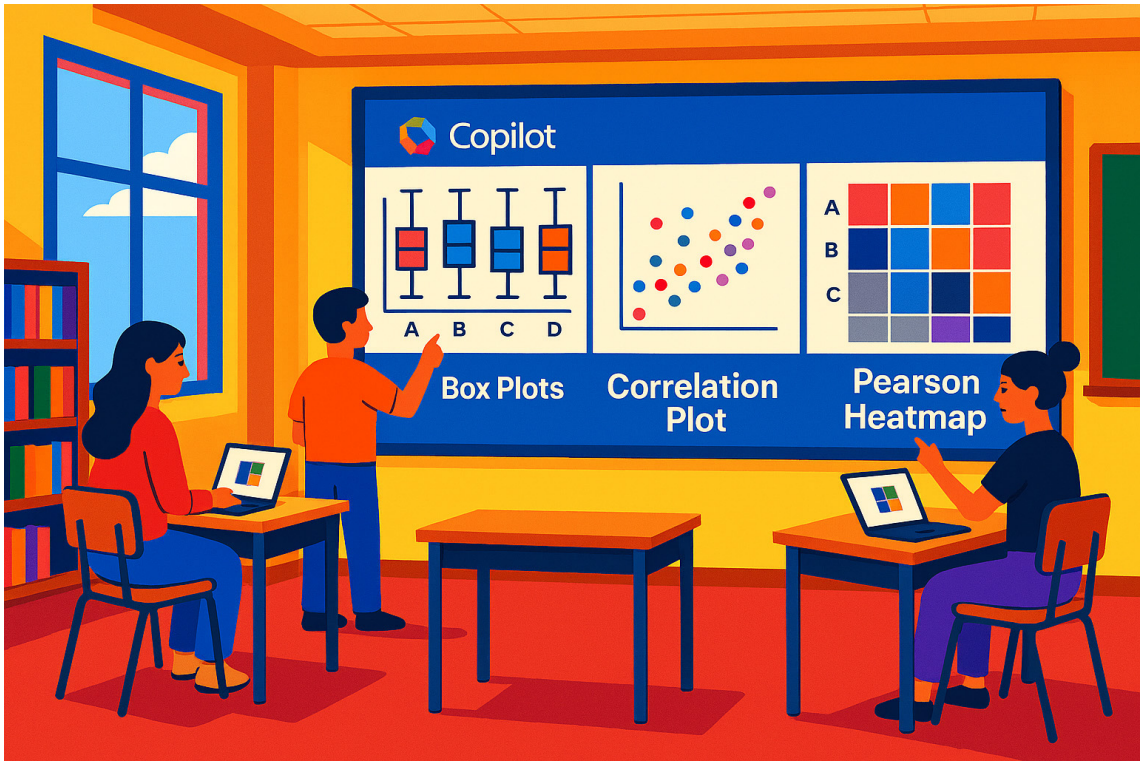
Leveraging Microsoft Copilot (GPT-5) for Calculations and Interactive Data Visualization



This file has the answers provided by Microsoft Copilot (GPT-5) to the questions in Table 1. It also contains the R scripts provided by Ciopilot to reproduce the images. For examples, sections Figure 2, 21 and 22, have the R scripts to reproduce these images, the reader just needs to change the path for the spreadsheets (Table S1 and Table S2)

Question 1: Calculate the amount of ethyl ethanoate formed when 2 moles of ethanoic acid and 3 moles of ethanol and 8 moles of water are allowed to come to equilibrium. The equilibrium constant for the reaction is 3.0.

Forrix Esterification jüibrium: erriuwhen CH_3OH (ethanoic acid) + C_2H_5OH (ethanol) $\Longleftrightarrow$ $CH_3COOC_2H_5$ (ethyl ethanoatc) + H_2O

Given 1: Gen'd: x moles fthanolo acof'med at' ul ethanoate.

Step 1: Express equilibrium concentrations in terms of x

$$\begin{aligned} \text{Ethanoic acid} &= 2-x \\ \text{Ethanol} &= 3-x \\ \text{Water} &= 8+x \\ \text{Ethyl ethanoate} &= x \end{aligned}$$

Step 2: Write equilibrium expression

$$K = \frac{[\text{((eterter]}^{\cdot}][\text{w-ater]}^{\prime}}{[\,[\text{acid]]}^{\prime}[\text{ethanol)]}} = x(8+x)$$

$$= x\,(8+x) = 3$$

Smpilss to: x_1 ($x) = x$.

Bring terms together: $'x^2 - 3x - Ex^2 + 8x - 18 = 0$

Multiplied by -1: $2x^2 - 23x + 18 = 0$

Applying the exprssion: $x = 23 \pm x = 3\,(2^{7\prime} - x_{\prime})\,(33) - 3$

$$x = 23 \pm \sqrt{529 - 144)} = 3.0$$

$$x = 23 \pm \sqrt{385}$$

Two roots x_1 and x_2 ane:

$$x = \frac{3}{4} = \frac{42.62}{4} \approx 10.66 \text{ (not possible, too large)}$$

$$x_2 = \frac{3}{4} = \frac{3.38}{4} \approx 0.845$$

Amount of ethyl ethanoate formed $= 0{,}345$ mol (approximately)

Amount of ethyl ethanoate formed $= 0.845$ mol (approximately)

Question 2: Calculate the pH of 0.025 M HI

HI is a **strong acid**, so it dissociates completely in water:

$$\mathrm{HI} \rightarrow \mathrm{H^+} + \mathrm{I^-}$$

For a strong acid, $[\mathrm{H^+}] = \text{initial concentration of HI}$.

Step 1: Calculate $[H^+]$

$$[\mathrm{H^+}] = 0.025\,\mathrm{M}$$

Step 2: Calculate pH

$$\mathrm{pH} = -\log[\mathrm{H^+}] = -\log(0.025)$$

$$\log(0.025) = \log(25 \times 10^{-3}) = \log(25) - 3 \approx 1.398 - 3 = -1.602$$

So:

pH=1.60

Question 3: Calculate the pH of 0.025 M $Sr(OH)_2$

$Sr(OH)_2$ is a strong base and dissociates completely:

$$Sr(OH)_2 \rightarrow Sr^{2+} + 2OH^-$$

Step 1: Calculate $[OH^-]$

$$[OH^-] = 2 \times 0.025 = 0.050\,M$$

Step 2: Calculate pOH

$$pOH = -\log(0.050) = -\log(5.0 \times 10^{-2}) = 2 - \log(5) \approx 2 - 0.699 = 1.301$$

Step 3: Calculate pH

$$pH = 14 - pOH = 14 - 1.301 = 12.70$$

Final Answer: pH ≈ 12.70

Q4: Calculate the pH of 2.0 M HF. Ka for HF = 6.6×10^{-4}

We have a weak acid (HF), so we need to use its dissociation equilibrium:

$$\mathbf{HF} \rightleftharpoons \mathbf{H^+} + \mathbf{F^-}$$

Given:

- $[HF]_{\text{initial}} = 2.0\,\text{M}$
- $K_a = 6.6 \times 10^{-4}$

Step 1: Set up ICE table

Let $x = [\mathbf{H^+}] = [\mathbf{F^-}]$ at equilibrium.

$$\begin{aligned} \text{Initial: } & [HF] = 2.0,\ [H^+] = 0,\ [F^-] = 0 \\ \text{Change: } & -x,\ +x,\ +x \\ \text{Equilibrium: } & [HF] = 2.0 - x,\ [H^+] = x,\ [F^-] = x \end{aligned}$$

Step 2: Write Ka expression

$$K_a = \frac{x^2}{2.0 - x} \approx \frac{x^2}{2.0} \text{ (since } x \ll 2.0)$$

$$6.6 \times 10^{-4} = \frac{x^2}{2.0}$$

Step 3: Solve for x

$$x^2 = (6.6 \times 10^{-4})(2.0) = 1.32 \times 10^{-3}$$

$$x = \sqrt{1.32 \times 10^{-3}} \approx 0.0363\,\text{M}$$

Step 4: Calculate pH

$$\text{pH} = -\log(0.0363) \approx 1.44$$

✅ Final Answer: pH ≈ 1.44

Question 5: Calculate the pH of 2.0 M CH_3NH_2. Kb for CH_3NH_2 = 4.38 × 10^{-4}

We have a **weak base** (methylamine, CH_3NH_2), so we use its base dissociation equilibrium:

$$CH_3NH_2 + H_2O \rightleftharpoons CH_3NH_3^+ + OH^-$$

Given:

- $[CH_3NH_2]_{initial} = 2.0\,M$
- $K_b = 4.38 \times 10^{-4}$

Step 1: ICE Table

Let $x = [OH^-]$ at equilibrium.

$$\begin{aligned} \text{Initial: } & [CH_3NH_2] = 2.0,\ [OH^-] = 0 \\ \text{Change: } & -x,\ +x \\ \text{Equilibrium: } & [CH_3NH_2] = 2.0 - x,\ [OH^-] = x \end{aligned}$$

Step 2: Write Kb expression

$$K_b = \frac{x^2}{2.0 - x} \approx \frac{x^2}{2.0} \, (\text{since } x \ll 2.0)$$

$$4.38 \times 10^{-4} = \frac{x^2}{2.0}$$

Step 3: Solve for x

$$x^2 = (4.38 \times 10^{-4})(2.0) = 8.76 \times 10^{-4}$$

$$x = \sqrt{8.76 \times 10^{-4}} \approx 0.0296 \, \text{M}$$

So:

$$[\text{OH}^-] \approx 0.0296 \, \text{M}$$

So:

$$[\mathrm{OH^-}] \approx 0.0296\,\mathrm{M}$$

Step 4: Calculate pOH

$$\mathrm{pOH} = -\log(0.0296) \approx 1.53$$

Step 5: Calculate pH

$$\mathrm{pH} = 14 - 1.53 = 12.47$$

✅ Final Answer: pH ≈ 12.47

Question 6: Calculate the pH of 0.25 M NH_4Cl. Kb for NH_3 = 1.8 × 10^{-5}

NH_4Cl is a salt of a weak base (NH_3) and a strong acid (HCl), so its solution is acidic due to hydrolysis of NH_4^+:

$$\mathrm{NH_4^+ + H_2O \rightleftharpoons NH_3 + H_3O^+}$$

Given:

- $[\mathrm{NH_4Cl}] = 0.25\,\mathrm{M}$
- $K_b(\mathrm{NH_3}) = 1.8 \times 10^{-5}$
- $K_w = 1.0 \times 10^{-14}$

Step 1: Calculate Ka for NH_4^+

$$K_a = \frac{K_w}{K_b} = \frac{1.0 \times 10^{-14}}{1.8 \times 10^{-5}} \approx 5.56 \times 10^{-10}$$

Step 2: Set up ICE table

Let $x = [\mathrm{H^+}]$ at equilibrium.

$$K_a = \frac{x^2}{0.25 - x} \approx \frac{x^2}{0.25}$$

$$5.56 \times 10^{-10} = \frac{x^2}{0.25}$$

Step 3: Solve for x

$$x^2 = (5.56 \times 10^{-10})(0.25) = 1.39 \times 10^{-10}$$

$$x = \sqrt{1.39 \times 10^{-10}} \approx 1.18 \times 10^{-5}\,\mathrm{M}$$

Step 4: Calculate pH

$$\text{pH} = -\log(1.18 \times 10^{-5}) \approx 4.93$$

✅ Final Answer: pH ≈ 4.93

Question 7: Calculate the pH of 0.25 M NaF. Ka for HF = 6.6 × 10⁻⁴

NaF is a salt of a strong base (NaOH) and a weak acid (HF), so its solution is basic due to hydrolysis of F^-:

$$F^- + H_2O \rightleftharpoons HF + OH^-$$

Given:

- $[NaF] = 0.25\,M$
- $K_a(HF) = 6.6 \times 10^{-4}$
- $K_w = 1.0 \times 10^{-14}$

Step 1: Calculate Kb for F^-

$$K_b = \frac{K_w}{K_a} = \frac{1.0 \times 10^{-14}}{6.6 \times 10^{-4}} \approx 1.52 \times 10^{-11}$$

Step 2: ICE table

Let $x = [OH^-]$ at equilibrium.

$$K_b = \frac{x^2}{0.25 - x} \approx \frac{x^2}{0.25}$$

$$1.52 \times 10^{-11} = \frac{x^2}{0.25}$$

Step 3: Solve for x

$$x^2 = (1.52 \times 10^{-11})(0.25) = 3.8 \times 10^{-12}$$

$$x = \sqrt{3.8 \times 10^{-12}} \approx 1.95 \times 10^{-6}\,\text{M}$$

So:

$$[\text{OH}^-] \approx 1.95 \times 10^{-6}\,\text{M}$$

Step 4: Calculate pOH

$$\text{pOH} = -\log(1.95 \times 10^{-6}) \approx 5.71$$

✅ Final Answer: pH ≈ 8.29

Question 8: How many mL of 0.20 M KOH are needed to neutralize 150 mL of 0.020 M H_2SO_4?

We have a neutralization reaction:

$$H_2SO_4 + 2KOH \longrightarrow K_2SO_4 + 2H_2O$$

Step 1: Moles of H_2SO_4

$$\text{moles of } H_2SO_4 = M \times V = 0.020 \times 0.150 = 0.003 \text{ mol}$$

Step 2: Moles of KOH required

From the balanced equation:

1 mol H_2SO_4 reacts with 2 mol KOH.

Step 3: Volume of KOH solution

$$\text{Volume} = \frac{\text{moles}}{\text{molarity}} = \frac{0.006}{0.20} = 0.03 \text{ L} = 30 \text{ mL}$$

✅ Final Answer: 30 mL of 0.20 M KOH are needed.

Question 9:How many mL of 0.20 M HCl are needed to neutralize 150 mL of 0.020 M $Ca(OH)_2$?

We have the neutralization reaction:

$$Ca(OH)_2 + 2HCl \longrightarrow CaCl_2 + 2H_2O$$

Step 1: Moles of $Ca(OH)_2$

$$\text{moles of } Ca(OH)_2 = M \times V = 0.020 \times 0.150 = 0.003 \text{ mol}$$

Step 2: Moles of HCl required

From the balanced equation:

1 mol $Ca(OH)_2$ reacts with 2 mol HCl.

$$\text{moles of HCl} = 0.003 \times 2 = 0.006 \text{ mol}$$

Step 3: Volume of HCl solution

$$\text{Volume} = \frac{\text{moles}}{\text{molarity}} = \frac{0.006}{0.20} = 0.03 \text{ L} = 30 \text{ mL}$$

✅ Final Answer: 30 mL of 0.20 M HCl are needed.

Question 10: If 25.0 mL of 0.25 M HNO_3 is combined with 15.0 mL of 0.25 M CH_3NH_2,

We have a **strong acid** (HNO_3) and a **weak base** (CH_3NH_2) reacting to form a conjugate acid ($CH_3NH_3^+$). This is a **buffer-like situation**, but since the strong acid and weak base are mixed in stoichiometric amounts, we first check which is in excess.

Step 1: Calculate moles

$$\text{HNO}_3\text{: } 0.025\,\text{L} \times 0.25\,\text{M} = 0.00625\,\text{mol}$$

$$\text{CH}_3\text{NH}_2\text{: } 0.015\,\text{L} \times 0.25\,\text{M} = 0.00375\,\text{mol}$$

Step 2: Reaction

$$\text{CH}_3\text{NH}_2 + \text{HNO}_3 \longrightarrow \text{CH}_3\text{NH}_3^+$$

The weak base is limiting:

$$\text{CH}_3\text{NH}_2 \text{ completely reacts: } 0.00375\,\text{mol}$$

Remaining HNO_3:

$$0.00625 - 0.00375 = 0.00250\,\text{mol (excess strong acid)}$$

Step 3: Total volume

$$V_{\text{total}} = 25.0\,\text{mL} + 15.0\,\text{mL} = 40.0\,\text{mL} = 0.040\,\text{L}$$

Step 4: [H^+] from excess HNO_3

$$[\text{H}^+] = \frac{0.00250}{0.040} = 0.0625\,\text{M}$$

Step 5: pH

$$\text{pH} = -\log(0.0625) \approx 1.20$$

✅ Final Answer: pH ≈ 1.20

(The conjugate acid $CH_3NH_3^+$ is present, but the strong acid dominates, so pH is controlled by excess HNO_3.)

Question 11: If 25.0 mL of 0.25 M NaOH is combined with 15.0 mL of 0.25 M CH_3COOH, what is the pH? Ka = 1.8×10^{-5}

We have a strong base (NaOH) and a weak acid (CH_3COOH) reacting to form its conjugate base (CH_3COO^-). This creates a buffer solution after neutralization.

Step 1: Calculate moles

$$\text{NaOH: } 0.025\,\text{L} \times 0.25\,\text{M} = 0.00625\,\text{mol}$$

$$\text{CH₃COOH: } 0.015\,\text{L} \times 0.25\,\text{M} = 0.00375\,\text{mol}$$

Step 2: Reaction

$$\text{CH₃COOH} + \text{OH}^- \longrightarrow \text{CH₃COO}^- + \text{H₂O}$$

Step 2: Reaction

$$\text{CH₃COOH} + \text{OH}^- \longrightarrow \text{CH₃COO}^- + \text{H₂O}$$

The weak acid is limiting:

$$\text{CH₃COOH completely reacts: } 0.00375\,\text{mol}$$

Remaining NaOH:

$$0.00625 - 0.00375 = 0.00250\,\text{mol (excess OH}^-\text{)}$$

Step 3: Total volume

$$V_{\text{total}} = 25.0\,\text{mL} + 15.0\,\text{mL} = 40.0\,\text{mL} = 0.040\,\text{L}$$

Step 4: [OH⁻] from excess NaOH

$$[\text{OH}^-] = \frac{0.00250}{0.040} = 0.0625\,\text{M}$$

Step 5: pOH

$$\text{pOH} = -\log(0.0625) \approx 1.20$$

Step 6: pH

$$\text{pH} = 14 - 1.20 = 12.80$$

✅ Final Answer: pH ≈ 12.80

(The excess strong base dominates, so the solution is strongly basic.)

