Debreceni Egyetem
Természettudományi és Technológiai Kar
Matematikai Intézet

Szakdolgozat

Alkalmazott matematika és a SageMath

készítette:

Szilágyi Bence

témavezető: Dr. Tengely Szabolcs

Debrecen, 2020

Tartalomjegyzék

Köszönetnyilvánítás

Először szeretném megköszönni a családomnak, hogy végig támogattak és biztattak a tanulmányaim során. Barátnőmnek, Mikó Anitának, hogy mindig átsegített a nehézségeken, és nap mint nap motivál az újabb célok elérésében. Szeretném megköszönni a barátaimnak (ide értve a teniszpályán kapott rengeteg támogatást is), akik mindig ott voltak mellettem, ha problémám volt, illetve segítettek kiszakadni a szürke hétköznapokból. Szintén szeretném megköszönni a tanáraimnak, illetve csoporttársaimnak, hogy segítséget nyújtottak az egyetemi tanulmányaim alatt.
Ezenkívül szeretném külön megköszönni Dr. Tengely Szabolcsnak, hogy vállalta a témavezetésem, és segített engem a teljes munkám ideje alatt; illetve Molnár Alexandrának, akivel egymást segítve jutottunk túl a szakdolgozatírás kezdeti nehézségein.

1. Bevezető

A szakdolgozatom a sportrangsorolással foglalkozik. Ebben a témakörben sokan dolgoztak már, legtöbben az amerikai sportokra alkalmazták. Korábban egy volt Debreceni Egyetemi hallgató, Félegyházi Dávid, is írt szakdolgozatot a témában, "Interaktív alkalmazások implementálása SageMath programcsomagba" címmel. Ő a TeamRanking algoritmusoknál a Colley-módszert, a Massey-módszert, illetve a Keener-módszert vette górcső alá. A sportrangsoroláson belül a PageRank algoritmust és az Élő-pontrendszert (angolul Elo vagy ELO rangsorolás) és azok matematikáját mutatom be. Ehhez a Premier League (Angol labdarúgó bajnokság első osztálya) 2015/2016-os és a 2019/2020-as kiírását használtam fel, mivel ez egy egyenlő feltételeket felvonultató (mindenki játszik mindenkivel oda-vissza vágós alapon) rendszer. Az algoritmusokat a SageMath nevű matematikai programcsomag segítségével állítottam elő és alkalmaztam.

2. PageRank algoritmus

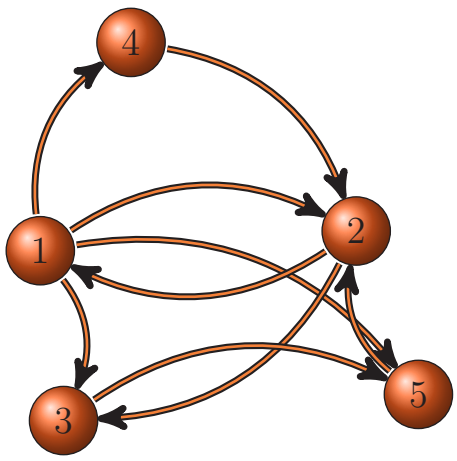
A Pagerank algoritmust Page és Brin találta ki 1998-ban [1], és a Google kereső-motorjának prototípusában használták. A cél egy weboldal népszerűségének vagy fontosságának becslése a háló összekapcsolása alapján. Ennek mögöttes oka: (i) a több bejövő hivatkozással rendelkező oldal fontosabb, mint a kevesebb bejövő hivatkozással ellátott oldal, (ii) szintén fontos egy olyan oldal, amelynek hivatkozása van egy olyan oldalról, amelyről ismert, hogy nagyon fontos.

2.1. Egyszerűsített PageRank algoritmus

Az oldalak közötti kapcsolatokat egy gráf ábrázolja. Egy csúcs egy weboldalt [5] ábrázol, az A oldalról a B oldalra mutató nyíl (irányított él) azt jelenti, hogy van egy link az A oldalról a B oldalra. A kimenő hivatkozások/linkek száma fontos paraméter. "Egy csúcs ki-foka" jelölést használjuk az oldalon lévő kimenő hivatkozások számának megjelölésére. Ezt a gráfot általában webgráfnak nevezik. A gráf minden csúcsát egy oldal azonosítja. $L(p)$-vel jelöljük a kimenő élek számát egy p oldalon.

2.2. Példa

Vegyünk egy öt csúcsú gráfot.



Az 1-es csúcsból mind a négy csúcsba mutat él. A 2-es csúcsból az egyesbe és a hármasba megy él. A 3-as csúcsból csak az ötösbe, a 4-es csúcsból csak a kettesbe, míg az 5-ös csúcsból is csak a kettesbe mutat él. Így a következőket írhatjuk fel: $L(1)$=4, $L(2)$=2, $L(3)$=$L(4)$=$L(5)$=1.

2.3. Egyszerűsített PageRank algoritmus matematikája

Folytatva az eddig leírtakat és a példát követve, legyen N az összes csúcs száma. Hozzunk létre egy $N \times N$-es mátrixot, ami legyen A. Az A mátrix az (i, j)-kel van meghatározva, ahol

$$a_{ij} = \begin{cases} \frac{1}{L(j)} & \text{ha van egy él j és i között,} \\ 0 & \text{egyébként .} \end{cases}$$

A példában az A mátrix 5×5-ös mátrix:

$$\begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 1 & 1 \\ \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 1 & 0 & 0 \end{bmatrix}$$

Az egyes oszlopokban az értékek összege egyenlő. Általában egy mátrixot oszlopsztochasztikusnak tekintünk, ha az értékei nem-negatívak, és az egyes oszlopok összege egyenlő 1-gyel. Az A mátrix oszlopsztochasztikus mátrixot hoz létre, feltéve, hogy minden csúcsnak van legalább egy kimenő éle.
Az egyszerűsített PageRank algoritmus:

- Inicializálja az x-et $N \times 1$-es oszlopvektorba nem-negatív komponensekkel, majd ismétlődően kicseréli x-et Ax-re, amíg az konvergál.

Az x-et nevezzük a PageRank vektornak. Ezt inicializáljuk egy oszlopvektornak, amelynek komponensei egyenlőek egymással.
A példában az x vektor komponensei az x_1, x_2, x_3, x_4 és az x_5. Így az x a következőképpen néz ki:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Ebben az esetben az algoritmus gyorsan konvergál, így hamar megkapjuk a végeredményt. Ezt a következő módon számolhatjuk ki [3]:
A P_i csúcs PageRank értékét $r(P_i)$-vel jelöljük, ami az összes olyan csúcs PageRank értékének az összege, amelyekből mutat él P_i-re.
Ekkor

$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|} \tag{2.3.1}$$

ahol B_{P_i} a P_i-re mutató élek száma és $|P_j|$ a kimenő élek száma a P_j csúcsból. A probléma az egyenlettel, hogy az $r(P_j)$ értékei, és a P_i csúcsba menő élekhez tartozó csúcsok PageRank értékei ismeretlenek. Ennek megoldására Brin és Page iteratív eljárást alkalmazott. Vagyis azt feltételezik, hogy a csúcsok egyenlő PageRank értékekkel rendelkeznek (Mondjuk 1/n. ahol n a csúcsok száma). Ezután az egyenlet szabályát követjük $r(P_i)$ kiszámításához minden egyes P_i csúcsra. A szabályt egymás után alkalmazzuk az előző iteratív értéket $r(P_j)$-re cserélve.
Még egy jelölés az iteratív eljárás meghatározása céljából:
Legyen $r_k + 1(P_i)$ a (P_i) csúcs PageRank értéke k+1 iterációnál.

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|} \tag{2.3.2}$$

Ezt a folyamatot az $r_0(P_i)$=1/n értékkel kezdjük meg az összes P_i csúcson és megismételjük annak a reményében, hogy a PageRank értékei néhány lépés után véges értékhez fog konvergálni.

Én a példa PageRank értékeinek kiszámításához a SageMath programot hívtam segítségül, amibe alapból be van építve az algoritmus. Íme a végeredmény:

Csúcs	PageRank érték
1	0.16784207409035679
2	0.32433429197730995
3	0.20350851483455765
4	0.06566644074420083
5	0.2386486783535748

A példánkban szereplő gráf esetén a 2-es csúcs rendelkezik a legnagyobb PageRank értékkel.

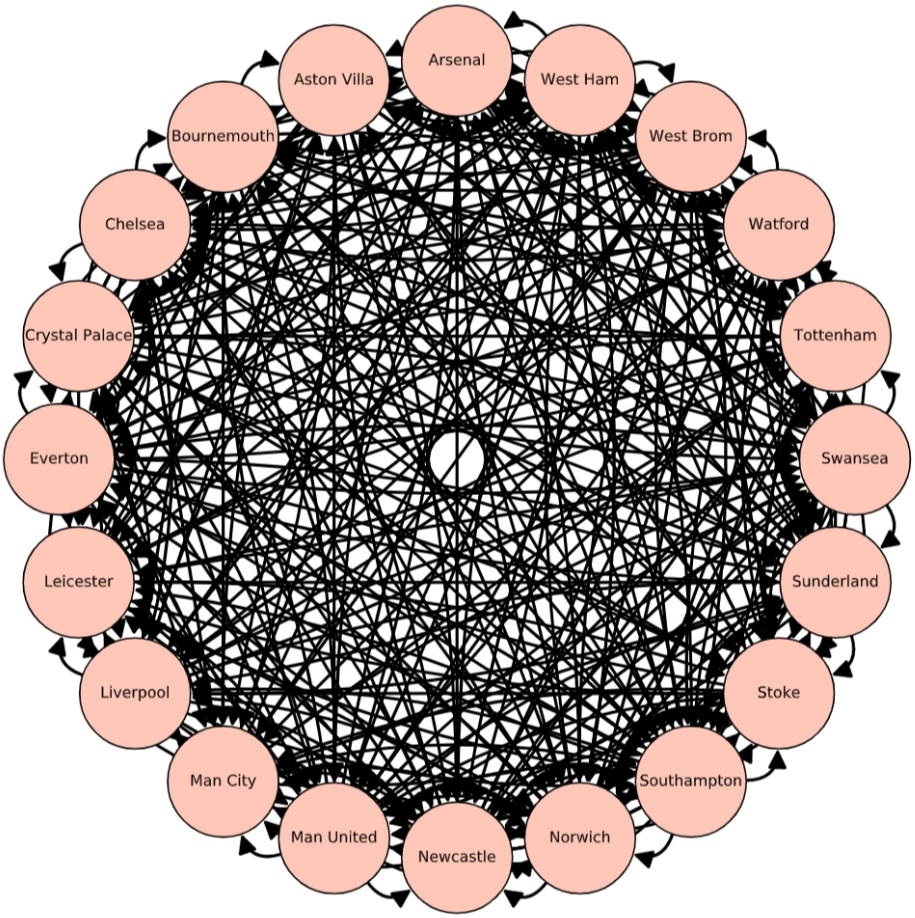
2.4. PageRank a Premier League-re

Ebben a részben bemutatom, miként alkalmaztam a PageRank algoritmust a Premier League-re. A bajnokságban húsz csapat szerepel, és mindenki játszik mindenkivel oda-vissza vágós alapon. Az eredmények alapján létrehoztam egy gráfot a SageMath program segítségével. Minden egyes mérkőzés esetében a vesztes csapat csúcsából indult él a győztes csapat csúcsáéba, illetve döntetlen esetén mindkét egyesület kapott egy-egy élt. Alapesetben a győzelemért járó él súlyozása 3, míg a döntetlenért járó élé 1 volt. Ezt utána továbbfejlesztettem a lőtt gólok súlyával. A legvégén pedig hozzáadtam az alábbi értékek súlyát is:

- hazai kaput nem találó lövés súlya
- vendég kaput nem találó lövés súlya
- hazai kaput találó lövés súlya
- vendég kaput találó lövés súlya
- hazai szabálytalanság súlya
- vendég szabálytalanság súlya
- hazai sárga lap súlya
- vendég sárga lap súlya
- hazai piros lap súlya
- vendég piros lap súlya.

Végezetül pedig kiszámoltam az egyes csapatok PageRank értékeit a SageMath segítségével.

Íme a létrehozott gráf:



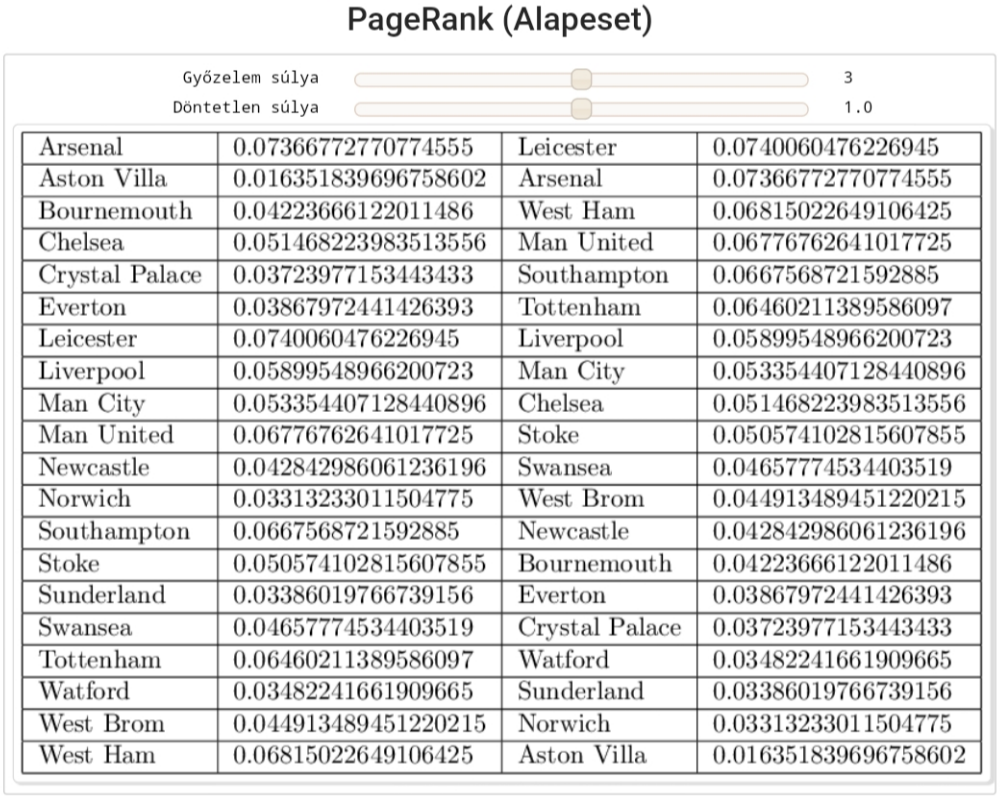
2.5. Program és Programkód

A SageMath [6] matematikai programcsomagban elkészült interaktív alkalmazás az alapesetre nézve. A programkód:

```
html("<h2 align=center>PageRank (Alapeset)</h2>")
@interact(layout=[['gys'],['gs'],['ds']])
def sliders(gys=slider(1,5,1, label='Győzelem súlya',default=3),
ds=slider(0,2,0.5.n(digits=2), label='Döntetlen súlya',default=1)):
    import networkx as nx
    import matplotlib.pyplot as plt
    from sage.graphs.graph_plot import GraphPlot
    options = {'vertex_size': 2500,'layout': 'circular',}
    G20=nx.DiGraph()
    y=[]
    import csv
    a = csv.reader(open('season-1516_csv.csv'))
    elek1=[]
    elek2=[]
    elek3=[]
    for k in a:
    if k[2]!='HomeTeam':
    y=y+[k[2]]
    if k[6]=='H':
etemp=[s for s in elek1 if s[0]==k[3] and s[1]==k[2]]
if etemp==[]:
elek1=elek1+[(k[3],k[2],gys)]
else:
suly=etemp[0][2]+gys
elek1=elek1+[(k[3],k[2],suly)]
    elif k[6]=='A':
etemp=[s for s in elek1 if s[0]==k[2] and s[1]==k[3]]
if etemp==[]:
elek1=elek1+[(k[2],k[3],gys)]
else:
suly=etemp[0][2]+gys
elek1=elek1+[(k[2],k[3],suly)]
    else:
etemp=[s for s in elek1 if (s[0]==k[2] and s[1]==k[3])]
if etemp==[]:
elek1=elek1+[(k[2],k[3],ds)]
```

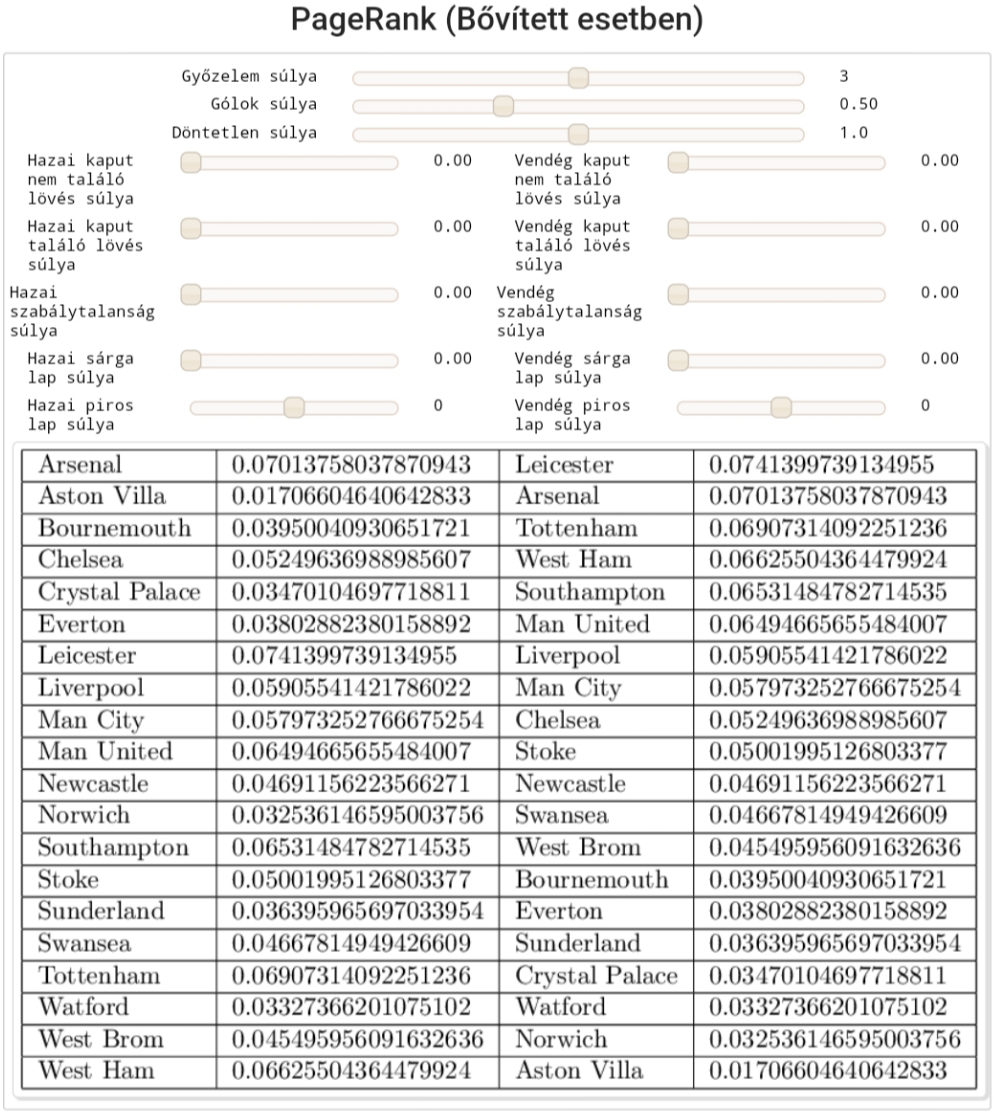
```
else:
suly=etemp[0][2]+ds
elek1=elek1+[(k[2],k[3],suly)]
etemp=[s for s in elek1 if (s[0]==k[3] and s[1]==k[2])]
if etemp==[]:
elek1=elek1+[(k[3],k[2],ds)]
else:
suly=etemp[0][2]+ds
elek1=elek1+[(k[3],k[2],suly)]
    G20.add_nodes_from(y)
    G20.add_weighted_edges_from(elek1)
    pr = nx.pagerank_numpy(G20, alpha=0.99)
    T=list(pr)
    T2=[(k,pr[k]) for k in T]
    T3=sorted(T2, key=lambda tup: -tup[1] )
    T2.sort()
    TU=[(T2[k][0],T2[k][1],T3[k][0],T3[k][1]) for k in
        [0..len(T2)-1]]
    pretty_print(table(rows=TU,frame=True))
```

