Escuela Universitaria Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

ANEXOS

Herramientas dinámicas para determinación de política de inventario óptima bajo distintas distribuciones de demanda

Dynamic tools for deciding on the best inventory policy under different demand distributions

425.18.94

Autor: Adrián Calleja Visiedo

Director: Luis Mariano Esteban Escaño

Fecha: 28/11/2018

INDICE DE CONTENIDO

ANEXO 1. (CÓDIGO PYTHON)

A continuación, se adjunta el código Python que emplea el programa diseñado en Jupyter Notebook junto a alguna breve explicación.

1.1. DATOS DE ENTRADA

Se comienza importando las librerías necesarias y cargando el archivo Excel con los datos de entrada.

```
import pandas as pd
import numpy as np
import math
from scipy.stats import norm
import random
import matplotlib.pyplot as plt
from scipy import stats

file = pd.ExcelFile('C:/Users/AdrianCalleja/Datos.xlsx')

print(file.sheet_names)
```

```
['Ventas', 'MaestroProductos', 'TablaDin']
```

1.2. FILTRACIÓN DE DATOS Y PRIMEROS CÁLCULOS

Seguidamente, se procede a realizar el filtrado de datos de la demanda según el percentil 95.

```
df1 = file.parse('Ventas')
df1 = df1.round()

df1_pivot = df1.pivot(index = 'idSecuencia', columns = 'idProducto', values = 'udsDemanda')

df1_pivot_perc = pd.DataFrame()
df1_pivot_perc = df1_pivot[:]

df1_pivot_perc_sin_nan = pd.DataFrame()
df1_pivot_perc_sin_nan = df1_pivot_perc[:]

for x in range (0, len(df1_pivot.columns)):
    Datos_antes_percentiles = df1_pivot.iloc[:, x]

    Promedio_prueba = Datos_antes_percentiles.sum()/len(Datos_antes_percentiles[Datos_antes_percentiles > 0])
    Std_prueba = Datos_antes_percentiles[Datos_antes_percentiles > 0].std()

    max1 = Datos_antes_percentiles.max()

    perc_inf = np.percentile(Datos_antes_percentiles, 0)
    perc_sup = np.percentile(Datos_antes_percentiles, 95)

    Datos_despues_percentiles = Datos_antes_percentiles[(Datos_antes_percentiles >= perc_inf) & (Datos_antes_percentiles
                                                                                                 <= perc_sup)]

    max2 = Datos_despues_percentiles.max()

    df1_pivot_perc.iloc[:, x] = Datos_despues_percentiles

    Promedio_prueba_perc = (Datos_despues_percentiles.sum()/
                            len(Datos_despues_percentiles[Datos_despues_percentiles > 0])).round()
    Std_prueba_perc = Datos_despues_percentiles[Datos_despues_percentiles > 0].std()

    df1_pivot_perc_sin_nan.iloc[:, x] = df1_pivot_perc.iloc[:, x].fillna(Promedio_prueba_perc)[:]
```

