

Batsmen Ranking

May 18, 2021

1 TOPSIS Ranking for Batsmen

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

```
[2]: attributes_data = pd.read_csv('data/batting_criteria.csv')
attributes_data
```

```
[2]:
```

	Name	Ranking	Ideally
0	SR	1	Higher
1	Avg	2	Higher
2	Runs	3	Higher
3	Inn	4	Higher
4	NO	5	Higher
5	6s	6	Higher
6	4s	7	Higher
7	100s	8	Higher
8	50s	9	Higher
9	Mat	10	Higher
10	HS	11	Higher
11	BF	12	Higher

```
[3]: 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)
```

```
[4]: weights = 2 * (n + 1 - ranks) / (n * (n + 1))
weights
```

```
[4]: array([0.15384615, 0.14102564, 0.12820513, 0.11538462, 0.1025641 ,
          0.08974359, 0.07692308, 0.06410256, 0.05128205, 0.03846154,
          0.02564103, 0.01282051])
```

```
[5]: original_dataframe = pd.read_csv('data/batsmen.csv')
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)
```

```
[5]:
```

	SR	Avg	Runs	Inn	NO	6s	4s	100s	50s	Mat	\
AB de Villiers	154.00	44.20	442.0	13.0	3.0	26.0	31.0	0.0	5.0	13.0	
Andre Russel	204.81	56.67	510.0	13.0	4.0	52.0	31.0	0.0	4.0	14.0	
Ben Stokes	124.24	20.50	123.0	9.0	3.0	4.0	8.0	0.0	0.0	9.0	
Chris Gayle	153.60	40.83	490.0	13.0	1.0	34.0	45.0	0.0	4.0	13.0	
Chris Lynn	139.65	31.15	405.0	13.0	0.0	22.0	41.0	0.0	4.0	13.0	
David Warner	143.86	69.20	692.0	12.0	2.0	21.0	57.0	1.0	8.0	12.0	
Faf Du Plessis	123.36	36.00	396.0	12.0	1.0	15.0	36.0	0.0	3.0	12.0	
Jonny Bairstow	157.24	55.63	445.0	10.0	2.0	18.0	48.0	1.0	2.0	10.0	
Jos Buttler	151.70	38.88	311.0	8.0	0.0	14.0	38.0	0.0	3.0	8.0	
Kane Williamson	120.00	22.29	156.0	9.0	2.0	5.0	12.0	0.0	1.0	9.0	
Kieron Pollard	156.74	34.88	279.0	14.0	6.0	22.0	14.0	0.0	1.0	16.0	
Marcus Stoinis	135.25	52.75	211.0	10.0	6.0	10.0	14.0	0.0	0.0	10.0	
Moeen Ali	165.41	27.50	220.0	10.0	2.0	17.0	16.0	0.0	2.0	11.0	
Quinton de Kock	132.91	35.27	529.0	16.0	1.0	25.0	45.0	0.0	4.0	16.0	
Shane Watson	127.56	23.41	398.0	17.0	0.0	20.0	42.0	0.0	3.0	17.0	
Steve Smith	116.00	39.88	319.0	10.0	2.0	4.0	30.0	0.0	3.0	12.0	

	HS	BF
AB de Villiers	82.0	287.0
Andre Russel	80.0	249.0
Ben Stokes	46.0	99.0
Chris Gayle	99.0	319.0
Chris Lynn	82.0	290.0
David Warner	100.0	481.0
Faf Du Plessis	96.0	321.0
Jonny Bairstow	114.0	283.0
Jos Buttler	89.0	205.0
Kane Williamson	70.0	130.0
Kieron Pollard	83.0	178.0

Marcus Stoinis	46.0	156.0
Moeen Ali	66.0	133.0
Quinton de Kock	81.0	398.0
Shane Watson	96.0	312.0
Steve Smith	73.0	275.0

```
[6]: divisors = np.empty(n)
for j in range(n):
    column = raw_data[:,j]
    # print(np.sqrt(column @ column))
    divisors[j] = np.sqrt(column @ column)

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

