a better sorting hat

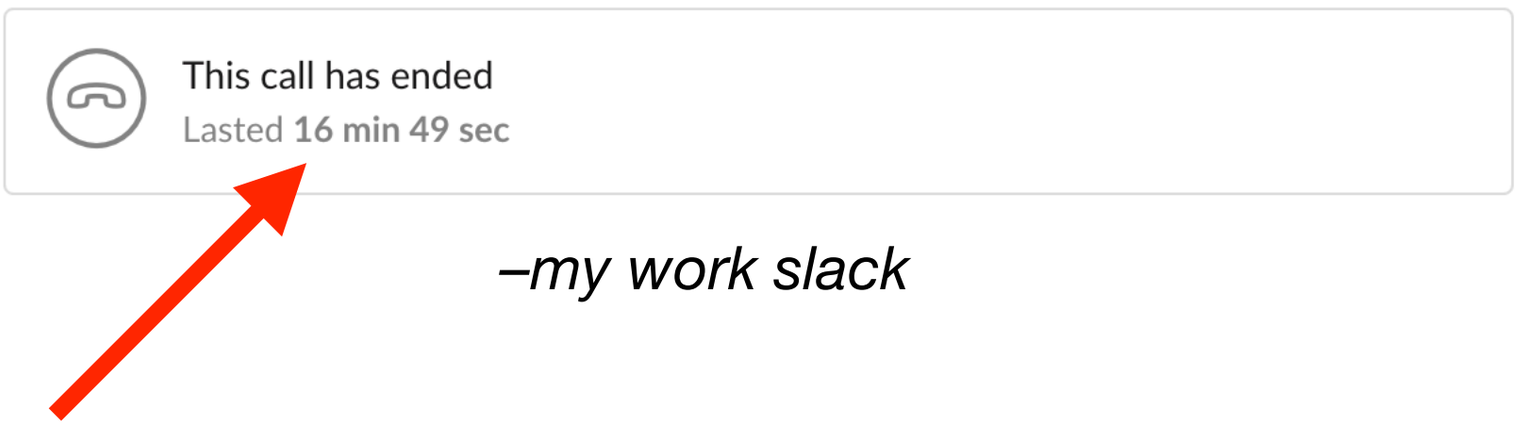
using python and PuLP to create optimal subgroups

casey woolfolk

@caseywoolfolk
(devict slack)

ccwoolfolk/optimalgroups
(github)

“Hmm, it may be simplest to jump on a quick call to organize into groups.”