Después se realizan cálculos básicos sobre la demanda.

```
36
37  GroupbyFechaArticulo = df1_pivot_perc_sin_nan.stack()[:]
38  GroupbyFechaArticulo.columns = ['idSecuencia', 'idProducto', 'udsDemanda']
39
40  ArticulosHistorico = df1['idProducto']
41  ArticulosHistorico = ArticulosHistorico.drop_duplicates()
42  ArticulosHistorico = ArticulosHistorico.sort_values()
43  ArticulosHistorico = ArticulosHistorico.reset_index()
44  ArticulosHistorico = ArticulosHistorico.drop(['index'], axis=1)
45  ArticulosHistorico['ColumnaComun'] = 1
46
47  FechasHistorico = df1['idSecuencia']
48  FechasHistorico = FechasHistorico.drop_duplicates()
49  FechasHistorico = FechasHistorico.sort_values()
50  FechasHistorico = FechasHistorico.reset_index()
51  FechasHistorico = FechasHistorico.drop(['index'], axis=1)
52  FechasHistorico['ColumnaComun'] = 1
53
54  HistoricoFechasArticulos = pd.merge(FechasHistorico, ArticulosHistorico, how = 'outer', on = 'ColumnaComun')
55  HistoricoFechasArticulos = HistoricoFechasArticulos.drop(['ColumnaComun'], axis=1)
56
57  ReconstruccionPedidos = pd.merge(HistoricoFechasArticulos, GroupbyFechaArticulo.to_frame(), how = 'left',
58                                   on = ['idProducto', 'idSecuencia'])
59  ReconstruccionPedidos = ReconstruccionPedidos.fillna(0)
60  ReconstruccionPedidos.columns = ['idSecuencia', 'idProducto', 'udsDemanda']
61
62  SumDemandaTotalArticulo = ReconstruccionPedidos.groupby(['idProducto']).sum()
63  SumDemandaTotalArticulo = SumDemandaTotalArticulo.drop(['idSecuencia'], axis=1)
64
65  CountDemandaTotalArticulo = ReconstruccionPedidos[ReconstruccionPedidos.udsDemanda > 0].groupby(['idProducto']).count()
66  CountDemandaTotalArticulo = CountDemandaTotalArticulo.drop(['idSecuencia'], axis=1)
67
```

```
68  DemandaMediaArticuloPorPedido = SumDemandaTotalArticulo/CountDemandaTotalArticulo
69
70  StdDemandaMediaArticuloPorPedido = ReconstruccionPedidos[ReconstruccionPedidos.udsDemanda >
71                                                           0].groupby(['idProducto']).std()
72  StdDemandaMediaArticuloPorPedido = StdDemandaMediaArticuloPorPedido.drop(['idSecuencia'], axis=1)
73
74  Tabla1 = pd.merge(SumDemandaTotalArticulo, CountDemandaTotalArticulo, how = 'outer', on = 'idProducto' )
75  Tabla1.columns = ['Demanda_Total', 'N pedidos']
76
77  Tabla1 = pd.merge(Tabla1, DemandaMediaArticuloPorPedido, how = 'outer', on = 'idProducto' )
78  Tabla1.columns = ['Demanda_Total','N pedidos', 'Demanda Promedio']
79
80  Tabla1 = pd.merge(Tabla1, StdDemandaMediaArticuloPorPedido, how = 'outer', on = 'idProducto' )
81  Tabla1.columns = ['Demanda_Total', 'N pedidos', 'Demanda Promedio', 'Desv Demanda']
82
83  df2 = file.parse('MaestroProductos')
84
85  df2 = df2.groupby(['idProducto']).sum()
86
87  Tabla2 = pd.merge(Tabla1, df2, how = 'outer', on = 'idProducto' )
88  Tabla2.columns = ['Demanda_Total', 'N_pedidos', 'Demanda_Promedio', 'Desv_Demanda',
89                    'Precio', 'Lt', 'dLt', 'Ce', 'Cm', 'Cs']
90
91  Tabla2['Valor'] = Tabla2.Demanda_Total * Tabla2.Precio
92  ValorTotalProductos = Tabla2['Valor'].sum()
93  Tabla2['Valor_%'] = ((Tabla2.Valor * 100)/ValorTotalProductos)
94  Tabla2 = Tabla2.sort_values('Valor_%', ascending=False)
```

1.3. ANÁLISIS DE PARETO

Y se realiza el análisis de Pareto junto con la asignación del nivel de servicio.

```
i = 0
valor = Tabla2['Valor_%'].head(i).sum(i)
Limite_Rank_A = 80
Limite_Rank_B = 95
Limite_Rank_C = 100

while valor < Limite_Rank_C:
    i = (i + 1)
    valor = Tabla2['Valor_%'].head(i).sum()
    Rank_C = i
    while valor < Limite_Rank_B:
        i = (i + 1)
        valor = Tabla2['Valor_%'].head(i).sum()
        Rank_B = i
        while valor < Limite_Rank_A:
            i = (i + 1)
            valor = Tabla2['Valor_%'].head(i).sum()
            Rank_A = i

Tabla2.reset_index(level=0, inplace=True)

Tabla2['Rank_ABC'] = 0
Tabla2.loc[0:Rank_A, 'Rank_ABC'] = 'A'
Tabla2.loc[Rank_A:Rank_B, 'Rank_ABC'] = 'B'
Tabla2.loc[Rank_B:Rank_C, 'Rank_ABC'] = 'C'

Tabla2['N_Servicio'] = 0
Tabla2.loc[0:Rank_A, 'N_Servicio'] = 99
Tabla2.loc[Rank_A:Rank_B, 'N_Servicio'] = 90
Tabla2.loc[Rank_B:Rank_C, 'N_Servicio'] = 80
```