```
[6]:
```

	SR	Avg	Runs	Inn	NO	6s \
AB de Villiers	0.264163	0.266301	0.277322	0.269260	0.264135	0.287807
Andre Russel	0.351320	0.341432	0.319987	0.269260	0.352180	0.575614
Ben Stokes	0.213114	0.123511	0.077173	0.186411	0.264135	0.044278
Chris Gayle	0.263477	0.245997	0.307438	0.269260	0.088045	0.376363
Chris Lynn	0.239548	0.187676	0.254107	0.269260	0.000000	0.243529
David Warner	0.246770	0.416924	0.434178	0.248548	0.176090	0.232460
Faf Du Plessis	0.211605	0.216897	0.248460	0.248548	0.088045	0.166043
Jonny Bairstow	0.269721	0.335166	0.279204	0.207123	0.176090	0.199251
Jos Buttler	0.260218	0.234249	0.195129	0.165699	0.000000	0.154973
Kane Williamson	0.205841	0.134295	0.097878	0.186411	0.176090	0.055348
Kieron Pollard	0.268863	0.210149	0.175052	0.289973	0.528271	0.243529
Marcus Stoinis	0.232000	0.317815	0.132387	0.207123	0.528271	0.110695
Moeen Ali	0.283735	0.165685	0.138034	0.207123	0.176090	0.188182
Quinton de Kock	0.227987	0.212499	0.331908	0.331397	0.088045	0.276738
Shane Watson	0.218809	0.141043	0.249715	0.352110	0.000000	0.221390
Steve Smith	0.198980	0.240274	0.200149	0.207123	0.176090	0.044278

	4s	100s	50s	Mat	HS	BF
AB de Villiers	0.222189	0.000000	0.354441	0.260889	0.245830	0.259994
Andre Russel	0.222189	0.000000	0.283552	0.280957	0.239834	0.225570
Ben Stokes	0.057339	0.000000	0.000000	0.180615	0.137905	0.089684
Chris Gayle	0.322533	0.000000	0.283552	0.260889	0.296795	0.288983
Chris Lynn	0.293863	0.000000	0.283552	0.260889	0.245830	0.262712
David Warner	0.408542	0.707107	0.567105	0.240820	0.299792	0.435740
Faf Du Plessis	0.258026	0.000000	0.212664	0.240820	0.287801	0.290795
Jonny Bairstow	0.344035	0.707107	0.141776	0.200683	0.341763	0.256371
Jos Buttler	0.272361	0.000000	0.212664	0.160547	0.266815	0.185710
Kane Williamson	0.086009	0.000000	0.070888	0.180615	0.209855	0.117767
Kieron Pollard	0.100344	0.000000	0.070888	0.321094	0.248828	0.161251
Marcus Stoinis	0.100344	0.000000	0.000000	0.200683	0.137905	0.141321
Moeen Ali	0.114678	0.000000	0.141776	0.220752	0.197863	0.120485

Quinton de Kock	0.322533	0.000000	0.283552	0.321094	0.242832	0.360550
Shane Watson	0.301031	0.000000	0.212664	0.341162	0.287801	0.282642
Steve Smith	0.215022	0.000000	0.212664	0.240820	0.218848	0.249123

```
[7]: raw_data *= weights
      pd.DataFrame(data=raw_data, index=candidates, columns=attributes)
```

	SR	Avg	Runs	Inn	NO	6s \
AB de Villiers	0.040640	0.037555	0.035554	0.031068	0.027091	0.025829
Andre Russel	0.054049	0.048151	0.041024	0.031068	0.036121	0.051658
Ben Stokes	0.032787	0.017418	0.009894	0.021509	0.027091	0.003974
Chris Gayle	0.040535	0.034692	0.039415	0.031068	0.009030	0.033776
Chris Lynn	0.036854	0.026467	0.032578	0.031068	0.000000	0.021855
David Warner	0.037965	0.058797	0.055664	0.028679	0.018061	0.020862
Faf Du Plessis	0.032555	0.030588	0.031854	0.028679	0.009030	0.014901
Jonny Bairstow	0.041496	0.047267	0.035795	0.023899	0.018061	0.017882
Jos Buttler	0.040034	0.033035	0.025017	0.019119	0.000000	0.013908
Kane Williamson	0.031668	0.018939	0.012549	0.021509	0.018061	0.004967
Kieron Pollard	0.041364	0.029636	0.022443	0.033458	0.054182	0.021855
Marcus Stoinis	0.035692	0.044820	0.016973	0.023899	0.054182	0.009934
Moeen Ali	0.043652	0.023366	0.017697	0.023899	0.018061	0.016888
Quinton de Kock	0.035075	0.029968	0.042552	0.038238	0.009030	0.024835
Shane Watson	0.033663	0.019891	0.032015	0.040628	0.000000	0.019868
Steve Smith	0.030612	0.033885	0.025660	0.023899	0.018061	0.003974

	4s	100s	50s	Mat	HS	BF
AB de Villiers	0.017091	0.000000	0.018176	0.010034	0.006303	0.003333
Andre Russel	0.017091	0.000000	0.014541	0.010806	0.006150	0.002892
Ben Stokes	0.004411	0.000000	0.000000	0.006947	0.003536	0.001150
Chris Gayle	0.024810	0.000000	0.014541	0.010034	0.007610	0.003705
Chris Lynn	0.022605	0.000000	0.014541	0.010034	0.006303	0.003368
David Warner	0.031426	0.045327	0.029082	0.009262	0.007687	0.005586
Faf Du Plessis	0.019848	0.000000	0.010906	0.009262	0.007380	0.003728
Jonny Bairstow	0.026464	0.045327	0.007271	0.007719	0.008763	0.003287
Jos Buttler	0.020951	0.000000	0.010906	0.006175	0.006841	0.002381
Kane Williamson	0.006616	0.000000	0.003635	0.006947	0.005381	0.001510
Kieron Pollard	0.007719	0.000000	0.003635	0.012350	0.006380	0.002067
Marcus Stoinis	0.007719	0.000000	0.000000	0.007719	0.003536	0.001812
Moeen Ali	0.008821	0.000000	0.007271	0.008490	0.005073	0.001545
Quinton de Kock	0.024810	0.000000	0.014541	0.012350	0.006226	0.004622
Shane Watson	0.023156	0.000000	0.010906	0.013122	0.007380	0.003624
Steve Smith	0.016540	0.000000	0.010906	0.009262	0.005611	0.003194

