欧冠淘汰赛抽签概率的研究

——基于蒙特卡洛模拟的算法

丁杰

二月份将开始欧冠淘汰赛的比赛，正值天朝人民的过年时间，想必广大球迷很是期待。我想，对于欧冠，大家最关心的毫无疑问的是比赛本身。当然，还有一点能够挑动球迷们神经的是淘汰赛的对阵情况。对于欧冠进入淘汰赛的各支球队之间的对阵概率，其实计算起来比较困难。作为一名纯种学渣，我考虑过计算对阵概率的解析解（Analysis Solution）。当然，在计算过程中碰到了我解决不了的问题（嗯，没错，我小学五年级数学是体育老师教的，这锅就让他来背，本泽马可以先休息下），最后不得不放弃这个方法。每当没有办法找到解析解的时候，大家自然而然的一个想法就是使用数值解（Numeric Solution）的方法。本文就是论述如何使用基于蒙特卡洛模拟的算法来得到数值解。

欧冠淘汰赛一共有十六支球队。小组赛中以小组第一身份出线的球队有：曼联（A 组，英超）、巴黎（B 组，法甲）、罗马（C 组，意甲）、巴萨（D 组，西甲）、利物浦（E 组，英超）、曼城（F 组，英超）、贝西克塔斯（G 组，土超）和热刺（H 组，英超）。小组赛中以小组第二身份出线的球队有：巴塞尔（A 组，瑞超）、拜仁（B 组，德甲）、切尔西（C 组，英超）、尤文（D 组，意甲）、塞维利亚（E 组，西甲）、矿工（F 组，乌超）、波尔图（G 组，葡超）和皇马（H 组，西甲）。淘汰赛抽签的规则大致如下：1.小组第一的球队对阵小组第二的球队；2.同个小组的球队之间回避（举例来说，曼联不能和小组赛同组的巴塞尔对阵）；3.同国联赛的球队规避（举例来说，巴萨能与同为西甲的皇马对决）。

我们的目标是计算一个 8×8 的矩阵，矩阵内的元素对应两支球队之间的对阵概率。矩阵的行对应小组第一出线的球队（曼联标记为 1，巴黎标记为 2，罗马标记为 3，巴萨标记为 4，利物浦标记为 5，曼城标记为 6，贝西克塔斯标记为 7，热刺标记为 8），矩阵的列对应小组第二出线的球队（同理，巴塞尔标记为 1，拜仁标记为 2，切尔西标记为 3，尤文标记为 4，塞维利亚标记为 5，矿工标记为 6，波尔图标记为 7，皇马标记为 8）。

具体的抽签流程如下：第一轮，先从小组第一的球队中抽出一支球队，然后从它对应的可能的（要考虑到规则 2 和规则 3 中的回避条款）小组第二对手中抽出一支球队，然后这两支球队成为对阵双方。第二轮，在剩下的球队中再次进行上一步操作。从理论上说，这样进行八轮就可以完成我们的抽签。是不是觉得简单，麻麻再也不用担心我的数学了？会不会觉得会打王者的小学生都会算这个概率？哈哈哈。其实里面还是有很多陷阱的。

让我们来看看这些陷阱。第一，虽然名义上说我们要八轮，但是实际上我们抽七轮就已经确定最后的对阵了。因为七轮过后小组第一和小组第二的球队都只剩下一支，所以肯定是他们之间对决。第二，假设六轮过后，最后剩下的小组第一的球队是巴萨和贝西克塔斯，而小组第二球队是矿工和皇马。实际上可以说，六轮后抽签就已经结束了。为什么？因为，**小组第一**的球队巴萨不能对阵皇马（同国联赛回避），所以他只能对阵矿工，由此可以推出贝西克塔斯对阵皇马。但是问题来了，在这种情况下，六轮后抽签就已经结束了，那么有没有可能五轮后，甚至四轮后抽签就已经结束了，或者至少部分对阵双方提前决定了呢？我们后面会讨论这个问题。第三，我们从另一个侧面来看第二个陷阱的。从**小组第二**球队皇马的角度来看，他不能对阵巴萨（同国联赛回避），所以他只能对阵贝西克塔斯，由此接着可以推出巴萨对阵矿工。