1.4. PARÁMETROS DE CONTROL

Ahora se pueden calcular los parámetros que controlarán las diferentes políticas.

```
Tabla2['Phi'] = norm.ppf(Tabla2.N_Servicio/100)
#Contar numero de entradas
N_fechas_distintas = int(len(set(df1.idSecuencia)))
N_articulos_distintos = int(len(set(df1.idProducto)))
Factor_Dias = Tabla2.N_pedidos/N_fechas_distintas

Tabla2['EOQ'] = ((2 * Tabla2.Demanda_Total * Tabla2.Ce) / (Tabla2.Cm))**(1/2)
Tabla2['EOQ'] = Tabla2['EOQ'].apply(np.round)

Tabla2['Pp'] = (Tabla2.Demanda_Promedio*Factor_Dias) * Tabla2.Lt + (Tabla2.Phi * (Tabla2.Lt * (Tabla2.Desv_Demanda**2) +
                                        (((Tabla2.Demanda_Promedio*Factor_Dias)**2)*(Tabla2.dLt**2)))**(1/2))
Tabla2['Pp'] = Tabla2['Pp'].apply(np.round)
Tabla2['Tp'] = round(360/((Tabla2.Demanda_Total * (Tabla2.N_pedidos/int(len(set(df1.idSecuencia)))))/Tabla2.EOQ))


Tabla2['Stock_de_Ciclo'] = (Tabla2.Demanda_Promedio*Factor_Dias) * (Tabla2.Lt + Tabla2.Tp)
Tabla2['Stock_de_Ciclo'] = Tabla2['Stock_de_Ciclo'].apply(np.round)
Tabla2['Stock_de_Ciclo_prueba'] = (Tabla2.Demanda_Promedio*Factor_Dias) * (Tabla2.Lt)
Tabla2['Stock_de_Ciclo_prueba'] = Tabla2['Stock_de_Ciclo_prueba'].apply(np.round)

Tabla2['Stock_de_Seguridad'] = Tabla2.Phi * ((((Tabla2.Lt + Tabla2.Tp) * (Tabla2.Desv_Demanda**2)+
                                        (((Tabla2.Demanda_Promedio*Factor_Dias)**2)*(Tabla2.dLt**2))))**(1/2))
Tabla2['Stock_de_Seguridad'] = Tabla2['Stock_de_Seguridad'].apply(np.round)

Tabla2['StockObjetivo'] = Tabla2.Stock_de_Ciclo + Tabla2.Stock_de_Seguridad
Tabla2['StockObjetivo'] = Tabla2['StockObjetivo'].apply(np.round)
Tabla2['StockObjetivo_prueba'] = Tabla2.Stock_de_Ciclo_prueba + Tabla2.Stock_de_Seguridad
Tabla2['StockObjetivo_prueba'] = Tabla2['StockObjetivo_prueba'].apply(np.round)
```

Se preparan algunas tablas para registrar valores durante las simulaciones.

```
TablaIndice = pd.DataFrame(Tabla2.idProducto)
N_articulos_distintos = int(len(set(TablaIndice.idProducto)))

TablaLt = pd.DataFrame()

TablaDemanda_1 = pd.DataFrame()
TablaDemanda_2 = pd.DataFrame()
TablaDemanda_3 = pd.DataFrame()
TablaDemanda_4 = pd.DataFrame()

TablaStock_1 = pd.DataFrame()
TablaStock_2 = pd.DataFrame()
TablaStock_3 = pd.DataFrame()
TablaStock_4 = pd.DataFrame()

TablaReabast_1 = pd.DataFrame()
TablaReabast_2 = pd.DataFrame()
TablaReabast_3 = pd.DataFrame()
TablaReabast_4 = pd.DataFrame()

TablaRoturas_1 = pd.DataFrame()
TablaRoturas_2 = pd.DataFrame()
TablaRoturas_3 = pd.DataFrame()
TablaRoturas_4 = pd.DataFrame()
```

1.5. SIMULACIÓN DEMANDA Y LEAD TIME

Y comienza la simulación de la demanda y lead time.

```
Rango_fin_bucles = N_articulos_distintos - 1

#SIMULAR DEMANDA y Lt
TablaDemandas = pd.DataFrame()
TablaLt = pd.DataFrame()

for x in range(0, Rango_fin_bucles):
    print ("Simulacion del articulo ", x)
    Articulo = TablaIndice['idProducto'][x]
    mu = Tabla2['Demanda_Promedio'][x]
    sigma = Tabla2['Desv_Demanda'][x]
    Lt = Tabla2['Lt'][x]
    dLt = Tabla2['dLt'][x]
    print ("Articulo: ", Articulo)

    N_fechas_distintas = int(len(set(df1.idSecuencia)))
    Factor_Dia = Tabla2['N_pedidos'][x]/N_fechas_distintas
    print("El factor dia es: ", Factor_Dia)

    contador_veces_demanda = 0

    lista_demanda = []
    lista_Lt = []

