1 Questions 1-3: Resource Allocation Problem

You are in charge of a company that makes two hot sauces: x_1 liters of Kapatio and x_2 liters of Zriracha. We will use optimization technique to find the "best" manufacturing strategy given our resource constraints.

First, we need to define what we mean by "best" strategy. In this scenario, the goal is to obtain the highest revenue possible. While doing so, there are resource constraints we must satisfy.

For example, in order to manufacture these two hot sauce products, different amount of peppers and vineger are needed. Also, we have only so much total resource available.

Ingridients	Kapatio	Zriracha	Total Available
Pepper	5	7	30
Vineger	4	2	12

1.1 Question 1: Resource Constraints

1.1.1 Question 1.a: Modeling Resource Usage

What is the equation for the amount of pepper needed to manufacture x_1 and x_2. What is the equation for the amount of vinegar? (Use Mathpix to write equations)

$$\begin{aligned} f(P) &= 5 * x1 + 7 * x2 \\ f(V) &= 4 * x1 + 2 * x2 \end{aligned} \tag{1}$$

1.1.2 Question 1.b: Resource Usage vs. Total Resource Constraint

Total amount of pepper needed cannot exceed total available. Write down the inequality expressing this relationship. Do the same for vinegar. These inequalities are your resource constraints. Additinally, variables x_1 and x_2 are non-negative: i.e. amount of manufactured goods cannot be negative.

Rewrite the system of constraint inequalities into a matrix inequality: $Ax \leq b$, where $x = (x_1, x_2)^T$. Arrange rows of A and b such that:

- Row 1: total pepper amount constraint
- Row 2: total vinegar amount constraint
- Row 3: Kapatio non-negativity constraint
- Row 4: Zriracha non-negativity constraint

Less than symbol in $Ax \leq b$ means element-wise.

$$\begin{bmatrix} 5 & 7 \\ 4 & 2 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 30 \\ 12 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

Sometimes, things just do not work as expected.

In the toy example code,

```
sns.lineplot(x='$z_1$', y='$z_2$', hue='constraint', data=pd.concat(boundary), ax=ax).axvline(1)
```

what seems strange about the plotting command? Why was the strange code necessary?

It seems strange that the hue is set to 'constraint' where usually hue is a vector of numbers. It is necessary though since we are dealing with LaTex code.

By dissecting the command below and reading the documentation, report what each of the following lines does:

- `((y1_grid >= 1) & (y2_grid >= 2)).astype(int)` (What is the output of running this command?)
- `origin='lower'`
- `extent=(y1_grid.min(), y1_grid.max(), y2_grid.min(), y2_grid.max())`
- `cmap='Greys'`
- `alpha=0.3`
- `aspect='equal'`

((y1__grid >= 1) & (y2__grid >= 2)).astype(int) Creates a grid that is black and white that outlines the area of constraint

origin='lower' Places index of the array in the lower left corner of the Axes and makes the vertical axis point upwards

extent=(y1__grid.min(), y1__grid.max(), y2__grid.min(), y2__grid.max()) The bounding box in data coordinates that the image will fill in this case it is for our z1__grid and z2__grid min and max