你会不会觉得我无聊，第二个陷阱和第三个陷阱在这个例子中实质是一样的，是不是有种凑字数要稿费的嫌隙。我要洗脱下我的罪名了。的确，在上个例子中，第二个陷阱和第三

个陷阱实质上是一致的，但是在更一般化情形中，两者并不一定一致。我们举个例子。假设前两轮结束后，我们的抽签结果是巴黎对阵皇马，贝西克塔斯对阵拜仁。那么在这种情况下，让我们来关注下小组第二的切尔西的处境。由于同组规避和同国联赛规避的条款存在，所以他在一开始可能的球队就只有巴黎、巴萨和贝西克塔斯，现在前两轮巴黎和贝西克塔斯都已经被抽走，所以他唯一可能的对阵对手是巴萨。这种情形对应我们的第三个陷阱，而并不对应我们的第二个陷阱。

我们还要讨论的一个问题是几轮过后才可能会出现第二个陷阱或者第三个陷阱呢？答案是两轮过后。为什么？根据欧冠规则，同个联赛原则上最多的欧冠名额是 3+1 个。其中三个是直接进入欧冠小组赛，还有一个参加附加赛，如果附加赛获胜就可以参加小组赛（为什么此刻的我想起了阿森纳，哈哈哈）。但是如果该联赛有一支其他的球队上一年获得欧联杯的冠军（去年欧联杯的冠军是英超球队曼联），那么也可以直接获得这一年参加欧冠小组赛的资格。所以从理论上说，某个联赛参加欧冠小组赛的球队最多可以是五支。如果这五支球队都表现优异从小组出线，那么该联赛有五支球队能够参加欧冠淘汰赛。如果其中四支球队小组第一而另一支球队小组第二或者其中四支球队小组第二而另一支球队小组第一，那么那支与众不同的球队不仅需要回避四支同国联赛的球队，还有回避同小组的球队，所以他需要回避的球队有五支，能够选择的球队就只有三支。这种最为极端的例子今年正好碰到了，切尔西就是那只 lucky dog（准确地说是倒霉蛋）。在这种最极端的情形下，就有可能两轮抽签后就决定了切尔西的对阵。

对于以上这个问题，有它的坏的一面，也有它的好的一面。坏的一面是，由于问题的复杂化，我们的代码的长度会变长；而好的一面是，我们考虑的已经是最极端的情形，所以对于别的年份的情况，我们只要改变初始数据集就可以了，而不用修改后面运算部分的任何代码。

下面介绍如何使用蒙特卡洛模拟的算法来算出对阵概率的数值解。第一步，我们先生成一个 8×8 元素全部为 0 的矩阵。第二步，我们根据规则抽出一次对阵结果（即 16 支球队的八场对阵结果）。如果两支球队之间对阵，则矩阵中相应的元素加 1；如果两支球队之间不对阵，则矩阵中相应的元素不变。第三步，我们不断重复第二步操作，直到我们完成 100,000 次蒙特卡罗模拟。第四步，我们将第三步最终得到矩阵中的每个元素的最后累积值除以蒙特卡洛模拟的次数，即 100,000 次，从而得到我们的对阵概率的数值解。

在以上操作中，最为困难的编程部分在于第二步。我大致编程思路如下。前两轮根据规则随便抽（我们上文已经分析，即使在最极端的情形下，前两轮也不会掉进第二个陷阱和第三个陷阱）。从第三轮开始，我们首先观察小组第一的任何一支球队会不会出现只剩下一个可以选择的对手的情形。如果有，直接把他们抽出来；如果没有，那么很好，我们进入下一步。我们的下一步是接着观察小组第二的任何一支球队会不会出现只剩下一个可以选择的对手的情形。如果有，直接把他们抽出来；如果没有，那么很好，我们可以在第三轮同前两轮一样根据规则（即在同组回避和同国联赛规避的条件下）随便抽。第四轮到第七轮的抽签思路同第三轮。在所有的情形下，第八轮的抽签是没有实际意义的，因为对阵结果在七轮以后已经完全决定。因此，我们从前七轮很自然地就推导出第八轮的抽签结果。

我们的概率估计结果如下：

表 1：各球队之间的对阵概率

	巴塞尔	拜仁	切尔西	尤文	塞维利亚	矿工	波尔图	皇马
曼联	0	0.14542	0	0.1834	0.18396	0.15456	0.14771	0.18495
巴黎	0.1082	0	0.29948	0.12748	0.12457	0.10854	0.10371	0.12802
罗马	0.15676	0.15104	0	0	0.19159	0.1609	0.15018	0.18953
巴萨	0.15027	0.14725	0.40256	0	0	0.152	0.14792	0

利物浦	0.16022	0.15287	0	0.19061	0	0.15902	0.15121	0.18607
曼城	0.15558	0.14788	0	0.18201	0.18194	0	0.1478	0.18479
贝西克塔斯	0.10816	0.10395	0.29796	0.12719	0.12839	0.10771	0	0.12664
热刺	0.16081	0.15159	0	0.18931	0.18955	0.15727	0.15147	0

