Formulario 4: DCL

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

Se quiere estudiar el efecto de cinco diferentes catalizadores (A, B, C, D y E) sobre el tiempo de reacción de un proceso químico. Cada lote de material sólo permite cinco corridas y cada corrida requiere aproximadamente 1.5 horas, por lo que sólo se pueden realizar cinco corridas diarias. El experimentador decide correr los experimentos con un diseño en cuadro latino para controlar activamente los lotes y días. Los datos obtenidos son:

		Día				
		1	**2**	**3**	**4**	**5**
Lote	1	A = 8	B = 7	D = 1	C = 7	E = 3
	2	C = 11	E = 2	A = 7	D = 3	B = 8
	3	B = 4	A = 9	C = 10	E = 1	D = 5
	4	D = 6	C = 8	E = 6	B = 6	A = 10
	5	E = 4	D = 2	B = 3	A = 8	C = 8

Creación y visualización del dataframe

```
lote = ["Lote 1", "Lote 2", "Lote 3" , "Lote 4", "Lote 5"]
dia = ["Lunes" , "Martes" , "Miercoles" , "Jueves" , "Viernes"]
catalizador = ["A", "B", "D","C","E", "C", "E", "A","D", "B", "B", "A", "C", "E", "D", "D", "C", "E", "
```

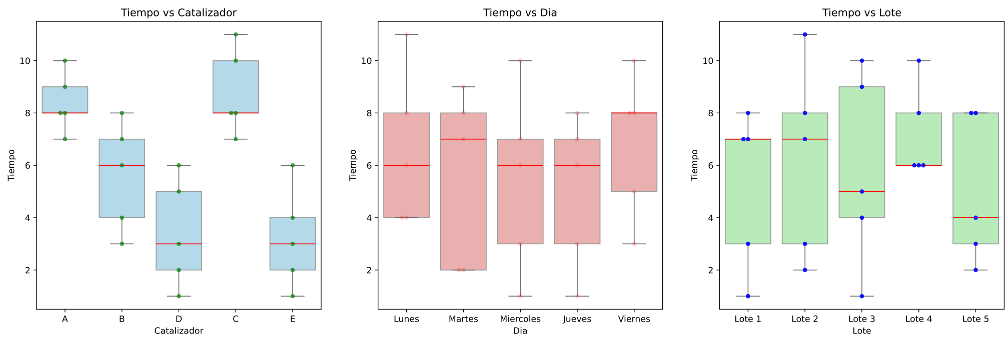
```
tiempo = [8, 7, 1, 7, 3, 11, 2, 7, 3, 8, 4, 9, 10, 1, 5 , 6, 8, 6, 6, 10, 4, 2, 3, 8, 8]

DCL = pd.DataFrame({
  "Lote": np.repeat(["Lote 1" , "Lote 2" , "Lote 3" , "Lote 4", "Lote 5"], 5),
  "Dia": dia * 5,
  "Catalizador": catalizador,
  "Tiempo": tiempo
})

print(DCL)
```

```
##        Lote        Dia Catalizador  Tiempo
## 0   Lote 1      Lunes           A       8
## 1   Lote 1     Martes           B       7
## 2   Lote 1  Miercoles           D       1
## 3   Lote 1     Jueves           C       7
## 4   Lote 1    Viernes           E       3
## 5   Lote 2      Lunes           C      11
## 6   Lote 2     Martes           E       2
## 7   Lote 2  Miercoles           A       7
## 8   Lote 2     Jueves           D       3
## 9   Lote 2    Viernes           B       8
## 10  Lote 3      Lunes           B       4
## 11  Lote 3     Martes           A       9
## 12  Lote 3  Miercoles           C      10
## 13  Lote 3     Jueves           E       1
## 14  Lote 3    Viernes           D       5
## 15  Lote 4      Lunes           D       6
## 16  Lote 4     Martes           C       8
## 17  Lote 4  Miercoles           E       6
## 18  Lote 4     Jueves           B       6
## 19  Lote 4    Viernes           A      10
## 20  Lote 5      Lunes           E       4
## 21  Lote 5     Martes           D       2
## 22  Lote 5  Miercoles           B       3
## 23  Lote 5     Jueves           A       8
## 24  Lote 5    Viernes           C       8
```