Question 12: A household cleaner contains ammonia. A 25.37 g sample of the cleaner is dissolved in water and made up to 250 cm^3. A 25.0 cm^3 portion of this solution requires 37.3 cm^3 of 0.360 mol dm^{-3} sulfuric acid for neutralization. What is the percentage by mass of ammonia in the cleaner?

1. **Calculate the moles of sulphuric acid used:**

$$\text{Moles of } H_2SO_4 = \text{Volume (dm}^3\text{)} \times \text{Concentration (mol dm}^{-3}\text{)}$$

$$\text{Volume of } H_2SO_4 = 37.3 \text{ cm}^3 = 0.0373 \text{ dm}^3$$

$$\text{Moles of } H_2SO_4 = 0.0373 \text{ dm}^3 \times 0.360 \text{ mol dm}^{-3} = 0.013428 \text{ mol}$$

2. **Determine the moles of ammonia neutralised:** The balanced equation for the reaction is:

$$2NH_3 + H_2SO_4 \rightarrow (NH_4)_2SO_4$$

Therefore, 1 mole of H_2SO_4 neutralises 2 moles of NH_3.

$$\text{Moles of } NH_3 = 2 \times 0.013428 \text{ mol} = 0.026856 \text{ mol}$$

3. **Calculate the mass of ammonia in the 25.0 cm^3 portion:**

$$\text{Mass of } NH_3 = \text{Moles} \times \text{Molar mass}$$

$$\text{Molar mass of } NH_3 = 17.03 \text{ g mol}^{-1}$$

$$\text{Mass of } NH_3 = 0.026856 \text{ mol} \times 17.03 \text{ g mol}^{-1} = 0.457 \text{ g}$$

4. **Calculate the mass of ammonia in the entire 250 cm^3 solution:**

$$\text{Mass of } NH_3 \text{ in } 250 \text{ cm}^3 = \frac{0.457 \text{ g}}{25.0 \text{ cm}^3} \times 250 \text{ cm}^3 = 4.57 \text{ g}$$

5. **Determine the percentage by mass of ammonia in the cleaner:**

$$\text{Percentage by mass} = \left(\frac{\text{Mass of } NH_3}{\text{Mass of cleaner}} \right) \times 100$$

$$\text{Percentage by mass} = \left(\frac{4.57 \text{ g}}{25.37 \text{ g}} \right) \times 100 = 18.01\%$$

So, the percentage by mass of ammonia in the cleaner is **18.01%**.

Would you like help with another question or any further explanation on this one?

Question 13: 8.492 g of Ammonium iron (II) sulphate crystals ($(NH_4)_2SO_4.FeSO_4.nH_2O$) were dissolved in water and the solution was made up to 250 cm^3 with distilled water and dilute sulphuric acid. A 25.0 cm^3 portion of the solution was further acidified and titrated against potassium permanganate (VII) solution of concentration 0.0150 mol dm^{-3}, requiring 22.5 cm^3 for neutralization to be achieved. Determine the value of *n*.

Solution:

1. **Calculate the moles of potassium manganate (VII) used:**

$$\text{Moles of } KMnO_4 = \text{Volume } (dm^3) \times \text{Concentration } (\text{mol } dm^{-3})$$

$$\text{Volume of } KMnO_4 = 22.5 \text{ cm}^3 = 0.0225 \text{ dm}^3$$

$$\text{Moles of } KMnO_4 = 0.0225 \text{ dm}^3 \times 0.0150 \text{ mol dm}^{-3} = 0.0003375 \text{ mol}$$

2. **Determine the moles of iron (II) sulphate neutralised:** The balanced equation for the reaction is:

$$5Fe^{2+} + MnO_4^- + 8H^+ \rightarrow 5Fe^{3+} + Mn^{2+} + 4H_2O$$

Therefore, 1 mole of MnO_4^- reacts with 5 moles of Fe^{2+}.

$$\text{Moles of } Fe^{2+} = 5 \times 0.0003375 \text{ mol} = 0.0016875 \text{ mol}$$

3. **Calculate the moles of iron (II) sulphate in the entire 250 cm^3 solution:**

$$\text{Moles of } Fe^{2+} \text{ in } 250 \text{ cm}^3 = \frac{0.0016875 \text{ mol}}{25.0 \text{ cm}^3} \times 250 \text{ cm}^3 = 0.016875 \text{ mol}$$

4. **Determine the moles of ammonium iron (II) sulphate crystals:**

$$\text{Moles of } (NH_4)_2SO_4.FeSO_4.nH_2O = 0.016875 \text{ mol}$$

5. **Calculate the molar mass of ammonium iron (II) sulphate crystals:**

$$\text{Molar mass} = \text{Mass}/\text{Moles}$$

$$\text{Molar mass} = 8.492\ \text{g}/0.016875\ \text{mol} = 503.6\ \text{g mol}^{-1}$$

6. **Determine the value of n:** The molar mass of ammonium iron (II) sulphate crystals $((\text{NH}_4)_2\text{SO}_4.\text{FeSO}_4.n\text{H}_2\text{O})$ can be calculated as:

$$\text{Molar mass} = 132.14\ \text{g mol}^{-1}((\text{NH}_4)_2\text{SO}_4) + 151.91\ \text{g mol}^{-1}(\text{FeSO}_4) + n \times 18.015\ \text{g mol}^{-1}(\text{H}_2$$

$$503.6\ \text{g mol}^{-1} = 284.05\ \text{g mol}^{-1} + n \times 18.015\ \text{g mol}^{-1}$$

$$n \times 18.015\ \text{g mol}^{-1} = 503.6\ \text{g mol}^{-1} - 284.05\ \text{g mol}^{-1} = 219.55\ \text{g mol}^{-1}$$

$$n = \frac{219.55\ \text{g mol}^{-1}}{18.015\ \text{g mol}^{-1}} = 12.19 \approx 12$$

So, the value of n is **12**.

Would you like to go through another question or need further clarification on this one?

Question 14: An aspirin tablet was ground using a glass rod, and 25 mL of a 1.0 mol/L NaOH solution was added to the powder. The mixture was then heated for 15 minutes to ensure complete reaction. After cooling, the excess NaOH was titrated with 20 mL of a 0.5 mol/L H_2SO_4 solution. Based on this data, calculate the mass of aspirin present in the tablet. (Molar mass of aspirin = 180.1574 g/mol)

Step 1: Initial NaOH moles

$$n_{\text{NaOH initial}} = 1.0 \times 0.025 = 0.025\,\text{mol}$$

Step 2: NaOH neutralized by H_2SO_4

$$n_{H_2SO_4} = 0.5 \times 0.020 = 0.010\,\text{mol}$$

$$n_{\text{NaOH leftover}} = 2 \times 0.010 = 0.020\,\text{mol}$$

Step 3: NaOH consumed by aspirin

$$n_{\text{NaOH reacted}} = 0.025 - 0.020 = 0.005\,\text{mol}$$

Step 4: Moles of aspirin Since 1 mol aspirin consumes 2 mol NaOH:

$$n_{\text{aspirin}} = \frac{0.005}{2} = 0.0025\,\text{mol}$$

Step 5: Mass of aspirin

$$m = 0.0025 \times 180.1574 = 0.4504\,\text{g}$$

Question 14: Calculate the pH of the following solutions: a) HCl, 0.1 mol/L b) Acetic acid, 0.1 mol/L (pKa = 4.75) c) How many times is the [H^+] concentration in the HCl solution greater than that in the acetic acid solution?

a) HCl (0.1 mol/L):

Since HCl is a strong acid, it dissociates completely:

$$[H^+] = 0.1\,\text{mol/L} \quad \Rightarrow \quad \text{pH} = -\log(0.1) = \mathbf{1.00}$$

b) Acetic acid (0.1 mol/L, pKa = 4.75):

$$\text{Ka} = 10^{-4.75} \approx 1.78 \times 10^{-5}$$

For a weak acid:

$$[H^+] \approx \sqrt{\text{Ka} \times C} = \sqrt{(1.78 \times 10^{-5})(0.1)} \approx 1.33 \times 10^{-3}\,\text{mol/L}$$

$$\text{pH} = -\log(1.33 \times 10^{-3}) \approx \mathbf{2.88}$$

c) Ratio of [H^+]:

$$\frac{[H^+]_{\text{HCl}}}{[H^+]_{\text{Acetic}}} = \frac{0.1}{1.33 \times 10^{-3}} \approx \mathbf{75}\ \text{times greater}$$

Question 15: How many moles of sodium ethanoate must be added to 1.00 dm^3 of 0.0100 mol dm^{-3} of ethanoic acid to produce a buffer solution of pH 5.8?

To achieve a buffer solution of pH = 5.8 starting with 1.00 dm^3 of 0.0100 mol·dm^{-3} ethanoic acid, we use the Henderson–Hasselbalch equation:

$$\mathrm{pH} = \mathrm{pKa} + \log\left(\frac{[salt]}{[acid]}\right)$$

Given:

- $\mathrm{Ka} = 1.86 \times 10^{-5} \rightarrow \mathrm{pKa} \approx 4.73$
- $\mathrm{pH} = 5.8$
- $[acid] = 0.0100\,\mathrm{mol}\backslash\mathrm{cdotp}\mathrm{dm}^{-3}$
- Volume = 1.00 dm^3

Steps:

1. Compute ratio:

$$\mathrm{pH} - \mathrm{pKa} = 5.8 - 4.73 = 1.07$$

2. Calculate required salt concentration:

$$[salt] = 11.74 \times 0.0100 = 0.1174\,\mathrm{mol}\backslash\mathrm{cdotp}\mathrm{dm}^{-3}$$

3. For 1.00 dm^3:

$$\text{moles of sodium ethanoate} = 0.1174\,\mathrm{mol}$$

✅ **Answer:** You need to add ≈ **0.117 mol of sodium ethanoate** to 1.00 dm^3 of 0.0100 mol·dm^{-3} ethanoic acid to achieve pH 5.8.

Q16: Calculate the pH of the following solutions: a) HCl, 0.1 mol/L
b) Acetic acid, 0.1 mol/L (pKa = 4.75)
c) How many times is the [H^+] concentration in the HCl solution greater than that in the acetic acid solution?

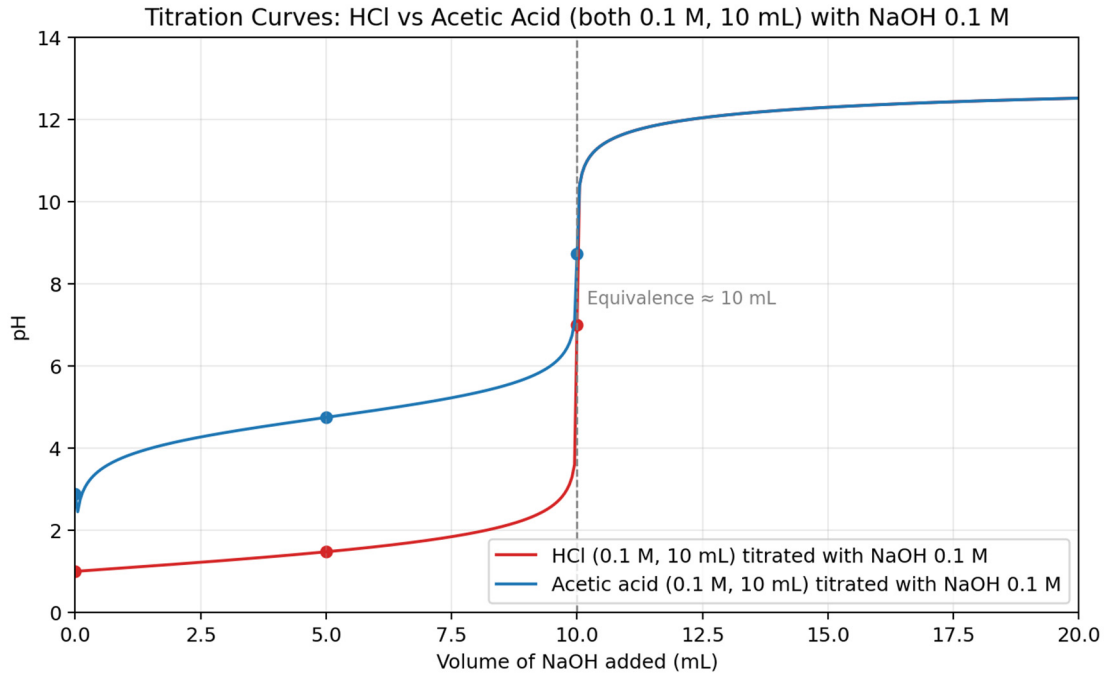


Figure S 1: Comparison of titration curves for 10 mL of 0.1 M HCl (strong acid) and 10 mL of 0.1 M acetic acid (weak acid, pKa = 4.75) titrated with 0.1 M NaOH. The strong acid shows a sharp pH rise near equivalence (pH ≈ 7), while the weak acid exhibits a buffer region and an equivalence point above pH 7 (≈ 8.7) due to acetate hydrolysis

Key Takeaways from the Titration Curves (Figure S 1**)**

Equivalence volume.
Both acids require **10.0 mL** of 0.100 M NaOH to reach equivalence because they contain the same initial amount of acid:
n = 0.100 M × 10.0 mL = 1.0×10^{-3} mol.

Initial pH.

- **HCl:** pH = 1.00, reflecting complete dissociation and [H^+] ≈ 0.10 M.
- **Acetic acid:** pH ≈ 2.87, obtained from the weak-acid equilibrium using Ka = $10^{-4.75}$.

Half-equivalence (5.0 mL of base).

- **HCl:** pH ≈ 1.48, governed only by dilution of the remaining strong acid.

- **Acetic acid:** pH = pKa = 4.75 exactly, according to the Henderson–Hasselbalch relation.

Equivalence point (10.0 mL).

- **HCl:** pH = 7.00, as the solution contains only NaCl (neglecting activity corrections).
- **Acetic acid:** pH ≈ 8.72, because acetate undergoes hydrolysis (Kb = Kw/Ka) and generates OH^-. The resulting acetate concentration is approximately 0.050 M.

How the Curves Were Computed

Let
V_0 = 0.010 L (initial acid volume),
Cb = 0.100 M (base concentration),
Vb (L) = added base volume, and
Vtot = V_0 + Vb.

---

1. Strong Acid (HCl)

Before equivalence (CbVb < nHCl):

$$[H^+] = \frac{n_{HCl} - C_b V_b}{V_{tot}}, pH = -\log_{10}[H^+].$$

At equivalence:

pH ≈ 7.00.

After equivalence (CbVb > nHCl):

$$[OH^-] = \frac{C_b V_b - n_{HCl}}{V_{tot}}, pH = 14 + \log_{10}[OH^-].$$

---

2. Weak Acid (Acetic Acid; pKa = 4.75, Ka = 10^{-4.75})

Initial solution (no base):

Solve the weak-acid dissociation expression:

$$K_a = \frac{x^2}{C_0 - x}, x = [H^+],$$

where $C_0 = 0.100$ M.

Buffer region (before equivalence):

Stoichiometry: $HA + OH^- \rightarrow A^-$

$$\text{pH} = pK_a + \log_{10}\left(\frac{n_{A^-}}{n_{HA}}\right) = pK_a + \log_{10}\left(\frac{C_b V_b}{n_{HAc} - C_b V_b}\right).$$

Equivalence point:

Acetate undergoes hydrolysis:

$$K_b = \frac{K_w}{K_a}.$$

Solve:

$$x^2 + K_b x - K_b C_{A^-} = 0, x = [OH^-],$$

where

$$C_{A^-} = \frac{n_{HAc}}{V_{tot}}.$$

After equivalence:

Excess strong base dominates:

$$[OH^-] = \frac{C_b V_b - n_{HAc}}{V_{tot}}, \text{pH} = 14 + \log_{10}[OH^-].$$

Compared with the strong acid, the acetic-acid titration curve begins at a higher pH and exhibits a pronounced **buffer region**, characterized by a gradual rise in pH until the midpoint of the titration (pH = pKa). The slope remains shallow up to the equivalence point because both the acid and its conjugate base are present simultaneously. At equivalence, the pH exceeds 7 due to **acetate hydrolysis**, which produces OH^- and shifts the solution basic.

In contrast, the strong-acid titration curve displays a steep rise with no buffer region and reaches pH 7 at equivalence because the salt formed (NaCl) does not undergo hydrolysis. The juxtaposition of these curves highlights fundamental differences in acid strength, buffer behavior, and the role of conjugate bases in determining the equivalence-point pH.

Q17. For 10 mL of HCl (0.1 mol/L) and 10 mL of acetic acid (0.1 mol/L, pKa = 4.75), each titrated with NaOH (0.1 mol/L).Present both titration curves on the same graph for comparison.

What you're seeing

- Equivalence volume (both):
  $n_{\text{acid}} = 0.010\,\text{L} \times 0.1\,\text{mol}\cdotp\text{L}^{-1} = 0.001\,\text{mol} \Rightarrow V_{\text{eq}} = \frac{0.001}{0.1} = 0.010\,\text{L} = 10\,\text{mL}$
- Strong acid (HCl): Sharp jump at equivalence; pH = 7 at V_{eq}.
- Weak acid (acetic acid): Buffer region up to V_{eq} with a gentler slope; pH = pK_a = 4.75 at half-equivalence (5 mL); pH ≈ 8.72 at equivalence (basic due to acetate hydrolysis).