```


Autor: Adrián Calleja Visiedo
425.18.94

```
    for y in range(0, 360):
        random_number = random.random()

        if Factor_Dia > random_number:
            Simulacion_demanda = stats.expon.rvs(scale=mu, size=1)
            Simulacion_demanda = Simulacion_demanda.round()
            while Simulacion_demanda <= 0:
                Simulacion_demanda = (np.random.normal(mu, sigma, 1))
                Simulacion_demanda = Simulacion_demanda.round()

            lista_demanda.append(Simulacion_demanda)

        else:
            lista_demanda.append(0)

        #Lead time
        Simulacion_Lt = (np.random.normal(Lt, dLt, 1))
        Simulacion_Lt = Simulacion_Lt.round()
        while Simulacion_Lt <= 0:
            Simulacion_Lt = (np.random.normal(Lt, dLt, 1))
            Simulacion_Lt = Simulacion_Lt.round()

        lista_Lt.append(Simulacion_Lt)

    TablaDemandas[Articulo] = lista_demanda

    TablaLt[Articulo] = lista_Lt


TablaDemandas = TablaDemandas.astype(int)
TablaLt = TablaLt.astype(int)
```

1.6. SIMULACIÓN POLÍTICAS

A partir de aquí comienzan las simulaciones de las diferentes políticas, comenzando por Stock Objetivo.

```
#=============================================================================================================================
#=========================================METODO STOCK OBJETIVO================================================================
#=============================================================================================================================
i = 0
for x in range(0, Rango_fin_bucles):
    print ("Simulacion del articulo ", x)
    Articulo = TablaIndice['idProducto'][i]
    mu = Tabla2['Demanda_Promedio'][i]
    sigma = Tabla2['Desv_Demanda'][i]
    Lt = Tabla2['Lt'][i]
    dLt = Tabla2['dLt'][i]
    print ("Articulo: ", Articulo)
    print ("mu es: ", mu)
    print ("sigma es: ", sigma)
    print ("el Lt es: ", Lt)
    print ("la dLt es: ", dLt)
    Stock_objetivo = Tabla2['StockObjetivo'][i]
    Stock_disponible = Stock_objetivo
    Stock_virtual = Stock_objetivo
    print ("El stock objetivo es: ", Stock_objetivo)
    N_fechas_distintas = int(len(set(df1.idSecuencia)))
    Factor_Dia = Tabla2['N_pedidos'][i]/N_fechas_distintas
    print("El factor dia es: ", Factor_Dia)
    contador_veces_demanda = 0
    contador_dias = 0
    Tp = Tabla2['Tp'][i]
    print("El tiempo entre pedidos es: ", Tp)
    Esperando_reabast = 0

    dia_emision_orden_reabast = 0
    Simulacion_Lt_dia_pedido = 0
    orden_reabast_emitida = 0
    recepciones_pendientes = 0
```

