



**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)	1
1.1. DATOS DE ENTRADA	1
1.2. FILTRACIÓN DE DATOS Y PRIMEROS CÁLCULOS	1
1.3. ANÁLISIS DE PARETO	3
1.4. PARÁMETROS DE CONTROL	3
1.5. SIMULACIÓN DEMANDA Y LEAD TIME	4
1.6. SIMULACIÓN POLÍTICAS	5
1.7. COSTES DE LAS POLÍTICAS	7
1.8. REDUCCIÓN DE INVERSIÓN	9

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.

```
1 import pandas as pd
2 import numpy as np
3 import math
4 from scipy.stats import norm
5 import random
6 import matplotlib.pyplot as plt
7 from scipy import stats
8
9 file = pd.ExcelFile('C:/Users/AdrianCalleja/Datos.xlsx')
10
11 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.

```
1 df1 = file.parse('Ventas')
2 df1 = df1.round()
3
4 df1_pivot = df1.pivot(index = 'idSecuencia', columns = 'idProducto', values = 'udsDemanda')
5
6 df1_pivot_perc = pd.DataFrame()
7 df1_pivot_perc = df1_pivot[:]
8
9 df1_pivot_perc_sin_nan = pd.DataFrame()
10 df1_pivot_perc_sin_nan = df1_pivot_perc[:]
11
12 for x in range(0, len(df1_pivot.columns)):
13     Datos_antes_percentiles = df1_pivot.iloc[:, x]
14
15     Promedio_prueba = Datos_antes_percentiles.sum()/len(Datos_antes_percentiles[Datos_antes_percentiles > 0])
16     Std_prueba = Datos_antes_percentiles[Datos_antes_percentiles > 0].std()
17
18     max1 = Datos_antes_percentiles.max()
19
20     perc_inf = np.percentile(Datos_antes_percentiles, 0)
21     perc_sup = np.percentile(Datos_antes_percentiles, 95)
22
23     Datos_despues_percentiles = Datos_antes_percentiles[(Datos_antes_percentiles >= perc_inf) & (Datos_antes_percentiles
24                                                         <= perc_sup)]
25
26     max2 = Datos_despues_percentiles.max()
27
28     df1_pivot_perc.iloc[:, x] = Datos_despues_percentiles
29
30     Promedio_prueba_perc = (Datos_despues_percentiles.sum()/
31                             len(Datos_despues_percentiles[Datos_despues_percentiles > 0])).round()
32     Std_prueba_perc = Datos_despues_percentiles[Datos_despues_percentiles > 0].std()
33
34     df1_pivot_perc_sin_nan.iloc[:, x] = df1_pivot_percSinNaN.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
69 DemandaMediaArticuloPorPedido = SumDemandaTotalArticulo/CountDemandaTotalArticulo
70
71 StdDemandaMediaArticuloPorPedido = ReconstruccionPedidos[ReconstruccionPedidos.udsDemanda >
72                                                         0].groupby(['idProducto']).std()
73 StdDemandaMediaArticuloPorPedido = StdDemandaMediaArticuloPorPedido.drop(['idSecuencia'], axis=1)
74
75 Tabla1 = pd.merge(SumDemandaTotalArticulo, CountDemandaTotalArticulo, how = 'outer', on = 'idProducto')
76 Tabla1.columns = ['Demanda_Total', 'N pedidos']
77
78 Tabla1 = pd.merge(Tabla1, DemandaMediaArticuloPorPedido, how = 'outer', on = 'idProducto')
79 Tabla1.columns = ['Demanda_Total', 'N pedidos', 'Demanda Promedio']
80
81 Tabla1 = pd.merge(Tabla1, StdDemandaMediaArticuloPorPedido, how = 'outer', on = 'idProducto')
82 Tabla1.columns = ['Demanda_Total', 'N pedidos', 'Demanda Promedio', 'Desv Demanda']
83
84 df2 = file.parse('MaestroProductos')
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.

```

97 i = 0
98 valor = Tabla2['Valor_%'].head(i).sum()
99 Limite_Rank_A = 80
100 Limite_Rank_B = 95
101 Limite_Rank_C = 100
102
103 while valor < Limite_Rank_C:
104     i = (i + 1)
105     valor = Tabla2['Valor_%'].head(i).sum()
106     Rank_C = i
107     while valor < Limite_Rank_B:
108         i = (i + 1)
109         valor = Tabla2['Valor_%'].head(i).sum()
110         Rank_B = i
111         while valor < Limite_Rank_A:
112             i = (i + 1)
113             valor = Tabla2['Valor_%'].head(i).sum()
114             Rank_A = i
115
116 Tabla2.reset_index(level=0, inplace=True)
117
118 Tabla2['Rank_ABC'] = 0
119 Tabla2.loc[0:Rank_A, 'Rank_ABC'] = 'A'
120 Tabla2.loc[Rank_A:Rank_B, 'Rank_ABC'] = 'B'
121 Tabla2.loc[Rank_B:Rank_C, 'Rank_ABC'] = 'C'
122
123 Tabla2['N_Servicio'] = 0
124 Tabla2.loc[0:Rank_A, 'N_Servicio'] = 99
125 Tabla2.loc[Rank_A:Rank_B, 'N_Servicio'] = 90
126 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.

