Codigos Anexados

Osarel

Caracterización de los factores de riesgo del riesgo soberano de la renta fija dominicana

Serie histórica de nodos de referencia (benchmark) de la curva de rendimiento gubernamental.

Estos datos incluyen los nodos de referencia tanto del Banco Central de la República Dominicana (BCRD) como del Ministerio de Hacienda denominados en pesos dominicanos. El conjunto de datos (`benchmarks`) se organiza con un índice basado en fechas, donde cada columna representa un nodo de referencia para un tramo específico de la curva de rendimiento.

```
import pandas as pd

pd.set_option("display.max_colwidth", None)
benchmarks = pd.read_csv(r".\aggregate.csv")
benchmarks.set_index(keys="date", inplace=True)
benchmarks_melted = benchmarks.reset_index().melt(id_vars="date", var_name="Nodos de referencia", value_name="
benchmarks_melted
```

	date	Nodos de referencia	YTM
0	2013-11-01	1-YR-BCRD	0.089165
1	2013-12-01	1-YR-BCRD	0.097277
2	2014-01-01	1-YR-BCRD	0.099050
3	2014-02-01	1-YR-BCRD	0.097428
4	2014-03-01	1-YR-BCRD	0.102157
...	...	...	...
2683	2024-02-28	9-YR-MHDOP	0.096657
2684	2024-03-27	9-YR-MHDOP	0.097305
2685	2024-04-24	9-YR-MHDOP	0.098394
2686	2024-05-29	9-YR-MHDOP	0.101556
2687	2024-06-26	9-YR-MHDOP	0.101102

Matriz de correlación de nodos de referencia.

Para identificar relaciones significativas entre los diferentes nodos de referencia, primero calculamos las primeras diferencias de las series temporales (benchmarks_delta). Una vez transformadas las series temporales, se procede a calcular la matriz de correlación (benchmarks_corr)

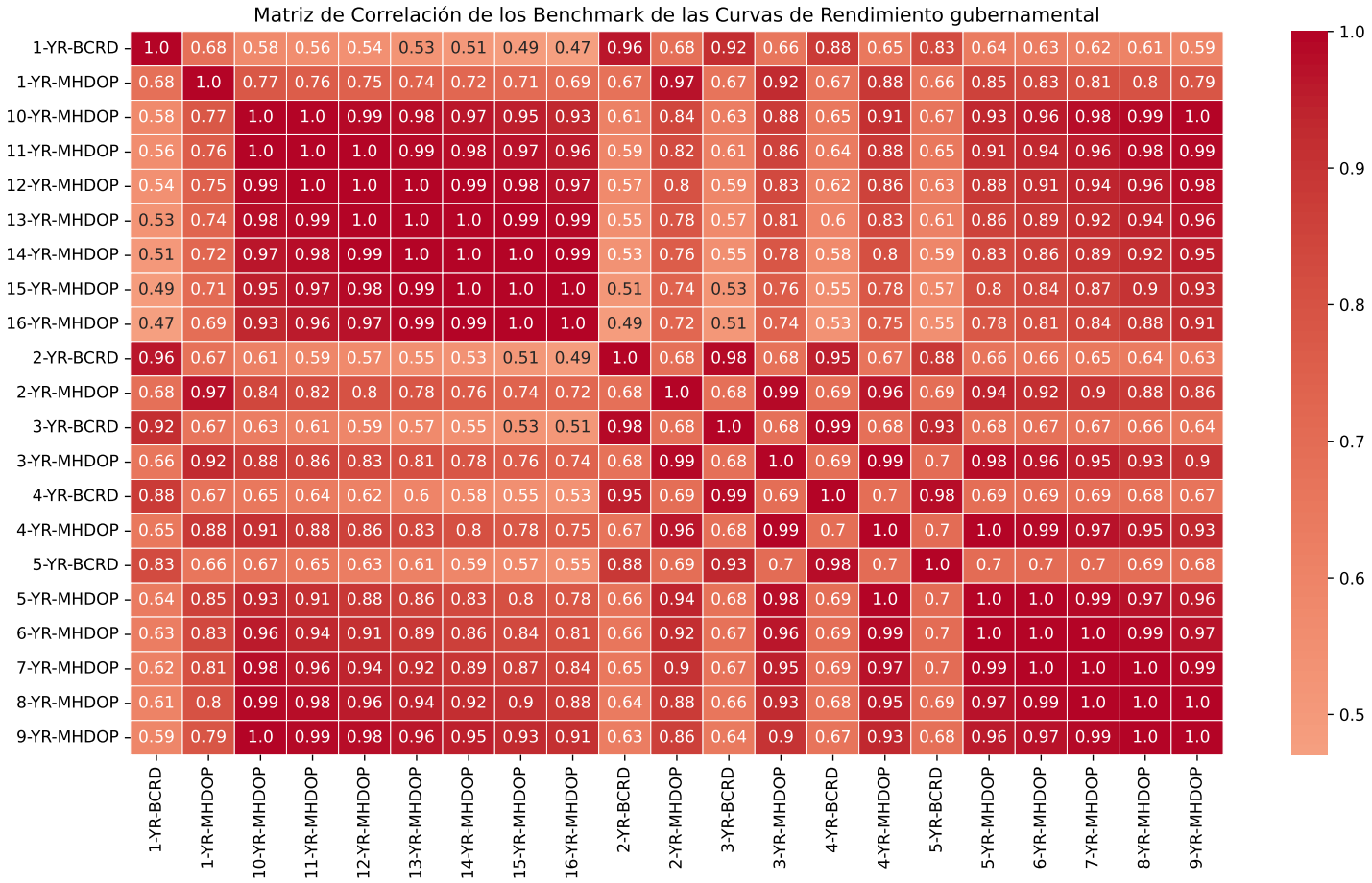
La matriz de correlación **R** es definida como:

$$R_{ij} = \frac{\text{Cov}(X_i, X_j)}{\sigma_{X_i}\sigma_{X_j}}$$

donde $\text{Cov}(X_i, X_j)$ es la covarianza entre los factores X_i y X_j, y σ_{X_i} es la desviación estándar de X_i. Esta matriz es esencial para medir la similitud entre las variables a nivel de correlación.

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 8))
```

```
benchmarks_delta = benchmarks.diff(1).dropna()
benchmarks_corr = benchmarks_delta.corr()
sns.heatmap(benchmarks_corr, annot=True, fmt=".2", cmap="coolwarm", center=0, linewidths=0.5, linecolor="white
plt.title("Matriz de Correlación de los Benchmark de las Curvas de Rendimiento gubernamental")
plt.show()
```

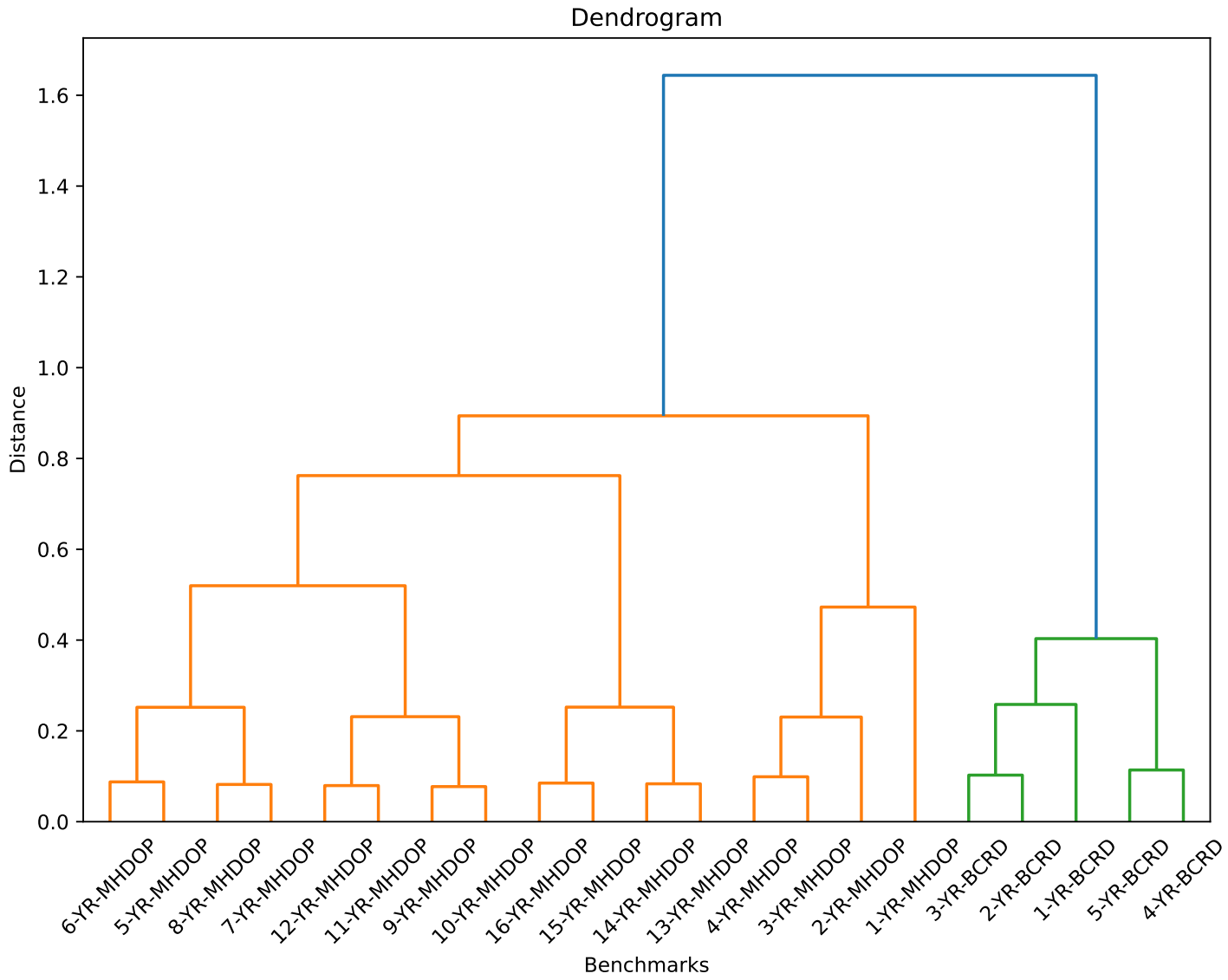


Análisis de Clúster Jerárquico

El siguiente paso metodológico involucra la aplicación de un clúster jerárquico la matriz de distancia, la cual se define como $ 1 - text{Cov}(X_i, X_j)$ El resultado del clúster jerárquico se representa mediante un dendograma, que visualiza cómo se agrupan los diferentes nodos de la curva de rendimiento en función de su similitud (correlación).

```
import numpy as np
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt

distance_matrix = 1 - abs(benchmarks_corr)

# Ejecutar cluster jerarquico
linked = sch.linkage(distance_matrix, method="complete")

# Plotear dendograma
plt.figure(figsize=(10, 7))
dendrogram = sch.dendrogram(
    linked, labels=benchmarks_corr.columns, orientation="top", distance_sort="descending", show_leaf_counts=Tr
)

plt.title("Dendrogram")
plt.xlabel("Benchmarks")
```

```
plt.ylabel("Distance")
plt.show()

# Create clusters based on the 0.98 correlation threshold
clusters = sch.fcluster(linked, 0.5, criterion="distance")
```



Se utilizó el método de enlace completo ("complete linkage") para determinar la distancia entre clústeres, definido como la distancia máxima entre los puntos de cada grupo.

Identificación de Clústeres

Finalmente, se identifican los clústeres de riesgo relevantes basados en un umbral de correlación de 0.95. Se asignan los valores de clústeres a cada nodo de referencia y se realiza una agrupación de los factores de riesgo por duración de macauly.