cmap='Greys' maps the scalar data to a color, in this case grey

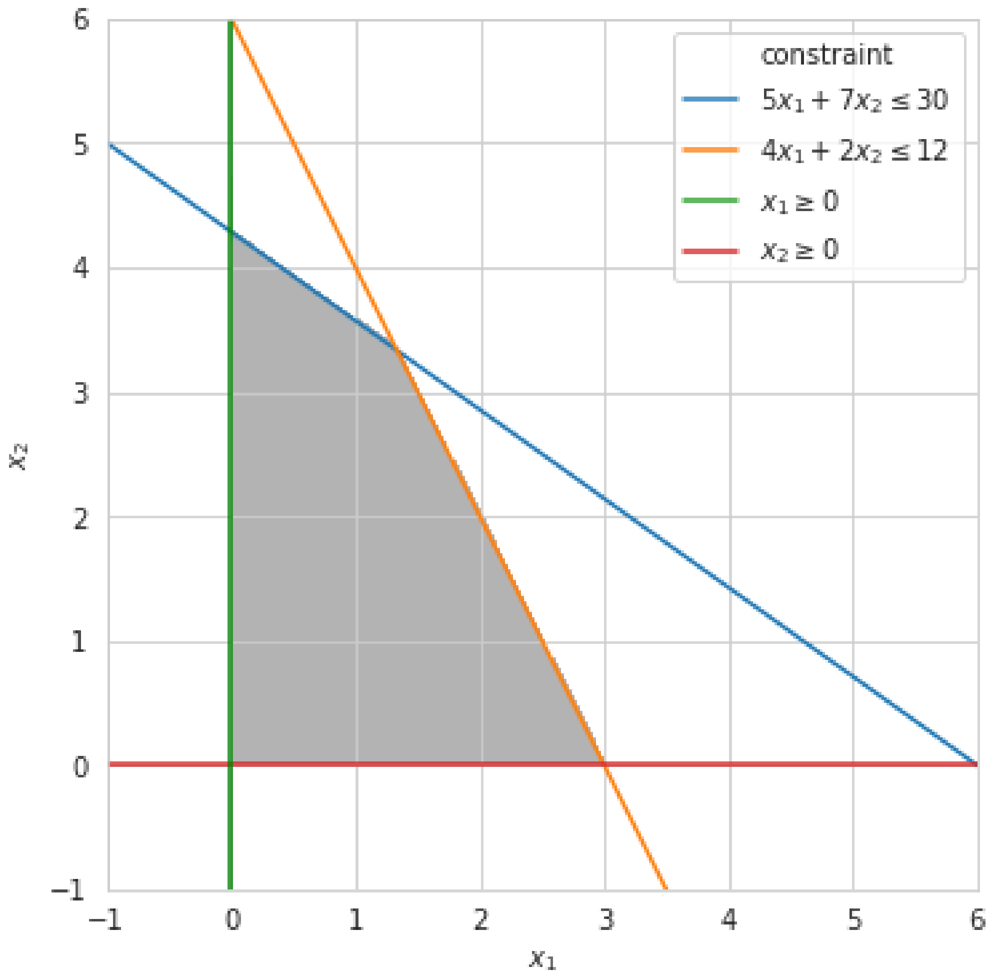
alpha=0.3 turns the boundary area grey by decreasing transparenty.

aspect='equal'ensures an aspect ratio of 1, pixels will be square

1.1.3 Question 1.e: Visualizing the Feasible Region

Finally, create a figure that shows constraint boundaries and the interior region shaded with a light grey color.

Your output will look like this:

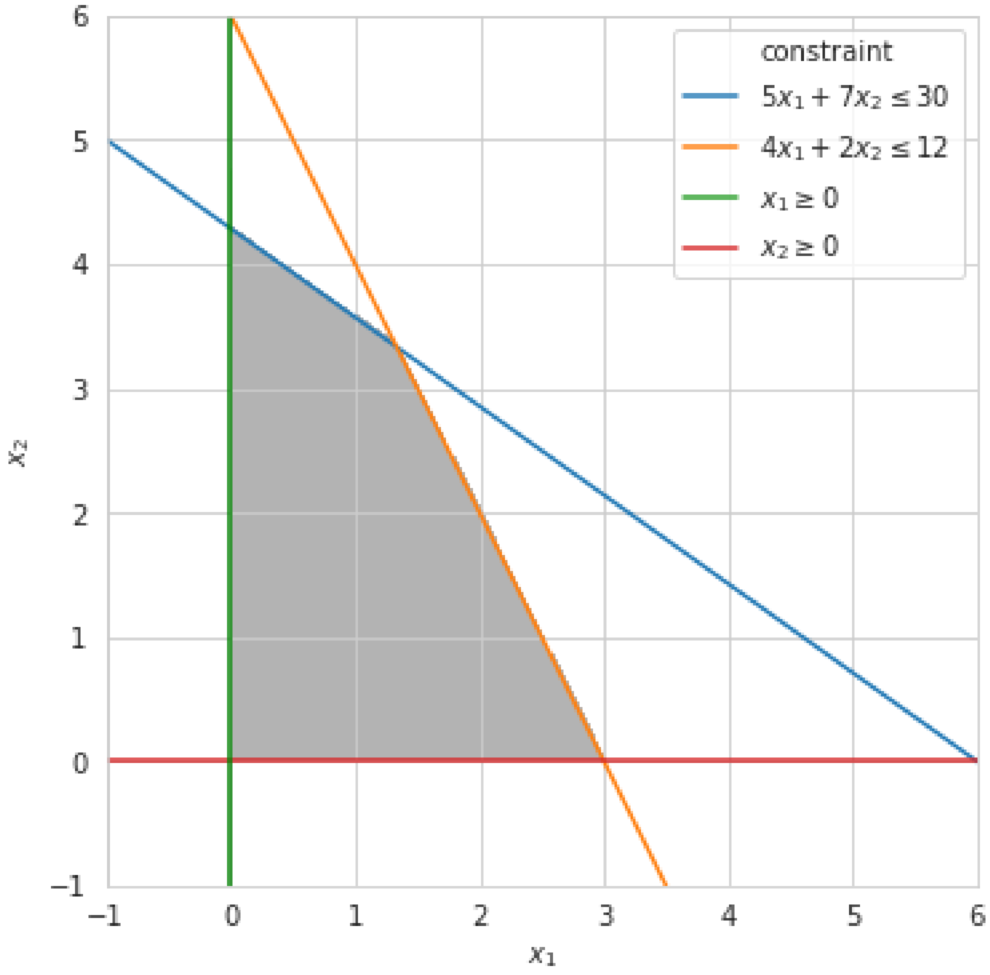


```
In [123]: fig, ax = plt.subplots(figsize=(6, 6))
          x1_grid, x2_grid = np.meshgrid(x1_line, x2_line)

          ax.imshow(
              (
                  ((x1_grid >= 0)&(x2_grid >= 0)&(5*x1_grid+7*x2_grid<=30)&(4*x1_grid+2*x2_grid<=12)).a
              ),
              origin='lower',
              extent=(x1_grid.min(), x1_grid.max(), x2_grid.min(), x2_grid.max()),
              cmap="Greys", alpha = 0.3, aspect='equal')
```

```
# ax = sns.lineplot(???).axvline(???)
ax = sns.lineplot(x='$x_1$',y='$x_2$',hue='constraint',data=pd.concat(boundary),ax=ax).axvline

plt.xlim(-1, 6)
plt.ylim(-1, 6)
plt.show()
```



In the context of linear programming, $Ax \leq b$ is called the *feasible region* (including the appropriate sections of the boundaries). Denote the (shaded) feasible region as set C. Points $(x_1, x_2) \in C$ satisfy all of the constraints.

Describe in plain words the feasible region in the context of hot sauce manufacturing. Specifically, which constraint is violated (if any) by a point at:

- $(x_1, x_2) = (4, 1)$
- $(x_1, x_2) = (0, 5)$

- $(x_1, x_2) = (3, 4)$

(4,1) = constraint 4*x1 + 2*x2 <= 12 is violated

(0,5) = constraint 5*x1 + 7*x2 <= 30 is violated

(3,4) = constraints 4*x1 + 2*x2 <= 12 AND 5*x1 + 7*x2 <= 30 are violated

1.2 Question 2: Objective Function

1.2.1 Question 2.a: Defining Objective Function

Suppose the hot sauces are sold at the same price: \$5 per liter.

What is the equation $f(x)$ for the total revenue as a function of x_1 and x_2?

The function $f(x)$ is called the objective function.

$$f(x) = 5x1 + 5x2 \tag{3}$$

1.2.2 Question 2.b: Direction of Steepest Increase

Since we want to maximize revenue, we want to increase our objective function as much as possible. Analogous to the minimization example given in a previous lecture, we can repeatedly move in the direction of function increase. In order to determine such direction, compute the gradient of $f(x)$ at $x = (0,0)^T$:

$$\nabla_x f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \end{pmatrix}$$

$$\begin{aligned} \frac{\partial f(x)}{\partial x_1} &= \frac{\partial\,(5x_1 + 5x_2)}{\partial x_1} = 5 \\ \frac{\partial f(x)}{\partial x_2} &= \frac{\partial\,(5x_1 + 5x_2)}{\partial x_2} = 5 \\ \nabla_x f(x) &= \left(\begin{array}{c} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \end{array} \right) = \left(\begin{array}{c} 5 \\ 5 \end{array} \right) \end{aligned} \tag{4}$$

