TODIM Ranking

May 28, 2021

1 TODIM Ranking

[1]:
```
import math                        # for sqrt and other functions
import numpy as np                 # for linear algebra
import pandas as pd                # for tabular output
from scipy.stats import rankdata # for ranking the candidates
```

1.1 Step 0 - Obtaining and preprocessing data

[2]:
```
bowlers_data = {
    'weights': '../data/bowling_criteria.csv',
    'scores': '../data/bowlers.csv',
}
batsmen_data = {
    'weights': '../data/batting_criteria.csv',
    'scores': '../data/batsmen.csv',
}
data = bowlers_data
```

[3]:
```
attributes_data = pd.read_csv(data['weights'])
attributes_data
```

[3]:

	Name	Ranking	Ideally
0	SR	1	Lower
1	Econ	2	Lower
2	Avg	3	Lower
3	Wkts	4	Higher
4	Runs	5	Lower
5	Inns	6	Higher
6	TBB	7	Higher
7	4w	8	Higher
8	Mat	9	Higher

[4]:
```
benefit_attributes = set()
attributes = []
ranks = []
n = 0
```

```
for i, row in attributes_data.iterrows():
    attributes.append(row['Name'])
    ranks.append(float(row['Ranking']))
    n += 1

    if row['Ideally'] == 'Higher':
        benefit_attributes.add(i)

ranks = np.array(ranks)
```

[5]:
```
weights = 2 * (n + 1 - ranks) / (n * (n + 1))
pd.DataFrame(data=weights, index=attributes, columns=['Weight'])
```

[5]:

	Weight
SR	0.200000
Econ	0.177778
Avg	0.155556
Wkts	0.133333
Runs	0.111111
Inns	0.088889
TBB	0.066667
4w	0.044444
Mat	0.022222

[6]:
```
original_dataframe = pd.read_csv(data['scores'])
candidates = original_dataframe['Name'].to_numpy()
raw_data = pd.DataFrame(original_dataframe, columns=attributes).to_numpy()

dimensions = raw_data.shape
m = dimensions[0]
n = dimensions[1]

pd.DataFrame(data=raw_data, index=candidates, columns=attributes)
```

[6]:

	SR	Econ	Avg	Wkts	Runs	Inns	TBB	4w	Mat
Andre Russell	16.45	9.51	26.09	11.0	287.0	12.0	181.0	0.0	14.0
Ben Stokes	16.83	11.23	31.50	6.0	189.0	6.0	101.0	0.0	9.0
Chris Morris	15.23	9.27	23.54	13.0	306.0	9.0	198.0	0.0	9.0
Dwayne Bravo	22.45	8.02	30.00	11.0	330.0	12.0	247.0	0.0	12.0
Imran Tahir	14.85	6.70	16.58	26.0	431.0	17.0	386.0	2.0	17.0
Jofra Archer	23.45	6.77	26.45	11.0	291.0	11.0	258.0	0.0	11.0
Kagiso Rabada	11.28	7.83	14.72	25.0	368.0	12.0	282.0	2.0	12.0
Keemo Paul	18.11	8.72	26.33	9.0	237.0	8.0	163.0	0.0	8.0
Lasith Malinga	16.81	9.77	27.38	16.0	438.0	12.0	269.0	2.0	12.0
Moeen Ali	25.00	6.76	28.17	6.0	169.0	9.0	150.0	0.0	11.0
Mohammad Nabi	21.88	6.65	24.25	8.0	194.0	8.0	175.0	1.0	8.0

Rashid Khan	21.18	6.28	22.18	17.0	377.0	15.0	360.0	0.0	15.0
Sam Curran	19.80	9.79	32.30	10.0	323.0	9.0	198.0	1.0	9.0
Sunil Narine	26.60	7.83	34.70	10.0	347.0	12.0	266.0	0.0	12.0
Trent Boult	22.80	8.58	32.60	5.0	163.0	5.0	114.0	0.0	5.0

1.2 Step 1 - Normalizing the Ratings And Weights

$$P_{ij} = \begin{cases} \frac{x_{ij}}{\sum_{k=1}^{m} x_{kj}} & \text{if } j \in J_1 \\ \frac{\frac{1}{x_{ij}}}{\sum_{k=1}^{m} \frac{1}{x_{kj}}} & \text{if } j \in J_2 \end{cases}$$

$$w_{rc} = \frac{w_c}{w_r}$$

and $w_r = \max\{w_c | c = 1, 2, \ldots, n\}$

where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

[7]:
```
for j in range(n):
    column = raw_data[:,j]
    if j in benefit_attributes:
        raw_data[:,j] /= sum(column)
    else:
        column = 1 / column
        raw_data[:,j] = column / sum(column)

pd.DataFrame(data=raw_data, index=candidates, columns=attributes)
```