有的同学可能会问的一个问题是，你这个结果靠谱不？我的答案是，妥妥的。由于每次蒙特卡洛模拟之间完美地服从 I.I.D.假设，而且也就只有 64 个参数，进行 100,000 次模拟我都觉得有点费电。但是毕竟 100,000 次的模拟是学术圈的套路，我们就按套路走。但是如果非要我用证据回答，那么我会继续拿出学术圈的套路——置信区间。我进行了 1000 次实验，每次实验进行 100,000 次蒙特卡洛模拟。然后计算 95%水平下的置信区间和 99%水平下置信区间。相应的结果如下：

表 2：概率的置信区间

面板 A：95%水平下的置信区间

	1	2	3	4
1	(0,0)	(14.7186,14.7324)	(0,0)	(18.3448,18.3604)
2	(10.7887,10.8013)	(0,0)	(29.9289,29.9474)	(12.6592,12.6719)
3	(15.8933,15.9076)	(15.1114,15.1259)	(0,0)	(0,0)
4	(15.2141,15.2289)	(14.7079,14.7221)	(40.1113,40.1305)	(0,0)
5	(15.9032,15.9172)	(15.1113,15.1253)	(0,0)	(18.9711,18.9864)
6	(15.4723,15.486)	(14.7133,14.7274)	(0,0)	(18.3502,18.3652)
7	(10.7828,10.7952)	(10.477,10.4888)	(29.932,29.9499)	(12.6565,12.6695)
8	(15.8975,15.9119)	(15.1125,15.1261)	(0,0)	(18.9748,18.9902)
	5	6	7	8
1	(18.3519,18.3674)	(15.4706,15.4851)	(14.7211,14.7348)	(18.3489,18.364)
2	(12.66,12.6734)	(10.7887,10.8015)	(10.471,10.4828)	(12.656,12.6692)
3	(18.9706,18.9848)	(15.9007,15.9151)	(15.1113,15.1248)	(18.9694,18.985)
4	(0,0)	(15.208,15.2223)	(14.7205,14.7344)	(0,0)
5	(0,0)	(15.8975,15.912)	(15.1073,15.1206)	(18.9663,18.9817)
6	(18.3489,18.3642)	(0,0)	(14.7169,14.7308)	(18.3548,18.3701)
7	(12.6598,12.6727)	(10.7846,10.7966)	(0,0)	(12.6608,12.6738)
8	(18.9657,18.9805)	(15.9013,15.9159)	(15.1046,15.119)	(0,0)

面板 B：99%水平下的置信区间

	1	2	3	4
1	(0,0)	(14.7164,14.7345)	(0,0)	(18.3424,18.3628)
2	(10.7867,10.8033)	(0,0)	(29.926,29.9503)	(12.6572,12.6739)
3	(15.8911,15.9099)	(15.1091,15.1282)	(0,0)	(0,0)
4	(15.2118,15.2312)	(14.7057,14.7243)	(40.1082,40.1335)	(0,0)
5	(15.901,15.9194)	(15.1091,15.1275)	(0,0)	(18.9687,18.9888)
6	(15.4701,15.4882)	(14.711,14.7297)	(0,0)	(18.3478,18.3675)
7	(10.7808,10.7971)	(10.4752,10.4907)	(29.9292,29.9527)	(12.6545,12.6715)
8	(15.8952,15.9142)	(15.1104,15.1283)	(0,0)	(18.9724,18.9926)

	5	6	7	8
1	(18.3495,18.3698)	(15.4683,15.4874)	(14.719,14.737)	(18.3465,18.3664)
2	(12.6579,12.6755)	(10.7867,10.8035)	(10.4691,10.4846)	(12.6539,12.6713)
3	(18.9683,18.987)	(15.8985,15.9174)	(15.1092,15.1269)	(18.967,18.9874)
4	(0,0)	(15.2058,15.2245)	(14.7184,14.7366)	(0,0)
5	(0,0)	(15.8952,15.9143)	(15.1053,15.1227)	(18.9639,18.9841)
6	(18.3465,18.3666)	(0,0)	(14.7148,14.7329)	(18.3524,18.3725)
7	(12.6578,12.6748)	(10.7827,10.7985)	(0,0)	(12.6587,12.6758)
8	(18.9634,18.9828)	(15.8991,15.9182)	(15.1023,15.1212)	(0,0)