Boxplot de la data

```
fig, axs = plt.subplots(1, 3, figsize = (20, 6))
axs[0].set_title("Tiempo vs Catalizador")
sns.boxplot(
  x = "Catalizador",
  y = "Tiempo",
  data = DCL,
  ax = axs[0],
  color = "skyblue",
  boxprops = dict(alpha = .7),
  medianprops = dict(color = "red")
)
sns.swarmplot(
  x = "Catalizador",
```

```
  y = "Tiempo",
  data = DCL,
  color = "green",
  alpha = 0.7,
  ax = axs[0]
)

axs[1].set_title("Tiempo vs Dia")
sns.boxplot(
  x = "Dia",
  y = "Tiempo",
  data = DCL,
  ax = axs[1],
  color = "lightcoral",
  boxprops = dict(alpha = .7),
  medianprops = dict(color = "red")
)
sns.swarmplot(
  x = "Dia",
  y = "Tiempo",
  data = DCL,
  color = "red",
  alpha = 0.2,
  ax = axs[1]
)

axs[2].set_title("Tiempo vs Lote")
sns.boxplot(
  x = "Lote",
  y = "Tiempo",
  data = DCL,
  ax = axs[2],
  color = "lightgreen",
  boxprops = dict(alpha = .7),
  medianprops = dict(color = "red")
)
sns.swarmplot(
  x = "Lote",
  y = "Tiempo",
  data = DCL,
  color = "blue",
  alpha = 0.9,
  ax = axs[2]
)
```



Análisis de varianza

```
modelo = ols(
  "Tiempo ~ Catalizador + Lote + Dia",
  data = DCL
).fit()
anova_table = sm.stats.anova_lm(modelo, typ=1)
print(anova_table)
```

```
##                df  sum_sq    mean_sq          F    PR(>F)
## Catalizador   4.0  141.44  35.360000  11.309168  0.000488
## Lote          4.0   15.44   3.860000   1.234542  0.347618
## Dia           4.0   12.24   3.060000   0.978678  0.455014
## Residual     12.0   37.52   3.126667        NaN       NaN
```

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

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                 Tiempo   R-squared:                       0.818
## Model:                            OLS   Adj. R-squared:                  0.637
## Method:                 Least Squares   F-statistic:                     4.507
## Date:                lun, 13 oct 2025   Prob (F-statistic):            0.00716
## Time:                        21:21:58   Log-Likelihood:                -40.548
## No. Observations:                  25   AIC:                             107.1
## Df Residuals:                      12   BIC:                             122.9
## Df Model:                          12
## Covariance Type:            nonrobust
## ====================================================================================
##                        coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------------
## Intercept            6.8400      1.275      5.364      0.000       4.062       9.618
## Catalizador[T.B]    -2.8000      1.118     -2.504      0.028      -5.237      -0.363
## Catalizador[T.C]     0.4000      1.118      0.358      0.727      -2.037       2.837
## Catalizador[T.D]    -5.0000      1.118     -4.471      0.001      -7.437      -2.563
## Catalizador[T.E]    -5.2000      1.118     -4.650      0.001      -7.637      -2.763
## Lote[T.Lote 2]       1.0000      1.118      0.894      0.389      -1.437       3.437
## Lote[T.Lote 3]       0.6000      1.118      0.537      0.601      -1.837       3.037
## Lote[T.Lote 4]       2.0000      1.118      1.788      0.099      -0.437       4.437
## Lote[T.Lote 5]      -0.2000      1.118     -0.179      0.861      -2.637       2.237
## Dia[T.Lunes]         1.6000      1.118      1.431      0.178      -0.837       4.037
```