Key values (25 °C)

- Initial pH (HCl 0.1 M): 1.00
- Initial pH (CH_3COOH 0.1 M): ≈ 2.88
  $\text{Ka} = 10^{-4.75}$; $[\text{H}^+] \approx \sqrt{K_a C} \Rightarrow \text{pH} \approx 2.88$
- Half-equivalence (acetic acid): pH = 4.75
- Equivalence pH (acetic acid): ≈ 8.72
  $C_{\text{acetate}} = \frac{0.001}{0.020} = 0.05\,\text{M},\ K_b = \frac{K_w}{K_a} \Rightarrow [\text{OH}^-] \approx \sqrt{K_b C}$

How the curves were computed

For each added base volume V_b (0 → 20 mL):

HCl (strong acid):

- Before eq.: $[\text{H}^+] = \frac{n_{\text{HCl}} - n_{\text{OH}}}{V_{\text{tot}}}$
- At eq.: $\text{pH} = 7$
- After eq.: $[\text{OH}^-] = \frac{n_{\text{OH}} - n_{\text{HCl}}}{V_{\text{tot}}},\ \text{pH} = 14 + \log_{10}[\text{OH}^-]$

CH_3COOH (weak acid):

- Initial (no base): solve $K_a = \dfrac{x^2}{C - x}$ for $x = [\mathrm{H^+}]$
- Buffer region ($0 < V_b < V_{\text{eq}}$): Henderson–Hasselbalch
  $$\mathrm{pH} = \mathrm{p}K_a + \log \frac{n_{\mathrm{A^-}}}{n_{\mathrm{HA}}} = \mathrm{p}K_a + \log \frac{n_{\mathrm{OH}}}{n_{\mathrm{HA},0} - n_{\mathrm{OH}}}$$
- At eq.: acetate solution, basic:
  $$[\mathrm{OH^-}] \approx \sqrt{K_b\, C_{\text{acetate}}},\ \mathrm{pH} = 14 + \log_{10}[\mathrm{OH^-}]$$
- After eq.: excess strong base dominates
  $$[\mathrm{OH^-}] = \frac{n_{\mathrm{OH}} - n_{\mathrm{HA},0}}{V_{\text{tot}}}$$

Q18 Generate histograms for ACPP_FLN*ESYK (Light/Heavy) and, split by AG and NAG groups. Display both linear and logarithmic scales, include KDE (kernel density estimation) curves, and overlay the NAG and AG distributions on the same axis for comparison. Show all images in the same figure.

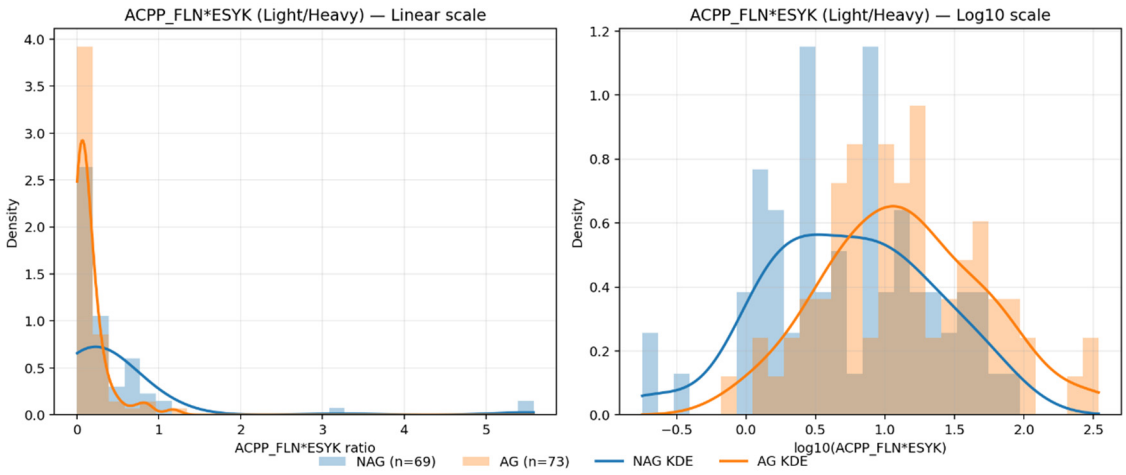


*Figure S 2: Overlayed histograms and Gaussian KDE curves for ACPP_FLN*ESYK (Light/Heavy) ratios in the NAG (blue) and AG (orange) prostate cancer groups. The left panel (linear scale) displays the raw ratio distribution and highlights absolute magnitude and potential outliers. The right panel (log_{10} scale) reduces skewness and compresses extreme values, improving visual comparison between groups. KDE curves (Gaussian kernel, Silverman bandwidth) provide smoothed density estimates that reveal underlying distributional structure beyond histogram binning effects.*

In Figure S 2, the two-panel visualization compares the same biomarker across NAG and AG groups using linear and logarithmic scales:

- **Linear scale (left):** Presents untransformed ratios, useful for examining absolute values and identifying extreme measurements.
- **Log_{10} scale (right):** Compresses the right tail and reduces skewness, allowing clearer comparison of group distributions.

Why KDE curves matter

Histograms are sensitive to bin size and can introduce visual artifacts. KDE smooths the distribution using a continuous probability density estimate, revealing structural features and group differences that may not be visible in histogram form.

Interpretation

The NAG and AG groups show overlapping distributions but differ in central tendency and spread. Log transformation clarifies these differences by stabilizing variance and

diminishing the influence of extreme values, improving interpretability for exploratory analysis and subsequent statistical modeling.

Q19: Show the same for CLU_EDALN*ETR

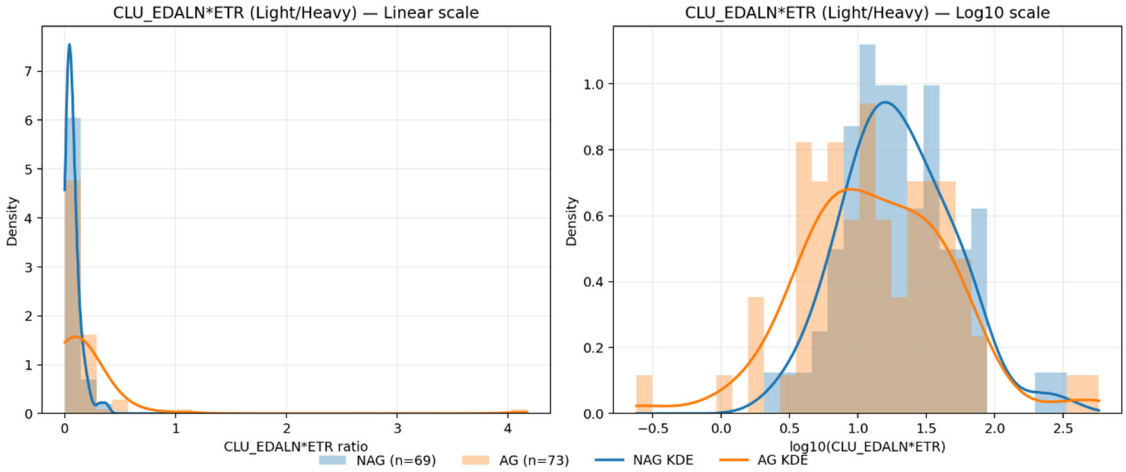


*Figure S 3Overlayed histograms and Gaussian KDE curves for CLU_EDALN*ETR (Light/Heavy) ratios in the NAG (blue) and AG (orange) prostate cancer groups. The left panel (linear scale) displays the raw ratio distribution and reveals the absolute spread and high-value tail. The right panel (log_{10} scale) reduces right-skew and enhances visibility of distributional differences. KDE curves (Gaussian kernel, Silverman bandwidth) provide smoothed density estimates that clarify underlying trends beyond histogram binning. Data: AG (n = 73), NAG (n = 69), from "Spreadsheet 1.xlsx."*

What is being compared

The Figure S 3 presents the CLU_EDALN*ETR biomarker ratio for the NAG and AG groups on linear and logarithmic scales. Both groups appear on the same axes, allowing direct visual assessment of overlap, separation, and variability.

Why two scales are used

- **Linear scale:** Preserves the raw magnitude of the ratios, enabling detection of large values and the full measurement range.
- **Log_{10} scale:** Compresses extreme values and mitigates right-skew, making differences in central tendency and modal structure easier to interpret—particularly relevant for biomarker intensities that span multiple orders of magnitude.

Role of KDE curves

Histograms can appear uneven due to bin-size sensitivity. Gaussian KDE smoothing (Silverman bandwidth) offers a continuous estimate of the distribution, revealing peaks, shoulders, and tail behavior that more accurately reflect the underlying data. Reading KDE curves alongside histogram bars strengthens interpretation of apparent group differences.

How to interpret the visualization

- A right-shifted KDE peak indicates higher ratios for that group.
- Broader KDE curves indicate greater variability; narrower curves signal tighter clustering.

- On the log scale, peak separation or overlap provides the most reliable visual cue when working with skewed or heavy-tailed biomarker data.

Q20: Show the same for age and Serum PSA

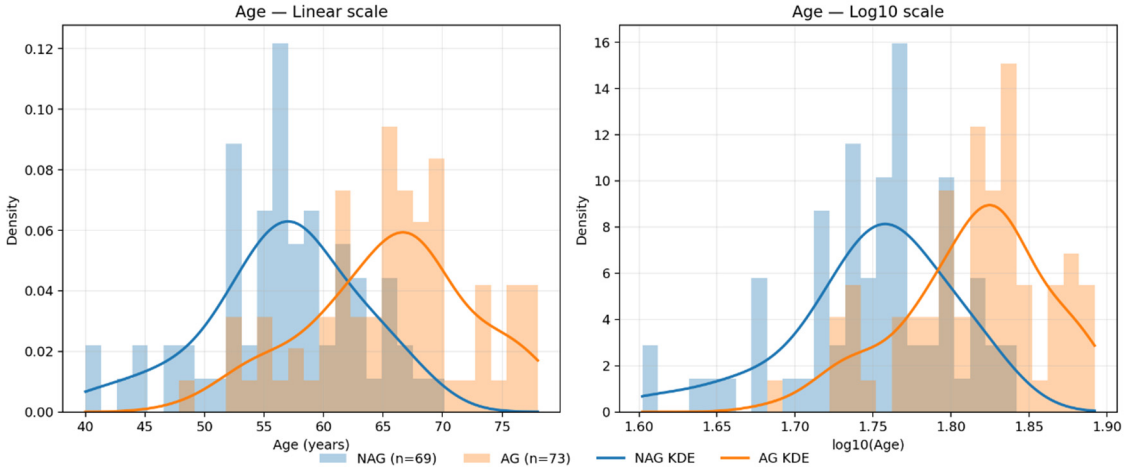


Figure S4: Overlayed histograms and Gaussian KDE curves for Age in NAG (blue) and AG (orange) groups. Left panel (linear scale) shows the absolute age distribution; right panel (log10 scale) compresses the range to emphasize shape differences. KDE curves (Silverman bandwidth) smooth the empirical distributions to highlight underlying density features

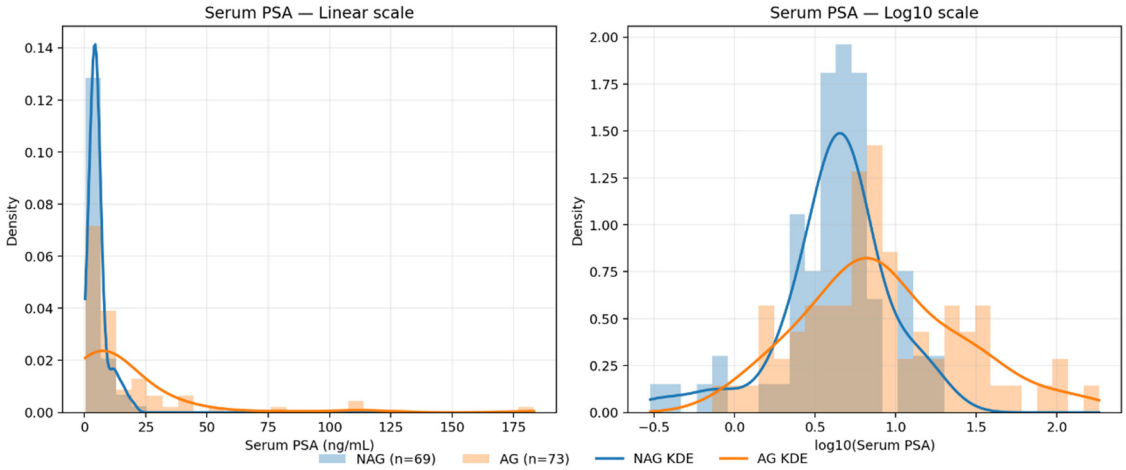


Figure S5: Overlayed histograms and Gaussian KDE curves for Serum PSA in NAG (blue) and AG (orange) groups. Left panel (linear scale) shows absolute PSA values and reveals high-value tails; right panel (log10 scale) reduces right-skew, making group differences in central tendency and spread easier to compare. KDE curves (Silverman bandwidth) smooth the empirical distributions.

Q21: Compare the expression levels of the glycopeptide ACPP_FLN*ESYK between aggressive (AG) and nonaggressive (NAG) prostate cancer groups using box plots.

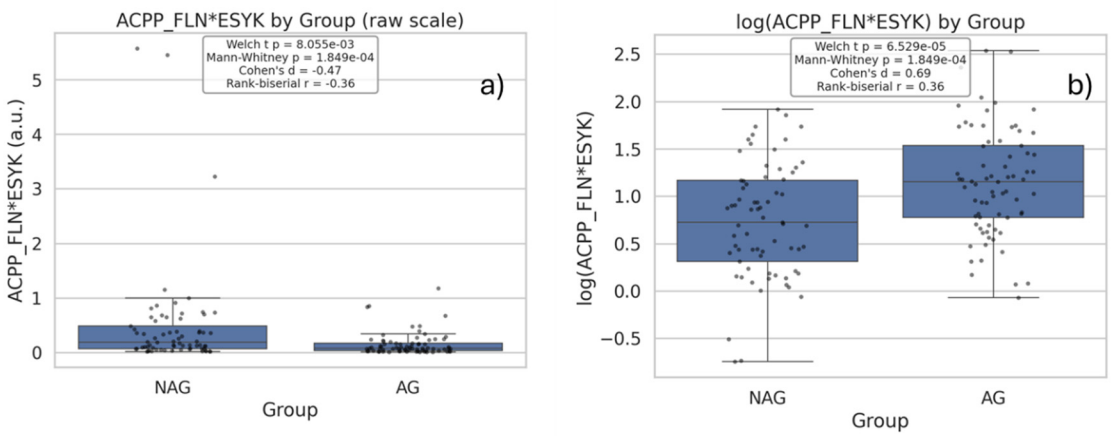


*Figure S 6: Comparison of ACPP_FLN*ESYK expression between non-aggressive (NAG) and aggressive (AG) prostate cancer groups using box plots. (a) Raw scale values with effect size and significance (Welch t p = 8.06×10^{-3}; Mann–Whitney p = 1.85×10^{-4}; Cohen's d = –0.47). (b) Log-transformed values showing improved separation (Welch t p = 6.53×10^{-5}; Mann–Whitney p = 1.85×10^{-4}; Cohen's d = 0.69)*

Q22: Compare the expression levels of the glycopeptide CLU_EDALN*ETR between aggressive (AG) and nonaggressive (NAG) prostate cancer groups using box plots

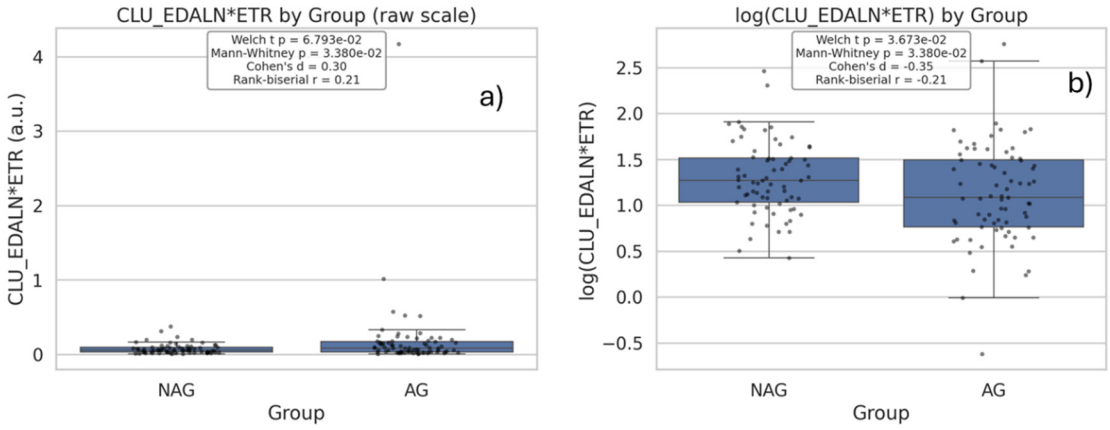


*Figure S 7:Comparison of CLU_EDALN*ETR expression between non-aggressive (NAG) and aggressive (AG) prostate cancer groups using box plots. (a) Raw scale values with statistical results (Welch t p = 6.79×10^{-2}; Mann–Whitney p = 3.38×10^{-2}; Cohen's d = 0.30). (b) Log-transformed values showing reversed effect direction (Welch t p = 3.67×10^{-2}; Mann–Whitney p = 3.38×10^{-2}; Cohen's d = –0.35).*

Q23: Compare age and serum PSA between aggressive (AG) and nonaggressive (NAG) prostate cancer groups using box plots

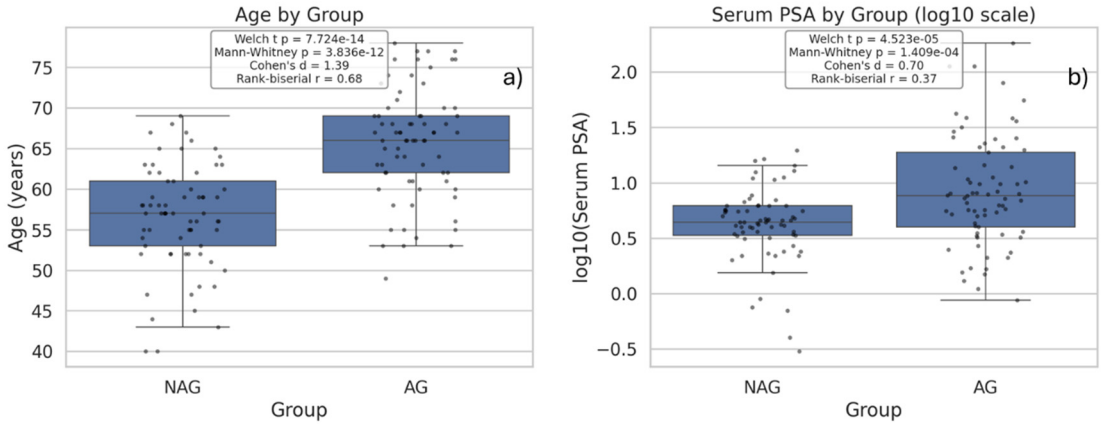


Figure S8: Comparison of age and serum PSA between non-aggressive (NAG) and aggressive (AG) prostate cancer groups using box plots. (a) Age shows a significant difference (Welch t p = 7.72×10^{-14}; Cohen's d = 1.39). (b) Serum PSA (log_{10} scale) also differs significantly (Welch t p = 4.52×10^{-5}; Cohen's d = 0.70

Q24: Build a correlation plot of ACPP_FLN*ESYK (y-axis) vs CLU_EDALN*ETR (x-axis), with the data split by AG and NAG groups, shown image in the same Figure, show Pearson correlations and correlation lines.

