Formulario 5: DCGL

Diseño de experimentos

Andrés Felipe Pico Zúñiga

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import seaborn as sns
import statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
import statsmodels.formula.api as smf
import scipy.stats as stats
```

Enunciado

Un experimentador estudia los efectos que tienen cinco Formulaciones (A, B, C, D, E) diferentes de la carga propulsora utilizada en los sistemas de expulsión de la tripulación de un avión basado en la rapidez de combustión. Cada formulación se hace con un lote de materia prima (Columnas: 1, 2, 3, 4, 5, 6) diferente que sólo alcanza para probar cinco formulaciones. Además, las formulaciones son preparadas por varios Operadores (filas: 1, 2, 3, 4, 5, 6), y puede haber diferencias sustanciales en las habilidades y experiencia en cada uno de ellos; por tanto, al parecer, hay dos factores perturbadores que serán "calculados en promedio" en dicho diseño: los lotes de materia prima y los operadores. El diseño apropiado para este problema consiste en probar cada formulación exactamente una vez en cada uno de los operadores. Para esto, se parte del diseño de un Cuadrado Latino. Luego de realizar el análisis del experimento de cuadrado latino anterior, se supone que existe un factor adicional, Los montajes de prueba (letras griegas), que podría ser importante. Este factor considera 5 montajes de prueba diferentes denotados por letras griegas. En la siguiente tabla se muestra el diseño cuadrado grecolatino resultante:

(Operador/Lote) 1 2 3 4 5 1 A =24 B =20 C =19 D =24 E =24 2 B =17 C =24 D =30 E =27 A =36 3 C =18 D =38 E =26 A =27 B =21 4 D =26 E =31 A =26 B =23 C =22 5 E =22 A =30 B =20 C =29 D =31

Creación y visualización del dataframe

```
data = {
  'Operador': [ 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5],
  'Lote': [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5],
  'Formulacion': ['A', 'B', 'C', 'D', 'E', 'B', 'C', 'D', 'E', 'A', 'C', 'D', 'E', 'A', 'B', 'D', 'E',
  'Montaje': [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
  'Respuesta': [24, 20, 19, 24, 24, 17, 24, 30, 27, 36, 18, 38, 26, 27, 21, 26, 31, 26, 23, 22, 22, 30,
}

DCGL = pd.DataFrame(data)
```

```
print(DCGL)
```

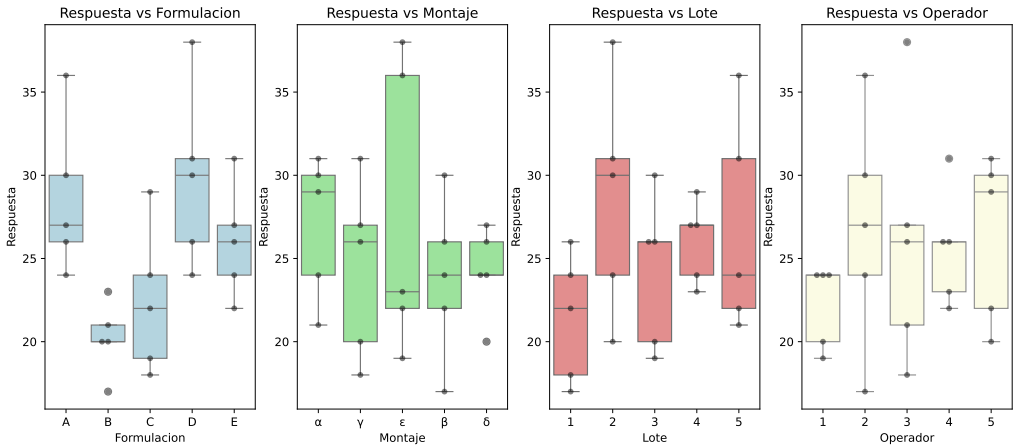
```
##     Operador  Lote Formulacion Montaje  Respuesta
## 0          1     1           A                 24
## 1          1     2           B                 20
## 2          1     3           C                 19
## 3          1     4           D                 24
## 4          1     5           E                 24
## 5          2     1           B                 17
## 6          2     2           C                 24
## 7          2     3           D                 30
## 8          2     4           E                 27
## 9          2     5           A                 36
## 10         3     1           C                 18
## 11         3     2           D                 38
## 12         3     3           E                 26
## 13         3     4           A                 27
## 14         3     5           B                 21
## 15         4     1           D                 26
## 16         4     2           E                 31
## 17         4     3           A                 26
## 18         4     4           B                 23
## 19         4     5           C                 22
## 20         5     1           E                 22
## 21         5     2           A                 30
## 22         5     3           B                 20
## 23         5     4           C                 29
## 24         5     5           D                 31
```

Boxplot de la data