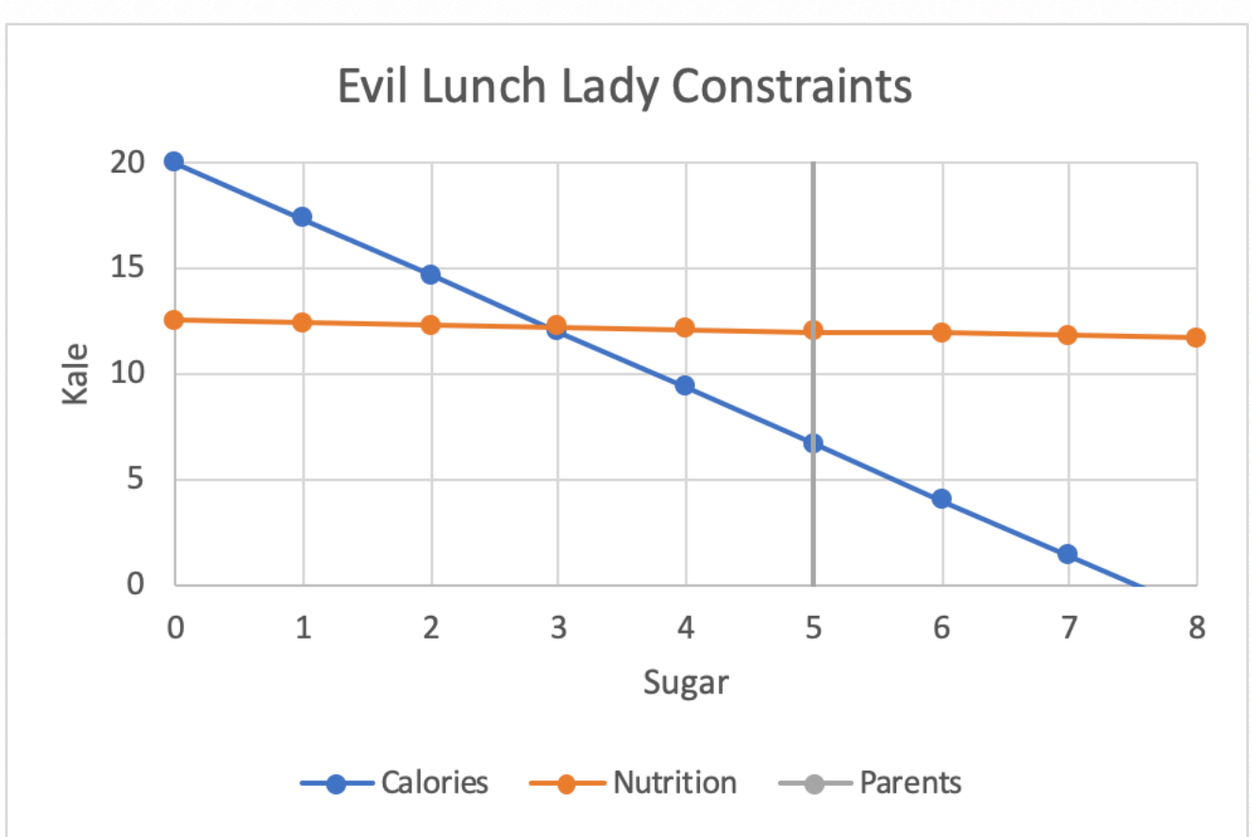
–my work slack

a simple lp

“

Linear programming (**LP**, also called **linear optimization**) is a method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships.

$$Calories : 200s + 75k \geq 1500$$
$$Nutrition : 10s + 100k \leq 1250$$
$$Parents : 1s + 0k \leq 5$$
$$MinimizeCost : 25s + 100k = c$$



our data set

Persons	Microbes and You	Tone it Down: Technical Writing for Creatives	Indoor Beekeeping	Trial by Lyre: Music and Jurisprudence in Ancient Rome
Anita	2	3	1	
Bob		2	1	3
Carlton	1	2	3	
Denise		2		1
Eugene		2		1
Francine	1	2		
Guy		2	1	
Horace		2		1
Innes	3	2	1	
Jorge	1	2	3	

our problem

$$minimize \sum_{n=1}^{nPeople} \left(\sum_{j=1}^{nClasses} choice_{i,j} \cdot cost_{i,j} \right)$$

- where choice is a binary variable and cost is a constant
- subject to:
  - every person in exactly one class
  - every class with any members must have a minimum number of members (specified by user)

code setup

```
"""CLI for optimal subgroups based on individual preferences"""
from argparse import ArgumentParser
import pandas as pd
import numpy as np
from pulp import LpProblem, LpMinimize, LpVariable, lpSum

NAN_VALUE = 10000 # Arbitrarily high value to force optimization away from empty choices


def optimize():
    """Run the optimizer"""
    args = get_args()
    rawdata = pd.read_excel(args.file_path, 'Sheet1', index_col=None, na_values=['NA'])
    prob = LpProblem("Optimal 10x Grouping", LpMinimize)

    alternatives = list(rawdata.columns[1:]) # Column labels
    persons = list(rawdata['Persons'])
    cost_matrix = rawdata.drop('Persons', 'columns').to_numpy() # Numpy array: [person][class]
    (n_persons, n_alternatives) = cost_matrix.shape

    costs = [NAN_VALUE if np.isnan(x) else x for x in cost_matrix.flatten()]
```

define lp variables

```
    # Create binary LpVariables for every person/alternative combination
    choices = []
    make_name = make_var_name_factory(persons, alternatives)
    for i in range(n_persons):
        choices.append([LpVariable(make_name(i, j), cat='Binary') for j in range(n_alternatives)])


    # Create binary LpVariables tracking if a group has any members
    has_membership = [LpVariable(f"{alternatives[j]}", cat='Binary') for j in range(n_alternatives)]
```