(Código Python)

```
    lista_stock = []
    lista_reabast = []
    lista_lt = []
    lista_y = []
    lista_recep = []
    lista_reab = []
    lista_dia_reab = []
    lista_roturas_stock = []
    lista_demanda = []

    for y in range(0, 360):
        random_number = random.random()

        Simulacion_demanda = TablaDemandas[Articulo][y]
        Simulacion_Lt = TablaLt[Articulo][y]

        if Simulacion_demanda > 0:
            lista_demanda.append(Simulacion_demanda)
            Stock_disponible = (Stock_disponible - Simulacion_demanda)
            Stock_virtual = Stock_virtual - Simulacion_demanda
            contador_veces_demanda = contador_veces_demanda + 1

            if Esperando_reabast == 1:
                for z in range (0,len(lista_dia_reab) + 5):
                    if (len(lista_dia_reab) > 0):
                        zz = 0
                        while zz < len(lista_dia_reab):
                            if (contador_dias == lista_dia_reab[zz]):
                                Stock_disponible = Stock_disponible + lista_reab[0]
                                recepciones_pendientes = recepciones_pendientes - lista_reab[0]
                                if recepciones_pendientes == 0:
                                    Esperando_reabast = 0
```

```
                                lista_reab.pop(0)
                                lista_dia_reab.pop(0)

                            else:
                                zz = zz + 1

            if (contador_dias % Tp == 0 and Stock_disponible < Stock_objetivo and contador_dias != 0):

                orden_reabast_emitida = Stock_objetivo - Stock_virtual
                Stock_virtual = Stock_virtual + orden_reabast_emitida

                Esperando_reabast = 1
                recepciones_pendientes = recepciones_pendientes + orden_reabast_emitida

                dia_emision_orden_reabast = contador_dias
                Simulacion_Lt_dia_pedido = Simulacion_Lt

                if (orden_reabast_emitida != 0):
                    lista_reab.append(orden_reabast_emitida)
                    lista_dia_reab.append(dia_emision_orden_reabast + Simulacion_Lt_dia_pedido)

            if Stock_disponible < 0 and lista_stock[-1] < 0:
                if Stock_disponible < lista_stock[-1]:
                    lista_roturas_stock.append(Stock_disponible - lista_stock[-1])
                else:
                    lista_roturas_stock.append(0)
            elif Stock_disponible < 0 and lista_stock[-1] >= 0:
                lista_roturas_stock.append(Stock_disponible)
            else:
                lista_roturas_stock.append(0)
```

```


        else:
            lista_roturas_stock.append(0)
            lista_demanda.append(0)

            if Esperando_reabast == 1:
                for z in range (0,len(lista_dia_reab) + 5):
                    if (len(lista_dia_reab) > 0):
                        zz =  0
                        while zz < len(lista_dia_reab):
                            if (contador_dias == lista_dia_reab[zz]):
                                Stock_disponible = Stock_disponible + lista_reab[0]
                                recepciones_pendientes = recepciones_pendientes - lista_reab[0]
                                if recepciones_pendientes == 0:
                                    Esperando_reabast = 0

                                lista_reab.pop(0)
                                lista_dia_reab.pop(0)

                            else:
                                zz = zz + 1
```


Autor: Adrián Calleja Visiedo
425.18.94

```
                orden_reabast_emitida = Stock_objetivo - Stock_virtual
                Stock_virtual = Stock_virtual + orden_reabast_emitida
                Esperando_reabast = 1
                recepciones_pendientes = recepciones_pendientes + orden_reabast_emitida
                dia_emision_orden_reabast = contador_dias
                Simulacion_Lt_dia_pedido = Simulacion_Lt

                if (orden_reabast_emitida != 0):
                    lista_reab.append(orden_reabast_emitida)
                    lista_dia_reab.append(dia_emision_orden_reabast + Simulacion_Lt_dia_pedido)

        lista_stock.append(Stock_disponible)

        if (contador_dias % Tp == 0 and Stock_disponible < Stock_objetivo):
            lista_reabast.append(orden_reabast_emitida)
        else:
            lista_reabast.append(0)

        lista_lt.append(Simulacion_Lt)
        lista_y.append(y)
        contador_dias = contador_dias + 1

    i = (i + 1)
    print ("i es: ", i)

    TablaStock_1[Articulo] = lista_stock
    TablaReabast_1[Articulo] = lista_reabast
    TablaRoturas_1[Articulo] = lista_roturas_stock
    TablaDemanda_1[Articulo] = lista_demanda
```

El código es prácticamente el mismo para el resto de políticas, solamente cambia la cantidad y la condición de reabastecimiento según lo explicado en la memoria, por lo que pasaremos directamente con los costes.

1.7. COSTES DE LAS POLÍTICAS

```
#============================TABLACOSTES_1=========================================
TablaDemanda_1 = TablaDemanda_1.astype(int)
TablaStock_1 = TablaStock_1.astype(int)
TablaReabast_1 = TablaReabast_1.astype(int)
TablaLt = TablaLt.astype(int)

TablaCostes_1 = pd.DataFrame()
Tabla2.set_index('idProducto', inplace = True)
TablaCostes_1['Media_stock'] = TablaStock_1.mean()
TablaCostes_1 = TablaCostes_1.astype(int)
TablaCostes_1['Cm'] = Tabla2.Cm
TablaCostes_1['Cm_Total'] = TablaCostes_1.Media_stock * TablaCostes_1.Cm
CountReabastArticulo = TablaReabast_1[TablaReabast_1 > 0].count()
TablaCostes_1['N_reabast'] = CountReabastArticulo
TablaCostes_1['Ce'] = Tabla2.Ce
TablaCostes_1['Ce_Total'] = TablaCostes_1.N_reabast * TablaCostes_1.Ce

TablaRoturas_1 = TablaRoturas_1.astype(int)
CountStockNegativo2 = TablaRoturas_1[TablaRoturas_1 < 0].count()
SumStockNegativo2 = TablaRoturas_1[TablaRoturas_1 < 0].sum()

TablaCostes_1['Roturas_stock_u'] = (SumStockNegativo2) * (-1)
TablaCostes_1['Cs'] = Tabla2.Cs
TablaCostes_1['Cs_Total'] = TablaCostes_1.Roturas_stock_u * TablaCostes_1.Cs
TablaCostes_1['Suma_Coste_Total'] = TablaCostes_1.Cm_Total + TablaCostes_1.Ce_Total + TablaCostes_1.Cs_Total
TablaCostes_1['Precio'] = Tabla2.Precio
TablaCostes_1['Inversion_stock_media'] = TablaCostes_1.Media_stock * TablaCostes_1.Precio
TablaCostes_1 = TablaCostes_1.round(2)

CosteEmisionInventario = TablaCostes_1['Ce_Total'].sum()
CosteMantenimientoInventario = TablaCostes_1['Cm_Total'].sum()
CosteRupturasInventario = TablaCostes_1['Cs_Total'].sum()
CosteTotalInventario = TablaCostes_1['Suma_Coste_Total'].sum()
InversionInventario = TablaCostes_1['Inversion_stock_media'].sum()
```