1.3 Question 3: Putting Pieces Together

1.3.1 Question 3.a: Standard Form of a Linear Programming Problem

Write down the so-called the *standard form* of a linear programming problem:

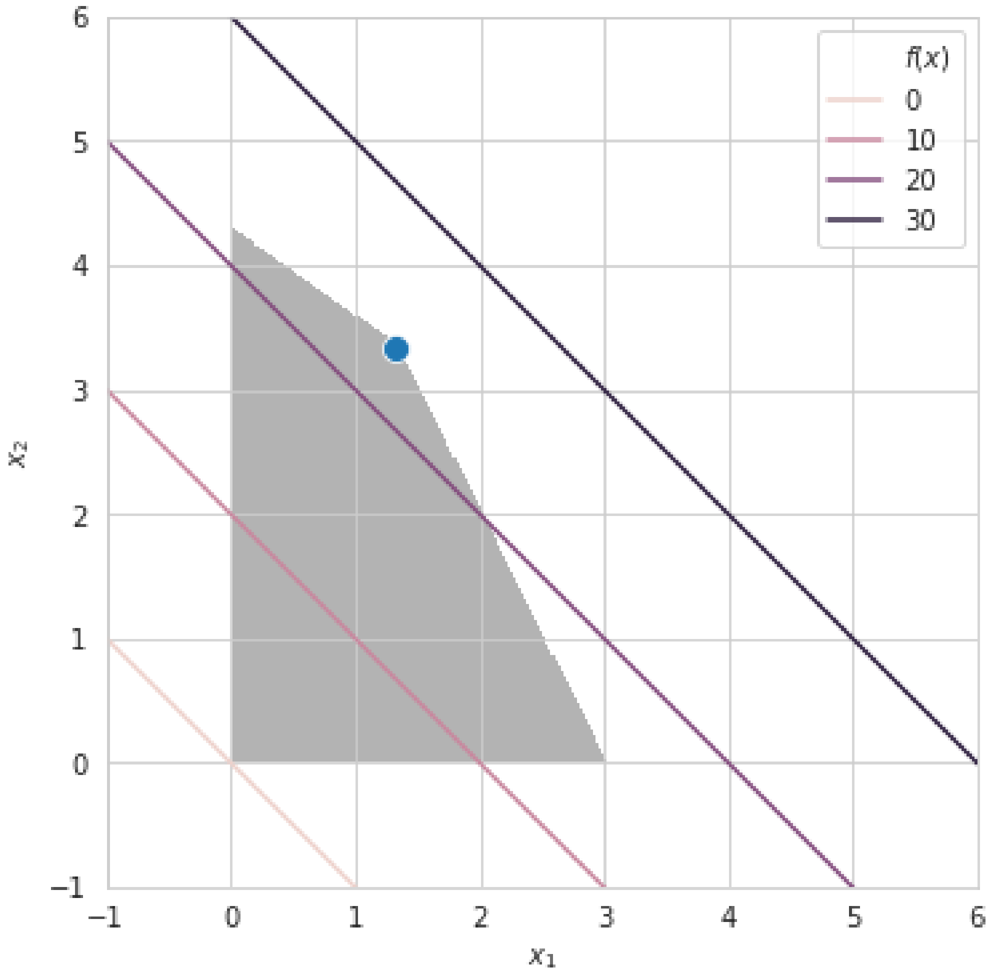
$$\max_x f(x)$$
$$\text{subject to } Ax \leq b$$

Specifically, write the obejective as an inner product of two vectors: $f(x) = c^T x$, and write the constraint as a vector inequality involving a matrix-vector prduct: $Ax \leq b$, where A is a 4-by-2 matrix.

$$\begin{aligned} c = \begin{bmatrix} 5 \\ 5 \end{bmatrix} &\text{ and } x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ \text{subject to } \begin{bmatrix} 5 & 7 \\ 4 & 2 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} &\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 30 \\ 12 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{5}$$