```
## Dia[T.Martes]        0.6000      1.118      0.537      0.601      -1.837       3.037
## Dia[T.Miercoles]     0.4000      1.118      0.358      0.727      -2.037       2.837
## Dia[T.Viernes]       1.8000      1.118      1.610      0.133      -0.637       4.237
## ==============================================================================
## Omnibus:                        1.780   Durbin-Watson:                   2.925
## Prob(Omnibus):                  0.411   Jarque-Bera (JB):                1.073
## Skew:                           0.133   Prob(JB):                        0.585
## Kurtosis:                       2.020   Cond. No.                         7.47
## ==============================================================================
##
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Prueba HSD de Tukey

```
# Parámetros
alpha = 0.05
k = 5  # Número de grupos
df_error =  modelo.df_resid  # Grados de libertad del error (N-k)

# Calcular el valor crítico del rango studentizado
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 = 3.13
```

```
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 = 12.0 es: q_critical = 4.51
```

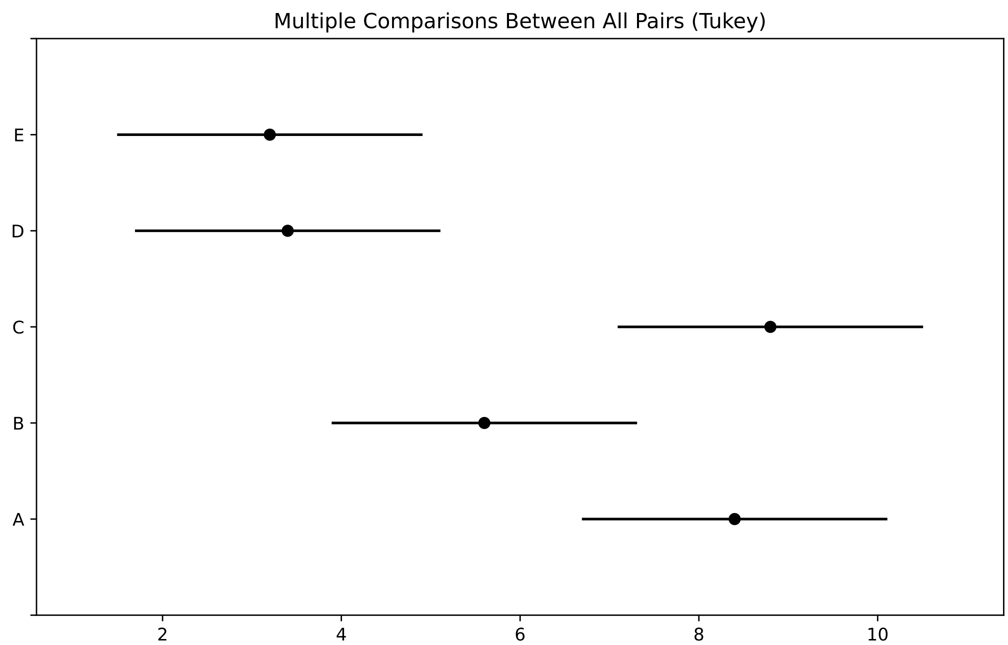
```
print(f'HSD teorico de la hipotesis principal es HSD = {HSD}')
```

```
## HSD teorico de la hipotesis principal es HSD = 3.5646077765231055
```

```
def generate_hsd(variable: str, alpha: float):
  tukey = pairwise_tukeyhsd(
    endog=DCL["Tiempo"],
    groups=DCL[variable],
    alpha=alpha
  )

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

aplha = 0.001

generate_hsd("Catalizador", alpha)
```