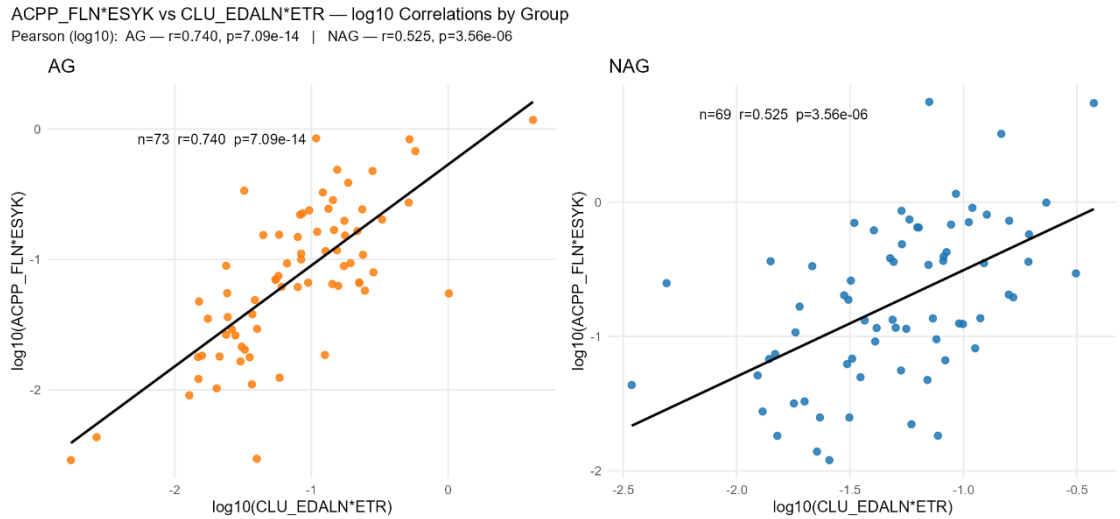


Figure S9: *Correlation of ACPP_FLN*ESYK (y) versus CLU_EDALN*ETR (x), split by AG and NAG groups.*

R scripts generate by Copilot, these scripts were used to generate the image

```
# ============================================================
# Correlation plots (log10): ACPP_FLN*ESYK (y) vs CLU_EDALN*ETR (x)
# Split by AG and NAG; show Pearson r & p, plus regression lines
# Output: single PNG saved next to the spreadsheet
# ============================================================

# If needed:
# install.packages(c("readxl", "dplyr", "ggplot2", "patchwork"))

library(readxl)
library(dplyr)
library(ggplot2)
library(patchwork)

# ---- Path to your Excel file ----
xlsx_path <- "C:\\Users\\marce\\Documents\\Digital\\Spreadsheet 1.xlsx"
out_dir   <- dirname(xlsx_path)
out_png   <- file.path(out_dir, "acpp_vs_clu_correlation_log10_AG_NAG.png")

# ---- Load and prepare data ----
df <- read_excel(xlsx_path)
names(df) <- trimws(names(df))  # normalize headers
```

```
# Robust column detection (falls back to regex if exact name not found)
locate_col <- function(exact, regex = NULL) {
  if (exact %in% names(df)) return(exact)
  if (!is.null(regex)) {
    hits <- grep(regex, names(df), ignore.case = TRUE, value = TRUE)
    if (length(hits) > 0) return(hits[1])
  }
  NA_character_
}

col_group <- locate_col("Prostate cancer group", "^Prostate\\s*cancer\\s*group$")
col_acpp  <- locate_col("ACPP_FLN*ESYK", "^ACPP.*ESYK")
col_clu   <- locate_col("CLU_EDALN*ETR", "^CLU.*ETR")

if (any(is.na(c(col_group, col_acpp, col_clu)))) {
  stop("Could not locate required columns: 'Prostate cancer group', 'ACPP_FLN*ESYK',
'CLU_EDALN*ETR'.")
}

# Keep only AG/NAG; coerce to numeric; drop non-finite; keep positives for log10
df_sub <- df |>
  filter(.data[[col_group]] %in% c("AG", "NAG")) |>
  transmute(
    Group = factor(.data[[col_group]], levels = c("AG", "NAG")),  # left->right order
    CLU   = suppressWarnings(as.numeric(.data[[col_clu]])),
    ACPP  = suppressWarnings(as.numeric(.data[[col_acpp]]))
  ) |>
  filter(is.finite(CLU), is.finite(ACPP)) |>
  filter(CLU > 0, ACPP > 0) |>                    # required for log10
  mutate(
    log10_CLU  = log10(CLU),
    log10_ACPP = log10(ACPP)
  )

# ---- Pearson stats (on log10-transformed data) ----
pearson_stats <- function(dat) {
  n <- nrow(dat)
  if (n < 3) {
    return(list(n = n, r = NA_real_, p = NA_real_, label = paste0("n=", n, " (too few for
Pearson)")))
  }
  ct <- suppressWarnings(cor.test(dat$log10_CLU, dat$log10_ACPP, method =
"pearson"))
  r  <- unname(ct$estimate)
  p  <- ct$p.value
  label <- paste0("n=", n, "  r=", sprintf("%.3f", r), "  p=", format.pval(p, digits = 3))
  list(n = n, r = r, p = p, label = label)
}
```

```
ag_dat  <- df_sub %>% filter(Group == "AG")
nag_dat <- df_sub %>% filter(Group == "NAG")

ag_stats  <- pearson_stats(ag_dat)
nag_stats <- pearson_stats(nag_dat)

# ---- Panel builder ----
make_panel <- function(data, panel_title, color_points, stats_label) {
  # Choose a readable annotation position (top-left in data space)
  x_pos <- quantile(data$log10_CLU, 0.02, na.rm = TRUE)
  y_pos <- quantile(data$log10_ACPP, 0.98, na.rm = TRUE)

  ggplot(data, aes(x = log10_CLU, y = log10_ACPP)) +
    geom_point(color = color_points, alpha = 0.85, size = 2.2) +
    geom_smooth(method = "lm", se = FALSE, color = "black", linewidth = 0.9) +
    annotate("text", x = x_pos, y = y_pos, hjust = 0,
             label = stats_label, size = 3.8) +
    labs(
      title = panel_title,
      x = "log10(CLU_EDALN*ETR)",
      y = "log10(ACPP_FLN*ESYK)"
    ) +
    theme_minimal(base_size = 12) +
    theme(panel.grid.minor = element_blank())
}

# ---- Build panels ----
p_ag  <- make_panel(ag_dat,  "AG",  color_points = "#ff7f0e", stats_label =
ag_stats$label)
p_nag <- make_panel(nag_dat, "NAG", color_points = "#1f77b4", stats_label =
nag_stats$label)

# ---- Combine and save ----
subtitle_txt <- paste0(
  "Pearson (log10):  AG — r=", ifelse(is.na(ag_stats$r), "NA", sprintf("%.3f",
ag_stats$r)),
  ", p=", ifelse(is.na(ag_stats$p), "NA", format.pval(ag_stats$p, digits = 3)),
  "   |   NAG — r=", ifelse(is.na(nag_stats$r), "NA", sprintf("%.3f", nag_stats$r)),
  ", p=", ifelse(is.na(nag_stats$p), "NA", format.pval(nag_stats$p, digits = 3))
)

combined <- (p_ag | p_nag) +
  plot_annotation(
    title = "ACPP_FLN*ESYK vs CLU_EDALN*ETR — log10 Correlations by Group",
    subtitle = subtitle_txt
  )

ggsave(out_png, plot = combined, width = 12, height = 5.6, dpi = 180)
message("Saved correlation figure to: ", out_png)
```

Q25: Build a correlation plot of age (x-axis) vs ACPP_FLN*ESYK (y-axis) and CLU_EDALN*ETR (y-axis), do it individually for AG and NAG groups, shown image in the same figure, shown Pearson correlations and a correlation lines and y-axis in log scale and x-axis linear scale

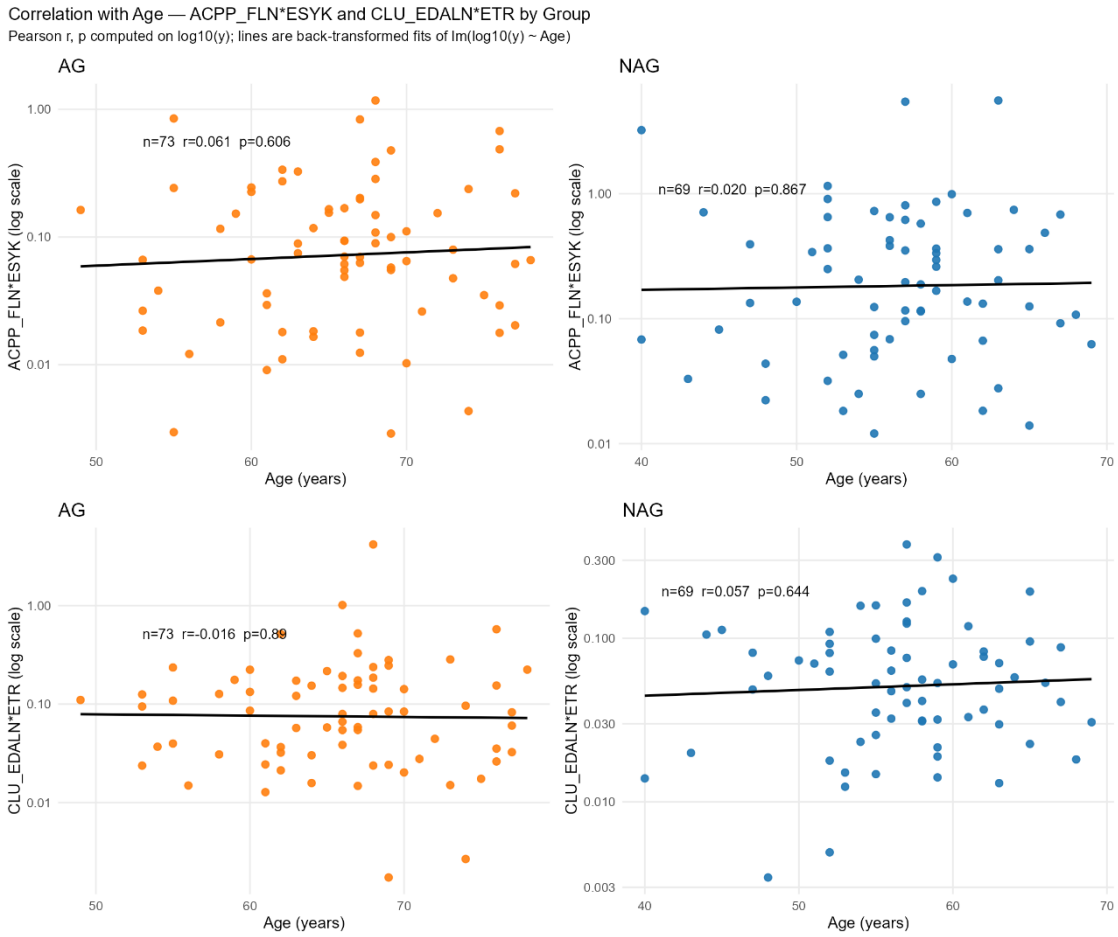


*Figure S10: Age vs. CLU_EDALNETR and Age vs. ACPP_FLNESYK, stratified by prostate cancer group (AG, NAG). **Top row**: AG; **bottom row**: NAG. **Left column**: Age vs. CLU_EDALNETR. **Right column**: Age vs. ACPP_FLNESYK. Each panel shows a least-squares trendline and annotates Pearson r and p-value for the within-group association.*

In Figure S10, each panel displays a scatter plot of **CLU_EDALN*ETR** and **ACPP_FLN*ESYK** (y), vs Age (x).
These were the R scripts provided by Copilot to generate the image.

```
# ============================================================
# Age (x, linear) vs ACPP_FLN*ESYK (y, log scale) and
# Age (x, linear) vs CLU_EDALN*ETR (y, log scale)
# Split by AG and NAG; show Pearson r & p (on log10(y)) and
# draw regression lines from lm(log10(y) ~ Age) back-transformed to y.
# Saves one PNG next to the spreadsheet.
# ============================================================
```

```
# If needed:
# install.packages(c("readxl", "dplyr", "ggplot2", "patchwork"))

library(readxl)
library(dplyr)
library(ggplot2)
library(patchwork)

# ---- Path to your Excel file ----
xlsx_path <- "C:\\Users\\marce\\Documents\\Digital\\Spreadsheet 1.xlsx"
out_dir   <- dirname(xlsx_path)
out_png   <- file.path(out_dir, "age_vs_acpp_and_clu_logY_by_group.png")

# ---- Load & normalize column names ----
df <- read_excel(xlsx_path)
names(df) <- trimws(names(df))  # normalize headers

# Robust column detection
locate_col <- function(exact, regex = NULL) {
  if (exact %in% names(df)) return(exact)
  if (!is.null(regex)) {
    hits <- grep(regex, names(df), ignore.case = TRUE, value = TRUE)
    if (length(hits) > 0) return(hits[1])
  }
  NA_character_
}

col_group <- locate_col("Prostate cancer group", "^Prostate\\s*cancer\\s*group$")
col_age   <- locate_col("Age", "^Age$")
col_acpp  <- locate_col("ACPP_FLN*ESYK", "^ACPP.*ESYK")
col_clu   <- locate_col("CLU_EDALN*ETR", "^CLU.*ETR")

if (any(is.na(c(col_group, col_age, col_acpp, col_clu)))) {
    stop("Could not locate required columns: 'Prostate cancer group', 'Age',
'ACPP_FLN*ESYK', 'CLU_EDALN*ETR'.")
}

# ---- Prepare data ----
df_sub <- df |>
  filter(.data[[col_group]] %in% c("AG", "NAG")) |>
  transmute(
    Group = factor(.data[[col_group]], levels = c("AG", "NAG")),  # columns order: AG,
NAG
    Age   = suppressWarnings(as.numeric(.data[[col_age]])),
    ACPP  = suppressWarnings(as.numeric(.data[[col_acpp]])),
    CLU   = suppressWarnings(as.numeric(.data[[col_clu]]))
  ) |>
  filter(is.finite(Age), is.finite(ACPP), is.finite(CLU)) |>
  # keep positives for log-scale on y
```

```
  mutate(
    ACPP = ifelse(ACPP > 0, ACPP, NA_real_),
    CLU  = ifelse(CLU  > 0, CLU,  NA_real_)
  ) |>
  tidyr::drop_na(ACPP, CLU)

# ---- Helper to compute Pearson on log10(y) and to build a panel ----
pearson_label <- function(x_age, y_raw) {
  # Pearson computed on log10(y) to match log y-axis
  ok <- is.finite(x_age) & is.finite(y_raw) & (y_raw > 0)
  x <- x_age[ok]; y <- y_raw[ok]
  n <- length(x)
  if (n < 3) {
    return(list(n = n, r = NA_real_, p = NA_real_, label = paste0("n=", n, " (too few)")))
  }
  ct <- suppressWarnings(cor.test(x, log10(y), method = "pearson"))
  r  <- unname(ct$estimate)
  p  <- ct$p.value
  label <- paste0("n=", n, "  r=", sprintf("%.3f", r), "  p=", format.pval(p, digits = 3))
  list(n = n, r = r, p = p, label = label)
}

# Build a panel for a single variable and single group
# y_name: "ACPP" or "CLU"; y_label: pretty y-axis label
make_panel <- function(dat_group, group_title, y_name, y_label, point_color =
"#1f77b4") {
  y <- dat_group[[y_name]]
  x <- dat_group$Age

  # Pearson on log10(y)
  st <- pearson_label(x, y)

  # Back-transformed regression line from lm(log10(y) ~ Age)
  # Create prediction frame only if we have >= 2 unique x
  pred_df <- NULL
  if (sum(is.finite(x) & is.finite(y)) >= 3 && length(unique(x)) >= 2) {
    fit <- lm(log10(y) ~ x)
    x_seq <- seq(min(x, na.rm = TRUE), max(x, na.rm = TRUE), length.out = 200)
    y_pred <- 10^(predict(fit, newdata = data.frame(x = x_seq)))
    pred_df <- data.frame(Age = x_seq, y_pred = y_pred)
  }

  # Annotation position: near top-left in data space
  x_pos <- quantile(x, 0.02, na.rm = TRUE)
  y_pos <- quantile(y, 0.95, na.rm = TRUE)

  g <- ggplot(dat_group, aes(x = Age, y = .data[[y_name]])) +
    geom_point(color = point_color, alpha = 0.9, size = 2.3) +
    { if (!is.null(pred_df)) geom_line(data = pred_df, aes(x = Age, y = y_pred),
                                       inherit.aes = FALSE, color = "black", linewidth = 0.9) } +
```

```
    scale_y_log10() +
    labs(
      title = group_title,
      x = "Age (years)",
      y = paste0(y_label, " (log scale)")
    ) +
    annotate("text", x = x_pos, y = y_pos, hjust = 0,
             label = st$label, size = 3.8, color = "black") +
    theme_minimal(base_size = 12) +
    theme(panel.grid.minor = element_blank())

  return(g)
}

# ---- Split data by group ----
ag_dat  <- df_sub %>% filter(Group == "AG")
nag_dat <- df_sub %>% filter(Group == "NAG")

# ---- Build 4 panels: rows = markers, columns = groups ----
# ACPP (row 1)
p_acpp_ag  <- make_panel(ag_dat,  "AG",  "ACPP", "ACPP_FLN*ESYK", point_color
= "#ff7f0e")
p_acpp_nag <- make_panel(nag_dat, "NAG", "ACPP", "ACPP_FLN*ESYK",
point_color = "#1f77b4")

# CLU (row 2)
p_clu_ag  <- make_panel(ag_dat,  "AG",  "CLU", "CLU_EDALN*ETR", point_color
= "#ff7f0e")
p_clu_nag <- make_panel(nag_dat, "NAG", "CLU", "CLU_EDALN*ETR",
point_color = "#1f77b4")

# ---- Combine and save ----
combined <- (p_acpp_ag | p_acpp_nag) / (p_clu_ag | p_clu_nag) +
  plot_annotation(
    title = "Correlation with Age — ACPP_FLN*ESYK and CLU_EDALN*ETR by
Group",
    subtitle = "Pearson r, p computed on log10(y); lines are back-transformed fits of
lm(log10(y) ~ Age)"
  )

ggsave(out_png, plot = combined, width = 12, height = 10, dpi = 180)
message("Saved figure to: ", out_png)
```

.