[7]:

	SR	Econ	Avg	Wkts	Runs	Inns	\
Andre Russell	0.075177	0.056158	0.064018	0.059783	0.062346	0.076433	
Ben Stokes	0.073479	0.047557	0.053023	0.032609	0.094673	0.038217	
Chris Morris	0.081199	0.057612	0.070953	0.070652	0.058474	0.057325	
Dwayne Bravo	0.055085	0.066592	0.055674	0.059783	0.054222	0.076433	
Imran Tahir	0.083277	0.079712	0.100737	0.141304	0.041515	0.108280	
Jofra Archer	0.052736	0.078887	0.063146	0.059783	0.061489	0.070064	
Kagiso Rabada	0.109633	0.068208	0.113466	0.135870	0.048623	0.076433	
Keemo Paul	0.068286	0.061246	0.063434	0.048913	0.075499	0.050955	
Lasith Malinga	0.073567	0.054664	0.061002	0.086957	0.040852	0.076433	
Moeen Ali	0.049466	0.079004	0.059291	0.032609	0.105877	0.057325	
Mohammad Nabi	0.056520	0.080311	0.068875	0.043478	0.092233	0.050955	
Rashid Khan	0.058388	0.085043	0.075303	0.092391	0.047462	0.095541	
Sam Curran	0.062457	0.054552	0.051710	0.054348	0.055397	0.057325	
Sunil Narine	0.046491	0.068208	0.048133	0.054348	0.051565	0.076433	
Trent Boult	0.054239	0.062246	0.051234	0.027174	0.109774	0.031847	

	TBB	4w	Mat
Andre Russell	0.054062	0.000	0.085366
Ben Stokes	0.030167	0.000	0.054878

```
Chris Morris     0.059140  0.000  0.054878
Dwayne Bravo     0.073775  0.000  0.073171
Imran Tahir      0.115293  0.250  0.103659
Jofra Archer     0.077061  0.000  0.067073
Kagiso Rabada    0.084229  0.250  0.073171
Keemo Paul       0.048686  0.000  0.048780
Lasith Malinga   0.080346  0.250  0.073171
Moeen Ali        0.044803  0.000  0.067073
Mohammad Nabi    0.052270  0.125  0.048780
Rashid Khan      0.107527  0.000  0.091463
Sam Curran       0.059140  0.125  0.054878
Sunil Narine     0.079450  0.000  0.073171
Trent Boult      0.034050  0.000  0.030488
```

[8]:
```
max_weight = max(weights)
weights /= max_weight

pd.DataFrame(data=weights, index=attributes, columns=['Weight'])
```

[8]:
```
        Weight
SR    1.000000
Econ  0.888889
Avg   0.777778
Wkts  0.666667
Runs  0.555556
Inns  0.444444
TBB   0.333333
4w    0.222222
Mat   0.111111
```

1.3 Step 2 - Calculating Dominance Degrees

For the contribution of each criteria, we have:

$$\Phi_c\left(A_i, A_j\right) = \begin{cases} \sqrt{\frac{(P_{ic}-P_{jc})w_{rc}}{\sum_{c=1}^{n} w_{rc}}} & \text{if } P_{ic} - P_{jc} > 0 \\ 0 & \text{if } P_{ic} - P_{jc} = 0 \\ -\frac{1}{\theta}\sqrt{\frac{\left(\sum_{c=1}^{n} w_{rc}\right)(P_{jc}-P_{ic})}{w_{rc}}} & \text{if } P_{ic} - P_{jc} < 0 \end{cases}$$

Combining all contributions, we get the dominance degrees:

$$\delta\left(A_i, A_j\right) = \sum_{c=1}^{n} \Phi_c\left(A_i, A_j\right)$$

Here $c = 1, 2, \ldots, n$, $i, j = 1, 2, \ldots, m$.