```
[8]: a_pos = np.zeros(n)
      a_neg = np.zeros(n)
      for j in range(n):
          column = raw_data[:,j]
```

```

max_val = np.max(column)
min_val = np.min(column)

# See if we want to maximize benefit or minimize cost (for PIS)
if j in benefit_attributes:
    a_pos[j] = max_val
    a_neg[j] = min_val
else:
    a_pos[j] = min_val
    a_neg[j] = max_val

pd.DataFrame(data=[a_pos, a_neg], index=["$A^*$", "$A^-"], columns=attributes)

```

```

[8]:
      SR      Avg      Runs      Inn      NO      6s      4s  \
$A^*$ 0.054049 0.058797 0.055664 0.040628 0.054182 0.051658 0.031426
$A^- 0.030612 0.017418 0.009894 0.019119 0.000000 0.003974 0.004411

      100s      50s      Mat      HS      BF
$A^*$ 0.045327 0.029082 0.013122 0.008763 0.005586
$A^- 0.000000 0.000000 0.006175 0.003536 0.001150

```

```

[9]: sp = np.zeros(m)
      sn = np.zeros(m)
      cs = np.zeros(m)

      for i in range(m):
          diff_pos = raw_data[i] - a_pos
          diff_neg = raw_data[i] - a_neg
          sp[i] = np.sqrt(diff_pos @ diff_pos)
          sn[i] = np.sqrt(diff_neg @ diff_neg)
          cs[i] = sn[i] / (sp[i] + sn[i])

      pd.DataFrame(data=zip(sp, sn, cs), index=candidates, columns=["$S^*$", "$S^-",
      ↪ "$C^*$"])

```

```

[9]:
      $S^*$      $S^-      $C^*$
AB de Villiers 0.070196 0.055112 0.439813
Andre Russel 0.056888 0.081165 0.587927
Ben Stokes 0.106526 0.027294 0.203959
Chris Gayle 0.076169 0.055195 0.420169
Chris Lynn 0.090296 0.040840 0.311430
David Warner 0.051679 0.090778 0.637231
Faf Du Plessis 0.088862 0.036606 0.291756
Jonny Bairstow 0.062814 0.069648 0.525796
Jos Buttler 0.095810 0.032634 0.254072
Kane Williamson 0.105748 0.019120 0.153120
Kieron Pollard 0.079607 0.062885 0.441323

```

Marcus Stoinis	0.087082	0.061924	0.415581
Moeen Ali	0.093227	0.029365	0.239533
Quinton de Kock	0.080278	0.053022	0.397761
Shane Watson	0.094307	0.041946	0.307855
Steve Smith	0.092551	0.033953	0.268397

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

```
[11]: cs_order = rank_according_to(cs)
        sp_order = rank_according_to(sp)
        sn_order = rank_according_to(sn)

        pd.DataFrame(data=zip(cs_order, sp_order[::-1], sn_order), index=range(1, m + 1),
                        columns=["$C*$$", "$S*$$", "$S~$"])
```

```
[11]:
```

	\$C*\$\$	\$S*\$\$	\$S~\$
1	David Warner	David Warner	David Warner
2	Andre Russel	Andre Russel	Andre Russel
3	Jonny Bairstow	Jonny Bairstow	Jonny Bairstow
4	Kieron Pollard	AB de Villiers	Kieron Pollard
5	AB de Villiers	Chris Gayle	Marcus Stoinis
6	Chris Gayle	Kieron Pollard	Chris Gayle
7	Marcus Stoinis	Quinton de Kock	AB de Villiers
8	Quinton de Kock	Marcus Stoinis	Quinton de Kock
9	Chris Lynn	Faf Du Plessis	Shane Watson
10	Shane Watson	Chris Lynn	Chris Lynn
11	Faf Du Plessis	Steve Smith	Faf Du Plessis
12	Steve Smith	Moeen Ali	Steve Smith
13	Jos Buttler	Shane Watson	Jos Buttler
14	Moeen Ali	Jos Buttler	Moeen Ali
15	Ben Stokes	Kane Williamson	Ben Stokes
16	Kane Williamson	Ben Stokes	Kane Williamson

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

The best candidate/alternative according to C* is David Warner
The preferences in descending order are David Warner, Andre Russel, Jonny Bairstow, Kieron Pollard, AB de Villiers, Chris Gayle, Marcus Stoinis, Quinton de Kock, Chris Lynn, Shane Watson, Faf Du Plessis, Steve Smith, Jos Buttler, Moeen Ali, Ben Stokes, Kane Williamson.