我们可以看到结果中置信区间都非常的窄，从而证明了我们估计结果的准确性。

还一种办法是看看 1000 次实验的分位点结果。我们关注了两组分位点结果。第一组是 2.5 百分位点和 97.5 百分位点（两者之间包含 95%的数据结果）。第二组是 0.5 百分位点和 99.5 百分位点（两者之间包含 99%的数据结果）。相应的结果如下：

表 3：概率的百分位点

面板 A：2.5 百分位点到 97.5 百分位点

	1	2	3	4
1	(0,0)	(14.518,14.9501)	(0,0)	(18.0989,18.598)
2	(10.589,10.988)	(0,0)	(29.6609,30.234)	(12.473,12.871)
3	(15.676,16.12)	(14.883,15.341)	(0,0)	(0,0)
4	(14.99,15.4471)	(14.492,14.933)	(39.813,40.433)	(0,0)
5	(15.692,16.133)	(14.8939,15.349)	(0,0)	(18.7309,19.218)
6	(15.261,15.703)	(14.497,14.942)	(0,0)	(18.107,18.6061)
7	(10.5989,10.987)	(10.294,10.671)	(29.6648,30.2231)	(12.461,12.864)
8	(15.673,16.1381)	(14.918,15.337)	(0,0)	(18.743,19.2222)
	5	6	7	8
1	(18.116,18.6041)	(15.246,15.697)	(14.516,14.944)	(18.1359,18.5991)
2	(12.468,12.878)	(10.598,10.997)	(10.2909,10.663)	(12.4588,12.868)
3	(18.752,19.201)	(15.682,16.123)	(14.908,15.323)	(18.741,19.2261)
4	(0,0)	(14.979,15.433)	(14.511,14.9462)	(0,0)
5	(0,0)	(15.685,16.138)	(14.9059,15.3271)	(18.726,19.216)
6	(18.1299,18.5992)	(0,0)	(14.4959,14.93)	(18.121,18.6001)
7	(12.4549,12.8591)	(10.6007,10.9791)	(0,0)	(12.4599,12.866)
8	(18.747,19.2131)	(15.6669,16.139)	(14.8949,15.3461)	(0,0)

面板 B：0.5 百分位点到 99.5 百分位点

	1	2	3	4
1	(0,0)	(14.454,15.02)	(0,0)	(18.0139,18.669)
2	(10.537,11.047)	(0,0)	(29.5609,30.308)	(12.414,12.917)
3	(15.576,16.212)	(14.813,15.405)	(0,0)	(0,0)
4	(14.915,15.5231)	(14.4539,15.0161)	(39.732,40.51)	(0,0)
5	(15.6479,16.224)	(14.859,15.4)	(0,0)	(18.663,19.294)
6	(15.187,15.77)	(14.428,14.999)	(0,0)	(18.038,18.668)

7	(10.5339,11.0711)	(10.232,10.717)	(29.5759,30.313)	(12.406,12.92)
8	(15.619,16.228)	(14.842,15.385)	(0,0)	(18.6749,19.287)
	5	6	7	8
1	(18.015,18.6911)	(15.1679,15.783)	(14.467,15.002)	(18.0719,18.6801)
2	(12.3729,12.9261)	(10.522,11.0471)	(10.254,10.744)	(12.3679,12.929)
3	(18.7059,19.305)	(15.613,16.186)	(14.849,15.388)	(18.6739,19.291)
4	(0,0)	(14.924,15.496)	(14.443,15.012)	(0,0)
5	(0,0)	(15.638,16.203)	(14.8399,15.402)	(18.636,19.297)
6	(18.026,18.6521)	(0,0)	(14.4239,14.978)	(18.0489,18.6601)
7	(12.412,12.97)	(10.535,11.0341)	(0,0)	(12.397,12.929)
8	(18.646,19.273)	(15.592,16.195)	(14.814,15.429)	(0,0)

同样我们可以看到，结果区间比较狭窄，再次证明了我们估计的结果的准确性。

另外有个同学在看到我的结论后提出的一个质疑是：你的代码能处理热球吗？哈哈哈，不得不为他的机智点赞。正如新闻中常常看到的某某球队有欧足联干爹，如果欧足联非要用热球作弊来操纵结果，这问题还是交给 FBI 来处理吧。