```
fig, axs = plt.subplots(1, 4, figsize=(15, 6))
axs[0].set_title('Respuesta vs Formulacion')

sns.boxplot(
  x="Formulacion",
  y="Respuesta",
  data=DCGL,
  ax=axs[0],
  color='lightblue'
)

sns.swarmplot(
  x="Formulacion",
  y="Respuesta",
  data=DCGL,
  color='black',
  alpha = 0.5,
  ax=axs[0]
)

axs[1].set_title('Respuesta vs Montaje')
```

```
sns.boxplot(
  x="Montaje",
  y="Respuesta",
  data=DCGL,
  ax=axs[1],
  color='lightgreen'
)

sns.swarmplot(
  x="Montaje",
  y="Respuesta",
  data=DCGL,
  color='black',
  alpha = 0.5,
  ax=axs[1]
)

axs[2].set_title('Respuesta vs Lote')

sns.boxplot(
  x="Lote",
  y="Respuesta",
  data=DCGL,
  ax=axs[2],
  color='lightcoral'
)

sns.swarmplot(
  x="Lote",
  y="Respuesta",
  data=DCGL,
  color='black',
  alpha = 0.5,
  ax=axs[2]
)

axs[3].set_title('Respuesta vs Operador')

sns.boxplot(
  x="Operador",
  y="Respuesta",
  data=DCGL,
  ax=axs[3],
  color='lightyellow'
)

sns.swarmplot(
  x="Operador",
  y="Respuesta",
  data=DCGL,
  color='black',
  alpha = 0.5,
  ax=axs[3]
```

```
)
```



Análisis de varianza

```
modelo = ols(
  'Respuesta ~ C(Formulacion) + C(Lote) + C(Operador) + C(Montaje)',
  data=DCGL
).fit()
anova_table = sm.stats.anova_lm(modelo)
print(anova_table)
```

```
##                   df  sum_sq  mean_sq          F    PR(>F)
## C(Formulacion)   4.0   330.0    82.50  10.000000  0.003344
## C(Lote)          4.0   150.0    37.50   4.545455  0.032930
## C(Operador)      4.0    68.0    17.00   2.060606  0.178311
## C(Montaje)       4.0    62.0    15.50   1.878788  0.207641
## Residual         8.0    66.0     8.25        NaN       NaN
```

```
print(modelo.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:              Respuesta   R-squared:                       0.902
## Model:                            OLS   Adj. R-squared:                  0.707
## Method:                 Least Squares   F-statistic:                     4.621
## Date:                sáb, 11 oct 2025   Prob (F-statistic):             0.0171
## Time:                        20:04:56   Log-Likelihood:                -47.608
## No. Observations:                  25   AIC:                             129.2
## Df Residuals:                       8   BIC:                             149.9
## Df Model:                          16
## Covariance Type:            nonrobust
## ===================================================================================
##                       coef    std err          t      P>|t|      [0.025      0.975]
## -----------------------------------------------------------------------------------
## Intercept           23.0000      2.369      9.711      0.000      17.538      28.462
## C(Formulacion)[T.B]  -8.4000      1.817     -4.624      0.002     -12.589      -4.211
## C(Formulacion)[T.C]  -6.2000      1.817     -3.413      0.009     -10.389      -2.011
```