```
benchmark_clusters = dict(zip(benchmarks_corr.columns, clusters))
cluster_labels = {1: "3-YR-BCRD", 2: "3-YR-MHDOP", 5: "5-YR-MHDOP", 3: "15-YR-MHDOP", 4: "10-YR-MHDOP"}
risk_factors = pd.DataFrame(list(benchmark_clusters.items()), columns=["benchmark", "cluster_id"])
risk_factors["factor_riesgo"] = risk_factors["cluster_id"].map(cluster_labels)
risk_factors["duracion_macauly_risk_factor"] = risk_factors["factor_riesgo"].apply(lambda x: int(x.split("-")[
risk_factors.sort_values("duracion_macauly_risk_factor", inplace=True)
risk_factors.rename(columns={'benchmark': 'Nodo de referencia', 'factor_riesgo': 'Factor de Riesgo', 'duracion
risk_factors
```

	benchmark	cluster_id	factor_riesgo	duracion_macauly_risk_factor
0	1-YR-BCRD	1	3-YR-BCRD	3
15	5-YR-BCRD	1	3-YR-BCRD	3
14	4-YR-MHDOP	2	3-YR-MHDOP	3
13	4-YR-BCRD	1	3-YR-BCRD	3
12	3-YR-MHDOP	2	3-YR-MHDOP	3
11	3-YR-BCRD	1	3-YR-BCRD	3
9	2-YR-BCRD	1	3-YR-BCRD	3
10	2-YR-MHDOP	2	3-YR-MHDOP	3
1	1-YR-MHDOP	2	3-YR-MHDOP	3
19	8-YR-MHDOP	5	5-YR-MHDOP	5
16	5-YR-MHDOP	5	5-YR-MHDOP	5
17	6-YR-MHDOP	5	5-YR-MHDOP	5
18	7-YR-MHDOP	5	5-YR-MHDOP	5
4	12-YR-MHDOP	4	10-YR-MHDOP	10
3	11-YR-MHDOP	4	10-YR-MHDOP	10
2	10-YR-MHDOP	4	10-YR-MHDOP	10
20	9-YR-MHDOP	4	10-YR-MHDOP	10
8	16-YR-MHDOP	3	15-YR-MHDOP	15
6	14-YR-MHDOP	3	15-YR-MHDOP	15
5	13-YR-MHDOP	3	15-YR-MHDOP	15
7	15-YR-MHDOP	3	15-YR-MHDOP	15

Método de sensibilización de duración y convexidad

1. Introducción a la estrcutura de datos del Indice y I y X: De Datos de Panel a Soluciones de Sistemas Lineales

El código parte de dos DataFrames, `I` y `X`, que contiene datos de panel sobre las características de riesgo tanto de un índice como del universo de instrumentos financieros a lo largo del tiempo.

- **DataFrame `I`**: Este DataFrame representa los datos del índice y contiene las siguientes columnas: `date` (fecha), `risk_factor` (factor de riesgo), `ponderacion` (ponderación del índice), `money_duration` (duración monetaria) y `money_convexity` (convexidad monetaria).

```
import pandas as pd
I = pd.read_csv(r'.\I.csv')
I
```

	Unnamed: 0	date	risk_factor	ponderacion	money_duration	money_convexity
0	0	2014-01-01	3-YR-MHDOP	0.386178	3.103263	13.360018
1	1	2014-01-01	5-YR-MHDOP	0.404128	6.487428	54.059120
2	2	2014-01-01	3-YR-BCRD	0.209694	1.875276	5.306952
3	3	2014-02-01	3-YR-BCRD	0.210101	1.821709	5.316268
4	4	2014-02-01	3-YR-MHDOP	0.309631	2.640783	9.607432
...	...	...	...	...	...	...
386	386	2023-11-01	5-YR-MHDOP	0.283510	6.008318	48.065294
387	387	2023-11-01	3-YR-BCRD	0.600105	1.997036	5.622338
388	388	2023-12-01	3-YR-MHDOP	0.124236	3.086137	12.188503
389	389	2023-12-01	5-YR-MHDOP	0.275015	5.982364	49.061479
390	390	2023-12-01	3-YR-BCRD	0.600749	1.954741	5.526715

- **DataFrame `X`**: Este DataFrame representa el universo de instrumentos y contiene las columnas `date` (fecha), `isin` (identificador del instrumento), `risk_factor` (factor de riesgo), `money_duration` (duración dinero), y `money_convexity` (convexidad DINERO).