A futási kép:



Arsenal	0.07366772770774555	Leicester	0.0740060476226945
Aston Villa	0.016351839696758602	Arsenal	0.07366772770774555
Bournemouth	0.04223666122011486	West Ham	0.06815022649106425
Chelsea	0.051468223983513556	Man United	0.06776762641017725
Crystal Palace	0.03723977153443433	Southampton	0.0667568721592885
Everton	0.03867972441426393	Tottenham	0.06460211389586097
Leicester	0.0740060476226945	Liverpool	0.05899548966200723
Liverpool	0.05899548966200723	Man City	0.053354407128440896
Man City	0.053354407128440896	Chelsea	0.051468223983513556
Man United	0.06776762641017725	Stoke	0.050574102815607855
Newcastle	0.042842986061236196	Swansea	0.04657774534403519
Norwich	0.03313233011504775	West Brom	0.044913489451220215
Southampton	0.0667568721592885	Newcastle	0.042842986061236196
Stoke	0.050574102815607855	Bournemouth	0.04223666122011486
Sunderland	0.03386019766739156	Everton	0.03867972441426393
Swansea	0.04657774534403519	Crystal Palace	0.03723977153443433
Tottenham	0.06460211389586097	Watford	0.03482241661909665
Watford	0.03482241661909665	Sunderland	0.03386019766739156
West Brom	0.044913489451220215	Norwich	0.03313233011504775
West Ham	0.06815022649106425	Aston Villa	0.016351839696758602

A futási kép a gólok súlyával bővítve:



Arsenal	0.07013758037870943	Leicester	0.0741399739134955
Aston Villa	0.01706604640642833	Arsenal	0.07013758037870943
Bournemouth	0.03950040930651721	Tottenham	0.06907314092251236
Chelsea	0.05249636988985607	West Ham	0.06625504364479924
Crystal Palace	0.03470104697718811	Southampton	0.06531484782714535
Everton	0.03802882380158892	Man United	0.06494665655484007
Leicester	0.0741399739134955	Liverpool	0.05905541421786022
Liverpool	0.05905541421786022	Man City	0.057973252766675254
Man City	0.057973252766675254	Chelsea	0.05249636988985607
Man United	0.06494665655484007	Stoke	0.05001995126803377
Newcastle	0.04691156223566271	Newcastle	0.04691156223566271
Norwich	0.032536146595003756	Swansea	0.04667814949426609
Southampton	0.06531484782714535	West Brom	0.045495956091632636
Stoke	0.05001995126803377	Bournemouth	0.03950040930651721
Sunderland	0.036395965697033954	Everton	0.03802882380158892
Swansea	0.04667814949426609	Sunderland	0.036395965697033954
Tottenham	0.06907314092251236	Crystal Palace	0.03470104697718811
Watford	0.03327366201075102	Watford	0.03327366201075102
West Brom	0.045495956091632636	Norwich	0.032536146595003756
West Ham	0.06625504364479924	Aston Villa	0.01706604640642833

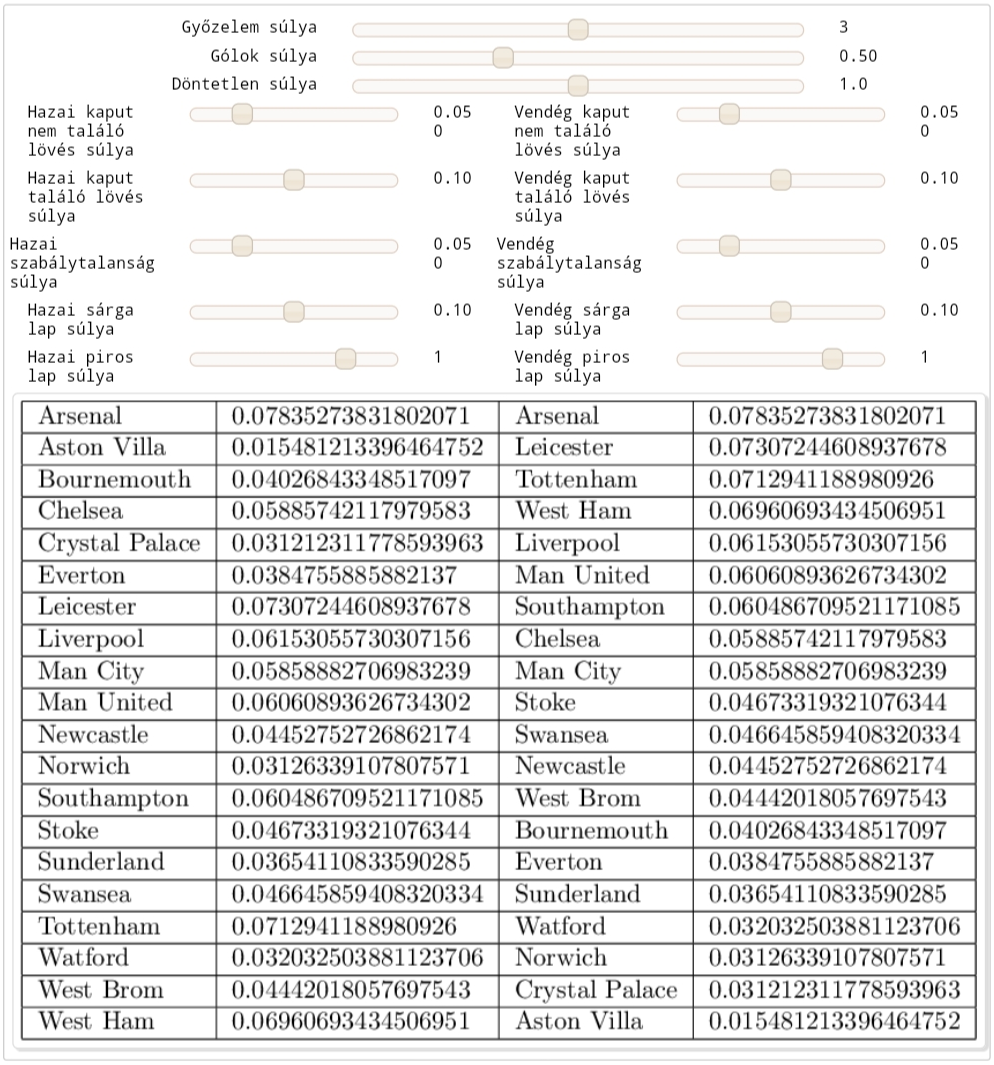
A teljes bővített esethez tartozó programkód:

```
html("<h2 align=center>PageRank (Teljes bővített esetben)</h2>")
@interact(layout=[['gys'],['gs'],['ds'],['hkntl','vkntl'],
['hktl','vktl'],['hsz','vsz'],['hs','vs'],['hp','vp']])
def sliders(gys=slider(1,5,1, label='Győzelem súlya',default=3),
gs=slider(0,1.5,0.25.n(digits=2), label='Gólok súlya',default=0.5),
ds=slider(0,2,0.5.n(digits=2), label='Döntetlen súlya',default=1),
hkntl=slider(0,0.2,0.025.n(digits=2),
label='Hazai kaput nem találó lövés  súlya',default=0.05),
vkntl=slider(0,0.2,0.025.n(digits=2),
label='Vendég kaput nem találó lövés súlya',default=0.05),
hktl=slider(0,0.2,0.05.n(digits=2),
label='Hazai kaput találó lövés súlya',default=0.1),
vktl=slider(0,0.2,0.05.n(digits=2),
label='Vendég kaput találó lövés súlya',default=0.1),
hsz=slider(0,0.2,0.025.n(digits=2),
label='Hazai szabálytalanság súlya',default=0.05),
vsz=slider(0,0.2,0.025.n(digits=2),
label='Vendég szabálytalanság súlya',default=0.05),
hs=slider(0,0.2,0.05.n(digits=2),
label='Hazai sárga lap súlya',default=0.1),
vs=slider(0,0.2,0.05.n(digits=2),
label='Vendég sárga lap súlya',default=0.1),
hp=slider(-2,2,1, label='Hazai piros lap súlya',default=1),
vp=slider(-2,2,1, label='Vendég piros lap  súlya',default=1)):
    import networkx as nx
    import matplotlib.pyplot as plt
    from sage.graphs.graph_plot import GraphPlot
    options = {'vertex_size': 2500,'layout': 'circular',}
    G20=nx.DiGraph()
    y=[]
    import csv
    a = csv.reader(open('season-1516_csv.csv'))
    elek1=[]
    elek2=[]
    elek3=[]
    for k in a:
    if k[2]!='HomeTeam':
    y=y+[k[2]]
    if k[6]=='H':
```

```
etemp=[s for s in elek1 if s[0]==k[3] and s[1]==k[2]]
if etemp==[]:
elek1=elek1+[(k[3],k[2],gys+gs*(Integer(k[4])-
Integer(k[5]))+hkntl*(Integer(k[11])-Integer(k[13]))
-vkntl*(Integer(k[12])-Integer(k[14]))+hktl*(Integer(k[13]))
-vktl*(Integer(k[14]))-hsz*(Integer(k[15]))
+vsz*(Integer(k[16]))-hs*(Integer(k[19]))+vs*(Integer(k[20]))
-hp*(Integer(k[21]))+vp*(Integer(k[22])))]
else:
suly=etemp[0][2]+(gys+gs*(Integer(k[4])-Integer(k[5]))
+hkntl*(Integer(k[11])-Integer(k[13]))-vkntl*(Integer(k[12])
-Integer(k[14]))+hktl*(Integer(k[13]))-vktl*(Integer(k[14]))
-hsz*(Integer(k[15]))+vsz*(Integer(k[16]))-hs*(Integer(k[19]))
+vs*(Integer(k[20]))-hp*(Integer(k[21]))+vp*(Integer(k[22])))
elek1=elek1+[(k[3],k[2],suly)]
    elif k[6]=='A':
etemp=[s for s in elek1 if s[0]==k[2] and s[1]==k[3]]
if etemp==[]:
elek1=elek1+[(k[2],k[3],gys+gs*(Integer(k[4])-Integer(k[5]))
-hkntl*(Integer(k[11])-Integer(k[13]))+vkntl*(Integer(k[12])
-Integer(k[14]))-hktl*(Integer(k[13]))+vktl*(Integer(k[14]))
+hsz*(Integer(k[15]))-vsz*(Integer(k[16]))+hs*(Integer(k[19]))
-vs*(Integer(k[20]))+hp*(Integer(k[21]))-vp*(Integer(k[22])))]
else:
suly=etemp[0][2]+(gys+gs*(Integer(k[5])-Integer(k[4]))
-hkntl*(Integer(k[11])-Integer(k[13]))+vkntl*(Integer(k[12])
-Integer(k[14]))-hktl*(Integer(k[13]))+vktl*(Integer(k[14]))
+hsz*(Integer(k[15]))-vsz*(Integer(k[16]))+hs*(Integer(k[19]))
-vs*(Integer(k[20]))+hp*(Integer(k[21]))-vp*(Integer(k[22])))
elek1=elek1+[(k[2],k[3],suly)]
    else:
etemp=[s for s in elek1 if (s[0]==k[2] and s[1]==k[3])]
if etemp==[]:
elek1=elek1+[(k[2],k[3],ds+gs*((Integer(k[4])+Integer(k[5]))/2))]
else:
suly=etemp[0][2]+ds+gs*((Integer(k[4])+Integer(k[5]))/2)
elek1=elek1+[(k[2],k[3],suly)]
etemp=[s for s in elek1 if (s[0]==k[3] and s[1]==k[2])]
if etemp==[]:
elek1=elek1+[(k[3],k[2],ds+gs*((Integer(k[4])+Integer(k[5]))/2))]
else:
```

```
suly=etemp[0][2]+ds+gs*((Integer(k[4])+Integer(k[5]))/2)
elek1=elek1+[(k[3],k[2],suly)]
    G20.add_nodes_from(y)
    G20.add_weighted_edges_from(elek1)
    pr = nx.pagerank_numpy(G20, alpha=0.99)
    T=list(pr)
    T2=[(k,pr[k]) for k in T]
    T3=sorted(T2, key=lambda tup: -tup[1] )
    T2.sort()
    TU=[(T2[k][0],T2[k][1],T3[k][0],T3[k][1]) for k in
        [0..len(T2)-1]]
    pretty_print(table(rows=TU,frame=True))
```