Q26: Generate a PCA score plot illustrating the distribution of periodic table elements based on their physicochemical properties (variables in columns D–K), using different colors to represent the categories specified in column L. Display the corresponding element symbols adjacent to each data point.

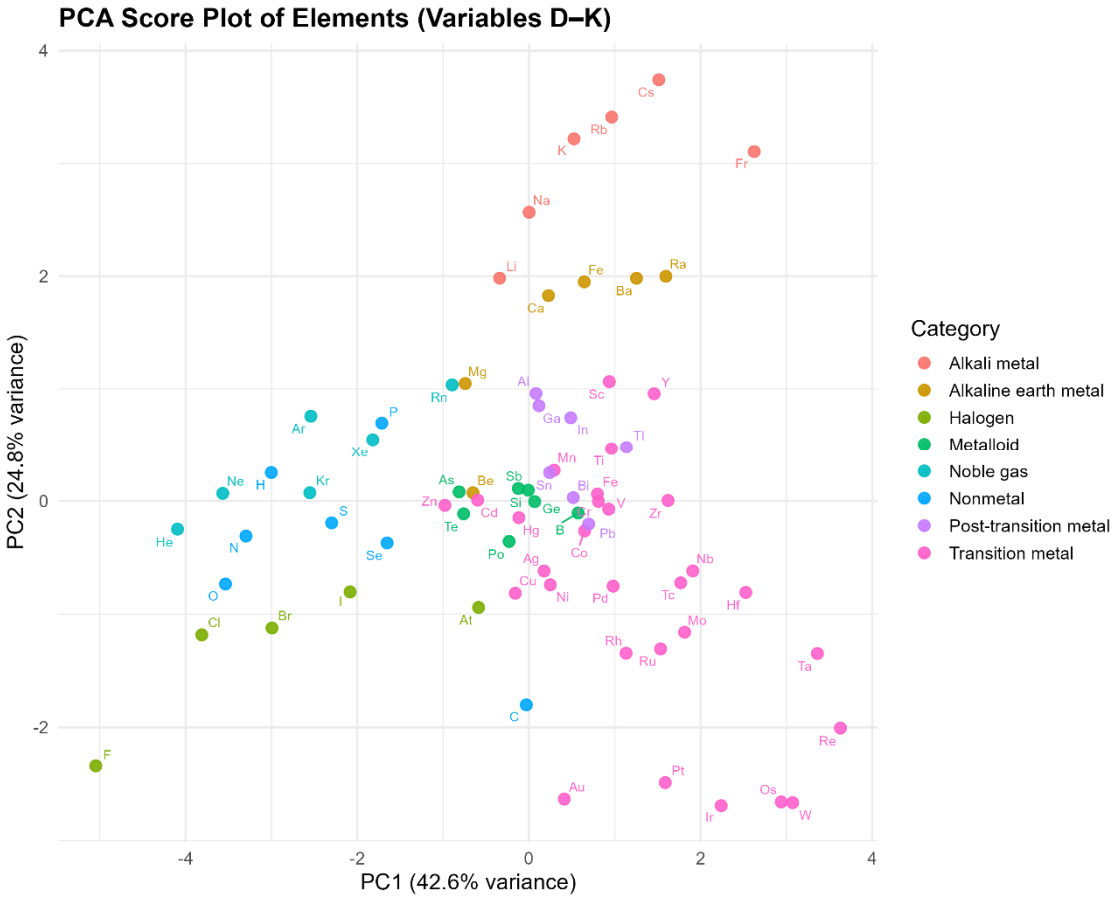


Figure S 11: ***PCA Score Plot of Physicochemical Properties for Chemical Elements.*** *The plot displays the first two principal components (PC1 and PC2) derived from seven standardized physicochemical variables: Atomic Mass, Electronegativity, Atomic Radius, Ionization Energy, Melting Point, Boiling Point, and Density. Each point represents an element, colored by its* ***GroupBlock*** *(e.g., Nonmetal, Halogen, Noble Gas, Transition Metal), and annotated with its chemical symbol.*

Explanation

This visualization uses **Principal Component Analysis (PCA)** to reduce multidimensional data into two components that capture the largest variance among the selected properties:

- **PC1 (horizontal axis)** explains the greatest variation in the dataset, largely influenced by size-related properties such as *Atomic Mass*, *Atomic Radius*, and *Density*. Elements with high PC1 scores tend to be heavier and denser (e.g., transition metals).
- **PC2 (vertical axis)** captures secondary variation, often linked to energetic properties like *Electronegativity* and *Ionization Energy*. Nonmetals and halogens typically score higher on PC2 due to their strong electron-attracting tendencies.

Color coding by GroupBlock reveals clear clustering:

- **Noble gases** cluster together due to similar low reactivity and high ionization energy.
- **Alkali metals** group on the opposite side, reflecting low ionization energy and large atomic radius.
- **Transition metals** form a dense cluster near the center, indicating intermediate values across most properties.

This plot helps visualize periodic trends and similarities among elements, making it easier to interpret chemical behavior in a multivariate context.

R scripts
These were the commands used in R to generate this image. These scripts were provided by Copilot.

```
# ----------- Load required packages -----------
library(readxl)
library(dplyr)
library(ggplot2)
library(ggrepel)

# ----------- Path to your spreadsheet -----------
file_path <- "C:/Users/marce/Documents/Digital/Spreadsheet 2.xlsx"

# Extract directory so we can save the plot in the same folder
save_dir <- dirname(file_path)

# ----------- Load dataset -----------
df <- read_excel(file_path)

# ----------- Select numeric variables D–K -----------
numeric_df <- df %>%
  select(4:11) %>%  # columns D–K
  mutate(across(everything(), as.numeric))

# Impute missing values with column mean
numeric_df <- numeric_df %>%
  mutate(across(everything(),
                ~ ifelse(is.na(.x), mean(.x, na.rm = TRUE), .x)))

# ----------- Extract symbols (column B) and categories (column L) -----------
```

```
symbols <- df[[2]]
categories <- df[[12]]

# ----------- PCA -----------
pca_res <- prcomp(numeric_df, scale. = TRUE)

scores <- as.data.frame(pca_res$x[, 1:2])
colnames(scores) <- c("PC1", "PC2")

scores$Symbol <- symbols
scores$Category <- as.factor(categories)

# Variance explained
pc1_var <- round(summary(pca_res)$importance[2,1] * 100, 1)
pc2_var <- round(summary(pca_res)$importance[2,2] * 100, 1)

# ----------- PCA Plot -----------
p <- ggplot(scores, aes(x = PC1, y = PC2, color = Category, label = Symbol)) +
  geom_point(size = 3, alpha = 0.9) +
  geom_text_repel(size = 3, max.overlaps = 50) +
  theme_minimal(base_size = 14) +
  labs(
    title = "PCA Score Plot of Elements (Variables D–K)",
    x = paste0("PC1 (", pc1_var, "% variance)"),
    y = paste0("PC2 (", pc2_var, "% variance)"),
    color = "Category"
  ) +
  theme(
    legend.position = "right",
    plot.title = element_text(face = "bold")
  )

# ----------- Save the figure to the same folder as the spreadsheet -----------
output_path <- file.path(save_dir, "PCA_elements_plot.png")

ggsave(output_path, p, width = 10, height = 8, dpi = 300)

# Confirmation
cat("✓ PCA figure saved at:\n", output_path, "\n")
```

Q27: Generate a PCA loading plot illustrating the relationships between the original variables and the principal components, emphasizing both the magnitude and direction of their contributions by representing each variable as a normalized vector within a unit-radius correlation circle.

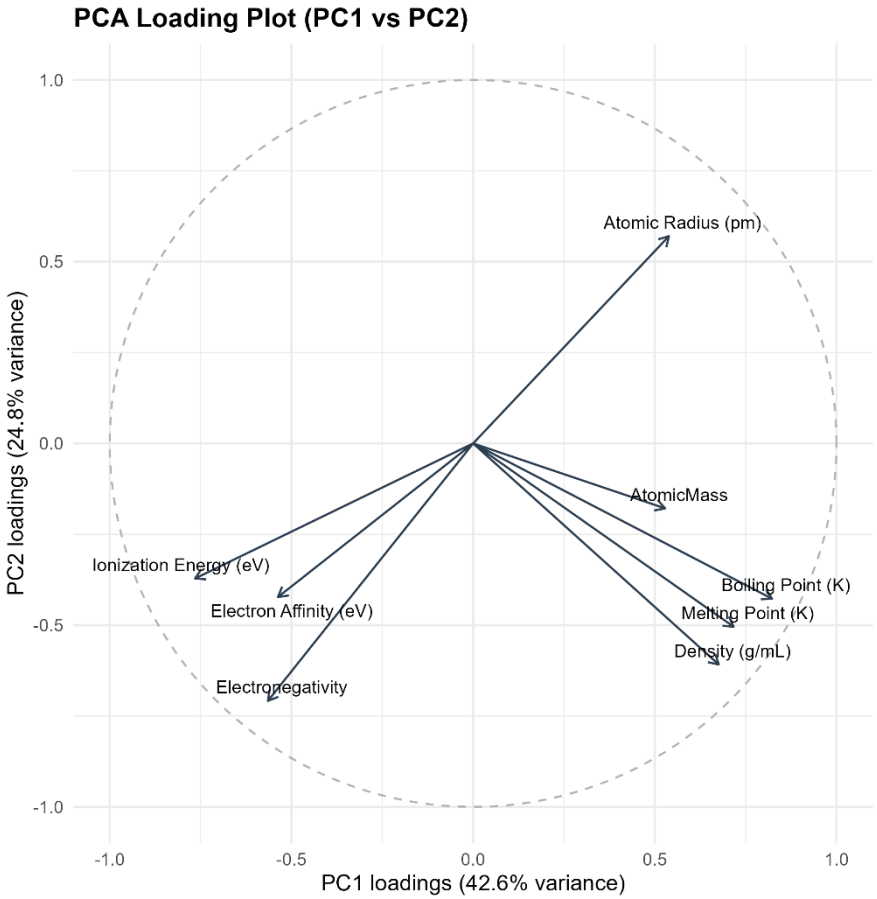


Figure S12: ***PCA Loading Plot (Correlation Circle) for Physicochemical Properties*** *The diagram shows the correlations of seven standardized physicochemical variables—Atomic Mass, Electronegativity, Atomic Radius, Ionization Energy, Melting Point, Boiling Point, and Density—with the first two principal components (PC1 and PC2). Each arrow represents a variable as a vector inside a unit circle, indicating its contribution and direction in the reduced PCA space.*

Explanation

Figure S12 is a **correlation circle**, a common way to interpret PCA loadings:

- **Unit circle**: The grey circle has radius 1, representing the maximum possible correlation. Vectors closer to the circle are well represented by PC1 and PC2; those near the center contribute less to these components.
- **Vectors (arrows)**:

    - The **direction** of each arrow shows how the variable aligns with the principal components. Variables pointing in similar directions are positively correlated; opposite directions indicate negative correlation.
    - The **length** of the arrow reflects how strongly the variable is captured by the first two PCs (communalities). Longer arrows = higher representation.
- **Color coding**: Warmer colors indicate stronger representation (longer vectors), cooler colors indicate weaker representation.
- **Interpretation of axes**:
    - **PC1** (horizontal) is dominated by size-related properties (*Atomic Mass*, *Density*, *Boiling Point*), as these vectors cluster toward the right.
    - **PC2** (vertical) is influenced by energetic properties (*Electronegativity*, *Ionization Energy*), which point upward.
- **Relationships among variables**:
    - *Atomic Mass*, *Density*, and *Boiling Point* cluster together, showing strong positive correlation.
    - *Electronegativity* and *Ionization Energy* share a similar direction, indicating they vary together but differently from size-related properties.
    - *Atomic Radius* points opposite to *Electronegativity*, reflecting their inverse relationship.

This visualization helps identify which properties drive the main dimensions of variation and how they relate to each other in multivariate space.

These were the R scripts for generating the loading plot. The scripts were provided by Microsoft Copilot

Q28: Create a Pearson correlation heatmap showing the relationships between the physicochemical properties of the elements in the periodic table

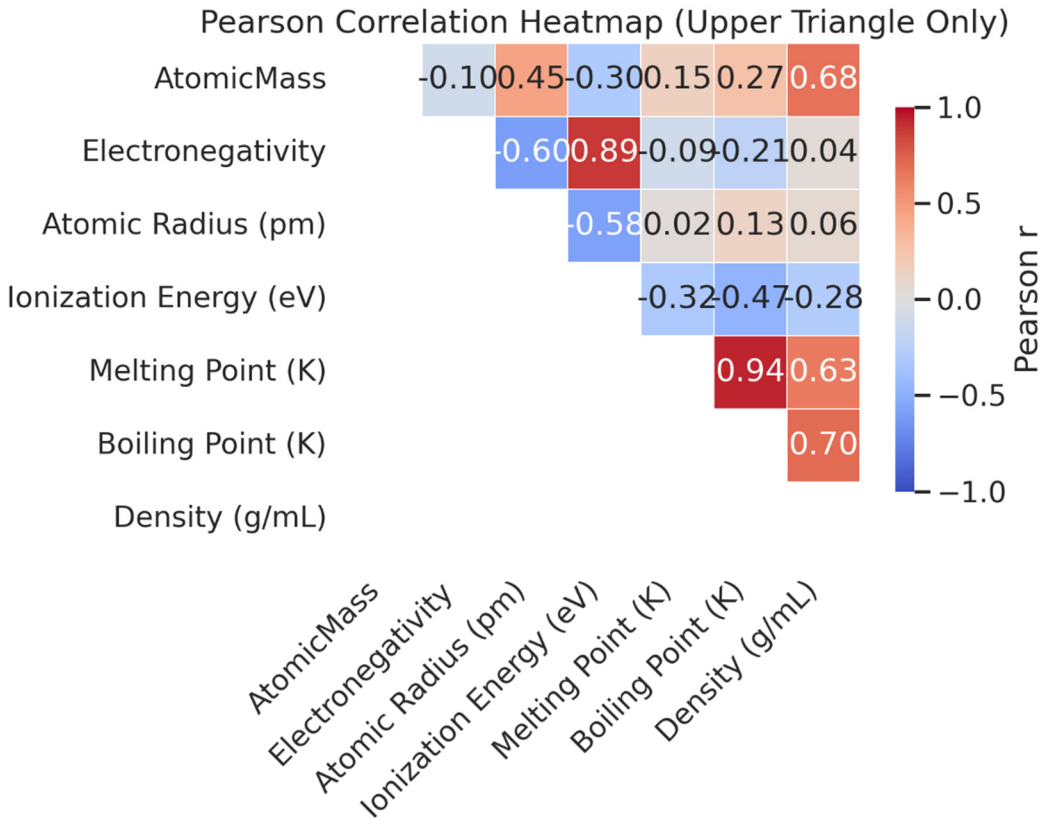


Figure S13: Upper-triangle heatmap of pairwise Pearson correlations among seven physicochemical properties of the elements. Cells display coefficients from −1 (strong negative) to +1 (strong positive). The lower triangle and diagonal were masked to eliminate duplicated information and self-correlations.

Q29: Construct a correlation plot comparing the Boiling Point and Melting Point of elements. Shown the elements symbol up the balls and the trend line with the Pearson correlation coefficient

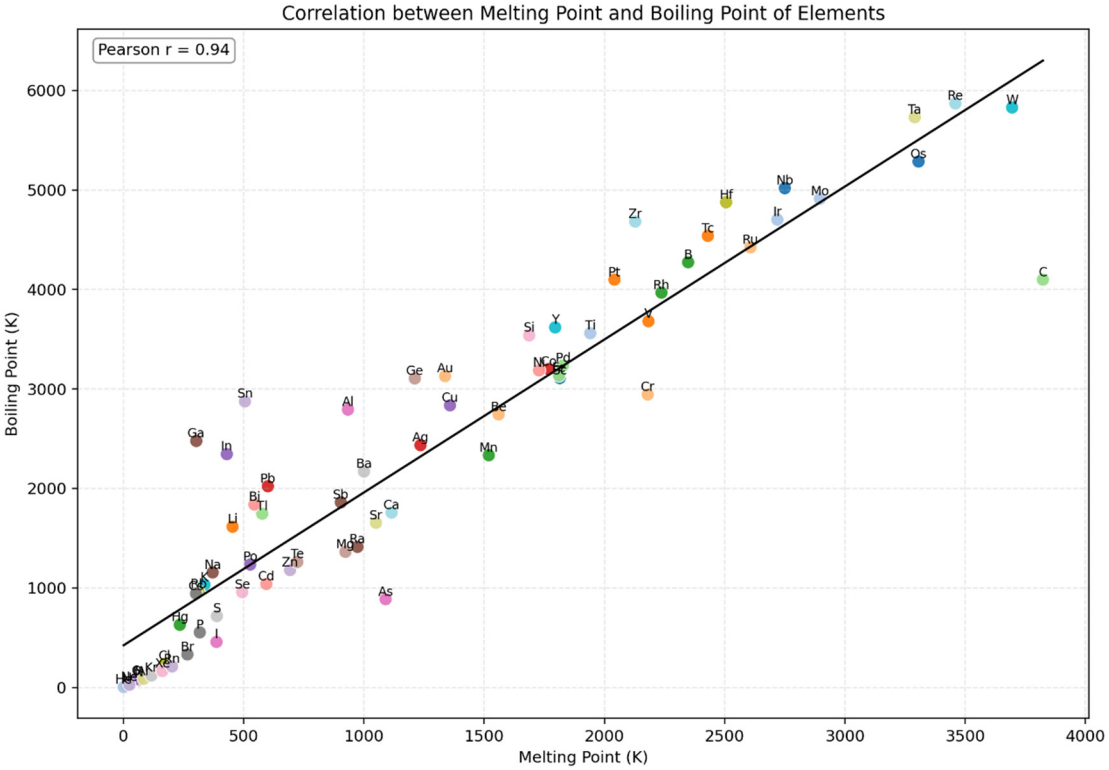


*Figure S 14**Linear Relationship Between Melting Point and Boiling Point of Elements***

Scatter plot of melting point (x-axis) versus boiling point (y-axis) shown in Figure S 14, with chemical symbols shown above each marker. The black line represents the best-fit linear regression, and the inset reports the Pearson correlation coefficient (r = 0.94), indicating a strong positive relationship: elements with higher melting points generally also exhibit higher boiling points.

