PCA on a Prioritization Matrix

This is an exploration of the application of Principal Component Analysis (https://en.wikipedia.org/wiki/Principal_component_analysis) to Prioritization Matrices (https://blog.transparentchoice.com/project-prioritization-matrix) in order to optimize the number of criteria (columns).

This process will produce criteria which, given the sample data, are maximally independent of each other.

The problem with it is that it will leave us with the task of interpreting the new criteria it creates.

We begin with a matrix with some representative projects and a bloated set of criteria.

In [29]:

```
import pandas as pd
import numpy as np

raw = pd.read_csv('data/input/pri-matrix.csv')
raw
```

Out[29]:

	Project Title	Cross-site relevance	Site directors want it	Site non-medical staff wants it	Site medical staff wants it	Builds on proven tools	Improves tools	Strengthens MOH partnerships	Strengthens donor partnerships	Other priority projects depend on this
0	Shire TB investigation	0	5	5	2	3	0	2	4	0
1	Moria DM program evaluation	4	1	1	0	1	5	5	1	1
2	White paper about the ring	4	5	3	0	0	4	4	5	0
3	New EMR for Mordor	0	3	5	3	4	3	0	4	3
4	Replace old EMR in Moria	0	4	5	4	5	2	4	4	1
5	Fancy new UI for EMR	5	0	0	3	1	5	0	0	1
6	Easier EMR set-up	4	0	4	4	3	5	0	0	3
7	Capacity-building for Some Fancy Tool	5	2	4	0	5	3	0	0	4
8	New form for Fellbeast Flu	5	4	0	2	4	0	5	2	0
9	Decentralizing data analysis	5	3	3	0	1	4	4	4	2
10	Integrating EMR with Some Fancy Tool	4	2	5	4	2	5	0	2	3
11	Deploying Rings of Power to all sites	5	2	0	0	2	2	0	5	0

In [38]:

```
m_original = raw.drop("Project Title", axis=1)
m = (m_original - m_original.mean()) / m_original.std()
```

We now calculate the full PCA decomposition matrix as

$$T_L = MV_L$$

Where V_L is the matrix whose columns are the eigenvectors of M^TM with only the columns corresponding to the largest L eigenvalues.

In [39]:

```
mtm = pd.DataFrame.transpose(m).to_numpy().dot(m.to_numpy())
w, v = np.linalg.eig(mtm)
# order the eigenvalues and eigenvectors according to eigenvalue magnitude, largest first
order = np.argsort(w)[::-1]
w = w[order]
v = v[:, order]
w
```

Out[39]:

```
array([35.25270065, 30.20781397, 11.94734806,  8.75033103,  7.37141379,
        2.7161016 ,  1.92484525,  0.44266153,  0.38678411])
```

Let's take the top 5 (i.e., $L = 5$), and then renormalize to get back to our $[0, 5]$ rating scheme.

In [42]:

```
l = 5
v_l = v[:, :l]
t_l = m.dot(v_l)
t_l_positive = (t_l - np.amin(t_l, axis=0))
t_l_0_5 = t_l_positive.multiply(5 / np.amax(t_l_positive, axis=0))
reduced = np.rint(t_l_0_5).astype(int)
reduced.index = raw["Project Title"]
reduced
```

Out[42]:

	0	1	2	3	4
Project Title					
Shire TB investigation	0	4	3	3	2
Moria DM program evaluation	3	0	3	3	5
White paper about the ring	1	0	5	3	3
New EMR for Mordor	2	5	4	3	2
Replace old EMR in Moria	1	5	3	4	4
Fancy new UI for EMR	5	1	2	5	2
Easier EMR set-up	5	4	3	4	3
Capacity-building for Some Fancy Tool	4	3	1	0	3
New form for Fellbeast Flu	1	1	0	3	4
Decentralizing data analysis	2	1	4	2	4
Integrating EMR with Some Fancy Tool	4	4	4	3	3
Deploying Rings of Power to all sites	2	0	2	3	0

Now we need to reinterpret the columns. We create a matrix that maps the old columns to the new ones. We color the cells according to the relative weight each old column has in the new column.

In [35]:

```
column_interpretation_matrix = pd.DataFrame(data=v_l, index=raw.columns.drop('Project Title'))
column_interpretation_matrix.style.background_gradient(cmap='Blues')
```

Out[35]:

	0	1	2	3	4
Cross-site relevance	0.300479	-0.418991	-0.186200	-0.269109	-0.034006
Site directors want it	-0.519936	0.049978	0.109291	-0.198548	0.077326
Site non-medical staff wants it	-0.079220	0.501703	0.417355	-0.230592	0.172557
Site medical staff wants it	0.097892	0.419099	-0.046033	0.722511	0.090038
Builds on proven tools	-0.087740	0.440788	-0.572998	-0.234139	0.084994
Improves tools	0.435787	-0.085750	0.531973	0.080385	0.222033
Strengthens MOH partnerships	-0.311303	-0.266170	-0.010630	0.035375	0.846576
Strengthens donor partnerships	-0.445498	-0.041355	0.391989	-0.053510	-0.403470
Other priority projects depend on this	0.361294	0.346982	0.109500	-0.497561	0.137766

This suggests that our new dimensions might be named as follows

In [36]:

```
new_columns = [
    "Future-facing",
    "Meet needs within existing system",
    "Improves data tools",
    "Site medical staff wants it",
    "Strengthens MOH partnerships"
]
```

Yielding the following prioritization matrix T_L

In [43]:

```
result = pd.DataFrame.from_records(reduced)
result.index = raw["Project Title"]
result.columns = new_columns
result.to_csv("data/output/pri-matrix-5.csv")
result
```

Out[43]:

	Future-facing	Meet needs within existing system	Improves data tools	Site medical staff wants it	Strengthens MOH partnerships
Project Title					
Shire TB investigation	0	4	3	3	2
Moria DM program evaluation	3	0	3	3	5
White paper about the ring	1	0	5	3	3
New EMR for Mordor	2	5	4	3	2
Replace old EMR in Moria	1	5	3	4	4
Fancy new UI for EMR	5	1	2	5	2
Easier EMR set-up	5	4	3	4	3
Capacity-building for Some Fancy Tool	4	3	1	0	3
New form for Fellbeast Flu	1	1	0	3	4
Decentralizing data analysis	2	1	4	2	4
Integrating EMR with Some Fancy Tool	4	4	4	3	3
Deploying Rings of Power to all sites	2	0	2	3	0

We note that the mapping has suffered from the process of interpretation—given these column names, these may not be exactly the values we would have chosen for those projects.

If we manually replace those calculated values with ones we choose based on the new column names, creating a new matrix T'_L, we can calculate a measure of the *interpretation error* for each column i as

$$norm(T'_i - T_i)$$

In [46]:

```
manual_raw = pd.read_csv("data/input/pri-matrix-reduced-manual.csv")
t_prime = manual_raw.drop("Project Title", axis=1)
np.linalg.norm(reduced.to_numpy() - t_prime, axis=0)
```

Out[46]:

```
array([1.73205081, 4.24264069, 3.46410162, 4.58257569, 5.        ])
```

Column names can then be refined by attempting to minimize these values.

From here, new projects can be evaluated in this new, smaller prioritization matrix format.