ESG_lag12_z
# 1. Obtener los residuales del modelo reducido
residuals = model_reduced.resid
```

```
# 2. Asegurar que ESG está disponible
esg = pd.to_numeric(df_changes['ESG_lag12_z'], errors='coerce')

# 3. Alinear índice y eliminar NaNs de ambos
resid_esg = pd.concat([residuals, esg], axis=1).dropna()
resid_esg.columns = ['residuals', 'ESG_lag12_z']

# 4. Regresión: residuales ~ ESG
X_esg = sm.add_constant(resid_esg['ESG_lag12_z'])
y_esg = resid_esg['residuals']
model_esg = sm.OLS(y_esg, X_esg).fit()

# 5. Mostrar resultados
print(model_esg.summary())
```

Anexo VI: Regresión por sectores

```
#ojo! hay que recuperar sector industrial que lo hemos perdido al hacer dummies

# 1. Reconstruir la columna 'sector' desde las dummies
sector_dummy_cols = [col for col in df_changes.columns if col.startswith('sector_') and '_lag12' in col]
df_changes['sector'] = df_changes[sector_dummy_cols].idxmax(axis=1)
df_changes['sector'] = df_changes['sector'].str.replace('sector_', '', regex=False).str.replace('_lag12', '', regex=False)

# 2. Recuperar sector base omitido (ej. Industrials) donde todas las dummies valen 0
cond = df_changes[sector_dummy_cols].sum(axis=1) == 0
df_changes.loc[cond, 'sector'] = 'Industrials'

# 3. Definir variables independientes del modelo reducido
X_cols_s = [
    'ND/Ebitda_lag12_z',
    'PTBV_lag12_z',
    'beta_lag12_z',
    'ASSETS_pct_change_5yr_lag12_z',
    'ROE_avg_last_60_lag12_z',
    'dummy_big_cap_lag12'
]

# 4. Filtrar dataset usable
df_sector = df_changes.dropna(subset=['ROE_z'] + X_cols_s + ['sector', 'ESG_lag12_z']).copy()
for col in X_cols_s:
    df_sector[col] = df_sector[col].astype(float)

# 5. Funciones auxiliares
def find_sig_variables(data_df, x_cols, y_col, thres=0.1):
    X1 = sm.add_constant(data_df[x_cols])
```

```
    y1 = data_df[y_col]
    model1 = sm.OLS(y1, X1).fit()
    sig_cols = model1.pvalues[model1.pvalues < thres].index.tolist()
    if "const" in sig_cols:
        sig_cols.remove('const')
    return sig_cols, model1.resid, model1.rsquared, model1.f_pvalue

def make_residuals(data_df, x_cols, y_col='ROE_z', second_col='ESG_lag12_z', thres=0.1):
    sig_vars, y2, r1, s1 = find_sig_variables(data_df, x_cols, y_col, thres)
    x2 = sm.add_constant(data_df[second_col])
    model2 = sm.OLS(y2, x2).fit()
    return model2.rsquared, model2.params.loc[second_col], model2.pvalues.loc[second_col],
model2.f_pvalue, r1, s1

# 6. Ejecutar análisis por sector
sector_res = pd.DataFrame()
for s in df_sector['sector'].unique():
    df_filt = df_sector[df_sector['sector'] == s].dropna(subset=['ESG_lag12_z', 'ROE_z'] +
X_cols_s)
    if df_filt.empty or len(df_filt) < 30:
        continue
    res_s = pd.Series(
        make_residuals(df_filt, X_cols_s, y_col='ROE_z', second_col='ESG_lag12_z'),
        index=[
            'rsquared',
            'param_ESG_lag12_z',
            'pvalue_ESG_lag12_z',
            'f_pvalue',
            'first_stage_r',
            'first_stage_f_pvalue'
        ],
        name=s
    )
    sector_res = pd.concat([sector_res, res_s], axis=1)
```

Anexo VII: Impacto ASG por sector