```
import pandas as pd
X = pd.read_csv(r'.\X.csv')
X
```

	Unnamed: 0	date	isin	risk_factor	money_duration	money_convexity	duration_to_convexity
0	4864	2023-12-01	DO1002227317	3-YR-BCRD	3.588172	15.395587	0.233065
1	4807	2023-11-01	DO1002227317	3-YR-BCRD	3.691170	15.505050	0.238062
2	4672	2023-08-01	DO1002226525	3-YR-BCRD	3.855543	17.608780	0.218956
3	4827	2023-11-01	DO1002226525	3-YR-BCRD	3.586627	13.697452	0.261846
4	4571	2023-06-01	DO1002226525	3-YR-BCRD	3.680351	15.847591	0.232234
...	...	...	...	...	...	...	...
4430	10	2013-11-01	DO1002245723	3-YR-BCRD	1.104257	1.383432	0.798201
4431	122	2014-03-01	DO1002245723	3-YR-BCRD	0.808227	0.467192	1.729965
4432	39	2013-12-01	DO1002245723	3-YR-BCRD	1.005797	1.359612	0.739768
4433	225	2014-06-01	DO1002245723	3-YR-BCRD	0.565692	0.472228	1.197923
4434	197	2014-05-01	DO1002245723	3-YR-BCRD	0.647015	0.470651	1.374724

El objetivo del ejercico es replicar las caracteristicas de riesgo del indice a lo largo del tiempo, encontrando un vector de ponderaciones β_k que minimice la diferencia entre la duración y la convexidad del índice y las correspondientes características del universo de instrumentos. Esto queda representado por la siguiente ecuación:

$$\beta_{k,t} = V_{k,t}^{-1} I_{k,t}, \quad \text{para cada } k \in \{1, 2, \dots, K\} \text{ y } t \in \{1, 2, \dots, T\}$$

Por lo que, por cada combinación específica de fecha y factor de riesgo (k, t) en nuestros datos de panel, se define un sistema de ecuaciones lineales que nos da un vector de ponderaciones β_k.

2. Explicación del Sistema Lineal y Resolución de Problemas

El código implementa un proceso iterativo para resolver un conjunto de sistemas de ecuaciones lineales. Matemáticamente, el problema se formula como:

$$\beta_k = V_k^{-1} I_k$$

Donde $ _k $ es el vector de ponderaciones para el factor de riesgo $ k $, $ V_k $ es una matriz que contiene la duración y la convexidad de los bonos seleccionados, y I_k es un vector con la duración y convexidad del índice para esa fecha y factor de riesgo.

El código considera diferentes casos según el número de bonos disponibles:

- **Un solo bono disponible**: La solución es trivial, y se asigna toda la ponderación a ese bono.
- **Dos bonos disponibles**: El sistema es sobredeterminado, y se distribuye la ponderación equitativamente entre los dos bonos.
- **Tres bonos disponibles**: Se construye una matriz $ A $ con las duraciones y convexidades de los tres bonos y se resuelve el sistema de ecuaciones lineales
$$A \times [b_1, b_2, b_3] = B$$

, donde $ B $ contiene la duración y convexidad del índice.

El sistema tendrá solución si y solo si la matriz $ A $ es invertible, y los coeficientes obtenidos son positivos. Esto garantiza que la ponderación para cada bono es válida en el contexto del problema financiero.

```
import numpy as np
import itertools
import pandas as pd
# Matriz I: DataFrame con columnas ['date', 'risk_factor', 'ponderacion', 'money_duration', 'money_convexity']
# Matriz X: DataFrame con columnas ['date', 'isin', 'risk_factor', 'money_duration', 'money_convexity']
I['date'] = pd.to_datetime(I['date'])
X['date'] = pd.to_datetime(X['date'])

# DataFrame vacío para almacenar resultados
resultados_df = pd.DataFrame(columns=['date', 'isin', 'b'])
# Obtener las combinaciones únicas de fechas y factores de riesgo en la matriz I
fechas_risk_factors = I[['date', 'risk_factor']].drop_duplicates()
```

```
# Iterar sobre cada combinación de fecha y factor de riesgo
for _, row in fechas_risk_factors.iterrows():
    fecha_t = row['date']
    risk_factor_t = row['risk_factor']