Para las tablas de costes 2, 3 y 4 el procedimiento es exactamente el mismo.

(Código Python)

A continuación, se calcula "TablaCostes_Minimos".

```
lista_indices = []
lista_indices = TablaCostes_1.index

TablaCostes_Minimos = pd.DataFrame()

TablaCostes_Minimos = TablaCostes_1.copy()
TablaCostes_Minimos['Politica_Ganadora'] = 'Temporal'

x = 0
for x in range(0, Rango_fin_bucles):
#TablaCostes_1
    if (TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]] < TablaCostes_1['Suma_Coste_Total'][lista_indices[x]]):
        TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_Minimos[TablaCostes_Minimos.index ==
                                                                                                 lista_indices[x]]
        TablaCostes_Minimos['Politica_Ganadora'][lista_indices[x]] = 'Politica_1'
    else:
        TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_1[TablaCostes_1.index ==
                                                                                           lista_indices[x]]
        TablaCostes_Minimos['Politica_Ganadora'][lista_indices[x]] = 'Politica_1'

#TablaCostes_2
    if (TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]] < TablaCostes_2['Suma_Coste_Total'][lista_indices[x]]):
        TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_Minimos[TablaCostes_Minimos.index ==
                                                                                                 lista_indices[x]]
    else:
        TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_2[TablaCostes_2.index ==
                                                                                           lista_indices[x]]
        TablaCostes_Minimos['Politica_Ganadora'][lista_indices[x]] = 'Politica_2'

#TablaCostes_3
    if (TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]] < TablaCostes_3['Suma_Coste_Total'][lista_indices[x]]):
        TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_Minimos[TablaCostes_Minimos.index ==
                                                                                                 lista_indices[x]]
    else:
        TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_3[TablaCostes_3.index ==
                                                                                           lista_indices[x]]
        TablaCostes_Minimos['Politica_Ganadora'][lista_indices[x]] = 'Politica_3'

#TablaCostes_4
    if (TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]] < TablaCostes_4['Suma_Coste_Total'][lista_indices[x]]):
        TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_Minimos[TablaCostes_Minimos.index ==
                                                                                                 lista_indices[x]]
    else:
        TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_4[TablaCostes_4.index ==
                                                                                           lista_indices[x]]
        TablaCostes_Minimos['Politica_Ganadora'][lista_indices[x]] = 'Politica_4'


TablaCostes_Minimos
```