最终的实际对阵结果为：热刺 v.s.尤文，曼城 v.s.巴塞尔，利物浦 v.s.波尔图，曼联 v.s.塞维利亚，巴黎 v.s.皇马，罗马 v.s.矿工，巴萨 v.s.切尔西，贝西克塔斯 v.s.拜仁。其中最为期待的对阵无疑是巴黎（球员有内少，老板阿联酋酋长有 10,000 亿美刀的资产）v.s.皇马（C 罗和贝尔两亿齐飞，中场 Kroos 和 Modric 真大腿）以及巴萨（前场四人组违约金 17 亿欧，可以说前场四亿妄为，后场特爹稳如狗）v.s.切尔西（虽然科特迪瓦刘德华不在了，但是毕竟还有比利时杨坤在）。从我们模拟的概率中可以看到，巴萨对阵切尔西的概率大约 40%，这种较大的概率事件最后成为了现实。巴黎对阵切尔西的概率也不小，达到 30%。虽然巴黎躲过了切尔西，但是碰到皇马明显是更倒霉的签运。

比赛时间是 26:45（凌晨 2 点 45 分）。考虑到广大读者中存在的年满 18 岁的老（Wei）年（Qiu）人（Mi），我给大家一个录像的网址链接，该网址中没有直接显示最终结果，所以第二天中午爬起来看录像和直播实际效果类似。链接如下：

https://www.zhibo8.cc/zuqiu/luxiang.htm

让我们来看下最新的赔率：曼城 1 赔 4（隐含夺冠概率为 1/4），拜仁 1 赔 5.5（隐含夺冠概率为 1/5.5），巴黎 1 赔 6（隐含夺冠概率为 1/6），巴萨 1 赔 8.5（隐含夺冠概率为 1/8.5），皇马 1 赔 9（隐含夺冠概率为 1/9），曼联 1 赔 15（隐含夺冠概率为 1/15），利物浦 1 赔 15（隐含夺冠概率为 1/15），尤文 1 赔 17（隐含夺冠概率为 1/17），热刺 1 赔 28（隐含夺冠概率为 1/28），切尔西 1 赔 34（隐含夺冠概率为 1/34）。知道我为什么之前把切尔西形容成倒霉蛋了吧。另外还想说一点，这些隐含夺冠概率加起来达到 108.45%。也就是说博彩公司的佣金高达大约 1/12。比起万分之五的交易成本，博彩公司有点太凶残了。从另一个角度来说，足球博彩的参与者的效用函数是明显凸的，除非他们能操纵比赛或者拥有重要的非公开信息。