1.3.2 Question 3.c: Plotting the optimal solution

```
In [128]: fig, ax = plt.subplots(figsize=(6, 6))
          x1_grid, x2_grid = np.meshgrid(x1_line, x2_line)
          ax.imshow(
              (
                  (A1[0,0]*x1_grid + A1[0,1]*x2_grid <= b1[0]) & # Pepper constraints
                  (A1[1,0]*x1_grid + A1[1,1]*x2_grid <= b1[1]) & # Vinegar constraints
                  (A1[2,0]*x1_grid + A1[2,1]*x2_grid <= b1[2]) &
                  (A1[3,0]*x1_grid + A1[3,1]*x2_grid <= b1[3])   # non-negativity constraints
              ),
              origin='lower',
              extent=(x1_grid.min(), x1_grid.max(), x2_grid.min(), x2_grid.max()),
              cmap="Greys", alpha = 0.3, aspect='equal'
          )
          sns.scatterplot(x='$x_1$', y='$x_2$', data=xstar1, ax=ax, s=100)
          sns.lineplot(x='$x_1$', y='$x_2$', hue='$f(x)$', data=f_vals, ax=ax)
          plt.xlim(-1, 6)
          plt.ylim(-1, 6)
          plt.show()
```



$f(x)$
0
10
20
30
x_1
x_2

1.3.3 Question 4.a: Define Constraints

To avoid eating the same foods, limit each food intake to be 2 or less. Also, one cannot consume less than zero servings. Furthermore, apply the nutritional constraints as specified in `nutritional_constraints.csv` (assume that the units are the same as food nutritional contents)

Note that a range constraints, e.g., $2000 \leq \text{total calories} \leq 2250$, can be written as two constraints: $\text{total calories} \leq 2250$ and $-\text{total calories} \leq -2000$. Hence, we can rewrite caloric intake constraints as

$$-(\text{calories in frozen broccoli})x_0 - (\text{calories in raw carrots})x_1 - \cdots - (\text{calories in bean bacon soup, w/watr})x_{63} = -c^T x \leq -2$$
$$\text{calories in frozen broccoli})x_0 + (\text{calories in raw carrots})x_1 + \cdots + (\text{calories in bean bacon soup, w/watr})x_{63} = c^T x \leq 22$$

where vector c contains calorie information for all 64 foods and x containts servings consumed of each food. Matrix U and vector w would be such that

$$U = \begin{pmatrix} -c^T \\ c^T \end{pmatrix} \text{ and } w = \begin{pmatrix} -2000 \\ 2250 \end{pmatrix},$$

and the matrix-vector inequality would be $Ux \leq w$. Range constraints of each food can be implemented similarly with identity matrices.

Denote nutritional content information from `foods` data frame as A and denote the `Min` and `Max` columns of `requirements` as vector b_L and b_U, respectively. Construct M and d in $Mx \leq d$ using I (identity matrix), A, b_L, b_U, and other constants, so that all the range constraints are expressed in $Mx \leq d$. (This is a theory question. No coding is involved)

Type your answer here, replacing this text.

1.3.4 Question 4.d: Interpreting the Results

State the results in the context of the problem. How much of each food was consumed? List the foods and their calculated amounts. What is the total cost of feeding one soldier?

The total cost for feeding one solider is $1.24

```
In [112]: foods["servings"] = xstar2.tolist()
          result = foods.iloc[:,[0,14]]
          print(result)
```

```
                     Name     servings
0         Frozen Broccoli  8.024228e-02
1            Carrots, Raw  2.240289e-01
2             Celery, Raw  1.401558e-11
3             Frozen Corn  3.148633e-12
4     Lettuce, Iceberg,Raw  3.018374e-11
..                    …            …
59       New Eng Clam Chwd  6.674082e-13
60             Tomato Soup  1.923350e-12
61  New Eng Clam Chwd, w/Mlk  3.453333e-13
62     Crm Mshrm Soup, w/Mlk  7.663001e-13
63   Bean Bacon Soup, w/Watr  7.326201e-13

[64 rows x 2 columns]
```