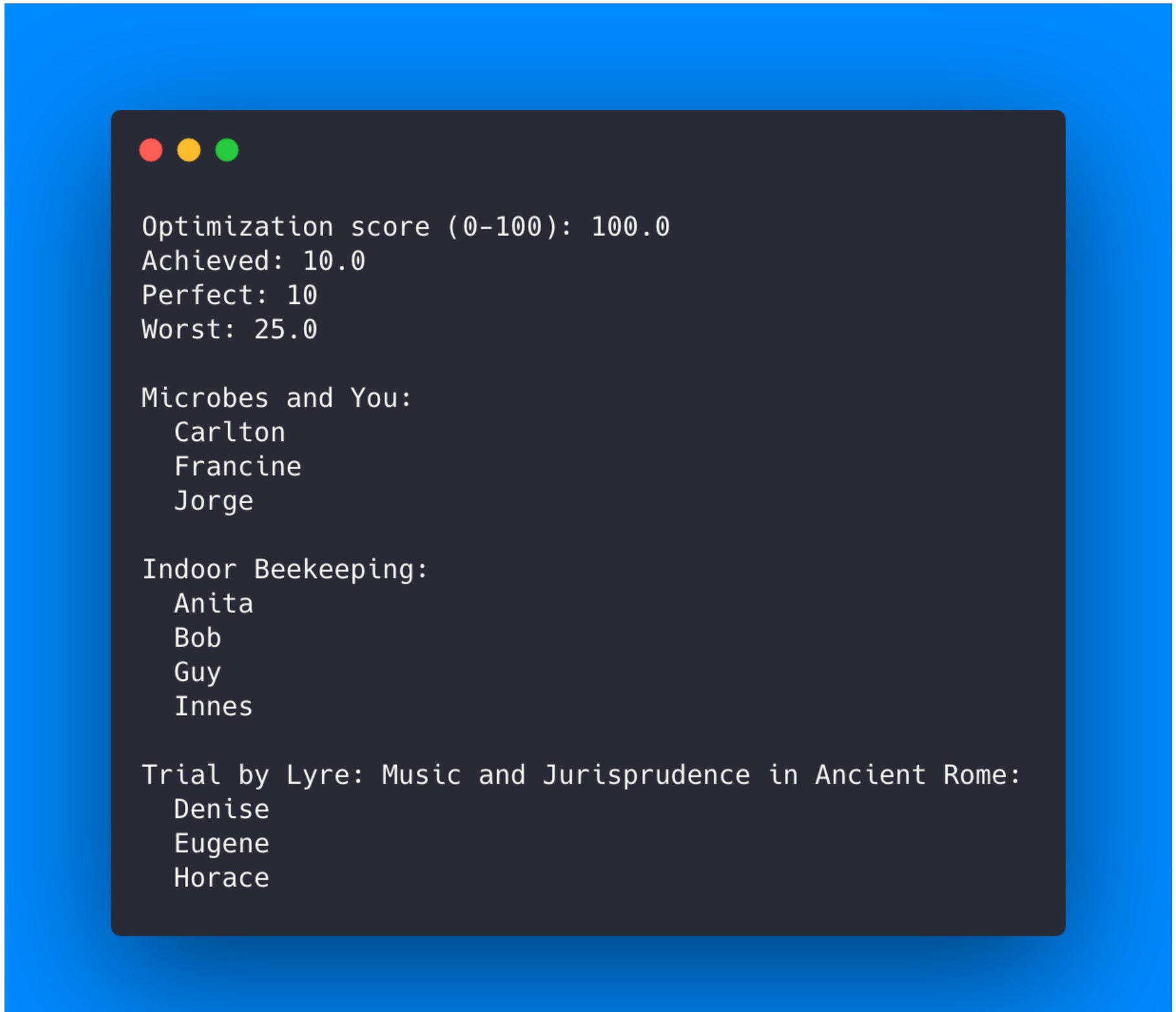
implement constraints

```
    # Add constraints
    # See https://cs.stackexchange.com/questions/12102/express-boolean-logic-operations-in-zero-one-
    #     integer-linear-programming-ilp
    # for a good explanation of logical operations (AND, OR, etc.) via linear constraints
    for i in range(n_persons):
        prob += sum(choices[i]) == 1 # Only one result per person

    for j in range(n_alternatives):
        # If the sum of every choice for the alternative is 0, has_membership must be 0
        prob += has_membership[j] <= sum([choices[i][j] for i in range(n_persons)])

        for i in range(n_persons):
            prob += has_membership[j] >= choices[i][j] # has_membership is 1 if any choice is 1

        # If a group has any members, enforce a minimum number
        prob += sum([choices[i][j] for i in range(n_persons)]) >= args.min * has_membership[j]
```

Solving

```
    # Define and calculate the optimization
    choices_full = np.array(choices).flatten()
    prob += lpSum([choices_full[i] * costs[i] for i in range(len(choices_full))])
    prob.solve()

    display_results(
        optimized_value=prob.objective.value(),
        persons=persons,
        alternatives=alternatives,
        choices=choices,
        cost_matrix=cost_matrix,
    )
```

results comparison (min 2)



```
Optimization score (0-100): 100.0
Achieved: 10.0
Perfect: 10
Worst: 25.0

Microbes and You:
  Carlton
  Francine
  Jorge

Indoor Beekeeping:
  Anita
  Bob
  Guy
  Innes

Trial by Lyre: Music and Jurisprudence in Ancient Rome:
  Denise
  Eugene
  Horace
```

results comparison (min 4)

```
Optimization score (0-100): 60.0
Achieved: 16.0
Perfect: 10
Worst: 25.0

Tone it Down: Technical Writing for Creatives:
  Carlton
  Denise
  Eugene
  Francine
  Horace
  Jorge

Indoor Beekeeping:
  Anita
  Bob
  Guy
  Innes
```

results comparison (min 5)

```
Optimization score (0-100): 53.0
Achieved: 17.0
Perfect: 10
Worst: 25.0

Tone it Down: Technical Writing for Creatives:
  Denise
  Eugene
  Francine
  Horace
  Jorge

Indoor Beekeeping:
  Anita
  Bob
  Carlton
  Guy
  Innes
```

ty