```
## C(Formulacion)[T.D]     1.2000      1.817      0.661      0.527      -2.989       5.389
## C(Formulacion)[T.E]    -2.6000      1.817     -1.431      0.190      -6.789       1.589
## C(Lote)[T.2]            7.2000      1.817      3.963      0.004       3.011      11.389
## C(Lote)[T.3]            2.8000      1.817      1.541      0.162      -1.389       6.989
## C(Lote)[T.4]            4.6000      1.817      2.532      0.035       0.411       8.789
## C(Lote)[T.5]            5.4000      1.817      2.973      0.018       1.211       9.589
## C(Operador)[T.2]        4.6000      1.817      2.532      0.035       0.411       8.789
## C(Operador)[T.3]        3.8000      1.817      2.092      0.070      -0.389       7.989
## C(Operador)[T.4]        3.4000      1.817      1.872      0.098      -0.789       7.589
## C(Operador)[T.5]        4.2000      1.817      2.312      0.050       0.011       8.389
## C(Montaje)[T. ]        -3.2000      1.817     -1.762      0.116      -7.389       0.989
## C(Montaje)[T. ]        -2.6000      1.817     -1.431      0.190      -6.789       1.589
## C(Montaje)[T. ]        -2.8000      1.817     -1.541      0.162      -6.989       1.389
## C(Montaje)[T. ]         0.6000      1.817      0.330      0.750      -3.589       4.789
## ==============================================================================
## Omnibus:                        0.138   Durbin-Watson:                   1.622
## Prob(Omnibus):                  0.933   Jarque-Bera (JB):                0.356
## Skew:                           0.045   Prob(JB):                        0.837
## Kurtosis:                       2.422   Cond. No.                         8.28
## ==============================================================================
##
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Prueba HSD de Tukey

```
# Parámetros
alpha = 0.05
k = len(DCGL['Formulacion'].unique())  # Número de grupos
df_error = 8  # Grados de libertad del error (N-k)
q_critical = stats.studentized_range.ppf(1 - alpha, k, df_error)
HSD = q_critical * np.sqrt(modelo.mse_resid / k)

print(f'Terminos del HSD')
```

```
## Terminos del HSD
```

```
print(f'MSE = {modelo.mse_resid:.2f}')
```

```
## MSE = 8.25
```

```
print(f"El rango studentizado para alpha = {alpha}, k = {k}, df_error = {df_error} es: q_critical = {q_
```

```
## El rango studentizado para alpha = 0.05, k = 5, df_error = 8 es: q_critical = 4.89
```