```

129 Tabla2['Phi'] = norm.ppf(Tabla2.N_Servicio/100)
130 #Contar numero de entradas
131 N_fechas_distintas = int(len(set(df1.idSecuencia)))
132 N_articulos_distintos = int(len(set(df1.idProducto)))
133 Factor_Dias = Tabla2.N_pedidos/N_fechas_distintas
134
135 Tabla2['EOQ'] = ((2 * Tabla2.Demanda_Total * Tabla2.Ce) / (Tabla2.Cm))**(1/2)
136 Tabla2['EOQ'] = Tabla2['EOQ'].apply(np.round)
137
138 Tabla2['Pp'] = (Tabla2.Demanda_Promedio*Factor_Dias) * Tabla2.Lt + (Tabla2.Phi * (Tabla2.Lt * (Tabla2.Desv_Demanda**2) +
139     (((Tabla2.Demanda_Promedio*Factor_Dias)**2)*(Tabla2.dLt**2))**(1/2))
140 Tabla2['Pp'] = Tabla2['Pp'].apply(np.round)
141 Tabla2['Tp'] = round(360/(((Tabla2.Demanda_Total * (Tabla2.N_pedidos/int(len(set(df1.idSecuencia)))))/Tabla2.EOQ))
142
143
144 Tabla2['Stock_de_Ciclo'] = (Tabla2.Demanda_Promedio*Factor_Dias) * (Tabla2.Lt + Tabla2.Tp)
145 Tabla2['Stock_de_Ciclo'] = Tabla2['Stock_de_Ciclo'].apply(np.round)
146 Tabla2['Stock_de_Ciclo_prueba'] = (Tabla2.Demanda_Promedio*Factor_Dias) * (Tabla2.Lt)
147 Tabla2['Stock_de_Ciclo_prueba'] = Tabla2['Stock_de_Ciclo_prueba'].apply(np.round)
148
149 Tabla2['Stock_de_Seguridad'] = Tabla2.Phi * (((Tabla2.Lt + Tabla2.Tp) * (Tabla2.Desv_Demanda**2) +
150     (((Tabla2.Demanda_Promedio*Factor_Dias)**2)*(Tabla2.dLt**2))**(1/2))
151 Tabla2['Stock_de_Seguridad'] = Tabla2['Stock_de_Seguridad'].apply(np.round)
152
153 Tabla2['StockObjetivo'] = Tabla2.Stock_de_Ciclo + Tabla2.Stock_de_Seguridad
154 Tabla2['StockObjetivo'] = Tabla2['StockObjetivo'].apply(np.round)
155 Tabla2['StockObjetivo_prueba'] = Tabla2.Stock_de_Ciclo_prueba + Tabla2.Stock_de_Seguridad
156 Tabla2['StockObjetivo_prueba'] = Tabla2['StockObjetivo_prueba'].apply(np.round)

```

Se preparan algunas tablas para registrar valores durante las simulaciones.