最后，你应该感谢上帝。宇宙存在了 137 亿年，地球存在了 46 亿年，人类存在了 500 万年，上帝却宠溺地把你与梅西安排在了同一个时代。

厦大金融系伪球迷 丁杰

R 代码：

```
##                          _oo0oo_
##                         o8888888o
##                         88" . "88
##                         (| -_- |)
##                         0\  =  /0
##                       ___/`---'\___
##                     .' \\|     |// '.
##                    / \\|||  :  |||// \
##                   / _|||||| -:- |||||- \
##                  |   | \\\  -  /// |   |
##                  | \_|  ''\---/''  |_/ |
##                  \  .-\__  '-'  ___/-. /
##                ___'. .'  /--.--\  `. .'___
##             ."" '<  `.___\_<|>_/___.' >' "".
##            | | :  `- \`.;`\ _ /`;.`/ - ` : | |
##            \  \ `_.   \_ __\ /__ _/   .-` /  /
##      =====`-.____`.___ \_____/___.-`___.-'=====
##
##      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
##                  佛祖保佑            永无 BUG
```

```
###################################
library(lubridate)
library(foreach)
library(doParallel)

set.seed(1234)
options(max.print=100000)
sink(file="C:/Users/Ding/Desktop/UEFA Champions League.txt",append=T)

###################################
#detectCores(logical = FALSE) # 8
cores <- detectCores(logical = FALSE)
cl <- makeCluster(cores)
registerDoParallel(cl, cores=cores)
clusterEvalQ(cl, library(lubridate))

n_1 <- 100000 # 100000 times Monte Carlos Simulations each experiment
n_2 <- 1000 # 1000 experiments

FUN_1 <- function(a,b) {
```

```
  result[a,1] <<- set_1[b]
  result[a,2] <<- set_2_2[[set_1[b]]]
  set_1 <<- setdiff(set_1,result[a,1])
  for(j in 1:8) {
    set_1_1[[j]] <<- setdiff(set_1_1[[j]],result[a,1])
  }
  set_2 <<- setdiff(set_2,result[a,2])
  for(j in 1:8) {
    set_2_2[[j]] <<- setdiff(set_2_2[[j]],result[a,2])
  }
}

FUN_2 <- function(a,b) {
  result[a,1] <<- set_1_1[[set_2[b]]]
  result[a,2] <<- set_2[b]
  set_1 <<- setdiff(set_1,result[a,1])
  for(j in 1:8) {
    set_1_1[[j]] <<- setdiff(set_1_1[[j]],result[a,1])
  }
  set_2 <<- setdiff(set_2,result[a,2])
  for(j in 1:8) {
    set_2_2[[j]] <<- setdiff(set_2_2[[j]],result[a,2])
  }
}

now()
result_all <- array(NA,dim=c(8,8,n_2))
foreach(x=1:n_2, .combine = rbind) %dopar% {
  result_cum <- matrix(0,nrow=8,ncol=8)

  for (k in 1:n_1) {
    result <- matrix(NA,nrow=8,ncol=2)
    set_1 <- 1:8
    set_2 <- 1:8
    set_1_1 <- list(setdiff(1:8, 1),setdiff(1:8, 2),setdiff(1:8, c(1,3,5,6,8)),setdiff(1:8, c(3,4)),
                    setdiff(1:8, c(4,5)),setdiff(1:8,6),setdiff(1:8,7),setdiff(1:8, c(4,8)))
    set_2_2 <- list(setdiff(1:8, c(1,3)),setdiff(1:8, 2),setdiff(1:8, c(3,4)),setdiff(1:8, c(4,5,8)),
                    setdiff(1:8, c(3,5)),setdiff(1:8, c(3,6)),setdiff(1:8, 7),setdiff(1:8, c(3,8)))
    for (i in 1:2) {
      result[i,1] <- sample(set_1,1)
      result[i,2] <- sample(set_2_2[[result[i,1]]],1)
      set_1 <- setdiff(set_1,result[i,1])
      for(j in 1:8) {
        set_1_1[[j]] <- setdiff(set_1_1[[j]],result[i,1])
```

```
    }
    set_2 <- setdiff(set_2,result[i,2])
    for(j in 1:8) {
      set_2_2[[j]] <- setdiff(set_2_2[[j]],result[i,2])
    }
}

if (length(set_2_2[[set_1[1]]])==1) {
    FUN_1(3,1)
} else if (length(set_2_2[[set_1[2]]])==1) {
    FUN_1(3,2)
} else if (length(set_2_2[[set_1[3]]])==1) {
    FUN_1(3,3)
} else if (length(set_2_2[[set_1[4]]])==1) {
    FUN_1(3,4)
} else if (length(set_2_2[[set_1[5]]])==1) {
    FUN_1(3,5)
} else if (length(set_2_2[[set_1[6]]])==1) {
    FUN_1(3,6)
} else if (length(set_1_1[[set_2[1]]])==1) {
    FUN_2(3,1)
} else if (length(set_1_1[[set_2[2]]])==1) {
    FUN_2(3,2)
} else if (length(set_1_1[[set_2[3]]])==1) {
    FUN_2(3,3)
} else if (length(set_1_1[[set_2[4]]])==1) {
    FUN_2(3,4)
