Formulario 3: DBCA - Parte I

Diseño de experimentos

Andrés Felipe Pico Zúñiga

```
import pandas as pd
import numpy as np
import scipy

from statsmodels.stats.multicomp import pairwise_tukeyhsd

data = "silo"
```

Enunciado

Problema 1: En una empresa se tienen varios silos para almacenar leche (tanques de 10,000 litros) Un aspecto crítico para que se conserve la leche es la temperatura de almacenamiento. Se sospecha que en algunos silos hay problemas, por ello durante cinco días se decide registrar la temperatura a cierta hora crítica. La temperatura de una día a otro puede ser una fuente de variabilidad que podría impactar la variabilidad total.

```
df = pd.read_csv(f"./data/{data}.csv")
print(df)
```

```
##      A    B    C    D    E
## 0  3.0  4.0  4.5  2.5  3.5
## 1  2.5  3.0  4.0  2.0  4.0
## 2  2.0  2.0  3.5  3.0  3.5
## 3  3.0  3.5  5.5  3.0  3.0
## 4  2.0  3.5  5.0  2.5  3.0
```

El diseño apropiado para este enunciado es el diseño en bloques (DBCA).

La hipótesis nula es que la temperatura media de almacenamiento es la misma en los silos.

Solución

Variables iniciales

```
alfa = 0.05           # Nivel de significancia
k = len(df.columns)   # Número de tratamientos
n = len(df)           # Número de muestras por tratamiento
N = k * n             # Número total de observaciones
```

Suma de cuadrados

```python
means = df.mean()
total_mean = np.mean(df.values)
sstr = n * sum((means - total_mean)**2)
print(f"Suma de cuadrados: {sstr}")
```

```
## Suma de cuadrados: 12.86
```

Varianza

```python
grados_libertad_tr = k - 1
error = N - k
cmtr = sstr / grados_libertad_tr
print(f"Varianza de los tratamientos: {cmtr}")
```

```
## Varianza de los tratamientos: 3.215
```

F calculada

```python
sse = sum(sum((df[col] - means[col])**2) for col in df.columns)
cme = sse / error
print(f"Cuadrado medio del error: {cme}")
```

```
## Cuadrado medio del error: 0.36
```

```python
f_calculated = cmtr / cme
print(f"F calculada: {f_calculated}")
```

```
## F calculada: 8.930555555555555
```

Valor teórico de F

```python
f_teoric = scipy.stats.f.isf(q=alfa, dfn=(k - 1), dfd=(N - k))
print(f"Valor teórico de F: {f_teoric}")
```

```
## Valor teórico de F: 2.8660814020156584
```

Prueba HSD de Tukey

```python
alfa = 0.001
df_long = df.melt(var_name='Silo', value_name='Temperatura')
tukey_results = pairwise_tukeyhsd(
  endog=df_long['Temperatura'],
  groups=df_long['Silo'],
  alpha=alfa
)

print(tukey_results)
```

```
## Multiple Comparison of Means - Tukey HSD, FWER=0.00
## ====================================================
## group1 group2 meandiff p-adj   lower   upper  reject
## ----------------------------------------------------
##      A      B      0.7 0.3773 -1.1084  2.5084  False
##      A      C      2.0 0.0003  0.1916  3.8084   True
```

```
##      A      D      0.1 0.9988 -1.7084  1.9084  False
##      A      E      0.9 0.1642 -0.9084  2.7084  False
##      B      C      1.3   0.02 -0.5084  3.1084  False
##      B      D     -0.6 0.5254 -2.4084  1.2084  False
##      B      E      0.2 0.9835 -1.6084  2.0084  False
##      C      D     -1.9 0.0006 -3.7084 -0.0916   True
##      C      E     -1.1 0.0605 -2.9084  0.7084  False
##      D      E      0.8 0.2552 -1.0084  2.6084  False
## ---------------------------------------------------
```

```
print("Temperatura promedio por silo:")
```

```
## Temperatura promedio por silo:
```

```
print(df.mean().sort_values(ascending=False))
```

```
## C    4.5
## E    3.4
## B    3.2
## D    2.6
## A    2.5
## dtype: float64
```

Prueba LSD

```
t_critico = scipy.stats.t.ppf(1 - alfa / 2, df=error)
mse = sse / error
lsd = t_critico * np.sqrt(mse * (2 / n))
print(f"Cuadrado Medio del Error (MSE): {mse:.4f}")
```

```
## Cuadrado Medio del Error (MSE): 0.3600
```

```
print(f"Valor t-crítico: {t_critico:.4f}")
```

```
## Valor t-crítico: 3.8495
```

```
print(f"Valor LSD con alfa={alfa}: {lsd:.4f}")
```

```
## Valor LSD con alfa=0.001: 1.4608
```

```
print("-" * 50)
```

```
## --------------------------------------------------
```

```
print("Comparaciones por pares:")
```

```
## Comparaciones por pares:
```

```
group_means = df.mean().to_dict()
groups = list(group_means.keys())

for i in range(len(groups)):
    for j in range(i + 1, len(groups)):
        group1 = groups[i]
        group2 = groups[j]

        mean_diff = abs(group_means[group1] - group_means[group2])

        print(f"\n- {group1} vs. {group2}:")
```

```
        print(f"  Diferencia de medias: {mean_diff:.4f}")

        if mean_diff > lsd:
            print(f"  Resultado: La diferencia es SIGNIFICATIVA (diferencia > LSD)")
        else:
            print(f"  Resultado: La diferencia NO es significativa (diferencia <= LSD)")
```

```
##
## - A vs. B:
##   Diferencia de medias: 0.7000
##   Resultado: La diferencia NO es significativa (diferencia <= LSD)
##
## - A vs. C:
##   Diferencia de medias: 2.0000
##   Resultado: La diferencia es SIGNIFICATIVA (diferencia > LSD)
##
## - A vs. D:
##   Diferencia de medias: 0.1000
##   Resultado: La diferencia NO es significativa (diferencia <= LSD)
##
## - A vs. E:
##   Diferencia de medias: 0.9000
##   Resultado: La diferencia NO es significativa (diferencia <= LSD)
##
## - B vs. C:
##   Diferencia de medias: 1.3000
##   Resultado: La diferencia NO es significativa (diferencia <= LSD)
##
## - B vs. D:
##   Diferencia de medias: 0.6000
##   Resultado: La diferencia NO es significativa (diferencia <= LSD)
##
## - B vs. E:
##   Diferencia de medias: 0.2000
##   Resultado: La diferencia NO es significativa (diferencia <= LSD)
##
## - C vs. D:
##   Diferencia de medias: 1.9000
##   Resultado: La diferencia es SIGNIFICATIVA (diferencia > LSD)
##
## - C vs. E:
##   Diferencia de medias: 1.1000
##   Resultado: La diferencia NO es significativa (diferencia <= LSD)
##
## - D vs. E:
##   Diferencia de medias: 0.8000
##   Resultado: La diferencia NO es significativa (diferencia <= LSD)
```