```
159 TablaIndice = pd.DataFrame(Tabla2.idProducto)
160 N_articulos_distintos = int(len(set(TablaIndice.idProducto)))
161
162 TablaLt = pd.DataFrame()
163
164 TablaDemanda_1 = pd.DataFrame()
165 TablaDemanda_2 = pd.DataFrame()
166 TablaDemanda_3 = pd.DataFrame()
167 TablaDemanda_4 = pd.DataFrame()
168
169 TablaStock_1 = pd.DataFrame()
170 TablaStock_2 = pd.DataFrame()
171 TablaStock_3 = pd.DataFrame()
172 TablaStock_4 = pd.DataFrame()
173
174 TablaReabast_1 = pd.DataFrame()
175 TablaReabast_2 = pd.DataFrame()
176 TablaReabast_3 = pd.DataFrame()
177 TablaReabast_4 = pd.DataFrame()
178
179 TablaRoturas_1 = pd.DataFrame()
180 TablaRoturas_2 = pd.DataFrame()
181 TablaRoturas_3 = pd.DataFrame()
182 TablaRoturas_4 = pd.DataFrame()
---
```

1.5. SIMULACIÓN DEMANDA Y LEAD TIME

Y comienza la simulación de la demanda y lead time.

```
185 Rango_fin_bucles = N_articulos_distintos - 1
186
187 #SIMULAR DEMANDA y Lt
188 TablaDemandas = pd.DataFrame()
189 TablaLt = pd.DataFrame()
190
191 for x in range(0, Rango_fin_bucles):
192     print ("Simulación del articulo ", x)
193     Articulo = TablaIndice['idProducto'][x]
194     mu = Tabla2['Demanda_Promedio'][x]
195     sigma = Tabla2['Desv_Demanda'][x]
196     Lt = Tabla2['Lt'][x]
197     dLt = Tabla2['dLt'][x]
198     print ("Articulo: ", Articulo)
199
200     N_fechas_distintas = int(len(set(df1.idSecuencia)))
201     Factor_Dia = Tabla2['N_pedidos'][x]/N_fechas_distintas
202     print("El factor dia es: ", Factor_Dia)
203
204     contador_veces_demanda = 0
205
206     lista_demanda = []
207     lista_Lt = []
208
```



```

208
209     for y in range(0, 360):
210         random_number = random.random()
211
212         if Factor_Dia > random_number:
213             Simulacion_demanda = stats.expon.rvs(scale=mu, size=1)
214             Simulacion_demanda = Simulacion_demanda.round()
215             while Simulacion_demanda <= 0:
216                 Simulacion_demanda = (np.random.normal(mu, sigma, 1))
217                 Simulacion_demanda = Simulacion_demanda.round()
218
219             lista_demanda.append(Simulacion_demanda)
220
221         else:
222             lista_demanda.append(0)
223
224         #Lead time
225         Simulacion_Lt = (np.random.normal(Lt, dLt, 1))
226         Simulacion_Lt = Simulacion_Lt.round()
227         while Simulacion_Lt <= 0:
228             Simulacion_Lt = (np.random.normal(Lt, dLt, 1))
229             Simulacion_Lt = Simulacion_Lt.round()
230
231         lista_Lt.append(Simulacion_Lt)
232
233     TablaDemandas[Articulo] = lista_demanda
234
235     TablaLt[Articulo] = lista_Lt
236
237
238     TablaDemandas = TablaDemandas.astype(int)
239     TablaLt = TablaLt.astype(int)

```

1.6. SIMULACIÓN POLÍTICAS

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

```

241 #=====
242 #=====METODO STOCK OBJETIVO=====
243 #=====
244 i = 0
245 for x in range(0, Rango_fin_bucles):
246     print("Simulacion del articulo ", x)
247     Articulo = TablaIndice['idProducto'][i]
248     mu = Tabla2['Demanda_Promedio'][i]
249     sigma = Tabla2['Desv_Demanda'][i]
250     Lt = Tabla2['Lt'][i]
251     dLt = Tabla2['dLt'][i]
252     print("Articulo: ", Articulo)
253     print("mu es: ", mu)
254     print("sigma es: ", sigma)
255     print("el Lt es: ", Lt)
256     print("la dLt es: ", dLt)
257     Stock_objetivo = Tabla2['StockObjetivo'][i]
258     Stock_disponible = Stock_objetivo
259     Stock_virtual = Stock_objetivo
260     print("El stock objetivo es: ", Stock_objetivo)
261     N_fechas_distintas = int(len(set(df1.idSecuencia)))
262     Factor_Dia = Tabla2['N_pedidos'][i]/N_fechas_distintas
263     print("El factor dia es: ", Factor_Dia)
264     contador_veces_demanda = 0
265     contador_dias = 0
266     Tp = Tabla2['Tp'][i]
267     print("El tiempo entre pedidos es: ", Tp)
268     Esperando_reabast = 0
269
270     dia_emision_orden_reabast = 0
271     Simulacion_Lt_dia_pedido = 0
272     orden_reabast_emitida = 0
273     recepciones_pendientes = 0

```