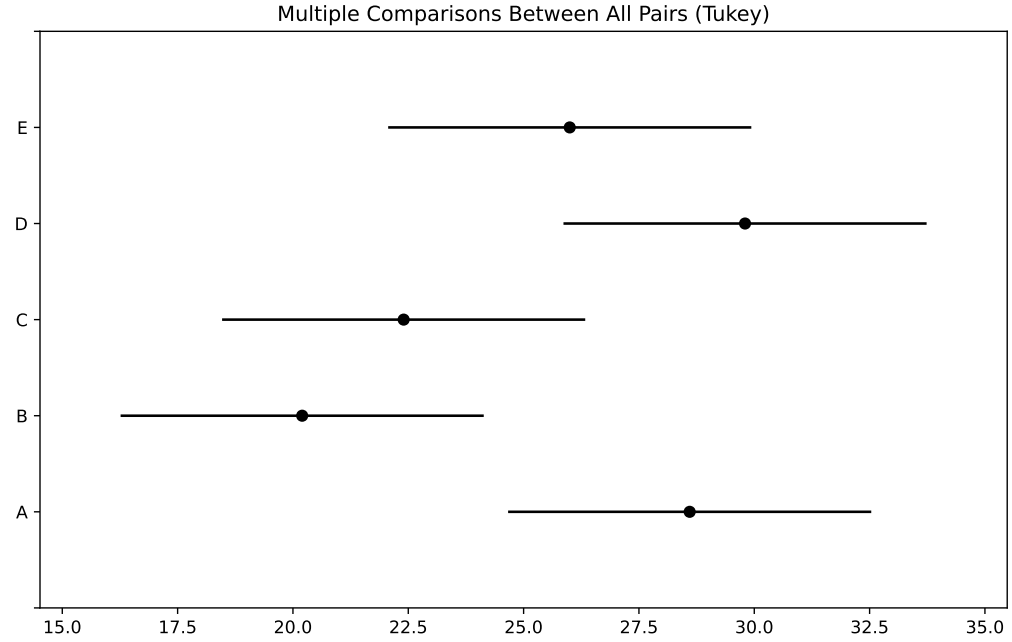
```
print(f'HSD teorico de la hipotesis principal es HSD = {HSD:.2f}')
```

```
## HSD teorico de la hipotesis principal es HSD = 6.28
```

```
def generate_hsd(variable: str):
  tukey = pairwise_tukeyhsd(
    endog=DCGL["Respuesta"],
    groups=DCGL[variable],
    alpha=alpha
  )
```

```
  tukey.plot_simultaneous()
  print(tukey.summary())

generate_hsd("Formulacion")
```

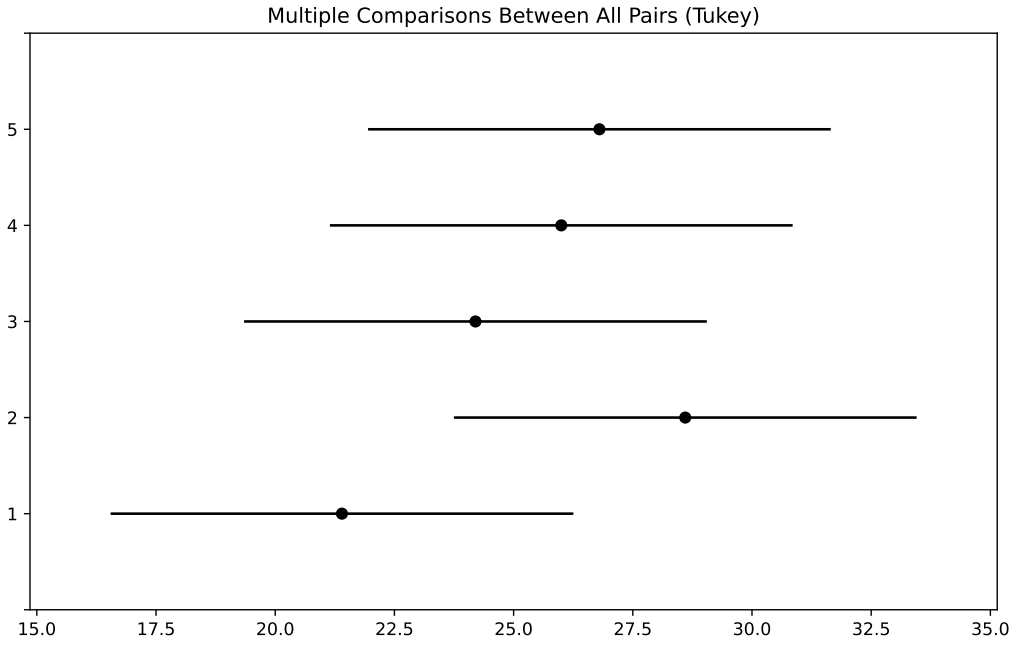
```
##  Multiple Comparison of Means - Tukey HSD, FWER=0.05
## ====================================================
## group1 group2 meandiff p-adj   lower    upper  reject
## ----------------------------------------------------
##      A      B     -8.4 0.0329 -16.2717 -0.5283   True
##      A      C     -6.2 0.1685 -14.0717  1.6717  False
##      A      D      1.2 0.9904  -6.6717  9.0717  False
##      A      E     -2.6 0.8574 -10.4717  5.2717  False
##      B      C      2.2 0.9161  -5.6717 10.0717  False
##      B      D      9.6 0.0123   1.7283 17.4717   True
##      B      E      5.8 0.2182  -2.0717 13.6717  False
##      C      D      7.4 0.0717  -0.4717 15.2717  False
##      C      E      3.6 0.6536  -4.2717 11.4717  False
##      D      E     -3.8 0.6077 -11.6717  4.0717  False
## ----------------------------------------------------
```




```
generate_hsd("Lote")
```