A teljes bővített esethez tartozó futási kép:



Arsenal	0.07835273831802071	Arsenal	0.07835273831802071
Aston Villa	0.015481213396464752	Leicester	0.07307244608937678
Bournemouth	0.04026843348517097	Tottenham	0.0712941188980926
Chelsea	0.05885742117979583	West Ham	0.06960693434506951
Crystal Palace	0.031212311778593963	Liverpool	0.06153055730307156
Everton	0.0384755885882137	Man United	0.06060893626734302
Leicester	0.07307244608937678	Southampton	0.060486709521171085
Liverpool	0.06153055730307156	Chelsea	0.05885742117979583
Man City	0.05858882706983239	Man City	0.05858882706983239
Man United	0.06060893626734302	Stoke	0.04673319321076344
Newcastle	0.04452752726862174	Swansea	0.046645859408320334
Norwich	0.03126339107807571	Newcastle	0.04452752726862174
Southampton	0.060486709521171085	West Brom	0.04442018057697543
Stoke	0.04673319321076344	Bournemouth	0.04026843348517097
Sunderland	0.03654110833590285	Everton	0.0384755885882137
Swansea	0.046645859408320334	Sunderland	0.03654110833590285
Tottenham	0.0712941188980926	Watford	0.032032503881123706
Watford	0.032032503881123706	Norwich	0.03126339107807571
West Brom	0.04442018057697543	Crystal Palace	0.031212311778593963
West Ham	0.06960693434506951	Aston Villa	0.015481213396464752

3. Élő-pontrendszer

Az Élő-pontrendszert [7] Élő Árpád Imre [8] (nemzetközileg ismertebb az Arpad Elo név) alkotta meg, aki magyar származású, de amerikai sakkozó, fizikaprofesszor és sportvezető volt. Élő a sakkban addig használt (Harkness-rendszer), a játékosok teljesítményére vonatkozó, értékelési rendszer modernizálásának és igazságosabbá tételének céljából alkotta meg a módszerét. Először az 1960-as években kezdték el alkalmazni az Egyesült Államokban, majd később (1970) a Nemzetközi Sakkszövetség is bevezette. Élő egy könyvet is írt e témában, ami a mai napig alapműnek számít a sakkjátékosok értékelésében. Ez a könyv a *The Rating of Chessplayers, Past and Present* címen jelent meg 1978-ban New Yorkban. Élő Árpád pontrendszere még most is használtban van.

A rendszer, a korábban használt szisztémát, ami majdhogynem szubjektíven odaítélt pontszámokról szólt, próbálta úgy meghatározni, hogy az független legyen mindentől, csak a teljesítmény számítson egy adott versenyen. Ezt statisztikai módszerrel kívánta elérni. Ennek lényege, hogy az elkövetkező játszma végkimenetelét, a játékosok addig elért eredményeiből származtatott játékerőt képviselő változó határozza meg. Ezt a változót normális eloszlású (Gauss-eloszlás) valószínűségi változónak nevezzük. Értelemszerűen, ha egy játékos úgy nyer, hogy egy nálánál magasabb rangsoroltat győz le, akkor nagyobb mértékben nő a pontszáma, mintha alacsonyabb pontszámút ver meg. A vereségre hasonló hatással van, tehát, ha jobb ellenféltől kap ki valaki, akkor kevésbé csökken az értéke, mintha rosszabbtól. Döntetlennél, ha a két játékos közel azonos erejű, akkor nem változik semmi, más esetben viszont az erősebb pontokat bukik, míg a gyengébb pontokat nyer. Így az Élő-pontszámok segítségével könnyen kalkulálható volt egy várható eredmény minden sakkjátszma előtt. A játékosok nagyon kedvelték ezt az új módszert, mert így hamar ki tudták számolni az új pontszámaikat.

3.1. Általánosított Élő-pontrendszer

Később a sakkban is egyre többen próbálták továbbfejleszteni az alapváltozatot. Ehhez nagy segítség volt a számítógép bevonása, mivel így bonyolultabb modelleket is létre lehetett hozni, amikből lényegesen nehezebb számítások adódtak, de így már tudták kezelni azokat. Ilyenek voltak Mark Glickman, illetve

Trueskill modelljei, sőt utóbbi rendszere már nemcsak a sakkban volt használható, hanem egyéb sportokban is. Megjelent például a golfban, góban, teniszben, csapatsportokban, de még a számítógépes játékokban is így határozzák meg a rangsorokat.

3.2. Az Élő-pontrendszer a labdarúgásban

Először Buchdahl (2003) használta jelentősen a labdarúgásra. A témában Lars Magnus Hvattum és Halvard Arntzen 2010-ben megalkotott egy átfogó tanulmányt [2]. A szerzők ugyan nem nyújtottak elegendő összehasonlítást más modellekkel, de Robbcrechts és Davis (2018) nemrégiben készített munkája bebizonyította, hogy a módszer megalapozott. Az egyik legfrissebb tanulmány [4], ami ezzel foglalkozik, az 2019-ben jelent meg Hubáček, Šourek és Železný reprezentálásában. Napjainkban a fogadóirodák, illetve bukmékerek is használják a sportfogadás területén, a mérkőzések végkimenetelének meghatározására. A FIFA (Nemzetközi Labdarúgó Szövetség) vezet egy Élő-ranglistát, ami a nemzeti labdarúgó-válogatottak alternatív ranglistája, és minden lejátszott mérkőzés után frissül. Ennek szintén az Élő-pontrendszer az alapja.

3.3. Az Élő-pontrendszer futballra alkalmazott matematikája

A már korábban említett tanulmány (Hvattum és Arntzen) segítségével szeretném bemutatni az Élő-rangsorolás matematikáját a labdarúgásra.
A korábban lejátszott mérkőzések eredményei alapján minden csapathoz hozzárendelhető ELO rangsor (más néven Élő-pontszám), amely a csapat aktuális erősségét mutatja. Legyen l_0^H és l_0^A a mérkőzés kezdetén lévő értéke a hazai és idegenbeli csapatnak. Definiáljunk egy olyan pontozási rendszert, ahol a győzelem 1 pontot ad, a döntetlen 0,5-öt, míg a vereség 0 pontot. Az ELO számításai azt feltételezik, hogy a szóban forgó mérkőzésen átlagosan a hazai és a vendégcsapatnak γ^H és γ^A pontokat kell szereznie, ahol

$$\gamma^H = \frac{1}{1 + c^{(l_0^A - l_0^H)/d}} \tag{3.3.1}$$

$$\gamma^A = 1 - \gamma^H = \frac{1}{1 + c^{(l_0^H - l_0^A)/d}}. \tag{3.3.2}$$

Most adjuk meg a hazai csapat aktuális pontját:

$$\alpha^H = \begin{cases} 1 & \text{ha a hazai csapat nyer,} \\ 0.5 & \text{ha a meccs döntetlen,} \\ 0 & \text{egyébként.} \end{cases}$$

Ekkor a vendég csapat aktuális pontja $\alpha^A = 1 - \alpha^H$. Az Élő-pontszám frissül a meccs után, és a hazai csapat új pontszáma a következő:

$$l_1^H = l_0^H + k(\alpha^H - \gamma^H). \tag{3.3.3}$$

A vendég csapat új pontszáma l_1^A, amit ugyanúgy kell kiszámolni, mint a hazait. A pontszám követi a csapatot, és frissül minden mérkőzés után.
Vegyük figyelembe, hogy az ELO rangsorok kiszámításához egy adott mérkőzés után szükség van néhány kezdeti érték megadására, minden csapat számára. Addig nem lehet várni, hogy az értékek megbízhatóak legyenek, míg elegendő számú korábbi mérkőzések eredményét nem veszik figyelembe.
Az eddig leírt módszerre úgy hivatkozunk, mint alapvető ELO rangsorolás. Ennek egy érdekes változata, amikor úgy frissítjük a k változót, hogy az függ a gólkülönbségtől. Így egy 3-0-ás győzelem nagyobb pontot fog generálni, mint egy 2-1-es. Erre egy lehetőség, hogy a (3.3.3) egyenletbe a k helyére a következő kifejezést írjuk:

$$k = k_0(1 + \delta)^\lambda, \tag{3.3.4}$$

ahol δ az abszolút gólkülönbség, és adottak a $k_0 > 0$ és $\lambda > 0$ állandó paraméterek. Ezt a megközelítést a cél alapú ELO rangsorolásnak nevezik.
Az alapvető ELO rangsorolási módszerben három paraméter található, c, d és a k. Amíg c és d könnyen értelmezhető úgy, hogy megfelelő értékeket adjon a rangsorolásnál, addig a k meghatározásánál óvatosnak kell lenni. Ha a k értéke túl alacsony, akkor nem fogja kellő képen változtatni a csapatok Élő-pontszámot, viszont ha túl nagy a k értéke, akkor megbízhatatlan lesz a rangsorolás, mert nagy mértékben fognak változni a pontszámok egyes mérkőzések után. A cél alapú ELO rangsorolásnál négy paraméter van, mivel k helyett k_0 és λ lesz. A paraméterek megadásánál a $c = 10$-et és a $d = 400$-at alkalmazzuk. A k értékét úgy kapjuk meg, hogy a λ-t 0-nak vesszük az alapestben, ekkor $k = 20$-at kapunk. A másik verzióban a $\lambda = 1$-et tekintjük és a $k_0 = 10$-et, ahonnan kapjuk a $k = 10(1 + \delta)$-t.