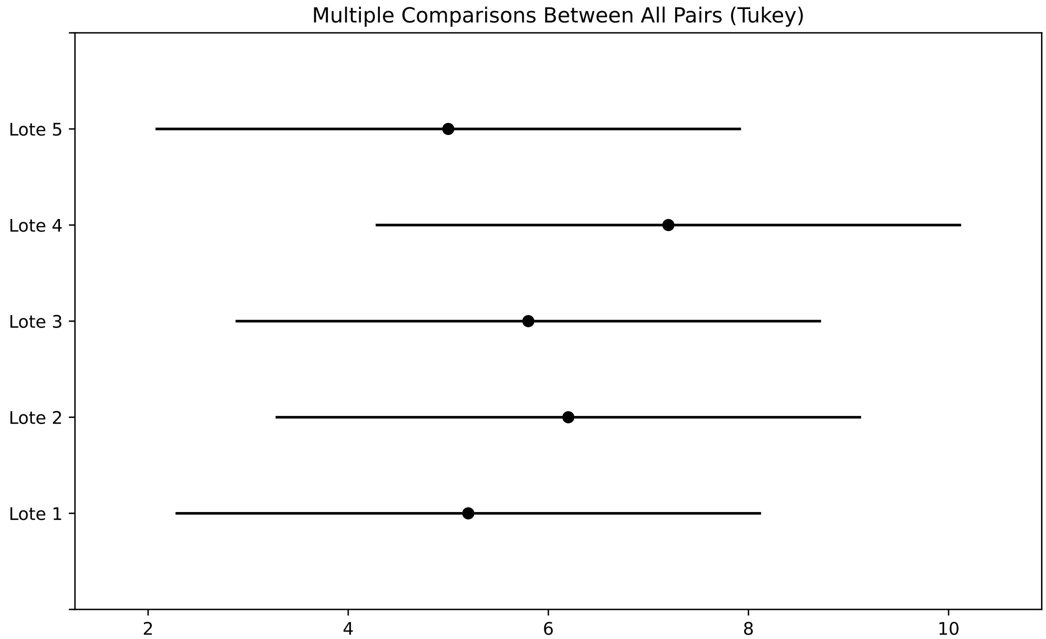
```
## Multiple Comparison of Means - Tukey HSD, FWER=0.05
## ====================================================
```

```
## group1 group2 meandiff p-adj   lower   upper  reject
## ---------------------------------------------------
##      A      B     -2.8 0.1423 -6.2171  0.6171  False
##      A      C      0.4 0.9965 -3.0171  3.8171  False
##      A      D     -5.0 0.0024 -8.4171 -1.5829   True
##      A      E     -5.2 0.0016 -8.6171 -1.7829   True
##      B      C      3.2 0.0733 -0.2171  6.6171  False
##      B      D     -2.2 0.3361 -5.6171  1.2171  False
##      B      E     -2.4 0.2578 -5.8171  1.0171  False
##      C      D     -5.4 0.0011 -8.8171 -1.9829   True
##      C      E     -5.6 0.0007 -9.0171 -2.1829   True
##      D      E     -0.2 0.9998 -3.6171  3.2171  False
## ---------------------------------------------------
```