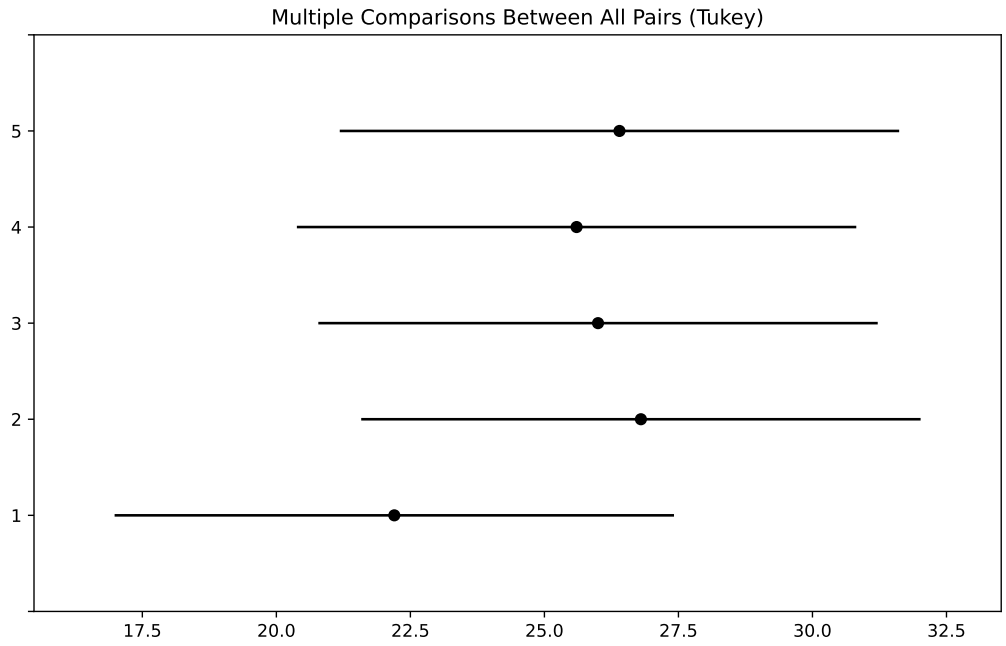
```
##  Multiple Comparison of Means - Tukey HSD, FWER=0.05
## ====================================================
## group1 group2 meandiff p-adj   lower    upper  reject
## ----------------------------------------------------
##      1      2      7.2 0.2129  -2.5056 16.9056  False
##      1      3      2.8 0.9069  -6.9056 12.5056  False
##      1      4      4.6 0.6236  -5.1056 14.3056  False
```

```
##      1      5      5.4 0.4762  -4.3056 15.1056  False
##      2      3     -4.4 0.6607 -14.1056  5.3056  False
##      2      4     -2.6 0.9271 -12.3056  7.1056  False
##      2      5     -1.8   0.98 -11.5056  7.9056  False
##      3      4      1.8   0.98  -7.9056 11.5056  False
##      3      5      2.6 0.9271  -7.1056 12.3056  False
##      4      5      0.8 0.9991  -8.9056 10.5056  False
## ----------------------------------------------------
```