Q30: Construct a correlation plot comparing Ionization Energy and Electron Affinity of elements. Shown the elements symbol up the balls and the trend line with the Pearson correlation coefficient

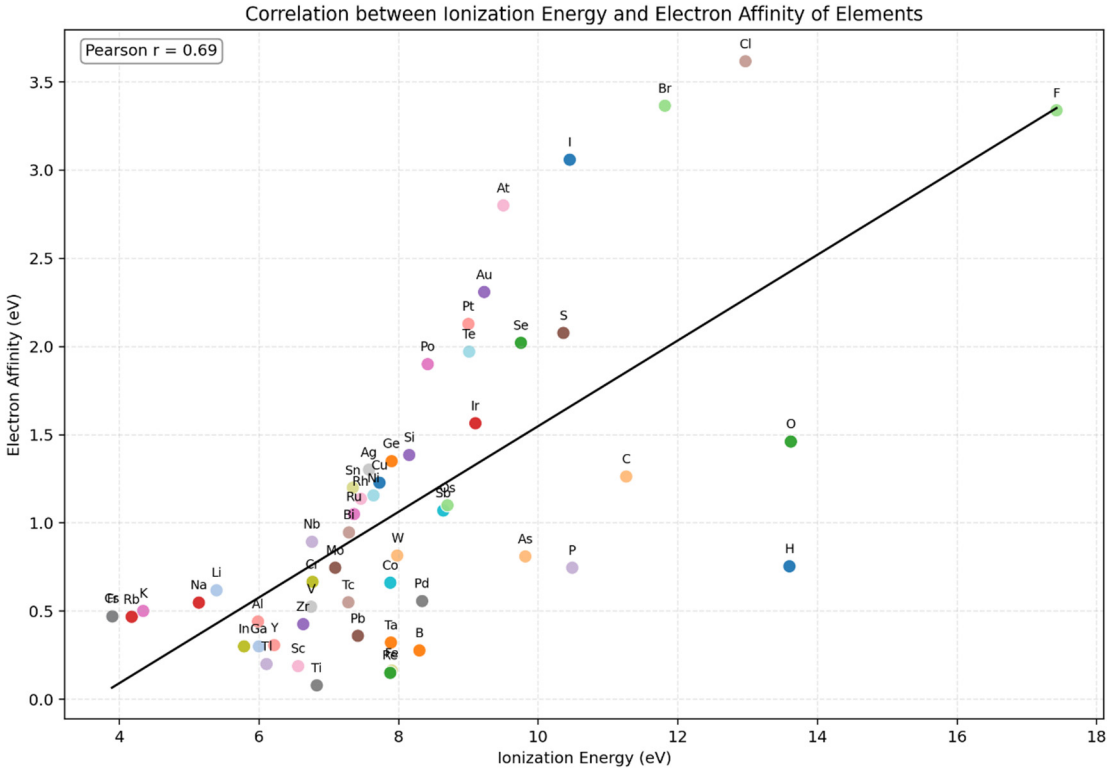


Figure S15: ***Correlation Between Ionization Energy and Electron Affinity of Elements***

Scatter plot (Figure S15) showing each element with its chemical symbol positioned above the marker. The black line represents the best-fit linear regression, and the inset reports the Pearson correlation coefficient (r = 0.69), indicating a moderate positive association. Elements with higher first ionization energies generally exhibit higher electron affinities, although group-specific behaviors (e.g., noble gases and certain metals) lead to noticeable deviations from linearity.

Q31: Construct a correlation plot comparing Ionization Energy and Electronegativity of elements. Shown the elements symbol up the balls and the trend line with the Pearson correlation coefficient

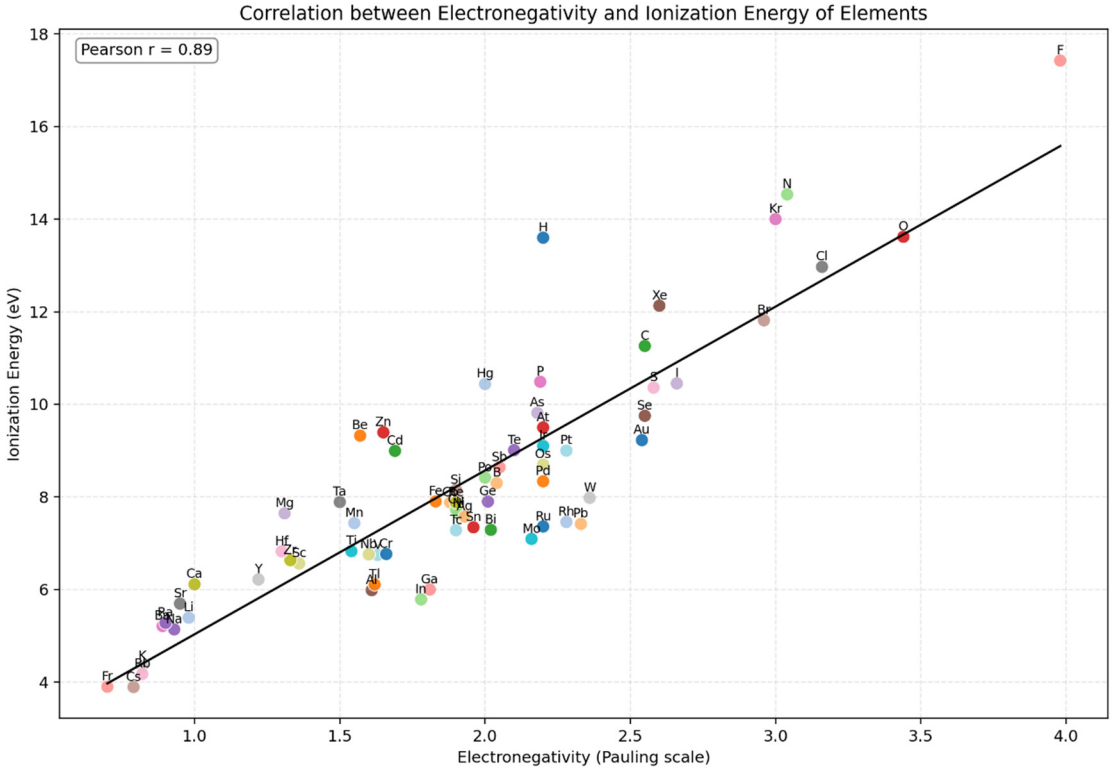


Figure S16: ***Correlation Between Electronegativity and Ionization Energy of Elements***

Scatter plot (Figure S16) showing each element with its chemical symbol above the marker. The black line represents the best-fit linear regression, and the inset reports the Pearson correlation coefficient (r = 0.89), indicating a strong positive association: elements with higher electronegativity (Pauling scale) generally exhibit higher first ionization energy (eV).

Q32: Does a comparison of physicochemical properties among groups (Column E; metals, nonmetals, and metalloids) using box plot, show all the box plots in the same image

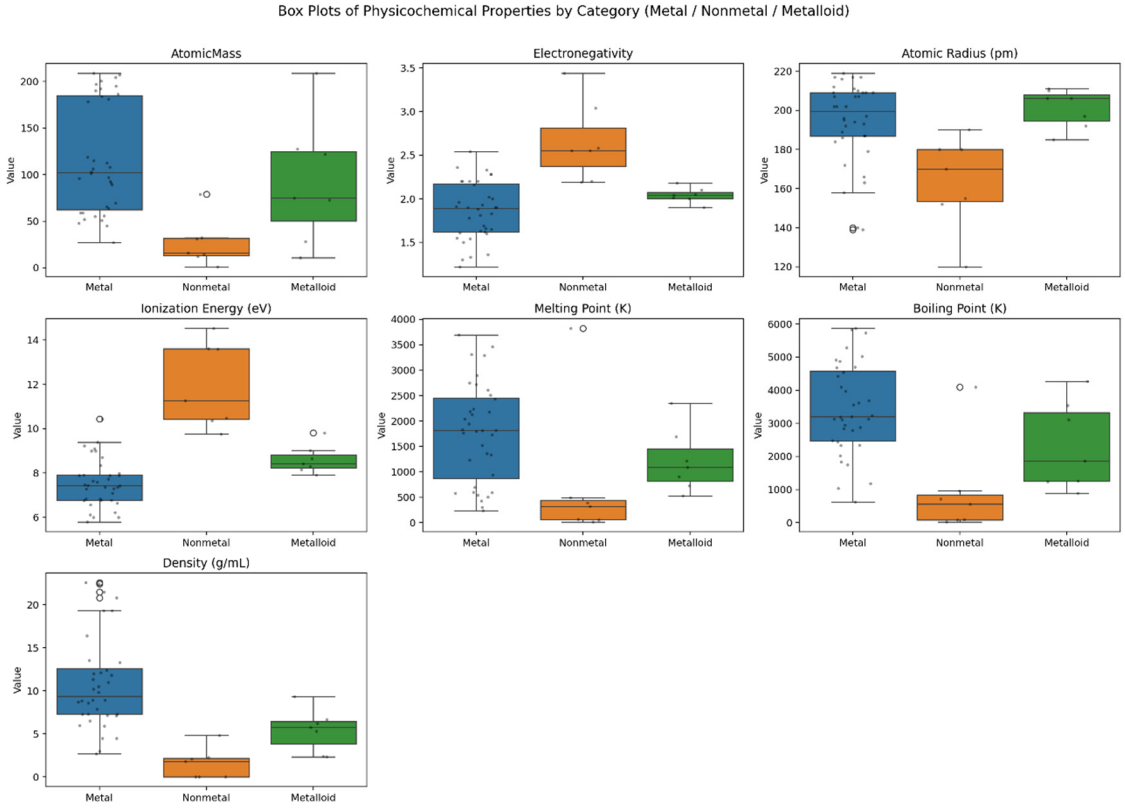


Figure S17: ***Box Plots of Physicochemical Properties by Element Category (Metal, Nonmetal, Metalloid)*** *Each panel compares the distribution of a specific property—Atomic Mass, Electronegativity, Atomic Radius, Ionization Energy, Melting Point, Boiling Point, and Density—across three categories:* ***Metals****,* ***Nonmetals****, and* ***Metalloids****. Boxes show the median and interquartile range (IQR), whiskers indicate variability outside the upper and lower quartiles, and points represent individual elements.*

Figure S17 summarizes how fundamental properties differ among the three major chemical categories:

- **Metals** generally exhibit:
  - **Higher Atomic Mass and Density**, reflecting their heavier nuclei and compact structures.
  - **Lower Electronegativity and Ionization Energy**, consistent with their tendency to lose electrons and form cations.
  - **Wide ranges in melting and boiling points**, with some extremely high values (e.g., refractory metals).
- **Nonmetals** tend to:
  - **Higher Electronegativity and Ionization Energy**, indicating strong electron-attracting ability.
  - **Lower Atomic Mass and Density**, as many are light elements or gases.

    - **Lower melting and boiling points**, except for network solids like carbon.
- **Metalloids** occupy intermediate positions:
    - Their properties often bridge metals and nonmetals, showing moderate electronegativity and ionization energy.
    - Physical properties (density, melting point) vary widely, reflecting mixed bonding characteristics.

Interpretation tip: Overlapping boxes suggest shared ranges, while distinct medians highlight systematic differences. Outliers often correspond to unique elements (e.g., noble gases for boiling point, tungsten for melting point).

Q33: Build a PLS-DA score plot using the variables in columns D–G, applying distinct colors and marker symbols to differentiate the prostate cancer groups listed in column B.

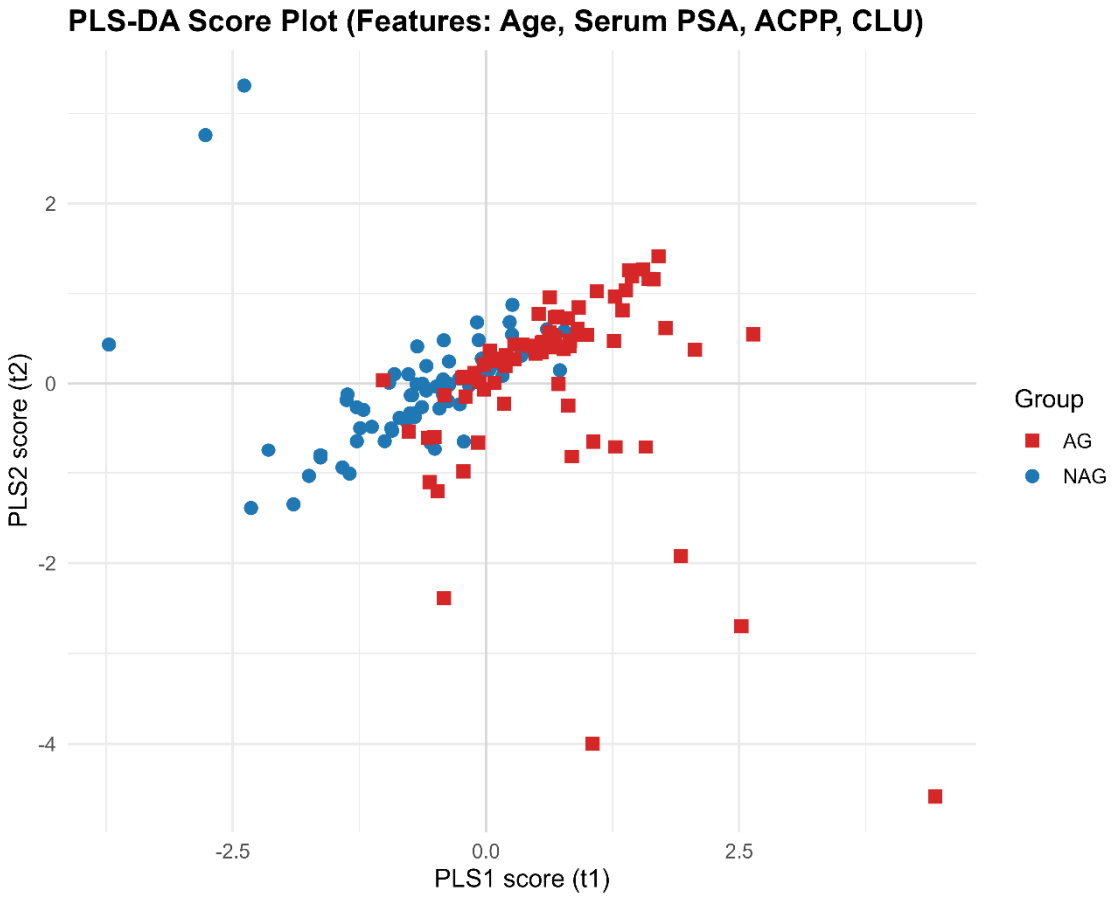


Figure S 18: PLS-DA score plot using columns D–G (Age, Serum PSA, ACPP_FLNESYK, CLU_EDALNETR) to discriminate prostate cancer groups (column B).

Samples were modeled with a **two-component PLS-DA (**Figure S 18**)** , projecting each case onto **Latent Variable 1 (LV1)** and **Latent Variable 2 (LV2)**. Points are styled by group: **AG** (blue circles, $n = 73$) and **NAG** (orange squares, $n = 69$). The axes show **X-scores** (PLS projection of the standardized predictors). Annotations include the **approximate $R^2(y) = 0.400$**, indicating that the two latent variables collectively explain ~40% of the variance in the binary group label (AG vs NAG). Axes are centered at zero to aid visual comparison; gridlines and a legend are included for readability.

R scripts provided by Copilot to generate Figure S18

```
# 1. LOAD LIBRARIES
library(ggplot2)
library(reshape2)

# 2. CREATE THE DATA
# Based on the results: TN=55, FP=14, FN=16, TP=57
```

```
cm_data <- matrix(c(55, 14, 16, 57), nrow = 2, byrow = TRUE)
colnames(cm_data) <- c("NAG", "AG")
rownames(cm_data) <- c("NAG", "AG")

# Melt the matrix for ggplot
cm_melted <- melt(cm_data)
colnames(cm_melted) <- c("Actual", "Predicted", "Value")

# 3. GENERATE THE PLOT
p_cm <- ggplot(cm_melted, aes(x = Predicted, y = Actual, fill = Value)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "#ebf2fb", high = "#2b7bba") +
  geom_text(aes(label = Value), color = "black", size = 8, fontface = "bold") +
  theme_minimal() +
  labs(title = "Confusion Matrix (5-Fold CV)",
       subtitle = "Aggressive (AG) vs Non-Aggressive (NAG)",
       x = "Predicted Group",
       y = "Actual Group") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 12),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        legend.position = "none")

# 4. SAVE THE IMAGE
output_cm_path <- "C:/Users/marce/Documents/Digital/Confusion_Matrix_Plot.png"
ggsave(output_cm_path, plot = p_cm, width = 6, height = 5, dpi = 300)

print(paste("Confusion matrix image saved to:", output_cm_path))
```

Q34: Generate a confusion matrix based on 5-fold cross-validation results, treating correctly classified aggressive (AG) samples as true positives and correctly classified non-aggressive (NAG) samples as true negatives. Build the model using two latent variables (LVs) and apply a classification threshold of 0.5.