```
274
275 lista_stock = []
276 lista_reabast = []
277 lista_lt = []
278 lista_y = []
279 lista_recep = []
280 lista_reab = []
281 lista_dia_reab = []
282 lista_roturas_stock = []
283 lista_demanda = []
284
285 for y in range(0, 360):
286     random_number = random.random()
287
288     Simulacion_demanda = TablaDemandas[Articulo][y]
289     Simulacion_Lt = TablaLt[Articulo][y]
290
291     if Simulacion_demanda > 0:
292         lista_demanda.append(Simulacion_demanda)
293         Stock_disponible = (Stock_disponible - Simulacion_demanda)
294         Stock_virtual = Stock_virtual - Simulacion_demanda
295         contador_veces_demanda = contador_veces_demanda + 1
296
297         if Esperando_reabast == 1:
298             for z in range(0, len(lista_dia_reab) + 5):
299                 if (len(lista_dia_reab) > 0):
300                     zz = 0
301                     while zz < len(lista_dia_reab):
302                         if (contador_dias == lista_dia_reab[zz]):
303                             Stock_disponible = Stock_disponible + lista_reab[0]
304                             recepciones_pendientes = recepciones_pendientes - lista_reab[0]
305                             if recepciones_pendientes == 0:
306                                 Esperando_reabast = 0
307
308                                 lista_reab.pop(0)
309                                 lista_dia_reab.pop(0)
310
311                             else:
312                                 zz = zz + 1
313
314             if (contador_dias % Tp == 0 and Stock_disponible < Stock_objetivo and contador_dias != 0):
315
316                 orden_reabast_emitida = Stock_objetivo - Stock_virtual
317                 Stock_virtual = Stock_virtual + orden_reabast_emitida
318
319                 Esperando_reabast = 1
320                 recepciones_pendientes = recepciones_pendientes + orden_reabast_emitida
321
322                 dia_emision_orden_reabast = contador_dias
323                 Simulacion_Lt_dia_pedido = Simulacion_Lt
324
325                 if (orden_reabast_emitida != 0):
326                     lista_reab.append(orden_reabast_emitida)
327                     lista_dia_reab.append(dia_emision_orden_reabast + Simulacion_Lt_dia_pedido)
328
329             if Stock_disponible < 0 and lista_stock[-1] < 0:
330                 if Stock_disponible < lista_stock[-1]:
331                     lista_roturas_stock.append(Stock_disponible - lista_stock[-1])
332                 else:
333                     lista_roturas_stock.append(0)
334             elif Stock_disponible < 0 and lista_stock[-1] >= 0:
335                 lista_roturas_stock.append(Stock_disponible)
336             else:
337                 lista_roturas_stock.append(0)
338
339
340     else:
341         lista_roturas_stock.append(0)
342         lista_demanda.append(0)
343
344         if Esperando_reabast == 1:
345             for z in range(0, len(lista_dia_reab) + 5):
346                 if (len(lista_dia_reab) > 0):
347                     zz = 0
348                     while zz < len(lista_dia_reab):
349                         if (contador_dias == lista_dia_reab[zz]):
350                             Stock_disponible = Stock_disponible + lista_reab[0]
351                             recepciones_pendientes = recepciones_pendientes - lista_reab[0]
352                             if recepciones_pendientes == 0:
353                                 Esperando_reabast = 0
354
355                                 lista_reab.pop(0)
356                                 lista_dia_reab.pop(0)
357
358                             else:
359                                 zz = zz + 1
```

