1 Cost-Benefit LP Problem: Phone Survey Example

In [4]:

```python
from pulp import LpMinimize, LpProblem, LpStatus, lpSum, LpVariable
import pandas as pd
```

The marketing group for a software development company plans to conduct a telephone survey to determine customer attitudes towards new potential products under development. In order to have a sufficient sample size to conduct the analysis, they need to complete surveys with at least 200 participents from each of 4 demographics: young males (under age 40), older males (over age 40), young females (under age 40) and older females (over age 40).

It costs $1.00 to make a daytime survey call and $1.50 to make an evening phone call. This cost is incurred whether or not anyone answers the phone.

Table 1 shows the probability of a given customer type answering the phone during each time of day (for example, if you make 100 daytime calls, you can expect to complete 10 surveys with young males, 15 surveys with older makes, etc.).

Finally, because of limited evening phone staffing, at most one-third of the phone calls places can be evening phone calls.

How should the marketing group conduct the telephone survey so as to meet the sample size requirements at the lowest possible cost?

Table 1:

Who Answers?	Daytime Calls	Evening Calls
Young Male	10%	20%
Older Male	15%	30%
Young Female	20%	40%
Older Female	35%	25%
No Answer	20%	5%

In [5]:

```python
Demos = ['YoungMale', 'OlderMale', 'YoungFemale', 'OlderFemale']
data = { 'Daytime': [.1, .15, .2, .35], 'Evening': [.2, .3, .4, .25]}
LpVariables = ['Daytime','Evening']
df = pd.DataFrame(data, index = Demos)
df
```

Out[5]:

	Daytime	Evening
YoungMale	0.10	0.20
OlderMale	0.15	0.30
YoungFemale	0.20	0.40
OlderFemale	0.35	0.25

In [6]:

```python
LpVariables
```

Out[6]:

```
['Daytime', 'Evening']
```

In [7]:

```python
required_participants = 200
costs = {'Daytime': 1, 'Evening': 1.5}
costs
```

Out[7]:

```
{'Daytime': 1, 'Evening': 1.5}
```

In [8]:
```
costs['Daytime']
```

Out[8]:
```
1
```

In [9]:
```
# Create the model (LpProblem class objects)
model = LpProblem("Phone_Survey", LpMinimize)
```

In [10]:
```
# Create the decision variables
calls = LpVariable.dicts('calls_', ['Daytime', 'Evening'], lowBound = 0, cat = 'I

calls
```

Out[10]:
```
{'Daytime': calls__Daytime, 'Evening': calls__Evening}
```

In [11]:
```
# Add the objective function to the model
obj_func = lpSum(calls[x] * costs[x] for x in ['Daytime', 'Evening'])
model += obj_func
model
```

Out[11]:
```
Phone_Survey:
MINIMIZE
1*calls__Daytime + 1.5*calls__Evening + 0.0
VARIABLES
0 <= calls__Daytime Integer
0 <= calls__Evening Integer
```

In [12]:
```
## note on indexing
df.index
```

Out[12]:
```
Index(['YoungMale', 'OlderMale', 'YoungFemale', 'OlderFemale'], dtype='object')
```

In [13]:
```
# Add the constraints to the model

for d in df.index :
    model += (lpSum(df.loc[d, x] * calls[x] for x in ['Daytime', 'Evening']) >=
```

In [111]:
```
model
```

Out[111]:
```
Phone_Survey:
MINIMIZE
1*calls__Daytime + 1.5*calls__Evening + 0.0
SUBJECT TO
YoungMale: 0.1 calls__Daytime + 0.2 calls__Evening >= 200

OlderMale: 0.15 calls__Daytime + 0.3 calls__Evening >= 200

YoungFemale: 0.2 calls__Daytime + 0.4 calls__Evening >= 200

OlderFemale: 0.35 calls__Daytime + 0.25 calls__Evening >= 200

VARIABLES
0 <= calls__Daytime Integer
0 <= calls__Evening Integer
```

In [112]:
```
## add ratio constraint
model += ((1/3)*calls['Daytime'] - (2/3)*calls['Evening'] >= 0, "ratio")

model
```

Out[112]:
```
Phone_Survey:
MINIMIZE
1*calls__Daytime + 1.5*calls__Evening + 0.0
SUBJECT TO
YoungMale: 0.1 calls__Daytime + 0.2 calls__Evening >= 200

OlderMale: 0.15 calls__Daytime + 0.3 calls__Evening >= 200

YoungFemale: 0.2 calls__Daytime + 0.4 calls__Evening >= 200

OlderFemale: 0.35 calls__Daytime + 0.25 calls__Evening >= 200

ratio: 0.333333333333 calls__Daytime - 0.666666666667 calls__Evening >= 0

VARIABLES
0 <= calls__Daytime Integer
0 <= calls__Evening Integer
```

In [113]:
```
model.solve()

LpStatus[model.status]
```

Out[113]:
```
'Optimal'
```

In [114]:
```
model.objective.value()
```

Out[114]:
```
1750.0
```

In [115]:
```
## Retrieve optimal values for decision variables
for v in model.variables(): print(f"{v.name}: {v.varValue}")
```

```
calls__Daytime: 1000.0
calls__Evening: 500.0
```

In [116]:
```
## Retrieve contraint values at optimal solution.  Note that value is relative to
for name, constraint in model.constraints.items(): print(f"{name}: {(constraint.v
```

```
YoungMale: 200.0
OlderMale: 300.0
YoungFemale: 400.0
OlderFemale: 475.0
ratio: 0.0
```

In [47]:
```
model.solver
```

Out[47]:
```
<pulp.apis.coin_api.PULP_CBC_CMD at 0x155281c16a0>
```

In [ ]:
```

```