```
#Gráfico: coeficiente ESG y p-valor por sector
fig, ax1 = plt.subplots(figsize=(10, 5))
sector_res.loc['param_ESG_lag12_z'].plot(kind='bar', ax=ax1, color='skyblue')
ax1.set_ylabel('Coef. ESG_lag12_z')
ax1.axhline(0, color='gray', linestyle='--')
ax2 = ax1.twinx()
sector_res.loc['pvalue_ESG_lag12_z'].plot(ax=ax2, color='red', marker='o', linestyle='None')
ax2.set_ylabel('P-valor')
plt.title('Impacto de ESG_lag12_z por sector (modelo sobre residuales)')
plt.show()
```

Anexo VIII: Regresión datos de panel con efectos fijos

```
!pip install linearmodels

from linearmodels.panel import PanelOLS

# Cargar datos
df = pd.read_excel('df_changes.xlsx', sheet_name='Sheet1')

# Extraer año
df['año'] = pd.to_datetime(df['fecha'], errors='coerce').dt.year

# Reconstruir columna 'sector' correctamente
sector_cols = [col for col in df.columns if 'sector_' in col and '_lag12' in col]
df['n_dummies_activas'] = df[sector_cols].sum(axis=1)
df['sector'] = df[sector_cols].idxmax(axis=1).str.replace('sector_', '').str.replace('_lag12', '')
df.loc[df['n_dummies_activas'] == 0, 'sector'] = 'Industrials'

# Variables independientes
vars_indep = [
    'ND/Ebitda_lag12_z', 'PTBV_lag12_z', 'beta_lag12_z',
    'ASSETS_pct_change_5yr_lag12_z', 'ROE_avg_last_60_lag12_z',
    'ESG_lag12_z', 'dummy_big_cap_lag12'
]

# Panel por sector
df_sector = df[['sector', 'año', 'ROE_z'] + vars_indep].dropna()
df_sector = df_sector.set_index(['sector', 'año'])

# Modelo
y = df_sector['ROE_z']
X = sm.add_constant(df_sector.drop(columns='ROE_z'))
model = PanelOLS(y, X, entity_effects=True)
results = model.fit()

print(results.summary)
```

```
#regresión panel efectos fijos dos pasos para residuales
!pip install linearmodels

import pandas as pd
import statsmodels.api as sm
from linearmodels.panel import PanelOLS

# Cargar datos
df = pd.read_excel('df_changes.xlsx', sheet_name='Sheet1')
df['año'] = pd.to_datetime(df['fecha'], errors='coerce').dt.year
```

```
# Reconstruir columna 'sector'
sector_cols = [col for col in df.columns if 'sector_' in col and '_lag12' in col]
df['n_dummies_activas'] = df[sector_cols].sum(axis=1)
df['sector'] = df[sector_cols].idxmax(axis=1).str.replace('sector_', '').str.replace('_lag12', '')
df.loc[df['n_dummies_activas'] == 0, 'sector'] = 'Industrials'


# Variables sin ESG
vars_sin_esg = [
    'ND/Ebitda_lag12_z', 'PTBV_lag12_z', 'beta_lag12_z',
    'ASSETS_pct_change_5yr_lag12_z', 'ROE_avg_last_60_lag12_z',
    'dummy_big_cap_lag12'
]

# Panel
df_panel = df[['sector', 'año', 'ROE_z', 'ESG_lag12_z'] + vars_sin_esg].dropna()
df_panel = df_panel.set_index(['sector', 'año'])

# Regresión sin ESG
y = df_panel['ROE_z']
X = sm.add_constant(df_panel[vars_sin_esg])
model_sin_esg = PanelOLS(y, X, entity_effects=True)
results_sin_esg = model_sin_esg.fit()

print("=== Resultados del modelo sin ESG ===")
print(results_sin_esg.summary)