```

363         orden_reabast_emitida = Stock_objetivo - Stock_virtual
364         Stock_virtual = Stock_virtual + orden_reabast_emitida
365         Esperando_reabast = 1
366         recepciones_pendientes = recepciones_pendientes + orden_reabast_emitida
367         dia_emision_orden_reabast = contador_dias
368         Simulacion_Lt_dia_pedido = Simulacion_Lt
369
370         if (orden_reabast_emitida != 0):
371             lista_reab.append(orden_reabast_emitida)
372             lista_dia_reab.append(dia_emision_orden_reabast + Simulacion_Lt_dia_pedido)
373
374         lista_stock.append(Stock_disponible)
375
376         if (contador_dias % Tp == 0 and Stock_disponible < Stock_objetivo):
377             lista_reabast.append(orden_reabast_emitida)
378         else:
379             lista_reabast.append(0)
380
381         lista_lt.append(Simulacion_Lt)
382         lista_y.append(y)
383         contador_dias = contador_dias + 1
384
385         i = (i + 1)
386         print ("i es: ", i)
387
388         TablaStock_1[Articulo] = lista_stock
389         TablaReabast_1[Articulo] = lista_reabast
390         TablaRoturas_1[Articulo] = lista_roturas_stock
391         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

```

1166 #=====TABLACOSTES_1=====
1167 TablaDemanda_1 = TablaDemanda_1.astype(int)
1168 TablaStock_1 = TablaStock_1.astype(int)
1169 TablaReabast_1 = TablaReabast_1.astype(int)
1170 TablaLt = TablaLt.astype(int)
1171
1172 TablaCostes_1 = pd.DataFrame()
1173 Tabla2.set_index('idProducto', inplace = True)
1174 TablaCostes_1['Media_stock'] = TablaStock_1.mean()
1175 TablaCostes_1 = TablaCostes_1.astype(int)
1176 TablaCostes_1['Cm'] = Tabla2.Cm
1177 TablaCostes_1['Cm_Total'] = TablaCostes_1.Media_stock * TablaCostes_1.Cm
1178 CountReabastArticulo = TablaReabast_1[TablaReabast_1 > 0].count()
1179 TablaCostes_1['N_reabast'] = CountReabastArticulo
1180 TablaCostes_1['Ce'] = Tabla2.Ce
1181 TablaCostes_1['Ce_Total'] = TablaCostes_1.N_reabast * TablaCostes_1.Ce
1182
1183 TablaRoturas_1 = TablaRoturas_1.astype(int)
1184 CountStockNegativo2 = TablaRoturas_1[TablaRoturas_1 < 0].count()
1185 SumStockNegativo2 = TablaRoturas_1[TablaRoturas_1 < 0].sum()
1186
1187 TablaCostes_1['Roturas_stock_u'] = (SumStockNegativo2) * (-1)
1188 TablaCostes_1['Cs'] = Tabla2.Cs
1189 TablaCostes_1['Cs_Total'] = TablaCostes_1.Roturas_stock_u * TablaCostes_1.Cs
1190 TablaCostes_1['Suma_Coste_Total'] = TablaCostes_1.Cm_Total + TablaCostes_1.Ce_Total + TablaCostes_1.Cs_Total
1191 TablaCostes_1['Precio'] = Tabla2.Precio
1192 TablaCostes_1['Inversion_stock_media'] = TablaCostes_1.Media_stock * TablaCostes_1.Precio
1193 TablaCostes_1 = TablaCostes_1.round(2)
1194
1195 CosteEmisionInventario = TablaCostes_1['Ce_Total'].sum()
1196 CosteMantenimientoInventario = TablaCostes_1['Cm_Total'].sum()
1197 CosteRupturasInventario = TablaCostes_1['Cs_Total'].sum()
1198 CosteTotalInventario = TablaCostes_1['Suma_Coste_Total'].sum()
1199 InversionInventario = TablaCostes_1['Inversion_stock_media'].sum()

```

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

A continuación, se calcula "TablaCostes_Minimos".

```
1 lista_indices = []
2 lista_indices = TablaCostes_1.index
3
4 TablaCostes_Minimos = pd.DataFrame()
5
6 TablaCostes_Minimos = TablaCostes_1.copy()
7 TablaCostes_Minimos['Politica_Ganadora'] = 'Temporal'
8
9 x = 0
10 for x in range(0, Rango_fin_bucles):
11     #TablaCostes_1
12     if (TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]] < TablaCostes_1['Suma_Coste_Total'][lista_indices[x]]):
13         TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_Minimos[TablaCostes_Minimos.index ==
14                                                                                                     lista_indices[x]]
15         TablaCostes_Minimos['Politica_Ganadora'][lista_indices[x]] = 'Politica_1'
16     else:
17         TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_1[TablaCostes_1.index ==
18                                                                                                     lista_indices[x]]
19         TablaCostes_Minimos['Politica_Ganadora'][lista_indices[x]] = 'Politica_1'
20
21     #TablaCostes_2
22     if (TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]] < TablaCostes_2['Suma_Coste_Total'][lista_indices[x]]):
23         TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_Minimos[TablaCostes_Minimos.index ==
24                                                                                                     lista_indices[x]]
25     else:
26         TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_2[TablaCostes_2.index ==
27                                                                                                     lista_indices[x]]
28         TablaCostes_Minimos['Politica_Ganadora'][lista_indices[x]] = 'Politica_2'
29
30     #TablaCostes_3
31     if (TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]] < TablaCostes_3['Suma_Coste_Total'][lista_indices[x]]):
32         TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_Minimos[TablaCostes_Minimos.index ==
33                                                                                                     lista_indices[x]]
34     else:
35         TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_3[TablaCostes_3.index ==
36                                                                                                     lista_indices[x]]
37         TablaCostes_Minimos['Politica_Ganadora'][lista_indices[x]] = 'Politica_3'
38
39     #TablaCostes_4
40     if (TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]] < TablaCostes_4['Suma_Coste_Total'][lista_indices[x]]):
41         TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_Minimos[TablaCostes_Minimos.index ==
42                                                                                                     lista_indices[x]]
43     else:
44         TablaCostes_Minimos[TablaCostes_Minimos.index == lista_indices[x]] = TablaCostes_4[TablaCostes_4.index ==
45                                                                                                     lista_indices[x]]
46         TablaCostes_Minimos['Politica_Ganadora'][lista_indices[x]] = 'Politica_4'
47
48
49 TablaCostes_Minimos
```