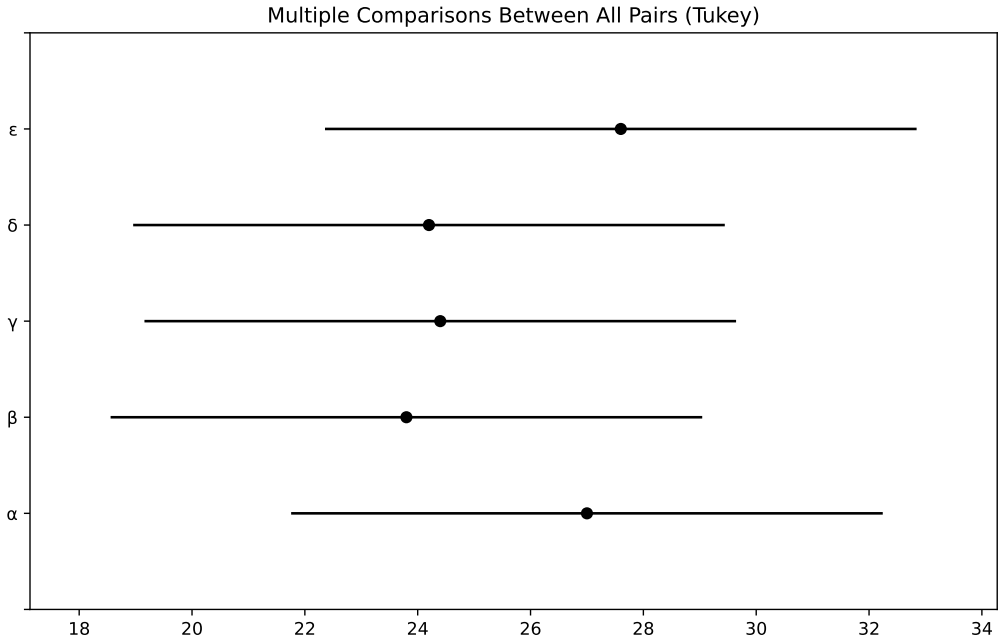

```
generate_hsd("Operador")
```

```
##  Multiple Comparison of Means - Tukey HSD, FWER=0.05
## ====================================================
## group1 group2 meandiff p-adj   lower    upper  reject
## ----------------------------------------------------
##      1      2      4.6  0.683  -5.8348 15.0348  False
##      1      3      3.8 0.8097  -6.6348 14.2348  False
##      1      4      3.4 0.8632  -7.0348 13.8348  False
##      1      5      4.2 0.7489  -6.2348 14.6348  False
##      2      3     -0.8 0.9993 -11.2348  9.6348  False
##      2      4     -1.2 0.9967 -11.6348  9.2348  False
##      2      5     -0.4    1.0 -10.8348 10.0348  False
##      3      4     -0.4    1.0 -10.8348 10.0348  False
##      3      5      0.4    1.0 -10.0348 10.8348  False
##      4      5      0.8 0.9993  -9.6348 11.2348  False
## ----------------------------------------------------
```




```
generate_hsd("Montaje")
```

```
##  Multiple Comparison of Means - Tukey HSD, FWER=0.05
## ====================================================
## group1 group2 meandiff p-adj   lower    upper  reject
## ----------------------------------------------------
##                  -3.2 0.8885 -13.6861  7.2861  False
##                  -2.6 0.9439 -13.0861  7.8861  False
##                  -2.8 0.9278 -13.2861  7.6861  False
##                   0.6 0.9998  -9.8861 11.0861  False
##                   0.6 0.9998  -9.8861 11.0861  False
##                   0.4    1.0 -10.0861 10.8861  False
##                   3.8 0.8123  -6.6861 14.2861  False
##                  -0.2    1.0 -10.6861 10.2861  False
##                   3.2 0.8885  -7.2861 13.6861  False
##                   3.4 0.8653  -7.0861 13.8861  False
## ----------------------------------------------------
```


Multiple Comparisons Between All Pairs (Tukey)
ε
δ
γ
β
α
18
20
22
24
26
28
30
32
34