# Guardar residuos
df_panel['residual'] = results_sin_esg.resids

# Regresión de residuales sobre ESG
y_resid = df_panel['residual']
X_resid = sm.add_constant(df_panel['ESG_lag12_z'])
model_resid_esg = PanelOLS(y_resid, X_resid, entity_effects=True)
results_resid_esg = model_resid_esg.fit()

print("\n=== Regresión de residuales sobre ESG ===")
print(results_resid_esg.summary)
```

Anexo IX: Interacciones ASG y sector para regresión datos panel

```
# Crear interacciones entre ESG y cada sector
sector_cols = [col for col in df.columns if 'sector_' in col and '_lag12' in col]
for col in sector_cols:
    sector_name = col.replace('sector_', '').replace('_lag12', '')
    df[f'ESG_x_{sector_name}'] = df['ESG_lag12_z'] * df[col]

# Extraer año y reconstruir sector
df['año'] = pd.to_datetime(df['fecha'], errors='coerce').dt.year
```

```
df['n_dummies_activas'] = df[sector_cols].sum(axis=1)
df['sector'] = df[sector_cols].idxmax(axis=1).str.replace('sector_', '').str.replace('_lag12', '')
df.loc[df['n_dummies_activas'] == 0, 'sector'] = 'Industrials'

# Variables base + interacciones ESG x sector
vars_interact = [
    'ND/Ebitda_lag12_z', 'PTBV_lag12_z', 'beta_lag12_z',
    'ASSETS_pct_change_5yr_lag12_z', 'ROE_avg_last_60_lag12_z',
    'dummy_big_cap_lag12'
]

# Añadir todas las interacciones ESG x sector
interact_cols = [f'ESG_x_{col.replace("sector_", "").replace("_lag12", "")}' for col in
sector_cols]
all_vars = vars_interact + interact_cols

# Crear panel
df_panel = df[['sector', 'año', 'ROE_z'] + all_vars].dropna()
df_panel = df_panel.set_index(['sector', 'año'])

# Ajustar el modelo con efectos fijos por sector
y = df_panel['ROE_z']
X = sm.add_constant(df_panel[all_vars])
model = PanelOLS(y, X, entity_effects=True)
results = model.fit()

print("=== Resultados con interacciones ESG x sector ===")
print(results.summary)
```

```
# Añadir explícitamente ESG × Industrials
df['ESG_x_Industrials'] = df['ESG_lag12_z'] * (df['sector'] == 'Industrials').astype(int)

# Volver a construir X sin constante (intercepto)
interact_cols_full = ['ESG_x_Industrials'] + [f'ESG_x_{col.replace("sector_",
"").replace("_lag12", "")}' for col in sector_cols]
all_vars = vars_interact + interact_cols_full

# Crear panel
df_panel = df[['sector', 'año', 'ROE_z'] + all_vars].dropna()
df_panel = df_panel.set_index(['sector', 'año'])

# Reajustar el modelo sin constante
X_no_const = df_panel[all_vars]  # sin sm.add_constant
y = df_panel['ROE_z']

model_full = PanelOLS(y, X_no_const, entity_effects=True)
results_full = model_full.fit()
```

```
print(results_full.summary)
```

Anexo X: Coeficiente ASG por cada sector según datos panel

```
# Extraer y graficar coeficientes ESG
coef_esg = results.params[interact_cols]
stderr = results.std_errors[interact_cols]
ci95 = 1.96 * stderr


plt.figure(figsize=(12, 6))  # ancho un poco mayor
coef_esg.plot(kind='bar', yerr=ci95, capsize=4, color='skyblue', edgecolor='black')
plt.axhline(0, color='gray', linestyle='--')
plt.title('Efecto estimado del ESG en cada sector (incluido Industrials)', fontsize=14)
plt.ylabel('Coeficiente ESG sobre ROE_z', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)  # rotación y alineación a la derecha
plt.tight_layout()
plt.show()
```