    # Filtrar las matrices I y X por la fecha y el factor de riesgo correspondientes
    I_t = I[(I['date'] == fecha_t) & (I['risk_factor'] == risk_factor_t)]
    X_t = X[(X['date'] == fecha_t) & (X['risk_factor'] == risk_factor_t)]

    ponderacion = I_t['ponderacion'].values[0]
    # Vector B para la fecha y factor de riesgo t
    if X_t.__len__() == 1:
        b = ponderacion
        isin_value = X_t['isin'].values[0]
        new_row = pd.DataFrame({'date': [fecha_t], 'isin': [isin_value], 'b': [b]})
        resultados_df = pd.concat([resultados_df, new_row], ignore_index=True)
        continue

    if X_t.__len__() == 2:
        for comb in itertools.combinations(X_t.index, 2):
            b = np.array([0.5, 0.5])
            b = b * ponderacion
            for i, bono in enumerate(comb):
                isin_value = X_t.loc[bono, 'isin']
                new_row = pd.DataFrame({'date': [fecha_t], 'isin': [isin_value], 'b': [b[i]]})
                resultados_df = pd.concat([resultados_df, new_row], ignore_index=True)
            break  # Detener el ciclo si se encuentra una solución
        continue

    for comb in itertools.combinations(X_t.index, 3):
        # Construir la matriz A con los bonos seleccionados
        money_duration_array = X_t.filter(items = comb, axis = 0)['money_duration'].values
        money_convexity_array = X_t.filter(items = comb, axis = 0)['money_convexity'].values
        A = np.array([
            money_duration_array,
            money_convexity_array,
            [1, 1, 1]
        ])
        B = np.array([I_t['money_duration'].values[0], I_t['money_convexity'].values[0], 1])
        # Resolver el sistema A * [b1, b2, b3] = B
        try:
            b = np.linalg.solve(A, B)
        except np.linalg.LinAlgError:
            continue  # Saltar si la matriz A no es invertible

        # Verificar si todos los coeficientes son positivos
        if np.all(b > 0):
            # Multiplicar los valores de b por la ponderación correspondiente
            ponderacion = I_t['ponderacion'].values[0]
            b = b * ponderacion

            # Guardar los resultados en el DataFrame
            for i, bono in enumerate(comb):
                isin_value = X_t.loc[bono, 'isin']
                new_row = pd.DataFrame({'date': [fecha_t], 'isin': [isin_value], 'b': [b[i]]})
                resultados_df = pd.concat([resultados_df, new_row], ignore_index=True)
            break  # Detener el ciclo si se encuentra una solución
    else:
        print(f"No se encontró una combinación que garantice que b1, b2, b3 sean positivos para la fecha {fecha
```

3. Explicación del DataFrame de Resultados

El DataFrame `resultados_df` almacena las soluciones del sistema de ecuaciones para cada combinación de fecha y factor de riesgo (k, t). Las columnas principales son:

- **`date`**: La fecha correspondiente a la solución encontrada.
- **`isin`**: El identificador del bono para el cual se ha encontrado una ponderación válida.
- **`b`**: La ponderación asignada a ese bono.

Cada fila del DataFrame representa una solución válida para una combinación específica de fecha y factor de riesgo. El proceso iterativo asegura que se encuentran soluciones que cumplen las condiciones del sistema de ecuaciones para cada combinación posible, siempre que existan suficientes bonos con características de riesgo distintas.

El código está diseñado para detenerse tan pronto como se encuentra una solución válida para tres bonos. Si no se encuentra una combinación que satisfaga las condiciones, el código lo reporta, lo que permite identificar posibles inconsistencias en los datos o en la modelización del problema.

```
resultados_df
```

	date	isin	b
0	2014-01-01	DO1005250829	0.012207
1	2014-01-01	DO1005241026	0.258747
2	2014-01-01	DO1005221523	0.115223
3	2014-01-01	DO1005252429	0.060234
4	2014-01-01	DO1005251025	0.074509
...	...	...	...
1148	2023-12-01	DO1005211227	0.122209
1149	2023-12-01	DO1005208124	0.061969
1150	2023-12-01	DO1002227317	0.031082
1151	2023-12-01	DO1002226525	0.050681
1152	2023-12-01	DO1002225527	0.518986