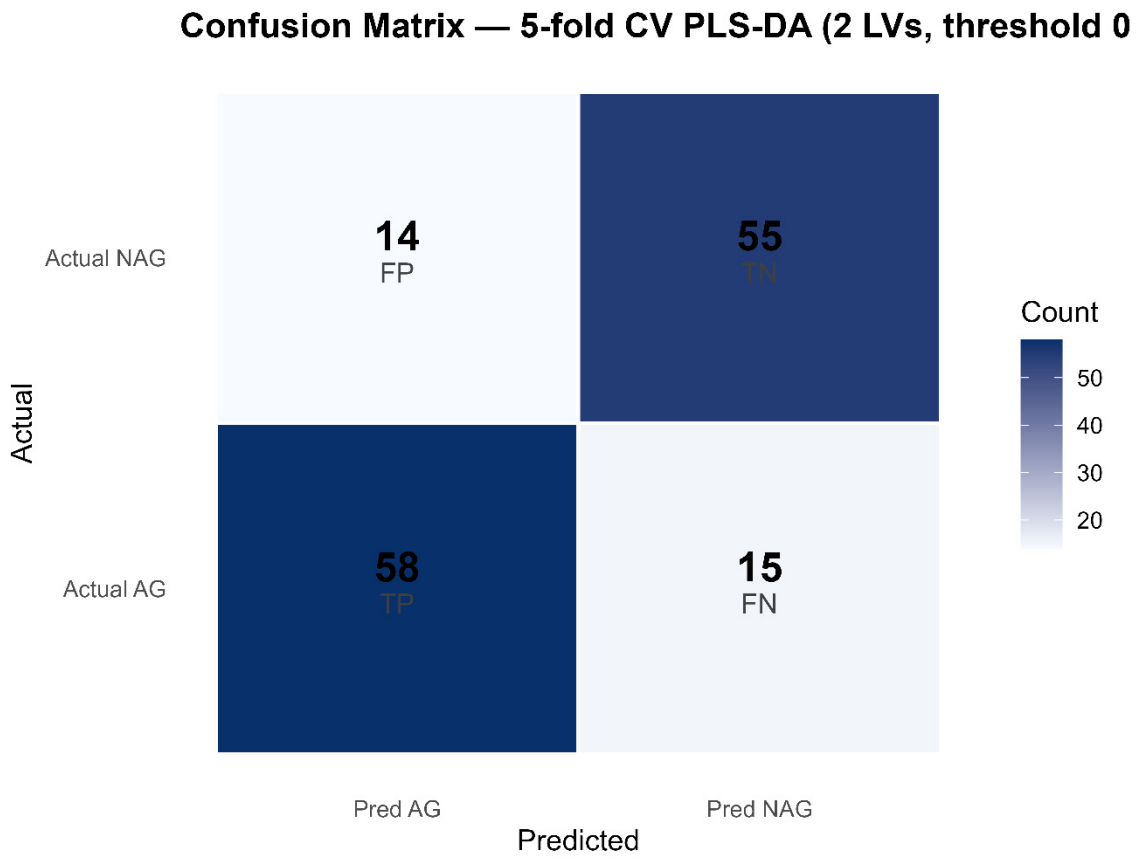


Figure S19: 5-fold stratified cross-validation confusion matrix for a two-component PLS-DA classifier built from columns D–G (Age, Serum PSA, ACPP_FLNESYK, CLU_EDALNETR), treating AG as the positive class and NAG as the negative class.

Aggregated over 5 folds, the matrix (Figure S19) shows **TP = 57**, **FN = 16**, **FP = 17**, **TN = 52** (total **n = 142**; AG = 73, NAG = 69). Overall **Accuracy = 0.768**, **Precision (PPV) = 0.770**, **Recall (Sensitivity) = 0.781**, **Specificity = 0.754**, and **F1 = 0.776**. Predictions were obtained by thresholding the PLS-DA output at **0.5** (AG = 1, NAG = 0) in each fold; predictors were **z-scored per training fold** before modeling.
The R scripts provided by Copilot to generate this image in R were:

```
# ============================================
# 5-fold CV PLS-DA + Confusion Matrix Image
# ============================================
```

```
# 1) Packages ------------------------------------------------------------
# If needed:
# install.packages(c("readxl", "pls", "ggplot2"))
library(readxl)
library(pls)
library(ggplot2)

# 2) Paths ---------------------------------------------------------------
xlsx_path <- "C:\\Users\\marce\\Documents\\Digital\\Spreadsheet 1.xlsx"

out_dir       <- dirname(xlsx_path)
pred_csv_path  <- file.path(out_dir, "plsda_cv5_predictions.csv")
cm_csv_path    <- file.path(out_dir, "plsda_cv5_confusion_matrix.csv")
cm_png_path    <- file.path(out_dir, "plsda_cv5_confusion_matrix.png")

# 3) Load and select columns ---------------------------------------------
df <- read_excel(xlsx_path)
names(df) <- trimws(names(df))

# By position: A=Sample ID, B=Group, D:G=features
sample_col_idx <- 1
group_col_idx  <- 2
x_cols_idx     <- 4:7

work <- df[, c(sample_col_idx, group_col_idx, x_cols_idx)]
work <- work[complete.cases(work), , drop = FALSE]
colnames(work)[1:2] <- c("SampleID", "Group")

# Make feature names formula-safe (x1..x4)
X_all <- work[, -(1:2), drop = FALSE]
colnames(X_all) <- paste0("x", seq_len(ncol(X_all)))
work <- cbind(work[, 1:2, drop = FALSE], X_all)

# Encode class: AG = 1 (positive), NAG = 0 (negative)
y <- ifelse(work$Group == "AG", 1L, 0L)

# 4) Stratified 5-fold split ---------------------------------------------
make_stratified_folds <- function(y, k = 5, seed = 42) {
  set.seed(seed)
  y <- as.factor(y)
  folds <- vector("list", k)
  idx_pos <- which(y == 1)
  idx_neg <- which(y == 0)
  idx_pos <- sample(idx_pos); idx_neg <- sample(idx_neg)

  split_idx <- function(idx, k) {
    n <- length(idx)
    if (n == 0) return(replicate(k, integer(0), simplify = FALSE))
    base <- floor(n / k); rem <- n %% k
```

```
    sizes <- rep(base, k); if (rem > 0) sizes[1:rem] <- sizes[1:rem] + 1
    starts <- cumsum(c(1, head(sizes, -1)))
    mapply(function(st, sz) idx[st:(st + sz - 1)], starts, sizes, SIMPLIFY = FALSE)
  }
  pos_folds <- split_idx(idx_pos, k)
  neg_folds <- split_idx(idx_neg, k)
  for (i in seq_len(k)) folds[[i]] <- sort(c(pos_folds[[i]], neg_folds[[i]]))
  folds
}
k <- 5
folds <- make_stratified_folds(y, k = k, seed = 42)

# 5) Cross-validated predictions (2 LVs, threshold 0.5) ----------------------
n <- nrow(work)
y_true <- y
y_prob <- rep(NA_real_, n)

for (i in seq_len(k)) {
  test_idx  <- folds[[i]]
  train_idx <- setdiff(seq_len(n), test_idx)

  train_df <- data.frame(y = y_true[train_idx], work[train_idx, colnames(X_all), drop =
FALSE])
  test_df  <- data.frame(work[test_idx,  colnames(X_all), drop = FALSE])

  # Fit PLS with 2 components (scaling learned on training fold)
  fit <- plsr(y ~ ., data = train_df, ncomp = 2, scale = TRUE, method = "oscorespls")

  # Continuous AG score for held-out fold
  y_hat <- as.vector(predict(fit, newdata = test_df, ncomp = 2))
  y_prob[test_idx] <- y_hat
}

# 0.5 threshold to get class labels
y_pred <- ifelse(y_prob >= 0.5, 1L, 0L)

# 6) Confusion matrix and metrics ------------------------------------------
TP <- sum(y_true == 1 & y_pred == 1)  # correctly classified AG
FN <- sum(y_true == 1 & y_pred == 0)  # AG -> NAG
FP <- sum(y_true == 0 & y_pred == 1)  # NAG -> AG
TN <- sum(y_true == 0 & y_pred == 0)  # correctly classified NAG

cm <- matrix(c(TP, FN, FP, TN), nrow = 2, byrow = TRUE)
dimnames(cm) <- list(
  Actual    = c("Actual AG", "Actual NAG"),
  Predicted = c("Pred AG",   "Pred NAG")
)

accuracy    <- (TP + TN) / (TP + TN + FP + FN)
sensitivity <- if ((TP + FN) > 0) TP / (TP + FN) else NA_real_  # AG recall
```

```
specificity <- if ((TN + FP) > 0) TN / (TN + FP) else NA_real_  # NAG TNR

# 7) Save numeric outputs ------------------------------------------------
# Per-sample predictions
pred_df <- data.frame(
  SampleID      = work$SampleID,
  Group         = work$Group,
  AG_true       = y_true,
  AG_pred_prob  = y_prob,
  AG_pred_label = y_pred
)
write.csv(pred_df, pred_csv_path, row.names = FALSE)

# Confusion matrix (long form) for record
cm_df <- data.frame(
  row = rep(c("Actual AG", "Actual NAG"), each = 2),
  col = rep(c("Pred AG", "Pred NAG"), times = 2),
  count = c(TP, FN, FP, TN)
)
write.csv(cm_df, cm_csv_path, row.names = FALSE)

# 8) High-resolution confusion matrix image (ggplot) -------------------------
# Build plotting frame with cell labels (TP, FN, FP, TN)
cm_plot <- data.frame(
  Actual    = factor(c("Actual AG", "Actual AG",  "Actual NAG", "Actual NAG"),
                     levels = c("Actual AG", "Actual NAG")),
  Predicted = factor(c("Pred AG",   "Pred NAG",   "Pred AG",    "Pred NAG"),
                     levels = c("Pred AG", "Pred NAG")),
  Count     = c(TP, FN, FP, TN),
  CellType  = c("TP", "FN", "FP", "TN")
)

# Ensure top-left cell is TP in the image (Actual on Y, Pred on X, AG row on top)
p_cm <- ggplot(cm_plot, aes(x = Predicted, y = Actual, fill = Count)) +
  geom_tile(color = "white", linewidth = 0.7) +
  # Counts in bold, cell type beneath in smaller grey
  geom_text(aes(label = Count), size = 6, fontface = "bold", color = "black", vjust = -
0.25) +
  geom_text(aes(label = CellType), size = 4, color = "grey25", vjust = 1.25) +
  scale_fill_gradient(low = "#f7fbff", high = "#08306b") +
  labs(
    title = "Confusion Matrix — 5-fold CV PLS-DA (2 LVs, threshold 0.5)",
    x = "Predicted",
    y = "Actual",
    fill = "Count"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    legend.position = "right",
    plot.title = element_text(face = "bold"),
```

```
    panel.grid = element_blank()
  )

# Save high-resolution PNG (600 dpi)
ggsave(filename = cm_png_path, plot = p_cm, width = 6.5, height = 5.0, dpi = 600, bg
= "white")

# 9) Console summary ------------------------------------------------------
cat("\n=== 5-fold CV PLS-DA (2 LVs) @ threshold 0.5 ===\n")
print(cm)
cat(sprintf("\nTP = %d | TN = %d | FP = %d | FN = %d\n", TP, TN, FP, FN))
cat(sprintf("Accuracy    = %.3f\n", accuracy))
cat(sprintf("Sensitivity = %.3f (AG recall)\n", sensitivity))
cat(sprintf("Specificity = %.3f (NAG true negative rate)\n", specificity))
cat("\nSaved predictions:     ", pred_csv_path, "\n", sep = "")
cat("Saved confusion matrix:", cm_csv_path,   "\n", sep = "")
cat("Saved CM image (PNG):  ", cm_png_path,   "\n", sep = "")
```

Figure 2

R scripts to build Figure 2
Those scripts were provided by Copilot

```
# ============================================
# Overlaid histograms (AG vs NAG), no KDE
# Variables: ACPP (Light/Heavy if available, else total), CLU_EDALN*ETR, Serum PSA
# Layout: rows = markers, columns = Linear (left) and log10 (right)
# Output: single PNG saved next to the spreadsheet
# ============================================

# If needed:
# install.packages(c("readxl", "dplyr", "ggplot2", "patchwork", "stringr"))

library(readxl)
library(dplyr)
library(ggplot2)
library(patchwork)
library(stringr)

# ---- Path to your Excel file ----
xlsx_path <- "C:\\Users\\marce\\Documents\\Digital\\Spreadsheet 1.xlsx"
out_dir   <- dirname(xlsx_path)
out_png   <- file.path(out_dir, "histograms_linear_and_log_all.png")

# ---- Load and prep data ----
df <- read_excel(xlsx_path)
names(df) <- trimws(names(df))  # remove leading/trailing spaces

# Helper to find a column by (case-insensitive) regex
find_col <- function(pattern) {
  hits <- grep(pattern, names(df), ignore.case = TRUE, value = TRUE)
  if (length(hits) > 0) hits[1] else NA_character_
}

# Identify columns
col_group <- if ("Prostate cancer group" %in% names(df)) {
  "Prostate cancer group"
} else {
  find_col("^Prostate\\s*cancer\\s*group$")
}

# Try ACPP Light/Heavy; otherwise fall back to a single ACPP column
col_acpp_light <- find_col("ACPP.*(Light|Lt\\b|\\bL\\b)")
col_acpp_heavy <- find_col("ACPP.*(Heavy|Hv\\b|\\bH\\b)")
col_acpp_total <- find_col("^ACPP.*ESYK")

# CLU and PSA
```

```
col_clu <- find_col("^CLU.*ETR")
col_psa <- if ("Serum PSA" %in% names(df)) "Serum PSA" else
find_col("^Serum\\s*PSA$")

# Build the variable list (each item: list(col_name, label))
vars <- list()
if (!is.na(col_acpp_light) && !is.na(col_acpp_heavy)) {
  vars <- append(vars, list(list(col = col_acpp_light, label = "ACPP_FLN*ESYK
(Light)")))
  vars <- append(vars, list(list(col = col_acpp_heavy, label = "ACPP_FLN*ESYK
(Heavy)")))
} else if (!is.na(col_acpp_total)) {
  vars <- append(vars, list(list(col = col_acpp_total, label = "ACPP_FLN*ESYK")))
}
if (!is.na(col_clu)) vars <- append(vars, list(list(col = col_clu, label =
"CLU_EDALN*ETR")))
if (!is.na(col_psa)) vars <- append(vars, list(list(col = col_psa, label = "Serum PSA")))

if (is.na(col_group) || length(vars) == 0) {
  stop("Required columns not found. Please check the sheet and column names.")
}

# Filter to AG/NAG and keep numeric
df_sub <- df |>
  filter(.data[[col_group]] %in% c("AG", "NAG"))

# ---- Colors and theme ----
c_nag <- "#1f77b4"
c_ag  <- "#ff7f0e"
base_theme <- theme_minimal(base_size = 12) +
  theme(panel.grid.minor = element_blank())

# Utility: compute Freedman–Diaconis binwidth from a numeric vector
fd_binwidth <- function(x) {
  x <- x[is.finite(x)]
  if (length(x) < 2 || diff(range(x)) == 0) return(NA_real_)
  nbins <- grDevices::nclass.FD(x)
  nbins <- max(nbins, 10)  # ensure enough bins
  diff(range(x)) / nbins
}

# Build one linear and one log10 plot per variable, then assemble
plots_linear <- list()
plots_log10  <- list()

for (v in vars) {
  col_name <- v$col
  label    <- v$label

  dat <- df_sub |>
```

```
    transmute(
      Group = factor(.data[[col_group]], levels = c("NAG", "AG")),
      value = suppressWarnings(as.numeric(.data[[col_name]]))
    ) |>
    filter(!is.na(value))

  # Split for binning fairness (FD on combined distribution)
  combined <- dat$value
  bw_lin   <- fd_binwidth(combined)

  # Prepare log10 (drop non-positive for log)
  dat_log <- dat |>
    filter(value > 0) |>
    mutate(value_log10 = log10(value))

  bw_log <- fd_binwidth(dat_log$value_log10)

  # Linear (no KDE): overlaid histograms with density
  p_lin <- ggplot(dat, aes(x = value, fill = Group, color = Group)) +
    geom_histogram(aes(y = after_stat(density)),
                   position = "identity", alpha = 0.35,
                   binwidth = bw_lin, boundary = min(combined, na.rm = TRUE)) +
    scale_fill_manual(values = c("NAG" = c_nag, "AG" = c_ag)) +
    scale_color_manual(values = c("NAG" = c_nag, "AG" = c_ag)) +
    labs(title = paste0(label, " — Linear Scale"),
         x = label, y = "Density", fill = "Group", color = "Group") +
    base_theme +
    theme(legend.position = "top")

  # Log10 (no KDE): plot transformed values
  p_log <- ggplot(dat_log, aes(x = value_log10, fill = Group, color = Group)) +
    geom_histogram(aes(y = after_stat(density)),
                   position = "identity", alpha = 0.35,
                   binwidth = bw_log, boundary = min(dat_log$value_log10, na.rm = TRUE))
+
    scale_fill_manual(values = c("NAG" = c_nag, "AG" = c_ag)) +
    scale_color_manual(values = c("NAG" = c_nag, "AG" = c_ag)) +
    labs(title = paste0(label, " — log10 Scale"),
         x = paste0("log10(", label, ")"), y = "Density",
         fill = "Group", color = "Group") +
    base_theme +
    theme(legend.position = "top")

  plots_linear[[length(plots_linear) + 1]] <- p_lin
  plots_log10[[length(plots_log10) + 1]]  <- p_log
}

# Stack linears in the left column and log10s in the right column
left_col  <- Reduce(`/`, plots_linear)
right_col <- Reduce(`/`, plots_log10)
```

```
combined  <- left_col | right_col

# Optional overall title with counts
n_ag  <- sum(df_sub[[col_group]] == "AG", na.rm = TRUE)
n_nag <- sum(df_sub[[col_group]] == "NAG", na.rm = TRUE)

combined <- combined + plot_annotation(
  title = paste0("Overlayed Histograms (no KDE) — NAG (n=", n_nag, ") vs AG (n=",
n_ag, ")")
)

# ---- Save single figure ----
ggsave(out_png, plot = combined, width = 14, height = 4.2 * length(vars), dpi = 180)
message("Saved combined figure to: ", out_png)
```

