Exercise Set 14

1 [Exam 2025] Thermal conductivities

You measure the thermal conductivity of two types of metals, Platinum (Pt) and Molybdenum (Mo). You can call the "type of metal" factor I. You measure them both at room temperature ("RT") and at liquid nitrogen temperature ("LN"). You can call the "temperature" factor J. Each possible configuration is measured 11 times. You find the following means and unbiased estimators for the standard deviation for your measurements. All thermal conductivities are given in units of W/m/K:

	$\bar{X}_{i,j,\bullet}$	$s_{i,j}$	N_S
Pt,RT	$\bar{X}_{1,1,\bullet} = 67$	$s_{1,1} = 11$	11
Pt,LN	$\bar{X}_{1,2,\bullet} = 83$	$s_{1,2} = 8$	11
Mo,RT	$\bar{X}_{2,1,\bullet} = 139$	$s_{2,1} = 14$	11
Mo,LN	$\bar{X}_{2,2,\bullet} = 211$	$s_{2,2} = 17$	11

{2p}a) Compute the partial/marginal means[1] for both factors and the total mean, and arrange[2] all means in a "table of means."

{2p} b) Compute the sum of the squared errors ($SS_{E,i,j}$) for each group and hence the total SS_E.

{1p} c) Compute the sum of the square difference between levels for each factor ($SS_{B,I}$ and $SS_{B,J}$).

{1p} d) Statistically speaking, which factor is the dominant one?

{5p} e) Create a complete two-factor ANOVA table *including interactions*. (Also include the Fisher statistic for each factor and for the interaction).

{2p} f) Test the null hypothesis "there is no interaction between the factors I and J" using a level of significance of 5%.

Now you focus on studying the dependence of the thermal conductivity, κ, of Platinum as a function of temperature T. You want to use linear regression to compare your data to different models.

{3p} g) For a model given by

$$\kappa = a + b_1 T$$

prove that the estimator for a, which minimizes the sum of the squared errors (SS_E) between the data and the model is given by $\hat{a} = \bar{\kappa} - \hat{b}_1 \bar{T}$ where $\hat{b}_1$ denotes the estimator for b_1

You have now taken data at 7 different temperatures. You fit your data with both models. The first contains only an offset and a linear slope (as above), the second additionally contains a term which is quadratic in temperature and has a pre-factor called b_2. You find the following SS_E for the two models:

[1] moyenne partielle [2] trier

Model	SS_E
$\kappa = a + b_1 T$	1255
$\kappa = a + b_1 T + b_2 T^2$	405

{1p} h) How many degrees of freedom does the error have in each model?

{2p} i) Is the second model statistically significantly better than the first model? Use a level of significance of 5%.

You have now automated your data acquisition with Python and successfully written a code that creates a numpy array called `measuredData`, where each element corresponds to a thermal conductivity.

{1p} j) However, you have to take a calibration factor into account, each measurement needs to be multiplied by 1.2. Which line of code would you write to produce an array called `calibratedData` that now contains data with the correct calibration?

Occasionally your measurement device malfunctions because of electrical spikes, and then produces erroneous outputs which give values larger than 1000. You have performed a measurement and created an array called `calibratedData` with the values $[68.4, 72.1, 1200.2, 80.4]$.

{1p} k) What output do you expect for the following line of code?

```
print(calibratedData[0:2])
```

{2p} l) Which line of code could you write to generate an output that lets you see (even for a large array) if your measurement contains an erroneous element (i.e. a value larger than 1000)? (*There are several good options.*)

{2p} m) To filter out such values and replace them by a `np.nan` ("not a number") value, your colleague has written the following function:

```
def myFilter(inputData):
    if inputData>1000.:
        return np.nan
    else:
        return inputData
```

You try to apply this function to your array by writing:

```
print(myFilter(calibratedData))
```

and you receive the following error message:

```
ValueError: The truth value of an array with more than one element is
ambiguous. Use a.any() or a.all()
```

How can you solve this issue so that you filter your data as desired? (*There are several possible solutions, and you are not forced to use your colleagues' function*).