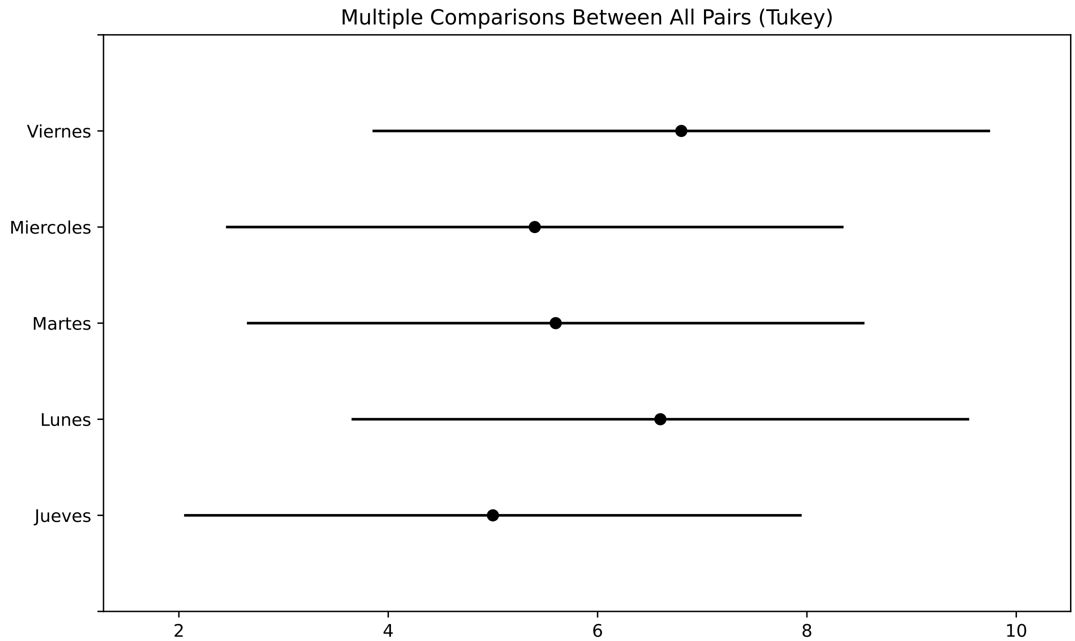

```
generate_hsd("Lote", alpha)
```

```
## Multiple Comparison of Means - Tukey HSD, FWER=0.05
## ==================================================
## group1 group2 meandiff p-adj   lower  upper  reject
## --------------------------------------------------
## Lote 1 Lote 2      1.0 0.9852 -4.8516 6.8516  False
## Lote 1 Lote 3      0.6 0.9979 -5.2516 6.4516  False
## Lote 1 Lote 4      2.0 0.8419 -3.8516 7.8516  False
## Lote 1 Lote 5     -0.2    1.0 -6.0516 5.6516  False
## Lote 2 Lote 3     -0.4 0.9996 -6.2516 5.4516  False
## Lote 2 Lote 4      1.0 0.9852 -4.8516 6.8516  False
## Lote 2 Lote 5     -1.2 0.9712 -7.0516 4.6516  False
## Lote 3 Lote 4      1.4 0.9504 -4.4516 7.2516  False
## Lote 3 Lote 5     -0.8 0.9936 -6.6516 5.0516  False
## Lote 4 Lote 5     -2.2 0.7916 -8.0516 3.6516  False
## --------------------------------------------------
```




```
generate_hsd("Dia", alpha)
```

```
##    Multiple Comparison of Means - Tukey HSD, FWER=0.05
## ==========================================================
##   group1    group2  meandiff p-adj   lower  upper  reject
## ----------------------------------------------------------
##    Jueves     Lunes      1.6  0.924 -4.3004 7.5004  False
##    Jueves    Martes      0.6  0.998 -5.3004 6.5004  False
##    Jueves Miercoles      0.4 0.9996 -5.5004 6.3004  False
##    Jueves   Viernes      1.8 0.8886 -4.1004 7.7004  False
##     Lunes    Martes     -1.0 0.9857 -6.9004 4.9004  False
##     Lunes Miercoles     -1.2 0.9721 -7.1004 4.7004  False
##     Lunes   Viernes      0.2    1.0 -5.7004 6.1004  False
##    Martes Miercoles     -0.2    1.0 -6.1004 5.7004  False
##    Martes   Viernes      1.2 0.9721 -4.7004 7.1004  False
## Miercoles   Viernes      1.4 0.9518 -4.5004 7.3004  False
## ----------------------------------------------------------
```



Prueba LSD

```
# Parámetros
alpha = 0.05
k = 5  # Número de grupos (Catalizador)
N = len(DCL)  # Número total de observaciones
df_error = modelo.df_resid  # Grados de libertad del error (N-k)

# Obtener el valor crítico t para la prueba LSD
t_critical = stats.t.ppf(1 - alpha / 2, df_error)

# Obtener el MSE del modelo
mse = modelo.mse_resid

# Calcular el tamaño de muestra promedio por grupo
n = DCL.groupby('Catalizador')['Tiempo'].count().mean()

# Calcular la LSD
LSD = t_critical * np.sqrt(2 * mse / n)

print(f'Términos de la LSD')
```

```
## Términos de la LSD
```

```
print(f'el valor de n = {n}')
```

```
## el valor de n = 5.0
```