3.4. Élő-pontrendszer a Premier League-re

Én a Premier League 2019/2020-as idényének a 28. fordulóig lejátszott mérkőzéseinek az eredményeit használtam fel Élő-rangsorolásom létrehozásához. Az eredmény megkapásához az előbb leírt modellt használtam, azon belül is a cél alapú ELO rangsorolást, némi módosítással. Annyi a változás, hogy a k értéke ugyan 20-ról indul, de változtatható, illetve a gólkülönbséget egy 'gólkülönbségi osztó' bevezetésének segítségéve számolom, ami szintén változtatható érték (alapbeállításban 300). Ez úgy van megvalósítva, hogy veszem az alapvető ELO rangsorolást, és hozzáadom a gólkülönbségért járó pontokat is, amit úgy számolok ki a hazai csapatra, hogy veszem az adott mérkőzésen rúgott góljainak a számát, és megszorzom a vendégcsapat Élő-pontszámának és a 'gólkülönbségi osztónak' a hányadosával. Az idegenben játszó együttes gólkülönbségért járó pontjait analóg módon számoljuk. A c és d paramétereket nem változtattam, azokat én is 10-nek és 400-nak definiáltam, illetve minden csapat 1500-as Élő-pontszámmal kezdett a bajnokságban. A program minden meccs után felülírta az Élő-pontszámokat, és a végén az aktuálisakat írta ki.

3.5. Program és Programkód

A programkód:

```
html("<h2 align=center>ÉLŐ pontszámítás</h2>")
@interact(layout=[['K'],['gk']])
def sliders(K=slider(10,30,1, label='gólkülönbségi szorzó',
default=20),gk=slider(0,800,50, label='gólkülönbségi osztó',
default=300)):
    cs=[]
    import csv
    a = csv.reader(open('E0.csv'))
    for k in a:
    if k[3]!='HomeTeam':
    cs=cs+[k[3]]
    L0=list(Set(cs))
    L=sorted(L0)
    H=[[k,1500] for k in L]
    A=[[k,1500] for k in L]
    c=10
    d=400
    import csv
    a = csv.reader(open('E0.csv'))
    y=[]
    YH=[]
    for k in a:
    if k[3]!='HomeTeam':
    y=y+[k[3]]
    lh=H[L.index(k[3])][1]
      la=A[L.index(k[4])][1]
    hg=k[5]
    ag=k[6]
    if k[7]=='H':
    x=1
    z=0
    YH=1/(1+c^((la-lh)/d))
    YA=1-YH
    elif k[7]=='A':
    x=0
    z=1
    YH=1/(1+c^((la-lh)/d))
```

```
YA=1-YH
else:
x=0.5
z=0.5
YH=1/(1+c^((la-lh)/d))
YA=1-YH
H[L.index(k[3])][1]=round(lh+K*(x-YH))+round(ZZ(hg)*la/gk)
A[L.index(k[4])][1]=round(la+K*(z-YA))+round(ZZ(ag)*lh/gk)
#print("H=",H)
#print("A=",A)
HR=sorted(H, key=lambda tup: -tup[1] )
AR=sorted(A, key=lambda tup: -tup[1] )
T=[(H[k][0],H[k][1],HR[k][0],HR[k][1],A[k][0],A[k][1],AR[k][0],
   AR[k][1]) for k in [0..len(H)-1]]
pretty_print(table(rows=T,frame=True))
```

A futási kép az alapbeállításokkal:

ÉLÖ pontszámítás

gólkülönbségi szorzó 20

gólkülönbségi osztó 300

Arsenal	1646	Liverpool	1775	Arsenal	1565	Liverpool	1733
Aston Villa	1579	Man City	1713	Aston Villa	1514	Man City	1710
Bournemouth	1565	Tottenham	1664	Bournemouth	1492	Leicester	1660
Brighton	1591	Man United	1663	Brighton	1532	Chelsea	1655
Burnley	1602	Leicester	1661	Burnley	1561	Southampton	1613
Chelsea	1594	Arsenal	1646	Chelsea	1655	Wolves	1600
Crystal Palace	1559	Everton	1624	Crystal Palace	1553	Sheffield United	1592
Everton	1624	Wolves	1624	Everton	1567	Tottenham	1573
Leicester	1661	Burnley	1602	Leicester	1660	Everton	1567
Liverpool	1775	Chelsea	1594	Liverpool	1733	Arsenal	1565
Man City	1713	Brighton	1591	Man City	1710	Man United	1563
Man United	1663	Sheffield United	1587	Man United	1563	Burnley	1561
Newcastle	1583	Newcastle	1583	Newcastle	1518	Crystal Palace	1553
Norwich	1549	Aston Villa	1579	Norwich	1470	Brighton	1532
Sheffield United	1587	Bournemouth	1565	Sheffield United	1592	West Ham	1522
Southampton	1548	West Ham	1562	Southampton	1613	Newcastle	1518
Tottenham	1664	Crystal Palace	1559	Tottenham	1573	Watford	1518
Watford	1556	Watford	1556	Watford	1518	Aston Villa	1514
West Ham	1562	Norwich	1549	West Ham	1522	Bournemouth	1492
Wolves	1624	Southampton	1548	Wolves	1600	Norwich	1470

A táblázat első oszlopa az otthon lejátszott meccsek Élő-pontszámát tartalmazza csapatokra lebontva és névsor szerint rendezve. A második oszlop szintén a hazai Élő-pontszámok, csak most érték szerint rangsorolva. A harmadik és negyedik oszlop pedig, az idegenben lejátszott mérkőzések Élő-pontszámát mutatja hasonló rendezésben, mint az első két oszlopban.

3.6. Eredmény meghatározás

Ebben a részben a bajnokságból hátralévő mérkőzések végkimenetelét próbáltam meghatározni az Élő-pontrendszer segítségével. Itt a korábban alkalmazott programot fejlesztettem tovább. Az előzőleg kiszámolt Élő-pontokat vettem figyelembe, valamint minden csapatnál az otthon illetve idegenben elért góljainak a listáját. Egy adott meccs eredményét úgy konstruáltam, hogy összehasonlítottam a hazai csapat aktuális Élő-pontszámát a vendég csapat aktuális Élő-pontszámával. Ha a hazaiaké lényegesen nagyobb (alapesetben a különbség 75 pontnál nagyobb), akkor valószínűleg ők fognak győzni. A konkrét eredményhez, vettem a csapatok átlag gólszámát (mindkét esetben), és az otthon játszó együttesnél kitoltam a határokat felfelé, míg az idegenben játszónál csökkentettem, és a program ezen intervallumokból generál egy-egy számot. Ha a vendégeké a lényegesen nagyobb, akkor ugyanez az eljárás, csak pepitában. Ha a két Élő-pontszám viszonylag közel van egymáshoz (a különbség 75 ponton belül van),

akkor a két csapat átlagban szerzett góljai számának a halmazát azonos mértékben módosítom, és innen kapjuk a végkimenetelt. A program minden mérkőzés után frissül, azaz az eredmények hozzáadódnak a szerzett gólok listájához, és új átlagok keletkeznek, valamint az Élő-pontok is újra számolódnak.