[9]:
```
# The loss attenuation factor
theta = 1.0
```

[10]:
```
phi = np.zeros((n, m, m))

weight_sum = sum(weights)

for c in range(n):
    for i in range(m):
        for j in range(m):
            pic = raw_data[i,c]
            pjc = raw_data[j,c]
            val = 0
            if pic > pjc:
                val = math.sqrt((pic - pjc) * weights[c] / weight_sum)
            if pic < pjc:
                val = -1.0 / theta * math.sqrt(weight_sum * (pjc - pic) /␣
 ↪weights[c])
            phi[c, i, j] = val
```

[11]:
```
delta = np.zeros((m, m))
for i in range(m):
    for j in range(m):
        delta[i,j] = sum(phi[:,i,j])

pd.DataFrame(data=delta, index=candidates, columns=candidates)
```

[11]:

	Andre Russell	Ben Stokes	Chris Morris	Dwayne Bravo	\
Andre Russell	0.000000	-0.256091	-0.948618	-0.640115	
Ben Stokes	-3.395140	0.000000	-2.367408	-3.153278	
Chris Morris	-1.681511	-0.280010	0.000000	-1.883396	
Dwayne Bravo	-1.480425	-0.635606	-1.023334	0.000000	
Imran Tahir	0.094481	-0.072683	0.124330	0.210453	
Jofra Archer	-1.569885	-0.568460	-0.722302	-0.775823	
Kagiso Rabada	-0.624253	-0.039962	0.203299	0.224265	
Keemo Paul	-2.566535	-0.895393	-1.996320	-2.520664	
Lasith Malinga	-1.293810	-0.294205	-0.724292	-0.329792	
Moeen Ali	-2.594644	-0.116315	-1.519322	-2.119307	
Mohammad Nabi	-2.412364	-0.652604	-1.831497	-2.249504	
Rashid Khan	-0.363674	-0.525040	-0.359261	0.066327	
Sam Curran	-2.622509	-0.672382	-1.230612	-2.336888	
Sunil Narine	-1.865025	-0.917318	-1.257069	-0.747582	
Trent Boult	-3.826860	-1.826479	-3.386515	-3.672372	

	Imran Tahir	Jofra Archer	Kagiso Rabada	Keemo Paul	\
Andre Russell	-6.620828	-0.812562	-4.983520	-0.333476	
Ben Stokes	-7.897602	-3.179277	-7.033048	-1.701473	

Chris Morris	-7.104803	-2.000113	-6.166372	-0.333710
Dwayne Bravo	-6.860415	-0.902784	-4.725282	-0.737200
Imran Tahir	0.000000	0.089699	-0.705105	0.024531
Jofra Archer	-6.753051	0.000000	-5.267292	-0.478002
Kagiso Rabada	-2.763542	-0.126805	0.000000	0.055154
Keemo Paul	-7.602526	-2.521952	-6.657522	0.000000
Lasith Malinga	-4.310707	-0.637133	-2.392892	-0.549940
Moeen Ali	-7.248964	-1.736133	-6.023291	-0.902745
Mohammad Nabi	-6.613377	-2.124177	-5.719613	-0.224017
Rashid Khan	-5.138686	-0.063239	-3.890964	-0.384254
Sam Curran	-6.862080	-2.596379	-5.808854	-0.901349
Sunil Narine	-6.913739	-1.185152	-4.613248	-0.929599
Trent Boult	-8.314198	-3.728651	-7.495802	-2.713294

	Lasith Malinga	Moeen Ali	Mohammad Nabi	Rashid Khan	\
Andre Russell	-3.329825	-0.739164	-2.546245	-2.951349	
Ben Stokes	-5.810701	-2.542547	-3.579048	-4.543259	
Chris Morris	-4.510778	-1.516449	-2.380317	-3.652038	
Dwayne Bravo	-3.541567	-0.908027	-2.760399	-3.355487	
Imran Tahir	0.410632	-0.197184	-0.230257	-0.012023	
Jofra Archer	-4.027618	-0.467397	-2.473359	-3.347820	
Kagiso Rabada	0.350569	-0.435901	-0.398507	-1.898623	
Keemo Paul	-5.224812	-1.864451	-2.736070	-4.146433	
Lasith Malinga	0.000000	-0.757209	-0.961624	-3.011334	
Moeen Ali	-5.029017	0.000000	-2.736111	-3.978705	
Mohammad Nabi	-4.593938	-1.299065	0.000000	-3.928230	
Rashid Khan	-2.368579	-0.364650	-1.996668	0.000000	
Sam Curran	-4.571475	-1.796194	-1.159252	-3.995724	
Sunil Narine	-3.554167	-1.180616	-2.980542	-3.502250	
Trent Boult	-6.406009	-2.904881	-4.638627	-5.064456	