Autor: Adrián Calleja Visiedo
425.18.94

1.8. REDUCCIÓN DE INVERSIÓN

Y el ahorro en inversión que supone cada política respecto a la actual.

```
TablaCostes_1['Sobrecoste'] = 'Temporal'
TablaCostes_2['Sobrecoste'] = 'Temporal'
TablaCostes_3['Sobrecoste'] = 'Temporal'
TablaCostes_4['Sobrecoste'] = 'Temporal'

TablaCostes_1['Ahorro_en_inversion'] = 'Temporal'
TablaCostes_2['Ahorro_en_inversion'] = 'Temporal'
TablaCostes_3['Ahorro_en_inversion'] = 'Temporal'
TablaCostes_4['Ahorro_en_inversion'] = 'Temporal'

x = 0
for x in range(0, Rango_fin_bucles):
#TablaCostes_1
    Sobrecoste = TablaCostes_1['Suma_Coste_Total'][lista_indices[x]] -
                         TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]]
    Ahorro_en_inversion = TablaCostes_Minimos['Inversion_stock_media'][lista_indices[x]] -
                         TablaCostes_1['Inversion_stock_media'][lista_indices[x]]

    TablaCostes_1['Sobrecoste'][lista_indices[x]] = Sobrecoste
    TablaCostes_1['Ahorro_en_inversion'][lista_indices[x]] = Ahorro_en_inversion

#TablaCostes_2
    Sobrecoste = TablaCostes_2['Suma_Coste_Total'][lista_indices[x]] -
                         TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]]
    Ahorro_en_inversion = TablaCostes_Minimos['Inversion_stock_media'][lista_indices[x]] -
                         TablaCostes_2['Inversion_stock_media'][lista_indices[x]]

    TablaCostes_2['Sobrecoste'][lista_indices[x]] = Sobrecoste
    TablaCostes_2['Ahorro_en_inversion'][lista_indices[x]] = Ahorro_en_inversion

#TablaCostes_3
    Sobrecoste = TablaCostes_3['Suma_Coste_Total'][lista_indices[x]] -
                         TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]]
    Ahorro_en_inversion = TablaCostes_Minimos['Inversion_stock_media'][lista_indices[x]] -
                         TablaCostes_3['Inversion_stock_media'][lista_indices[x]]

    TablaCostes_3['Sobrecoste'][lista_indices[x]] = Sobrecoste
    TablaCostes_3['Ahorro_en_inversion'][lista_indices[x]] = Ahorro_en_inversion

#TablaCostes_4
    Sobrecoste = TablaCostes_4['Suma_Coste_Total'][lista_indices[x]] -
                         TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]]
    Ahorro_en_inversion = TablaCostes_Minimos['Inversion_stock_media'][lista_indices[x]] -
                         TablaCostes_4['Inversion_stock_media'][lista_indices[x]]

    TablaCostes_4['Sobrecoste'][lista_indices[x]] = Sobrecoste
    TablaCostes_4['Ahorro_en_inversion'][lista_indices[x]] = Ahorro_en_inversion
```

(Código Python)

Para después obtener los ratios.

```
TablaCostes_1['Ratio_Ahorro_en_inversion_Sobrecoste'] = TablaCostes_1['Ahorro_en_inversion'] / TablaCostes_1['Sobrecoste']
TablaCostes_2['Ratio_Ahorro_en_inversion_Sobrecoste'] = TablaCostes_2['Ahorro_en_inversion'] / TablaCostes_2['Sobrecoste']
TablaCostes_3['Ratio_Ahorro_en_inversion_Sobrecoste'] = TablaCostes_3['Ahorro_en_inversion'] / TablaCostes_3['Sobrecoste']
TablaCostes_4['Ratio_Ahorro_en_inversion_Sobrecoste'] = TablaCostes_4['Ahorro_en_inversion'] / TablaCostes_4['Sobrecoste']

TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'] = 0.00
TablaCostes_Minimos['Politica_Ratio_Mayor'] = 'Actual'

x = 0
for x in range (0, Rango_fin_bucles):
#TablaCostes_1
    if (TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] <
                            TablaCostes_1['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]):
        TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] =
                            TablaCostes_1['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]
        TablaCostes_Minimos['Politica_Ratio_Mayor'][lista_indices[x]] = 'Politica_1'
    else:
        pass

#TablaCostes_2
    if (TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] <
                            TablaCostes_2['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]):
        TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] =
                            TablaCostes_2['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]
        TablaCostes_Minimos['Politica_Ratio_Mayor'][lista_indices[x]] = 'Politica_2'
    else:
        pass

#TablaCostes_3
    if (TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] <
                            TablaCostes_3['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]):
        TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] =
                            TablaCostes_3['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]
        TablaCostes_Minimos['Politica_Ratio_Mayor'][lista_indices[x]] = 'Politica_3'
    else:
        pass

#TablaCostes_4
    if (TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] <
                            TablaCostes_4['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]):
        TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] =
                            TablaCostes_4['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]
        TablaCostes_Minimos['Politica_Ratio_Mayor'][lista_indices[x]] = 'Politica_4'
    else:
        pass
```