3.7. Program és Programkód 2

A programkód:

```
cs=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
if k[3]!='HomeTeam':
cs=cs+[k[3]]
L0=list(Set(cs))
L=sorted(L0)

e=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[0]:
        e=e+[ZZ(k[5])]
E0=list(Set(e))
E=sorted(E0)
e1=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[1]:
        e1=e1+[ZZ(k[5])]
E01=list(Set(e1))
E1=sorted(E01)
e2=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[2]:
        e2=e2+[ZZ(k[5])]
E02=list(Set(e2))
E2=sorted(E02)
e3=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
```

```
    if k[3]==L[3]:
        e3=e3+[ZZ(k[5])]
E03=list(Set(e3))
E3=sorted(E03)
e4=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[4]:
        e4=e4+[ZZ(k[5])]
E04=list(Set(e4))
E4=sorted(E04)
e5=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[5]:
        e5=e5+[ZZ(k[5])]
E05=list(Set(e5))
E5=sorted(E05)
e6=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[6]:
        e6=e6+[ZZ(k[5])]
E06=list(Set(e6))
E6=sorted(E06)
e7=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[7]:
        e7=e7+[ZZ(k[5])]
E07=list(Set(e7))
E7=sorted(E07)
e8=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[8]:
```

```
        e8=e8+[ZZ(k[5])]
E08=list(Set(e8))
E8=sorted(E08)
e9=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[9]:
        e9=e9+[ZZ(k[5])]
E09=list(Set(e9))
E9=sorted(E09)
e10=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[10]:
        e10=e10+[ZZ(k[5])]
E010=list(Set(e10))
E10=sorted(E010)
e11=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[11]:
        e11=e11+[ZZ(k[5])]
E011=list(Set(e11))
E11=sorted(E011)
e12=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[12]:
        e12=e12+[ZZ(k[5])]
E012=list(Set(e12))
E12=sorted(E012)
e13=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[13]:
        e13=e13+[ZZ(k[5])]
```

```
E013=list(Set(e13))
E13=sorted(E013)
e14=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[14]:
        e14=e14+[ZZ(k[5])]
E014=list(Set(e14))
E14=sorted(E014)
e15=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[15]:
        e15=e15+[ZZ(k[5])]
E015=list(Set(e15))
E15=sorted(E015)
e16=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[16]:
        e16=e16+[ZZ(k[5])]
E016=list(Set(e16))
E16=sorted(E016)
e17=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[17]:
        e17=e17+[ZZ(k[5])]
E017=list(Set(e17))
E17=sorted(E017)
e18=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[18]:
        e18=e18+[ZZ(k[5])]
E018=list(Set(e18))
```

```
E18=sorted(E018)
e19=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[3]==L[19]:
        e19=e19+[ZZ(k[5])]
E019=list(Set(e19))
E19=sorted(E019)

EJ0=[e,e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12,e13,e14,e15,e16,e17,
     e18,e19]
EJ=[E,E1,E2,E3,E4,E5,E6,E7,E8,E9,E10,E11,E12,E13,E14,E15,E16,E17,
    E18,E19]
HG=[[L[k],EJ[k]] for k in [0..19]]

e=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[0]:
        e=e+[ZZ(k[6])]
E0=list(Set(e))
E=sorted(E0)
e1=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[1]:
        e1=e1+[ZZ(k[6])]
E01=list(Set(e1))
E1=sorted(E01)
e2=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[2]:
        e2=e2+[ZZ(k[6])]
E02=list(Set(e2))
E2=sorted(E02)
e3=[]
```

```
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[3]:
        e3=e3+[ZZ(k[6])]
E03=list(Set(e3))
E3=sorted(E03)
e4=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[4]:
        e4=e4+[ZZ(k[6])]
E04=list(Set(e4))
E4=sorted(E04)
e5=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[5]:
        e5=e5+[ZZ(k[6])]
E05=list(Set(e5))
E5=sorted(E05)
e6=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[6]:
        e6=e6+[ZZ(k[6])]
E06=list(Set(e6))
E6=sorted(E06)
e7=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[7]:
        e7=e7+[ZZ(k[6])]
E07=list(Set(e7))
E7=sorted(E07)
e8=[]
import csv
```

```
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[8]:
        e8=e8+[ZZ(k[6])]
E08=list(Set(e8))
E8=sorted(E08)
e9=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[9]:
        e9=e9+[ZZ(k[6])]
E09=list(Set(e9))
E9=sorted(E09)
e10=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[10]:
        e10=e10+[ZZ(k[6])]
E010=list(Set(e10))
E10=sorted(E010)
e11=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[11]:
        e11=e11+[ZZ(k[6])]
E011=list(Set(e11))
E11=sorted(E011)
e12=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[12]:
        e12=e12+[ZZ(k[6])]
E012=list(Set(e12))
E12=sorted(E012)
e13=[]
import csv
a = csv.reader(open('E0.csv'))
```

```
for k in a:
    if k[4]==L[13]:
        e13=e13+[ZZ(k[6])]
E013=list(Set(e13))
E13=sorted(E013)
e14=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[14]:
        e14=e14+[ZZ(k[6])]
E014=list(Set(e14))
E14=sorted(E014)
e15=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[15]:
        e15=e15+[ZZ(k[6])]
E015=list(Set(e15))
E15=sorted(E015)
e16=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[16]:
        e16=e16+[ZZ(k[6])]
E016=list(Set(e16))
E16=sorted(E016)
e17=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[17]:
        e17=e17+[ZZ(k[6])]
E017=list(Set(e17))
E17=sorted(E017)
e18=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
```

```
    if k[4]==L[18]:
        e18=e18+[ZZ(k[6])]
E018=list(Set(e18))
E18=sorted(E018)
e19=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
    if k[4]==L[19]:
        e19=e19+[ZZ(k[6])]
E019=list(Set(e19))
E19=sorted(E019)

EJA0=[e,e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12,e13,e14,e15,e16,
      e17,e18,e19]
EJA=[E,E1,E2,E3,E4,E5,E6,E7,E8,E9,E10,E11,E12,E13,E14,E15,E16,E17,
     E18,E19]
AG=[[L[k],EJA[k]] for k in [0..19]]

hcs=[]
import csv
a = csv.reader(open('S_jo.csv'))
for k in a:
if k[2]!='Home Team':
hcs=hcs+[k[2]]
vcs=[]
import csv
a = csv.reader(open('S_jo.csv'))
for k in a:
if k[3]!='Away Team':
vcs=vcs+[k[3]]

ph=[[hcs[k],vcs[k]] for k in [0..len(hcs)-1]]

html("<h2 align=center>ÉLŐ pontszámítás</h2>")
@interact(layout=[['K'],['gk']])
def sliders(K=slider(10,30,1, label='gólkülönbségi szorzó',default=20),
gk=slider(0,800,50, label='gólkülönbségi osztó',default=300)):
    cs=[]
    import csv
    a = csv.reader(open('E0.csv'))
```

```
for k in a:
    if k[3]!='HomeTeam':
        cs=cs+[k[3]]
L0=list(Set(cs))
L=sorted(L0)
H=[[k,1500] for k in L]
A=[[k,1500] for k in L]
c=10
d=400
#K=20
import csv
a = csv.reader(open('E0.csv'))
y=[]
YH=[]
for k in a:
    if k[3]!='HomeTeam':
        y=y+[k[3]]
        lh=H[L.index(k[3])][1]
        la=A[L.index(k[4])][1]
        hg=k[5]
        ag=k[6]
        if k[7]=='H':
            x=1
            z=0
            YH=1/(1+c^((la-lh)/d))
            YA=1-YH
        elif k[7]=='A':
            x=0
            z=1
            YH=1/(1+c^((la-lh)/d))
            YA=1-YH
        else:
            x=0.5
            z=0.5
            YH=1/(1+c^((la-lh)/d))
            YA=1-YH
        H[L.index(k[3])][1]=round(lh+K*(x-YH))+round(ZZ(hg)*la/gk)
        A[L.index(k[4])][1]=round(la+K*(z-YA))+round(ZZ(ag)*lh/gk)
print(H)
print(A)
HR=sorted(H, key=lambda tup: -tup[1] )
```

```
AR=sorted(A, key=lambda tup: -tup[1] )
T=[(H[k][0],H[k][1],HR[k][0],HR[k][1],A[k][0],A[k][1],AR[k][0],
   AR[k][1]) for k in [0..len(H)-1]]
#pretty_print(table(rows=T,frame=True))
HG0=[[L[k],EJ0[k]] for k in [0..19]]
AG0=[[L[k],EJA0[k]] for k in [0..19]]
dPL={L[0]:[HG0[0],AG0[0]],L[1]:[HG0[1],AG0[1]],L[2]:[HG0[2],
     AG0[2]],L[3]:[HG0[3],AG0[3]],L[4]:[HG0[4],AG0[4]],L[5]:
     L[8]:[HG0[8],AG0[8]],L[9]:[HG0[9],AG0[9]],L[10]:[HG0[10],
     AG0[10]],L[11]:[HG0[11],AG0[11]],L[12]:[HG0[12],AG0[12]],
     L[13]:[HG0[13],AG0[13]],L[14]:[HG0[14],AG0[14]],L[15]:
     AG0[7]],L[18]:[HG0[18],AG0[18]],L[19]:[HG0[19],AG0[19]]}
szamossaghg2=[len(dPL[L[k]][0][1]) for k in [0..19]]
szamossagag2=[len(dPL[L[k]][1][1]) for k in [0..19]]
osszeghg2=[sum([ZZ(s) for s in dPL[L[k]][0][1]]) for k in [0..19]]
osszegag2=[sum([ZZ(s) for s in dPL[L[k]][1][1]]) for k in [0..19]]
atlagh2=[[L[k],((osszeghg2[k]/szamossaghg2[k]).round("toward"))]
for k in [0..19]]
atlaga2=[[L[k],((osszegag2[k]/szamossagag2[k]).round("toward"))]
for k in [0..19]]
hgol=[]
vgol=[]
H=[['Arsenal', 1646], ['Aston Villa', 1579], ['Bournemouth',
    1565], ['Brighton', 1591], ['Burnley', 1602], ['Chelsea',
    1594], ['Crystal Palace', 1559], ['Everton', 1624],
   ['Leicester', 1661], ['Liverpool', 1775], ['Man City', 1713],
   ['Man United', 1663], ['Newcastle', 1583], ['Norwich', 1549],
   ['Tottenham', 1664], ['Watford', 1556], ['West Ham', 1562],

A=[['Arsenal', 1565], ['Aston Villa', 1514], ['Bournemouth',
    1492], ['Brighton', 1532], ['Burnley', 1561], ['Chelsea',
    1655], ['Crystal Palace', 1553], ['Everton', 1567],
   ['Leicester', 1660], ['Liverpool', 1733], ['Man City', 1710],
   ['Man United', 1563], ['Newcastle', 1518], ['Norwich', 1470],
   ['Tottenham', 1573], ['Watford', 1518], ['West Ham', 1522],
c=10
d=400
cs=[]
import csv
a = csv.reader(open('E0.csv'))
for k in a:
```

```
if k[3]!='HomeTeam':
cs=cs+[k[3]]
L0=list(Set(cs))
L=sorted(L0)
for h in [0..len(ph)-1]:
    for k in [0..len(H)-1]:
        for i in [0..len(H)-1]:
            if ph[h][0]==H[k][0] and ph[h][1]==A[i][0]:
                b=ZZ(H[k][1])-ZZ(A[i][1])
                #print(b)
                #print H[k][0],A[i][0]
                if b>75:
                    #print(1)
                    hgol=hgol+[ZZ.random_element(
                    ((ZZ(atlagh2[k][1]))*(1/2)).round("toward"),
                    2*ZZ(atlagh2[k][1])+1)]
                    vgol=vgol+[ZZ.random_element(0,
                    ((ZZ(atlaga2[i][1]))*(3/2)+1).
                    round("toward"))]
                elif b<-75:
                    #print(2)
                    hgol=hgol+[ZZ.random_element(0,
                    ((ZZ(atlagh2[k][1]))*(3/2)+1).
                    round("toward"))]
                    vgol=vgol+[ZZ.random_element(
                    ((ZZ(atlaga2[i][1])*((1/2)).round("toward"),
                    2*ZZ(atlaga2[i][1])+1)]
                else:
                    #print(3)
                    hgol=hgol+[ZZ.random_element(
                    (ZZ(atlagh2[k][1])*(1/2)).round("toward"),
                    (ZZ((atlagh2[k][1]))*(3/2)+1).
                    round("toward"))]
                    vgol=vgol+[ZZ.random_element(
                    (ZZ(atlaga2[i][1])*(1/2)).round("toward"),
                    ((3/2)*ZZ(atlaga2[i][1])+1).round("toward"))]
                lh=H[L.index(ph[h][0])][1]
                la=A[L.index(ph[h][1])][1]
                if ZZ(hgol[h])>ZZ(vgol[h]):
                    x=1
                    z=0
```

```
                    YH=1/(1+c^((la-lh)/d))
                    YA=1-YH
                elif ZZ(hgol[h])<ZZ(vgol[h]):
                    x=0
                    z=1
                    YH=1/(1+c^((la-lh)/d))
                    YA=1-YH
                else:
                    x=0.5
                    z=0.5
                    YH=1/(1+c^((la-lh)/d))
                    YA=1-YH
                H[L.index(ph[h][0])][1]=round(lh+K*(x-YH))
                    +round(ZZ(hgol[h])*la/gk)
                A[L.index(ph[h][1])][1]=round(la+K*(z-YA))
                +round(ZZ(vgol[h])*lh/gk)
print hgol
print vgol
print (H)
print (A)
html("<h2 align=center>Az új Élő-pontszámok</h2>")
HR=sorted(H, key=lambda tup: -tup[1] )
AR=sorted(A, key=lambda tup: -tup[1] )
T=[(H[k][0],H[k][1],HR[k][0],HR[k][1],A[k][0],A[k][1],AR[k][0],
AR[k][1]) for k in [0..len(H)-1]]
pretty_print(table(rows=T,frame=True))
html("<h3 align=center>27. forduló</h3>")
T2=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [0..0]]
pretty_print(table(rows=T2,frame=True))
html("<h3 align=center>28. forduló</h3>")
T3=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [1..10]]
pretty_print(table(rows=T3,frame=True))
html("<h3 align=center>29. forduló</h3>")
T4=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [11..20]]
pretty_print(table(rows=T4,frame=True))
html("<h3 align=center>30. forduló</h3>")
T5=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [21..30]]
pretty_print(table(rows=T5,frame=True))
html("<h3 align=center>31. forduló</h3>")
T6=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [31..40]]
pretty_print(table(rows=T6,frame=True))
```

```
html("<h3 align=center>32. forduló</h3>")
T7=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [41..50]]
pretty_print(table(rows=T7,frame=True))
html("<h3 align=center>33. forduló</h3>")
T8=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [51..60]]
pretty_print(table(rows=T8,frame=True))
html("<h3 align=center>34. forduló</h3>")
T9=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [61..70]]
pretty_print(table(rows=T9,frame=True))
html("<h3 align=center>35. forduló</h3>")
T10=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [71..80]]
pretty_print(table(rows=T10,frame=True))
html("<h3 align=center>36. forduló</h3>")
T11=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [81..90]]
pretty_print(table(rows=T11,frame=True))
html("<h3 align=center>37. forduló</h3>")
T12=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [91..100]]
pretty_print(table(rows=T12,frame=True))
html("<h3 align=center>38. forduló</h3>")
T13=[(ph[k][0],hgol[k],vgol[k],ph[k][1]) for k in [101..110]]
pretty_print(table(rows=T13,frame=True))
```