} else if (length(set_1_1[[set_2[5]]])==1) {
    FUN_2(3,5)
} else if (length(set_1_1[[set_2[6]]])==1) {
    FUN_2(3,6)
} else {
    result[3,1] <- sample(set_1,1)
    result[3,2] <- sample(set_2_2[[result[3,1]]],1)
    set_1 <- setdiff(set_1,result[3,1])
    for(j in 1:8) {
      set_1_1[[j]] <- setdiff(set_1_1[[j]],result[3,1])
    }
    set_2 <- setdiff(set_2,result[3,2])
    for(j in 1:8) {
      set_2_2[[j]] <- setdiff(set_2_2[[j]],result[3,2])
    }
}
```

```
if (length(set_2_2[[set_1[1]]])==1) {
   FUN_1(4,1)
} else if (length(set_2_2[[set_1[2]]])==1) {
   FUN_1(4,2)
} else if (length(set_2_2[[set_1[3]]])==1) {
   FUN_1(4,3)
} else if (length(set_2_2[[set_1[4]]])==1) {
   FUN_1(4,4)
} else if (length(set_2_2[[set_1[5]]])==1) {
   FUN_1(4,5)
} else if (length(set_1_1[[set_2[1]]])==1) {
   FUN_2(4,1)
} else if (length(set_1_1[[set_2[2]]])==1) {
   FUN_2(4,2)
} else if (length(set_1_1[[set_2[3]]])==1) {
   FUN_2(4,3)
} else if (length(set_1_1[[set_2[4]]])==1) {
   FUN_2(4,4)
} else if (length(set_1_1[[set_2[5]]])==1) {
   FUN_2(4,5)
} else {
   result[4,1] <- sample(set_1,1)
   result[4,2] <- sample(set_2_2[[result[4,1]]],1)
   set_1 <- setdiff(set_1,result[4,1])
   for(j in 1:8) {
      set_1_1[[j]] <- setdiff(set_1_1[[j]],result[4,1])
   }
   set_2 <- setdiff(set_2,result[4,2])
   for(j in 1:8) {
      set_2_2[[j]] <- setdiff(set_2_2[[j]],result[4,2])
   }
}

if (length(set_2_2[[set_1[1]]])==1) {
   FUN_1(5,1)
} else if (length(set_2_2[[set_1[2]]])==1) {
   FUN_1(5,2)
} else if (length(set_2_2[[set_1[3]]])==1) {
   FUN_1(5,3)
} else if (length(set_2_2[[set_1[4]]])==1) {
   FUN_1(5,4)
} else if (length(set_1_1[[set_2[1]]])==1) {
   FUN_2(5,1)
} else if (length(set_1_1[[set_2[2]]])==1) {
```

```
   FUN_2(5,2)
} else if (length(set_1_1[[set_2[3]]])==1) {
   FUN_2(5,3)
} else if (length(set_1_1[[set_2[4]]])==1) {
   FUN_2(5,4)
} else {
   result[5,1] <- sample(set_1,1)
   result[5,2] <- sample(set_2_2[[result[5,1]]],1)
   set_1 <- setdiff(set_1,result[5,1])
   for(j in 1:8) {
      set_1_1[[j]] <- setdiff(set_1_1[[j]],result[5,1])
   }
   set_2 <- setdiff(set_2,result[5,2])
   for(j in 1:8) {
      set_2_2[[j]] <- setdiff(set_2_2[[j]],result[5,2])
   }
}

if (length(set_2_2[[set_1[1]]])==1) {
   FUN_1(6,1)
} else if (length(set_2_2[[set_1[2]]])==1) {
   FUN_1(6,2)
} else if (length(set_2_2[[set_1[3]]])==1) {
   FUN_1(6,3)
} else if (length(set_1_1[[set_2[1]]])==1) {
   FUN_2(6,1)
} else if (length(set_1_1[[set_2[2]]])==1) {
   FUN_2(6,2)
} else if (length(set_1_1[[set_2[3]]])==1) {
   FUN_2(6,3)
} else {
   result[6,1] <- sample(set_1,1)
   result[6,2] <- sample(set_2_2[[result[6,1]]],1)
   set_1 <- setdiff(set_1,result[6,1])
   for(j in 1:8) {
      set_1_1[[j]] <- setdiff(set_1_1[[j]],result[6,1])
   }
   set_2 <- setdiff(set_2,result[6,2])
   for(j in 1:8) {
      set_2_2[[j]] <- setdiff(set_2_2[[j]],result[6,2])
   }
}

if (length(set_2_2[[set_1[1]]])==1) {
```

```
        FUN_1(7,1)
      } else if (length(set_2_2[[set_1[2]]])==1) {
        FUN_1(7,2)
      } else if (length(set_2_2[[set_1[3]]])==1) {
        FUN_1(7,3)
      } else if (length(set_1_1[[set_2[1]]])==1) {
        FUN_2(7,1)
      } else if (length(set_1_1[[set_2[2]]])==1) {
        FUN_2(7,2)
      } else if (length(set_1_1[[set_2[3]]])==1) {
        FUN_2(7,3)
      } else {
        result[7,1] <- sample(set_1,1)
        result[7,2] <- sample(set_2_2[[result[7,1]]],1)
        set_1 <- setdiff(set_1,result[7,1])
        for(j in 1:8) {
          set_1_1[[j]] <- setdiff(set_1_1[[j]],result[7,1])
        }
        set_2 <- setdiff(set_2,result[7,2])
        for(j in 1:8) {
          set_2_2[[j]] <- setdiff(set_2_2[[j]],result[7,2])
        }
      }

      result[8,1] <- setdiff(1:8,result[1:7,1])