1.8. REDUCCIÓN DE INVERSIÓN

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

```

1  TablaCostes_1['Sobrecoste'] = 'Temporal'
2  TablaCostes_2['Sobrecoste'] = 'Temporal'
3  TablaCostes_3['Sobrecoste'] = 'Temporal'
4  TablaCostes_4['Sobrecoste'] = 'Temporal'
5
6  TablaCostes_1['Ahorro_en_inversion'] = 'Temporal'
7  TablaCostes_2['Ahorro_en_inversion'] = 'Temporal'
8  TablaCostes_3['Ahorro_en_inversion'] = 'Temporal'
9  TablaCostes_4['Ahorro_en_inversion'] = 'Temporal'
10
11 x = 0
12 for x in range(0, Rango_fin_bucles):
13     #TablaCostes_1
14     Sobrecoste = TablaCostes_1['Suma_Coste_Total'][lista_indices[x]] -
15                 TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]]
16     Ahorro_en_inversion = TablaCostes_Minimos['Inversion_stock_media'][lista_indices[x]] -
17                         TablaCostes_1['Inversion_stock_media'][lista_indices[x]]
18
19     TablaCostes_1['Sobrecoste'][lista_indices[x]] = Sobrecoste
20     TablaCostes_1['Ahorro_en_inversion'][lista_indices[x]] = Ahorro_en_inversion
21
22     #TablaCostes_2
23     Sobrecoste = TablaCostes_2['Suma_Coste_Total'][lista_indices[x]] -
24                 TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]]
25     Ahorro_en_inversion = TablaCostes_Minimos['Inversion_stock_media'][lista_indices[x]] -
26                         TablaCostes_2['Inversion_stock_media'][lista_indices[x]]
27
28     TablaCostes_2['Sobrecoste'][lista_indices[x]] = Sobrecoste
29     TablaCostes_2['Ahorro_en_inversion'][lista_indices[x]] = Ahorro_en_inversion
30
31
32     #TablaCostes_3
33     Sobrecoste = TablaCostes_3['Suma_Coste_Total'][lista_indices[x]] -
34                 TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]]
35     Ahorro_en_inversion = TablaCostes_Minimos['Inversion_stock_media'][lista_indices[x]] -
36                         TablaCostes_3['Inversion_stock_media'][lista_indices[x]]
37
38     TablaCostes_3['Sobrecoste'][lista_indices[x]] = Sobrecoste
39     TablaCostes_3['Ahorro_en_inversion'][lista_indices[x]] = Ahorro_en_inversion
40
41     #TablaCostes_4
42     Sobrecoste = TablaCostes_4['Suma_Coste_Total'][lista_indices[x]] -
43                 TablaCostes_Minimos['Suma_Coste_Total'][lista_indices[x]]
44     Ahorro_en_inversion = TablaCostes_Minimos['Inversion_stock_media'][lista_indices[x]] -
45                         TablaCostes_4['Inversion_stock_media'][lista_indices[x]]
46
47     TablaCostes_4['Sobrecoste'][lista_indices[x]] = Sobrecoste
48     TablaCostes_4['Ahorro_en_inversion'][lista_indices[x]] = Ahorro_en_inversion