A 27. fordulóból hátra lévő mérkőzés eredménye:

27. forduló

Liverpool	6	0	West Ham

Illetve a további fordulók eredménye:

28. forduló

Bournemouth	1	4	Chelsea
Aston Villa	0	1	Sheffield United
Brighton	0	1	Crystal Palace
Everton	1	0	Man United
Man City	3	1	Arsenal
Newcastle	0	0	Burnley
Norwich	0	2	Leicester
Tottenham	1	0	Wolves
Watford	1	1	Liverpool
West Ham	1	0	Southampton

29. forduló

Arsenal	1	0	West Ham
Burnley	0	0	Tottenham
Chelsea	0	1	Everton
Crystal Palace	1	1	Watford
Leicester	4	0	Aston Villa
Liverpool	3	1	Bournemouth
Man United	2	2	Man City
Sheffield United	0	0	Norwich
Southampton	0	0	Newcastle
Wolves	1	1	Brighton

30. forduló

Bournemouth	1	1	Crystal Palace
Aston Villa	0	1	Chelsea
Brighton	0	1	Arsenal
Everton	0	3	Liverpool
Man City	5	0	Burnley
Newcastle	1	0	Sheffield United
Norwich	0	0	Southampton
Tottenham	1	0	Man United
Watford	0	2	Leicester
West Ham	3	1	Wolves

31. forduló

Burnley	1	0	Watford
Chelsea	1	2	Man City
Leicester	4	1	Brighton
Liverpool	4	0	Crystal Palace
Man United	4	0	Sheffield United
Newcastle	2	0	Aston Villa
Norwich	1	1	Everton
Southampton	0	0	Arsenal
Tottenham	3	1	West Ham
Wolves	0	1	Bournemouth

32. forduló

Bournemouth	1	1	Newcastle
Arsenal	2	0	Norwich
Aston Villa	1	1	Wolves
Brighton	1	1	Man United
Crystal Palace	1	1	Burnley
Everton	0	3	Leicester
Man City	1	2	Liverpool
Sheffield United	0	0	Tottenham
Watford	0	1	Southampton
West Ham	1	2	Chelsea

33. forduló

Burnley	0	0	Sheffield United
Chelsea	1	1	Watford
Leicester	4	1	Crystal Palace
Liverpool	6	1	Aston Villa
Man United	1	1	Bournemouth
Newcastle	2	0	West Ham
Norwich	1	1	Brighton
Southampton	1	1	Man City
Tottenham	3	1	Everton
Wolves	1	0	Arsenal

34. forduló

Bournemouth	1	1	Tottenham
Arsenal	2	2	Leicester
Aston Villa	0	0	Man United
Brighton	1	4	Liverpool
Crystal Palace	0	4	Chelsea
Everton	1	0	Southampton
Man City	4	1	Newcastle
Sheffield United	1	0	Wolves
Watford	0	0	Norwich
West Ham	1	0	Burnley

35. forduló

Bournemouth	1	4	Leicester
Aston Villa	1	1	Crystal Palace
Brighton	1	1	Man City
Liverpool	5	0	Burnley
Man United	1	0	Southampton
Norwich	1	0	West Ham
Sheffield United	0	1	Chelsea
Tottenham	1	0	Arsenal
Watford	1	0	Newcastle
Wolves	1	0	Everton

36. forduló

Arsenal	1	3	Liverpool
Burnley	0	0	Wolves
Chelsea	1	0	Norwich
Crystal Palace	0	0	Man United
Everton	0	1	Aston Villa
Leicester	3	1	Sheffield United
Man City	5	0	Bournemouth
Newcastle	0	0	Tottenham
Southampton	0	0	Brighton
West Ham	4	0	Watford

37. forduló

Bournemouth	1	1	Southampton
Aston Villa	0	1	Arsenal
Brighton	1	1	Newcastle
Liverpool	1	1	Chelsea
Man United	1	0	West Ham
Norwich	0	1	Burnley
Sheffield United	1	1	Everton
Tottenham	3	3	Leicester
Watford	1	3	Man City
Wolves	0	1	Crystal Palace

38. forduló

Arsenal	3	1	Watford
Burnley	0	1	Brighton
Chelsea	1	1	Wolves
Crystal Palace	1	1	Tottenham
Everton	1	1	Bournemouth
Leicester	2	1	Man United
Man City	4	0	Norwich
Newcastle	0	3	Liverpool
Southampton	1	0	Sheffield United
West Ham	2	1	Aston Villa

A futási kép az alapbeállításokkal:

Az új Élő-pontszámok

Arsenal	1704	Liverpool	1915	Arsenal	1579	Liverpool	1849
Aston Villa	1563	Man City	1837	Aston Villa	1522	Leicester	1772
Bournemouth	1584	Leicester	1778	Bournemouth	1529	Chelsea	1761
Brighton	1588	Tottenham	1761	Brighton	1562	Man City	1759
Burnley	1600	Man United	1725	Burnley	1566	Southampton	1601
Chelsea	1603	Arsenal	1704	Chelsea	1761	Crystal Palace	1592
Crystal Palace	1568	West Ham	1661	Crystal Palace	1592	Wolves	1587
Everton	1625	Wolves	1628	Everton	1582	Tottenham	1585
Leicester	1778	Everton	1625	Leicester	1772	Everton	1582
Liverpool	1915	Newcastle	1625	Liverpool	1849	Arsenal	1579
Man City	1837	Chelsea	1603	Man City	1759	Sheffield United	1578
Man United	1725	Burnley	1600	Man United	1555	Burnley	1566
Newcastle	1625	Sheffield United	1598	Newcastle	1525	Brighton	1562
Norwich	1558	Brighton	1588	Norwich	1461	Man United	1555
Sheffield United	1598	Bournemouth	1584	Sheffield United	1578	Bournemouth	1529
Southampton	1575	Southampton	1575	Southampton	1601	Newcastle	1525
Tottenham	1761	Crystal Palace	1568	Tottenham	1585	Aston Villa	1522
Watford	1564	Watford	1564	Watford	1516	Watford	1516
West Ham	1661	Aston Villa	1563	West Ham	1493	West Ham	1493
Wolves	1628	Norwich	1558	Wolves	1587	Norwich	1461

A bajnokság végén kapott Élő-pontszámok.