	Sam Curran	Sunil Narine	Trent Boult
Andre Russell	-1.719993	-0.674008	-0.528739
Ben Stokes	-3.274775	-2.997130	-0.744730
Chris Morris	-1.472675	-1.949796	-0.524199
Dwayne Bravo	-1.781258	-0.267294	-0.428851
Imran Tahir	0.208900	0.261912	-0.138995
Jofra Archer	-1.651956	-0.793466	-0.442198
Kagiso Rabada	0.278570	0.277966	-0.128276
Keemo Paul	-2.907840	-2.495736	-0.387368
Lasith Malinga	-0.032746	-0.289125	-0.548951
Moeen Ali	-2.607814	-1.924619	-0.121989
Mohammad Nabi	-1.387018	-2.217326	-0.049529
Rashid Khan	-1.737908	0.152185	-0.319922
Sam Curran	0.000000	-2.024779	-0.611952
Sunil Narine	-2.149481	0.000000	-0.820291
Trent Boult	-4.468290	-3.412196	0.000000

1.4 Step 3 - Calculate ratings from the normalised dominance degree values

$$\zeta_i = \frac{\sum_{j=1}^{m} \delta\left(A_i, A_j\right) - \delta_{\min}}{\delta_{\max} - \delta_{\min}}$$

where

$$\delta_{\min} = \min_i \sum_{j=1}^{m} \delta\left(A_i, A_j\right)$$

$$\delta_{\max} = \max_i \sum_{j=1}^{m} \delta\left(A_i, A_j\right)$$

and $i, j = 1, 2, \ldots, m$

[12]:
```
delta_sums = np.zeros(m)
for i in range(m):
    delta_sums[i] = sum(delta[i,:])
pd.DataFrame(data=delta_sums,index=candidates,columns=['Sum'])
```

[12]:

	Sum
Andre Russell	-27.084533
Ben Stokes	-52.219415
Chris Morris	-35.456166
Dwayne Bravo	-29.407928
Imran Tahir	0.068690
Jofra Archer	-29.338628
Kagiso Rabada	-5.026045
Keemo Paul	-44.523623
Lasith Malinga	-16.133760
Moeen Ali	-38.658977
Mohammad Nabi	-35.302259
Rashid Khan	-17.294333
Sam Curran	-37.190432
Sunil Narine	-32.616078
Trent Boult	-61.858631

[13]:
```
delta_min = min(delta_sums)
delta_max = max(delta_sums)
pd.DataFrame(data=[delta_min, delta_max], columns=['Value'], index=['Minimum',␣
↪'Maximum'])
```

[13]:

	Value
Minimum	-61.858631
Maximum	0.068690

[14]:
```
ratings = (delta_sums - delta_min) / (delta_max - delta_min)
pd.DataFrame(data=ratings, index=candidates, columns=['Rating'])
```

[14]:

	Rating
Andre Russell	0.561531
Ben Stokes	0.155654
Chris Morris	0.426346
Dwayne Bravo	0.524013
Imran Tahir	1.000000
Jofra Archer	0.525132
Kagiso Rabada	0.917730
Keemo Paul	0.279925
Lasith Malinga	0.738363
Moeen Ali	0.374627
Mohammad Nabi	0.428831
Rashid Khan	0.719623
Sam Curran	0.398341
Sunil Narine	0.472208
Trent Boult	0.000000

1.5 Step 4 - Create ranking based on the calculated ζ_i values

[15]:
```
def rank_according_to(data):
    ranks = (rankdata(data) - 1).astype(int)
    storage = np.zeros_like(candidates)
    storage[ranks] = candidates
    return storage[::-1]
```

[16]:
```
result = rank_according_to(ratings)
pd.DataFrame(data=result, index=range(1, m + 1), columns=['Name'])
```

[16]:

	Name
1	Imran Tahir
2	Kagiso Rabada
3	Lasith Malinga
4	Rashid Khan
5	Andre Russell
6	Jofra Archer
7	Dwayne Bravo
8	Sunil Narine
9	Mohammad Nabi
10	Chris Morris
11	Sam Curran
12	Moeen Ali
13	Keemo Paul
14	Ben Stokes
15	Trent Boult

[17]:
```
print("The best candidate/alternative according to C* is " + result[0])
print("The preferences in descending order are " + ", ".join(result) + ".")
```

```
The best candidate/alternative according to C* is Imran Tahir
The preferences in descending order are Imran Tahir, Kagiso Rabada, Lasith
Malinga, Rashid Khan, Andre Russell, Jofra Archer, Dwayne Bravo, Sunil Narine,
Mohammad Nabi, Chris Morris, Sam Curran, Moeen Ali, Keemo Paul, Ben Stokes,
Trent Boult.
```