```

Para después obtener los ratios.

```
1 TablaCostes_1['Ratio_Ahorro_en_inversion_Sobrecoste'] = TablaCostes_1['Ahorro_en_inversion'] / TablaCostes_1['Sobrecoste']
2 TablaCostes_2['Ratio_Ahorro_en_inversion_Sobrecoste'] = TablaCostes_2['Ahorro_en_inversion'] / TablaCostes_2['Sobrecoste']
3 TablaCostes_3['Ratio_Ahorro_en_inversion_Sobrecoste'] = TablaCostes_3['Ahorro_en_inversion'] / TablaCostes_3['Sobrecoste']
4 TablaCostes_4['Ratio_Ahorro_en_inversion_Sobrecoste'] = TablaCostes_4['Ahorro_en_inversion'] / TablaCostes_4['Sobrecoste']
5
6 TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'] = 0.00
7 TablaCostes_Minimos['Politica_Ratio_Mayor'] = 'Actual'
8
9 x = 0
10 for x in range(0, Rango_fin_bucles):
11     #TablaCostes_1
12     if (TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] <
13         TablaCostes_1['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]):
14         TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] =
15             TablaCostes_1['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]
16         TablaCostes_Minimos['Politica_Ratio_Mayor'][lista_indices[x]] = 'Politica_1'
17     else:
18         pass
19
20     #TablaCostes_2
21     if (TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] <
22         TablaCostes_2['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]):
23         TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] =
24             TablaCostes_2['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]
25         TablaCostes_Minimos['Politica_Ratio_Mayor'][lista_indices[x]] = 'Politica_2'
26     else:
27         pass
28
29     #TablaCostes_3
30     if (TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] <
31         TablaCostes_3['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]):
32         TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] =
33             TablaCostes_3['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]
34         TablaCostes_Minimos['Politica_Ratio_Mayor'][lista_indices[x]] = 'Politica_3'
35     else:
36         pass
37
38     #TablaCostes_4
39     if (TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] <
40         TablaCostes_4['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]):
41         TablaCostes_Minimos['Ratio_Ahorro_en_inversion_Mayor'][lista_indices[x]] =
42             TablaCostes_4['Ratio_Ahorro_en_inversion_Sobrecoste'][lista_indices[x]]
43         TablaCostes_Minimos['Politica_Ratio_Mayor'][lista_indices[x]] = 'Politica_4'
44     else:
45         pass
46
```

Y poder compararlos.

```
1 TablaCostes_Minimos_b = pd.DataFrame()
2 TablaCostes_Minimos_b = TablaCostes_Minimos.copy()
3
4 TablaCostes_Minimos_b = TablaCostes_Minimos_b.sort_values('Ratio_Ahorro_en_inversion_Mayor', ascending=False)
5
6 lista_indices_TablaCostes_Minimos_b = []
7 lista_indices_TablaCostes_Minimos_b = TablaCostes_Minimos_b.index
8
9 TablaReduccion_Inversion = pd.DataFrame()
10 TablaReduccion_Inversion = TablaCostes_Minimos_b.copy()
11
12 lista_Coste_Almacenaje = []
13 lista_Inversion_Almacenaje = []
14
15 x = 0
16 for x in range(0, 250):
17     Valor_Coste_Almacenaje = TablaReduccion_Inversion['Suma_Coste_Total'].sum()
18     Valor_Inversion_Almacenaje = TablaReduccion_Inversion['Inversion_stock_medio'].sum()
19     lista_Coste_Almacenaje.append(Valor_Coste_Almacenaje)
20     lista_Inversion_Almacenaje.append(Valor_Inversion_Almacenaje)
21
22     #Politica_1
23     if (TablaCostes_Minimos_b.loc[TablaCostes_Minimos_b.index ==
24         lista_indices_TablaCostes_Minimos_b[x], 'Politica_Ratio_Mayor'] == 'Politica_1').any():
25         TablaReduccion_Inversion[TablaReduccion_Inversion.index ==
26             lista_indices_TablaCostes_Minimos_b[x]] = TablaCostes_1[TablaCostes_1.index ==
27                 lista_indices_TablaCostes_Minimos_b[x]]
28         TablaReduccion_Inversion.loc[TablaReduccion_Inversion.index ==
29             lista_indices_TablaCostes_Minimos_b[x], 'Politica_Ganadora'] = 'Politica_1'
30     else:
31         pass
32
```

```

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