A korábban lejátszott mérkőzések végkimeneteleiből és az általam generált eredményekből készült bajnoki tabella (egyben végeredmény):

Helyezés	Csapatok	LM	GY	D	V	LG	KG	GK	Pontszám
1	Liverpool	38	35	3	0	102	22	80	108
2	Man City	38	25	6	7	99	39	60	81
3	Leicester	38	24	7	7	87	37	50	79
4	Chelsea	38	19	8	11	62	45	17	65
5	Tottenham	38	16	13	9	48	43	5	61
6	Man United	38	14	13	11	52	37	15	55
7	Arsenal	38	13	15	10	51	47	4	54
8	Sheffield United	38	12	14	12	33	36	-3	50
9	Wolves	38	11	16	11	44	42	2	49
10	Everton	38	13	9	16	43	55	-12	48
11	Burnley	38	13	9	16	36	52	-16	48
12	Southampton	38	12	10	16	38	53	-15	46
13	Crystal Palace	38	10	15	13	32	49	-17	45
14	Newcastle	38	11	12	15	32	51	-19	45
15	West Ham	38	11	6	21	43	63	-20	39
16	Brighton	38	7	16	15	40	54	-14	37
17	Bournemouth	38	8	12	18	36	60	-24	36
18	Watford	38	6	13	19	30	60	-30	31
19	Aston Villa	38	8	7	23	39	71	-32	31
20	Norwich	38	5	11	22	27	63	-36	26

Illetve még néhány újabb szimuláció után kapott Élő-pontszámok és tabellák:

Arsenal	1727	Liverpool	1894	Arsenal	1605	Liverpool	1804
Aston Villa	1584	Man City	1860	Aston Villa	1539	Man City	1799
Bournemouth	1577	Man United	1767	Bournemouth	1490	Leicester	1796
Brighton	1568	Leicester	1752	Brighton	1560	Chelsea	1784
Burnley	1603	Tottenham	1749	Burnley	1557	Southampton	1625
Chelsea	1594	Arsenal	1727	Chelsea	1784	Man United	1610
Crystal Palace	1554	West Ham	1690	Crystal Palace	1549	Wolves	1610
Everton	1616	Wolves	1639	Everton	1562	Arsenal	1605
Leicester	1752	Everton	1616	Leicester	1796	Tottenham	1602
Liverpool	1894	Burnley	1603	Liverpool	1804	Sheffield United	1572
Man City	1860	Sheffield United	1598	Man City	1799	Everton	1562
Man United	1767	Chelsea	1594	Man United	1610	Brighton	1560
Newcastle	1556	Aston Villa	1584	Newcastle	1535	Burnley	1557
Norwich	1532	Bournemouth	1577	Norwich	1472	Crystal Palace	1549
Sheffield United	1598	Watford	1576	Sheffield United	1572	Aston Villa	1539
Southampton	1563	Brighton	1568	Southampton	1625	West Ham	1538
Tottenham	1749	Southampton	1563	Tottenham	1602	Newcastle	1535
Watford	1576	Newcastle	1556	Watford	1522	Watford	1522
West Ham	1690	Crystal Palace	1554	West Ham	1538	Bournemouth	1490
Wolves	1639	Norwich	1532	Wolves	1610	Norwich	1472

Helyezes	Csapatok	LM	GY	D	V	LG	KG	GK	Pontszam
1.	Liverpool	38	35	1	2	93	29	64	106
2.	Man City	38	28	4	6	103	35	68	88
3.	Leicester	38	24	7	7	86	37	49	79
4.	Chelsea	38	19	7	12	66	49	17	64
5.	Man United	38	17	12	9	62	38	24	63
6.	Tottenham	38	16	12	10	61	45	16	60
7.	Arsenal	38	15	14	9	54	48	6	59
8.	Wolves	38	13	15	10	45	40	5	54
9.	Sheffield United	38	11	16	11	32	37	−5	49
10.	Southampton	38	14	7	17	39	57	−18	49
11.	Burnley	38	13	7	18	39	54	−15	46
12.	West Ham	38	13	6	19	50	65	−15	45
13.	Everton	38	12	8	18	42	56	−14	44
14.	Crystal Palace	38	8	14	16	30	51	−21	38
15.	Brighton	38	8	13	17	38	55	−17	37
16.	Newcastle	38	8	13	17	29	54	−25	37
17.	Aston Villa	38	8	10	20	41	66	−25	34
18.	Bournemouth	38	9	6	23	33	63	−30	33
19.	Watford	38	6	14	18	31	58	−27	32
20.	Norwich	38	5	10	23	25	62	−37	25

Arsenal	1678	Liverpool	1913	Arsenal	1590	Liverpool	1815
Aston Villa	1577	Man City	1822	Aston Villa	1523	Man City	1788
Bournemouth	1545	Tottenham	1741	Bournemouth	1523	Chelsea	1750
Brighton	1578	Man United	1727	Brighton	1557	Leicester	1744
Burnley	1578	Leicester	1720	Burnley	1548	Wolves	1635
Chelsea	1588	Arsenal	1678	Chelsea	1750	Tottenham	1612
Crystal Palace	1589	West Ham	1671	Crystal Palace	1592	Southampton	1608
Everton	1564	Wolves	1641	Everton	1577	Crystal Palace	1592
Leicester	1720	Sheffield United	1626	Leicester	1744	Arsenal	1590
Liverpool	1913	Newcastle	1591	Liverpool	1815	Man United	1580
Man City	1822	Crystal Palace	1589	Man City	1788	Sheffield United	1579
Man United	1727	Chelsea	1588	Man United	1580	Everton	1577
Newcastle	1591	Brighton	1578	Newcastle	1544	Brighton	1557
Norwich	1576	Burnley	1578	Norwich	1462	Burnley	1548
Sheffield United	1626	Aston Villa	1577	Sheffield United	1579	Newcastle	1544
Southampton	1527	Norwich	1576	Southampton	1608	Watford	1536
Tottenham	1741	Watford	1568	Tottenham	1612	West Ham	1534
Watford	1568	Everton	1564	Watford	1536	Aston Villa	1523
West Ham	1671	Bournemouth	1545	West Ham	1534	Bournemouth	1523
Wolves	1641	Southampton	1527	Wolves	1635	Norwich	1462

Helyezes	Csapatok	LM	GY	D	V	LG	KG	GK	Pontszam
1.	Liverpool	38	36	1	1	95	23	72	109
2.	Man City	38	28	4	6	94	35	59	88
3.	Leicester	38	22	9	7	74	34	40	75
4.	Tottenham	38	17	11	10	60	44	16	62
5.	Chelsea	38	17	10	11	60	50	10	61
6.	Man United	38	15	12	11	56	41	15	57
7.	Wolves	38	13	17	8	47	40	7	56
8.	Arsenal	38	12	17	9	48	48	0	53
9.	Sheffield United	38	11	16	11	39	40	−1	49
10.	Crystal Palace	38	11	15	12	32	45	−13	48
11.	Newcastle	38	11	12	15	29	48	−19	45
12.	West Ham	38	11	9	18	47	62	−15	42
13.	Burnley	38	11	8	19	37	51	−14	41
14.	Southampton	38	12	5	21	39	61	−22	41
15.	Everton	38	10	10	18	38	52	−14	40
16.	Brighton	38	8	14	16	37	49	−12	38
17.	Aston Villa	38	9	7	22	39	66	−27	34
18.	Bournemouth	38	8	9	21	31	56	−25	33
19.	Watford	38	6	15	17	31	56	−25	33
20.	Norwich	38	6	11	21	27	59	−32	29

Arsenal	1689	Liverpool	1891	Arsenal	1552	Liverpool	1818
Aston Villa	1558	Man City	1842	Aston Villa	1506	Man City	1798
Bournemouth	1570	Leicester	1773	Bournemouth	1509	Chelsea	1742
Brighton	1573	Tottenham	1758	Brighton	1533	Leicester	1741
Burnley	1611	Man United	1746	Burnley	1579	Southampton	1625
Chelsea	1588	Arsenal	1689	Chelsea	1742	Wolves	1620
Crystal Palace	1579	West Ham	1661	Crystal Palace	1572	Tottenham	1608
Everton	1615	Wolves	1646	Everton	1559	Sheffield United	1594
Leicester	1773	Sheffield United	1625	Leicester	1741	Burnley	1579
Liverpool	1891	Everton	1615	Liverpool	1818	Crystal Palace	1572
Man City	1842	Burnley	1611	Man City	1798	Man United	1563
Man United	1746	Newcastle	1589	Man United	1563	Everton	1559
Newcastle	1589	Chelsea	1588	Newcastle	1542	Arsenal	1552
Norwich	1543	Crystal Palace	1579	Norwich	1451	Newcastle	1542
Sheffield United	1625	Brighton	1573	Sheffield United	1594	Brighton	1533
Southampton	1567	Bournemouth	1570	Southampton	1625	West Ham	1526
Tottenham	1758	Southampton	1567	Tottenham	1608	Watford	1511
Watford	1539	Aston Villa	1558	Watford	1511	Bournemouth	1509
West Ham	1661	Norwich	1543	West Ham	1526	Aston Villa	1506
Wolves	1646	Watford	1539	Wolves	1620	Norwich	1451

Helyezes	Csapatok	LM	GY	D	V	LG	KG	GK	Pontszam
1.	Liverpool	38	35	2	1	93	23	70	107
2.	Man City	38	27	4	7	104	36	68	85
3.	Leicester	38	24	6	8	82	32	50	78
4.	Tottenham	38	17	12	9	61	43	18	63
5.	Chelsea	38	18	8	12	59	47	12	62
6.	Man United	38	17	10	11	52	41	11	61
7.	Wolves	38	13	17	8	45	38	7	56
8.	Sheffield United	38	14	13	11	36	34	2	55
9.	Burnley	38	15	6	17	39	52	−13	51
10.	Arsenal	38	11	16	11	48	52	−4	49
11.	Southampton	38	12	11	15	40	58	−18	47
12.	Crystal Palace	38	11	13	14	30	46	−16	46
13.	Everton	38	12	9	17	40	51	−11	45
14.	Newcastle	38	11	11	16	29	48	−19	44
15.	West Ham	38	10	10	18	46	64	−18	40
16.	Brighton	38	8	11	19	37	53	−16	35
17.	Bournemouth	38	8	10	20	31	59	−28	34
18.	Aston Villa	38	7	8	23	37	66	−29	29
19.	Watford	38	5	14	19	26	55	−29	29
20.	Norwich	38	5	9	24	26	63	−37	24

Irodalomjegyzék

[1] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30:107–117, 1998.

[2] Lars Magnus Hvattum and Halvard Arntzen. Using elo ratings for match result prediction in association football. *International Journal of Forecasting*, 26(3):460 – 470, 2010. Sports Forecasting.

[3] Amy N. Langville and Carl D. Meyer. *Google's PageRank and beyond: the science of search engine rankings*. Princeton University Press, Princeton, NJ, 2012. Paperback edition of the 2006 original.

[4] Gustav Šourek Ondřej Hubáček and Filip Železný. Score-based soccer match outcome modeling - an experimental review. In *MathSport International 2019 Conference*, pages 164–172, Athens, Greece, July 2019.

[5] Kenneth Shum. Notes on pagerank algorithm, March 2013. http://home.ie.cuhk.edu.hk/ wkshum/papers/pagerank.pdf.

[6] W. A. Stein et al. *Sage Mathematics Software (Version 9.0)*. The Sage Development Team, 2020. http://www.sagemath.org.

[7] Wikipédia. Élő-pontrendszer – wikipédia, 2018. `https://hu.wikipedia.org/wiki/%C3%89l%C5%91-pontrendszer`.

[8] Wikipédia. Élő Árpád – wikipédia, 2018. `https://hu.wikipedia.org/wiki/%C3%89l%C5%91_%C3%81rp%C3%A1d`.