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

```
## MSE = 3.13
```

```
print(f"El valor crítico t para alpha = {alpha/2}, df_error = {df_error} es: t_critical = {t_critical:.
```

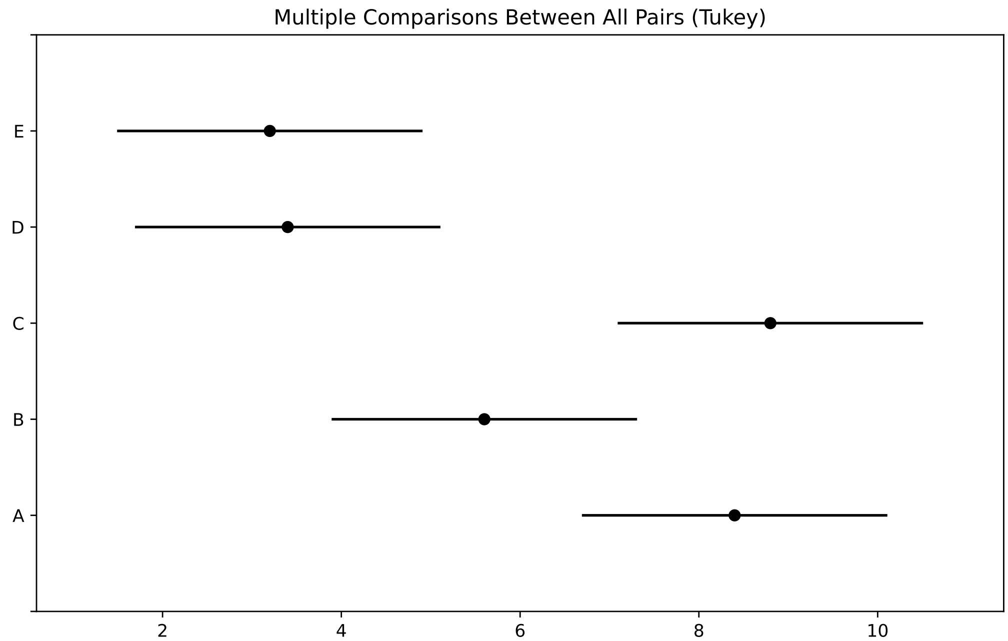
```
## El valor crítico t para alpha = 0.025, df_error = 12.0 es: t_critical = 2.18
```

```
print(f'LSD teórico para la hipótesis principal es LSD = {LSD:.2f}')
```

```
## LSD teórico para la hipótesis principal es LSD = 2.44
```

```
def generate_lsd(variable: str):
  tukey = pairwise_tukeyhsd(
    endog=DCL["Tiempo"],
    groups=DCL[variable],
    alpha=alpha
  )

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

generate_hsd("Catalizador", alpha)
```

```
## Multiple Comparison of Means - Tukey HSD, FWER=0.05
## ====================================================
## group1 group2 meandiff p-adj   lower   upper  reject
## ----------------------------------------------------
##      A      B     -2.8 0.1423 -6.2171  0.6171  False
##      A      C      0.4 0.9965 -3.0171  3.8171  False
##      A      D     -5.0 0.0024 -8.4171 -1.5829   True
##      A      E     -5.2 0.0016 -8.6171 -1.7829   True
##      B      C      3.2 0.0733 -0.2171  6.6171  False
##      B      D     -2.2 0.3361 -5.6171  1.2171  False
##      B      E     -2.4 0.2578 -5.8171  1.0171  False
##      C      D     -5.4 0.0011 -8.8171 -1.9829   True
##      C      E     -5.6 0.0007 -9.0171 -2.1829   True
##      D      E     -0.2 0.9998 -3.6171  3.2171  False
## ----------------------------------------------------
```



Multiple Comparisons Between All Pairs (Tukey)
E
D
C
B
A
2
4
6
8
10