Figure 3

The R scripts provided by Copilot to generate Figure 3

```
# ============================================
# Box plots (AG vs NAG), Linear and log10 panels, no KDE (box plots only)
# Variables: ACPP (Light/Heavy if available, else total), CLU_EDALN*ETR, Serum PSA
# Output: single PNG saved next to the spreadsheet
# ============================================

# If needed:
# install.packages(c("readxl", "dplyr", "ggplot2", "patchwork", "stringr"))

library(readxl)
library(dplyr)
library(ggplot2)
library(patchwork)
library(stringr)

# ---- Path to your Excel file ----
xlsx_path <- "C:\\Users\\marce\\Documents\\Digital\\Spreadsheet 1.xlsx"
out_dir   <- dirname(xlsx_path)
out_png   <- file.path(out_dir, "histograms_linear_and_log_all_boxplots.png")

# ---- Load and prep data ----
df <- read_excel(xlsx_path)
names(df) <- trimws(names(df))  # remove leading/trailing spaces

# Helper: find first matching column by (case-insensitive) regex
find_col <- function(pattern) {
  hits <- grep(pattern, names(df), ignore.case = TRUE, value = TRUE)
  if (length(hits) > 0) hits[1] else NA_character_
}

# Identify columns
col_group <- if ("Prostate cancer group" %in% names(df)) {
  "Prostate cancer group"
} else {
  find_col("^Prostate\\s*cancer\\s*group$")
}

# Try ACPP Light/Heavy; otherwise fall back to a single ACPP column
col_acpp_light <- find_col("ACPP.*(Light|Lt\\b|\\bL\\b)")
col_acpp_heavy <- find_col("ACPP.*(Heavy|Hv\\b|\\bH\\b)")
col_acpp_total <- find_col("^ACPP.*ESYK")

# CLU and PSA
col_clu <- find_col("^CLU.*ETR")
col_psa <- if ("Serum PSA" %in% names(df)) "Serum PSA" else
find_col("^Serum\\s*PSA$")
```

```
# Build the variable list (list of lists: each with 'col' and 'label')
vars <- list()
if (!is.na(col_acpp_light) && !is.na(col_acpp_heavy)) {
  vars <- append(vars, list(list(col = col_acpp_light, label = "ACPP_FLN*ESYK
(Light)")))
  vars <- append(vars, list(list(col = col_acpp_heavy, label = "ACPP_FLN*ESYK
(Heavy)")))
} else if (!is.na(col_acpp_total)) {
  vars <- append(vars, list(list(col = col_acpp_total, label = "ACPP_FLN*ESYK")))
}
if (!is.na(col_clu)) vars <- append(vars, list(list(col = col_clu, label =
"CLU_EDALN*ETR")))
if (!is.na(col_psa)) vars <- append(vars, list(list(col = col_psa, label = "Serum PSA")))

if (is.na(col_group) || length(vars) == 0) {
  stop("Required columns not found. Please check the sheet and column names.")
}

# Filter to AG/NAG and keep numeric
df_sub <- df |>
  filter(.data[[col_group]] %in% c("AG", "NAG"))

# ---- Styling ----
c_nag <- "#1f77b4"
c_ag  <- "#ff7f0e"
base_theme <- theme_minimal(base_size = 12) +
  theme(
    panel.grid.minor = element_blank(),
    legend.position = "none"  # x-axis shows groups; legend not needed
  )

# (Optional) toggle for notched box plots:
use_notches <- FALSE

# Build one linear and one log10 plot per variable, then assemble
plots_linear <- list()
plots_log10  <- list()

for (v in vars) {
  col_name <- v$col
  label    <- v$label

  dat <- df_sub |>
    transmute(
      Group = factor(.data[[col_group]], levels = c("NAG", "AG")),
      value = suppressWarnings(as.numeric(.data[[col_name]]))
    ) |>
    filter(!is.na(value))
```

```
  # --- Linear box plot ---
  p_lin <- ggplot(dat, aes(x = Group, y = value, fill = Group)) +
    geom_boxplot(width = 0.6, notch = use_notches,
                 outlier.shape = 16, outlier.alpha = 0.35) +
    scale_fill_manual(values = c("NAG" = c_nag, "AG" = c_ag)) +
    labs(
      title = paste0(label, " — Linear Scale"),
      x = NULL, y = label
    ) +
    base_theme

  # --- Log10 box plot (drop non-positive before log transform) ---
  dat_log <- dat |>
    filter(value > 0) |>
    mutate(value_log10 = log10(value))

  p_log <- ggplot(dat_log, aes(x = Group, y = value_log10, fill = Group)) +
    geom_boxplot(width = 0.6, notch = use_notches,
                 outlier.shape = 16, outlier.alpha = 0.35) +
    scale_fill_manual(values = c("NAG" = c_nag, "AG" = c_ag)) +
    labs(
      title = paste0(label, " — log10 Scale"),
      x = NULL, y = paste0("log10(", label, ")")
    ) +
    base_theme

  plots_linear[[length(plots_linear) + 1]] <- p_lin
  plots_log10[[length(plots_log10) + 1]]  <- p_log
}

# Stack linears in the left column and log10s in the right column
left_col  <- Reduce(`/`, plots_linear)
right_col <- Reduce(`/`, plots_log10)
combined  <- left_col | right_col

# Overall title with counts
n_ag  <- sum(df_sub[[col_group]] == "AG", na.rm = TRUE)
n_nag <- sum(df_sub[[col_group]] == "NAG", na.rm = TRUE)

combined <- combined + plot_annotation(
  title = paste0("Box Plots — NAG (n=", n_nag, ") vs AG (n=", n_ag, ") — Linear and
log10")
)

# ---- Save single figure ----
ggsave(out_png, plot = combined, width = 12, height = 3.6 * length(vars), dpi = 180)
message("Saved combined figure to: ", out_png)
```

Figure 21



Figure S 20: **Figure 1: Principal Component Analysis (PCA) of Chemical Elements. Left: PCA Score Plot** illustrating the distribution and clustering of 73 chemical elements based on their physicochemical properties. Elements are color-coded by their respective chemical categories (Group Block), with individual symbols labeled for identification. **Right: PCA Loading Plot** showing the contribution of each property to the first two principal components.

Figure S20 was generated in R using the scripts provided by Gemini
The R scripts were shown below

```
# --- 1. Install and Load Necessary Libraries ---
# Uncomment the following lines if you need to install the packages:
# install.packages(c("readxl", "ggplot2", "ggrepel", "patchwork"))

library(readxl)
library(ggplot2)
library(ggrepel)
library(patchwork)

# --- 2. Set Paths ---
# Note: In R, use forward slashes (/) or double backslashes (\\) for paths
file_path <- "C:/Users/marce/Documents/Digital/Spreadsheet 2.xlsx"
output_path <- "C:/Users/marce/Documents/Digital/PCA_Combined_Plots.png"

# --- 3. Load and Prepare Data ---
df <- read_excel(file_path)

# Variables: Columns D to K (indices 4 to 11)
# Category: Column L (index 12)
# Symbol: Column B (index 2)
pca_vars <- df[, 4:11]
category <- df[[12]]
symbols  <- df[[2]]

# Handle missing values: PCA requires complete data.
# Here we fill missing values with the column mean (Imputation).
```

```
pca_vars_imputed <- as.data.frame(lapply(pca_vars, function(x) {
  x[is.na(x)] <- mean(x, na.rm = TRUE)
  return(x)
}))

# --- 4. Perform PCA ---
# scale. = TRUE is crucial to normalize units (e.g., Boiling Point vs Density)
pca_res <- prcomp(pca_vars_imputed, scale. = TRUE)

# Calculate explained variance for axis labels
vars_pct <- round((pca_res$sdev^2 / sum(pca_res$sdev^2)) * 100, 1)

# --- 5. Generate Score Plot ---
score_data <- as.data.frame(pca_res$x)
score_data$GroupBlock <- category
score_data$Symbol <- symbols

p1 <- ggplot(score_data, aes(x = PC1, y = PC2, color = GroupBlock)) +
  geom_point(size = 3, alpha = 0.7) +
  geom_text_repel(aes(label = Symbol), size = 3, show.legend = FALSE) +
  labs(title = "PCA Score Plot",
       subtitle = "Distribution of Elements",
       x = paste0("PC1 (", vars_pct[1], "%)"),
       y = paste0("PC2 (", vars_pct[2], "%)"),
       color = "Group Block") +
  theme_minimal() +
  theme(legend.position = "bottom")

# --- 6. Generate Loading Plot ---
loadings <- as.data.frame(pca_res$rotation)
loadings$Variable <- rownames(loadings)

p2 <- ggplot(loadings, aes(x = PC1, y = PC2)) +
  geom_segment(aes(xend = PC1, yend = PC2), x = 0, y = 0,
               arrow = arrow(length = unit(0.2, "cm")), color = "red") +
  geom_text_repel(aes(label = Variable), color = "darkblue", size = 4) +
  geom_hline(yintercept = 0, linetype = "dashed", alpha = 0.5) +
  geom_vline(xintercept = 0, linetype = "dashed", alpha = 0.5) +
  labs(title = "PCA Loading Plot",
       subtitle = "Variable Contributions",
       x = paste0("PC1 (", vars_pct[1], "%)"),
       y = paste0("PC2 (", vars_pct[2], "%)")) +
  theme_minimal()

# --- 7. Combine and Save ---
# Side-by-side layout using patchwork
combined_plot <- p1 + p2 + plot_annotation(title = "PCA Analysis of Periodic Table
Elements")

ggsave(output_path, combined_plot, width = 16, height = 8, dpi = 300)
```

```
cat("Success! The plot has been saved to:", output_path)
```

Figure 22:

ChatGPT
R scripts provided by ChatGPT to generate the PLS-DA score plot

```
library(readxl)
library(pls)
library(ggplot2)

# -------------------------
# 1. Load data
# -------------------------
file_path <- "C:/Users/marce/Documents/Digital/Spreadsheet 1.xlsx"
data <- read_excel(file_path)

# Column B = groups
Y <- as.factor(data[[2]])

# Columns D–G = predictors
X <- as.matrix(data[, 4:7])

# -------------------------
# 2. Prepare dummy matrix for PLS-DA
# -------------------------
Y_dummy <- model.matrix(~ Y - 1)

# Autoscale predictors
X_scaled <- scale(X)

# -------------------------
# 3. Fit PLS model (PLS-DA)
# -------------------------
pls_model <- plsr(Y_dummy ~ X_scaled, ncomp = 2)

# Extract scores
scores_df <- as.data.frame(scores(pls_model)[, 1:2])
colnames(scores_df) <- c("comp1", "comp2")
scores_df$Group <- Y

# -------------------------
# 4. Plot
# -------------------------
p <- ggplot(scores_df,
            aes(x = comp1, y = comp2,
                color = Group,
                shape = Group)) +
  geom_point(size = 3) +
  stat_ellipse(level = 0.95, linewidth = 0.8) +
  labs(x = "PLS Component 1",
       y = "PLS Component 2",
```

```
      title = "PLS-DA Score Plot (Columns D–G)") +
  theme_classic(base_size = 14)

# -------------------------
# 5. Save High-Resolution TIFF
# -------------------------
output_path <- "C:/Users/marce/Documents/Digital/PLSDA_ScorePlot.tiff"

ggsave(filename = output_path,
       plot = p,
       device = "tiff",
       dpi = 600,
       width = 8,
       height = 6,
       units = "in",
       compression = "lzw")
```

ChatGPT
R scripts provided by ChatGPT to generate Confusion Matrix

```
# Install if needed
# install.packages(c("readxl", "pls", "caret", "ggplot2"))

library(readxl)
library(pls)
library(caret)
library(ggplot2)

# -------------------------
# 1. Load data
# -------------------------
file_path <- "C:/Users/marce/Documents/Digital/Spreadsheet 1.xlsx"
data <- read_excel(file_path)

# Column B = groups
Y <- as.factor(data[[2]])

# Ensure AG = positive class
Y <- factor(Y, levels = c("NAG", "AG"))

# Columns D–G = predictors
X <- as.matrix(data[, 4:7])

# Autoscale predictors
X_scaled <- scale(X)

# Convert to binary (AG = 1)
Y_binary <- ifelse(Y == "AG", 1, 0)

# -------------------------
# 2. 5-Fold Cross-Validation
# -------------------------
set.seed(123)
folds <- createFolds(Y_binary, k = 5)

predicted_prob <- rep(NA, length(Y_binary))

for(i in 1:5) {

  test_idx <- folds[[i]]
  train_idx <- setdiff(seq_along(Y_binary), test_idx)

  X_train <- X_scaled[train_idx, ]
  Y_train <- Y_binary[train_idx]
  X_test  <- X_scaled[test_idx, ]

  # PLS model with 2 latent variables
```

```
  pls_model <- plsr(Y_train ~ X_train, ncomp = 2)

  preds <- predict(pls_model, newdata = X_test, ncomp = 2)
  predicted_prob[test_idx] <- preds[, , 1]
}

# -------------------------
# 3. Apply threshold = 0.5
# -------------------------
predicted_class <- ifelse(predicted_prob >= 0.5, 1, 0)

# -------------------------
# 4. Compute Confusion Matrix
# -------------------------
TP <- sum(predicted_class == 1 & Y_binary == 1)
TN <- sum(predicted_class == 0 & Y_binary == 0)
FP <- sum(predicted_class == 1 & Y_binary == 0)
FN <- sum(predicted_class == 0 & Y_binary == 1)

conf_df <- data.frame(
  Actual = c("AG", "AG", "NAG", "NAG"),
  Predicted = c("AG", "NAG", "AG", "NAG"),
  Count = c(TP, FN, FP, TN)
)

# -------------------------
# 5. Plot Confusion Matrix
# -------------------------
p <- ggplot(conf_df, aes(x = Predicted, y = Actual, fill = Count)) +
  geom_tile(color = "black") +
  geom_text(aes(label = Count), size = 6, fontface = "bold") +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(title = "Confusion Matrix (5-fold CV, 2 LVs, Threshold = 0.5)",
       x = "Predicted Class",
       y = "Actual Class") +
  theme_classic(base_size = 14)

# -------------------------
# 6. Save High-Resolution Image
# -------------------------
output_path <-
"C:/Users/marce/Documents/Digital/Confusion_Matrix_5fold_PLSDA.tiff"

ggsave(filename = output_path,
       plot = p,
       device = "tiff",
       dpi = 600,
       width = 6,
       height = 5,
       units = "in",
```

```
        compression = "lzw")

print("Confusion matrix image saved successfully.")
```

Copilot
R scripts provided by Copilot to generate the PLS-DA score plot

```
# 1. INSTALL AND LOAD NECESSARY PACKAGES
packages <- c("readxl", "ggplot2", "pls")
new_packages <- packages[!(packages %in% installed.packages()[,"Package"])]
if(length(new_packages)) install.packages(new_packages)

library(readxl)
library(pls)
library(ggplot2)

# 2. SET FILE PATHS
# Note: Use forward slashes (/) in R to avoid "hex digit" errors
file_path <- "C:/Users/marce/Documents/Digital/Spreadsheet 1.xlsx"
output_path <- "C:/Users/marce/Documents/Digital/PLS_DA_Score_Plot.png"

# 3. LOAD DATA
df <- read_excel(file_path)

# 4. PREPARE DATA
# Column B (2) is the Group
# Columns D to G (4 to 7) are the features
Y_group <- as.factor(df[[2]])
X_matrix <- as.matrix(df[, 4:7])

# For PLS-DA, we create a dummy-coded matrix of the groups
Y_dummy <- model.matrix(~ Y_group - 1)

# 5. RUN PLS-DA
# Using the 'pls' package's plsr function
pls_model <- plsr(Y_dummy ~ X_matrix, ncomp = 2, scale = TRUE)

# 6. EXTRACT SCORES FOR PLOTTING
# pls_model$scores contains the coordinates for the samples
plot_data <- as.data.frame(pls_model$scores[, 1:2])
colnames(plot_data) <- c("Component1", "Component2")
plot_data$Group <- Y_group

# 7. CREATE HIGH-RESOLUTION PLOT
p <- ggplot(plot_data, aes(x = Component1, y = Component2, color = Group, shape =
Group)) +
  geom_hline(yintercept = 0, linetype = "dotted", color = "darkgrey") +
  geom_vline(xintercept = 0, linetype = "dotted", color = "darkgrey") +
  geom_point(size = 4, alpha = 0.8) +
  theme_minimal(base_size = 14) +
  labs(
    title = "PLS-DA Score Plot",
    subtitle = "Variables: Age, Serum PSA, ACPP_FLN*ESYK, CLU_EDALN*ETR",
    x = "Component 1",
```

```
    y = "Component 2"
  ) +
  scale_color_manual(values = c("#E41A1C", "#377EB8")) + # Custom distinct colors
  theme(
    legend.position = "right",
    plot.title = element_text(face = "bold"),
    panel.border = element_rect(colour = "black", fill=NA, size=1)
  )

# 8. SAVE IMAGE (High Resolution 300 DPI)
ggsave(output_path, plot = p, width = 8, height = 6, dpi = 300)

print(paste("Plot successfully saved to:", output_path))
```

R scripts provided by Gemini to generate the confusion matrix in R

```
# 1. LOAD LIBRARIES
library(ggplot2)
library(reshape2)

# 2. CREATE THE DATA
# Based on the results: TN=55, FP=14, FN=16, TP=57
cm_data <- matrix(c(55, 14, 16, 57), nrow = 2, byrow = TRUE)
colnames(cm_data) <- c("NAG", "AG")
rownames(cm_data) <- c("NAG", "AG")

# Melt the matrix for ggplot
cm_melted <- melt(cm_data)
colnames(cm_melted) <- c("Actual", "Predicted", "Value")

# 3. GENERATE THE PLOT
p_cm <- ggplot(cm_melted, aes(x = Predicted, y = Actual, fill = Value)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "#ebf2fb", high = "#2b7bba") +
  geom_text(aes(label = Value), color = "black", size = 8, fontface = "bold") +
  theme_minimal() +
  labs(title = "Confusion Matrix (5-Fold CV)",
       subtitle = "Aggressive (AG) vs Non-Aggressive (NAG)",
       x = "Predicted Group",
       y = "Actual Group") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 12),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12),
        legend.position = "none")

# 4. SAVE THE IMAGE
output_cm_path <- "C:/Users/marce/Documents/Digital/Confusion_Matrix_Plot.png"
ggsave(output_cm_path, plot = p_cm, width = 6, height = 5, dpi = 300)

print(paste("Confusion matrix image saved to:", output_cm_path))
```