Y poder compararlos.

```
TablaCostes_Minimos_b = pd.DataFrame()
TablaCostes_Minimos_b = TablaCostes_Minimos.copy()

TablaCostes_Minimos_b = TablaCostes_Minimos_b.sort_values('Ratio_Ahorro_en_inversion_Mayor', ascending=False)

lista_indices_TablaCostes_Minimos_b = []
lista_indices_TablaCostes_Minimos_b = TablaCostes_Minimos_b.index

TablaReduccion_Inversion = pd.DataFrame()
TablaReduccion_Inversion = TablaCostes_Minimos.copy()

lista_Coste_Almacenaje = []
lista_Inversion_Almacenaje = []

x = 0
for x in range (0, 250):
    Valor_Coste_Almacenaje = TablaReduccion_Inversion['Suma_Coste_Total'].sum()
    Valor_Inversion_Almacenaje = TablaReduccion_Inversion['Inversion_stock_media'].sum()
    lista_Coste_Almacenaje.append(Valor_Coste_Almacenaje)
    lista_Inversion_Almacenaje.append(Valor_Inversion_Almacenaje)

#Politica_1
    if (TablaCostes_Minimos_b.loc[TablaCostes_Minimos_b.index ==
                                  lista_indices_TablaCostes_Minimos_b[x],'Politica_Ratio_Mayor'] == 'Politica_1').any():
        TablaReduccion_Inversion[TablaReduccion_Inversion.index ==
                                 lista_indices_TablaCostes_Minimos_b[x]] = TablaCostes_1[TablaCostes_1.index ==
                                                                                         lista_indices_TablaCostes_Minimos_b[x]]
        TablaReduccion_Inversion.loc[TablaReduccion_Inversion.index ==
                                     lista_indices_TablaCostes_Minimos_b[x],'Politica_Ganadora'] = 'Politica_1'
    else:
        pass
```


Autor: Adrián Calleja Visiedo
425.18.94

```
32
33  #Politica_2
34      if (TablaCostes_Minimos_b.loc[TablaCostes_Minimos_b.index ==
35                                    lista_indices_TablaCostes_Minimos_b[x],'Politica_Ratio_Mayor'] == 'Politica_2').any():
36          TablaReduccion_Inversion[TablaReduccion_Inversion.index ==
37                                   lista_indices_TablaCostes_Minimos_b[x]] = TablaCostes_2[TablaCostes_2.index ==
38                                                                           lista_indices_TablaCostes_Minimos_b[x]]
39          TablaReduccion_Inversion.loc[TablaReduccion_Inversion.index ==
40                                       lista_indices_TablaCostes_Minimos_b[x],'Politica_Ganadora'] = 'Politica_2'
41      else:
42          pass
43
44  #Politica_3
45      if (TablaCostes_Minimos_b.loc[TablaCostes_Minimos_b.index ==
46                                    lista_indices_TablaCostes_Minimos_b[x],'Politica_Ratio_Mayor'] == 'Politica_3').any():
47          TablaReduccion_Inversion[TablaReduccion_Inversion.index ==
48                                   lista_indices_TablaCostes_Minimos_b[x]] = TablaCostes_3[TablaCostes_3.index ==
49                                                                           lista_indices_TablaCostes_Minimos_b[x]]
50          TablaReduccion_Inversion.loc[TablaReduccion_Inversion.index ==
51                                       lista_indices_TablaCostes_Minimos_b[x],'Politica_Ganadora'] = 'Politica_3'
52      else:
53          pass
54
```

```
55  #Politica_4
56      if (TablaCostes_Minimos_b.loc[TablaCostes_Minimos_b.index ==
57                                    lista_indices_TablaCostes_Minimos_b[x],'Politica_Ratio_Mayor'] == 'Politica_4').any():
58          TablaReduccion_Inversion[TablaReduccion_Inversion.index ==
59                                   lista_indices_TablaCostes_Minimos_b[x]] = TablaCostes_4[TablaCostes_4.index ==
60                                                                           lista_indices_TablaCostes_Minimos_b[x]]
61          TablaReduccion_Inversion.loc[TablaReduccion_Inversion.index ==
62                                       lista_indices_TablaCostes_Minimos_b[x],'Politica_Ganadora'] = 'Politica_4'
63      else:
64          pass
65
66
67  TablaComparacion_Coste_Inversion = pd.DataFrame()
68  TablaComparacion_Coste_Inversion['Coste'] = lista_Coste_Almacenaje
69  TablaComparacion_Coste_Inversion['Inversion'] = lista_Inversion_Almacenaje
70  TablaComparacion_Coste_Inversion
```

Relación de documentos

(_) Memoria ..... 45 páginas

(X) Anexos ..... 11 páginas

La Almunia, a 28 del 11 de 2018

Firmado: Adrián Calleja Visiedo