      result[8,2] <- setdiff(1:8,result[1:7,2])
      for (i in 1:8) {
        result_cum[result[i,1],result[i,2]] <- result_cum[result[i,1],result[i,2]]+1
      }
      #print(k)
    }
    #print(x)
    #print(now())
    result_all[,,x] <- result_cum/n_1
}

now()
stopCluster(cl)

###################################
#Note: Firstly, you need to delete useless information in "UEFA Champions League.txt" by hand.
result_matrix <- as.matrix(read.table(file="C:/Users/Ding/Desktop/UEFA Champions League.txt"))

n_1 <- 100000 # 100000 times Monte Carlos Simulations each experiment
```

```
n_2 <- 1000 # 1000 experiments
result_all <- array(NA,dim=c(8,8,n_2))
for (i in 1:n_2) {
   result_all[,,i] <- result_matrix[((i-1)*8+1):(i*8),]
}

result_mean <- matrix(NA,nrow=8,ncol=8)
for (i in 1:8) {
   for (j in 1:8) {
      result_mean[i,j] <- mean(result_all[i,j,])
   }
}

result_sd <- matrix(NA,nrow=8,ncol=8)
for (i in 1:8) {
   for (j in 1:8) {
      result_sd[i,j] <- sd(result_all[i,j,])
   }
}

result_all[,,1]

###################################
#### Confidence Interval

ci_95 <- data.frame(cbind(rep(NA,8),rep(NA,8),rep(NA,8),rep(NA,8),
                          rep(NA,8),rep(NA,8),rep(NA,8),rep(NA,8)))
#ci denotes confidence interval

colnames(ci_95) <- as.character(1:8)

# 95% confidence interval
for (i in 1:8) {
   for (j in 1:8) {
      ci_95[i,j] <- paste0("(",round((result_mean+qnorm(.025)*result_sd/sqrt(n_2))[i,j]*100,4),",",

round((result_mean+qnorm(.975)*result_sd/sqrt(n_2))[i,j]*100,4),")")
   }
}
ci_95 # in percentage

####
ci_99 <- data.frame(cbind(rep(NA,8),rep(NA,8),rep(NA,8),rep(NA,8),
                          rep(NA,8),rep(NA,8),rep(NA,8),rep(NA,8)))
```

```
colnames(ci_99) <- as.character(1:8)

# 99% confidence interval
for (i in 1:8) {
   for (j in 1:8) {
      ci_99[i,j] <- paste0("(",round((result_mean+qnorm(.005)*result_sd/sqrt(n_2))[i,j]*100,4),",",

round((result_mean+qnorm(.995)*result_sd/sqrt(n_2))[i,j]*100,4),")")
   }
}
ci_99 # in percentage

##################################
#### Quantile

# From .025th quantile to .975th quantile
quantile_1 <- data.frame(cbind(rep(NA,8),rep(NA,8),rep(NA,8),rep(NA,8),
                               rep(NA,8),rep(NA,8),rep(NA,8),rep(NA,8)))

colnames(quantile_1) <- as.character(1:8)

for (i in 1:8) {
   for (j in 1:8) {
      quantile_1[i,j] <- paste0("(",round(quantile(result_all[i,j,],.025)*100,4),",",
                                round(quantile(result_all[i,j,],.975)*100,4),")")
   }
}
quantile_1 # in percentage

#### From .005th quantile to .995th quantile
quantile_2 <- data.frame(cbind(rep(NA,8),rep(NA,8),rep(NA,8),rep(NA,8),
                               rep(NA,8),rep(NA,8),rep(NA,8),rep(NA,8)))

colnames(quantile_2) <- as.character(1:8)

for (i in 1:8) {
   for (j in 1:8) {
      quantile_2[i,j] <- paste0("(",round(quantile(result_all[i,j,],.005)*100,4),",",
                                round(quantile(result_all[i,j,],.995)*100,4),")")
   }
}
quantile_2 # in percentage

##################################
```

```
sink()
rm(list=ls())

1/4+1/5.5+1/6+1/8.5+1/9+1/15+1/15+1/17+1/28+1/34

###################################
## It takes 22.56139 hours to run the code above.
## Lasciate ogni speranza, voi ch'entrate!
```

备注：

1) set_1 表示小组第一的剩余可抽的球队；set_2 表示小组第二的剩余可抽的球队。
2) set_1_1 是一个长度为 8 的列表，每一个元素都是一个向量。元素 i 这个向量表示小组第二的第 i 支球队能够对阵的小组第一的球队编号。
3) set_2_2 也是一个长度为 8 的列表，每一个元素都是一个向量。元素 i 这个向量表示小组第一的第 i 支球队能够对阵的小组第二的球队编号。
4) 如果你不怕电脑累，你可以增加 n_1 和 n_2 的数值。必然存在某个正数，只要 n_1 和 n_2 大于该数字，能够满足你任意设定的精确性要求。
5) 关于代码效率问题。R 语言的循环速度的确坑爹。主要应对方法是去掉显式循环、使用并行或者 RCPP。我们主要使用 foreach 函数进行并行计算来实现代码提速。学院提供的 HPC 有八核，所以能较大幅度地提升代码速度。如有兴趣，可以在此代码上继续修改。作者感谢厦大物理系大佬黄晓妍仙女和波大通信工程专业巨头朱天之男神在并行方面的提供的宝贵的建议和意见。