棒球勝率分析

高嘉妤、柯堯珹、趙友誠、吳承恩

Table of contents

請利用連結 https://www.cpbl.com.tw/standings/season 中的資料，

利用 Bradley-Terry model 分析各個球隊的戰績。

python beautifulsoup4 程式碼

```python
# conda install anaconda::beautifulsoup4
# conda install anaconda::requests
# conda install anaconda::pandas
import requests
from bs4 import BeautifulSoup
import pandas as pd
url = "https://www.cpbl.com.tw/standings/season"
response = requests.get(url,headers = {"User-Agent":"Mozilla/5.0"})
response.encoding = 'utf-8'

soup = BeautifulSoup(response.text, 'html.parser') # parse html

# Because the format of table " 球隊對戰戰績" differs from all the others, we handle this seperately.

# find table " 球隊對戰戰績" by locating its upper layer first
caption_div = soup.find('div',{'class': 'record_table_caption'}, string=" 球隊對戰戰績")

# then the table itself
table = caption_div.find_next('div', {'class': 'RecordTable'}).find('table')

# retrieve column names
headers = []
for th in table.find_all('th'):   #extract all column names( th )
    # the structure here differs from the rest of the element( 2 levels )
    if th.find('div', class_='rank'):
        headers.append('排名')
        headers.append('球隊')
    # deal with the second level
    else:
        header = th.get_text(strip=True)
        headers.append(header)

# table.find_all to extract all cell data( tr )
rows = []
```

```
for tr in table.find_all('tr')[1:]:  # skip the first row( column name )
    row = []
    # the structure here is 2-level, too
    sticky = tr.find('td', class_='sticky')
    if sticky:
        rank = sticky.find('div', class_='rank').get_text(strip=True)
        team_name = sticky.find('div', class_='team-w-trophy').get_text(strip=True)
        row.append(rank)
        row.append(team_name)
    # handle the rest of the table
    for td in tr.find_all('td')[1:]:  # skip the first row( column name )
        cell = td.get_text(strip=True)
        row.append(cell)
    rows.append(row)
df = pd.DataFrame(rows, columns=headers)
df.to_csv("".join([" 球隊對戰戰績",'.csv']), index=False, encoding='utf-8-sig')
```

使用 Bradley-Terry model 分析

資料前處理

```
winlose <- data.table::fread(" 球隊對戰戰績.csv")
library(dplyr)
library(tidyr)
library(BradleyTerry2)
teams <- winlose$球隊
matches <- winlose[,8:13]

long_format <- matches %>%
  mutate(球隊 = teams) %>%
  pivot_longer(cols = -球隊, names_to = " 對戰球隊", values_to = " 戰績") %>%
  drop_na()

results <- long_format %>%
  separate(戰績, into = c(" 勝", " 和", " 敗"), sep = "-", convert = TRUE)

win_matrix <- results %>%
  mutate(勝隊 = 球隊, 敗隊 = 對戰球隊, 勝數 = 勝) %>%
  select(勝隊, 敗隊, 勝數) %>%
  pivot_wider(names_from = 敗隊, values_from = 勝數, values_fill = 0)

win_matrix <- as.data.frame(win_matrix)
rownames(win_matrix) <- win_matrix$勝隊
win_matrix <- win_matrix[,-which(names(win_matrix)=='勝隊')]
Head2head <- countsToBinomial(win_matrix)
names(Head2head)[3:4] <- c('Win','Lose')

tie_matrix <- results %>%
  mutate(隊伍 1 = 球隊, 隊伍 2 = 對戰球隊, 和數 = 和) %>%
  select(隊伍 1, 隊伍 2, 和數) %>%
  pivot_wider(names_from = 隊伍 2, values_from = 和數, values_fill = 0)
tie_matrix <- as.data.frame(tie_matrix)
rownames(tie_matrix) <- tie_matrix$隊伍 1
tie_matrix <- tie_matrix[,-which(names(tie_matrix) == '隊伍 1')]
```

Bradley-Terry model

```
library(VGAM)
fit <- vglm(Brat(as.matrix(win_matrix)) ~1, brat(refgp = 1), trace = FALSE, crit = "coef")
fit_ties <- vglm(Brat(as.matrix(win_matrix), as.matrix(tie_matrix)) ~1, bratt(refgp = 1), trace = FALSE,crit

summary(fit)
```

```

Call:
vglm(formula = Brat(as.matrix(win_matrix)) ~ 1, family = brat(refgp = 1), 
    trace = FALSE, crit = "coef")

Coefficients: 
              Estimate Std. Error z value Pr(>|z|)  
(Intercept):1  -0.3458     0.3413  -1.013   0.3109  
(Intercept):2  -0.4869     0.3431  -1.419   0.1558  
(Intercept):3  -0.4869     0.3431  -1.419   0.1558  
(Intercept):4  -0.6550     0.3451  -1.898   0.0577 .
(Intercept):5  -0.7729     0.3475  -2.224   0.0261 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors:  5 

Names of linear predictors: loglink(alpha2), loglink(alpha3), loglink(alpha4), 
loglink(alpha5), loglink(alpha6)

Log-likelihood: -319.6255 on 0 degrees of freedom

Number of Fisher scoring iterations: 4 

No Hauck-Donner effect found in any of the estimates
```

```
summary(fit_ties)
```

```

Call:
vglm(formula = Brat(as.matrix(win_matrix), as.matrix(tie_matrix)) ~ 
    1, family = bratt(refgp = 1), trace = FALSE, crit = "coef")

Coefficients: 
              Estimate Std. Error z value Pr(>|z|)    
(Intercept):1  -0.3470     0.3419  -1.015   0.3101    
(Intercept):2  -0.5170     0.3428  -1.508   0.1314    
(Intercept):3  -0.5170     0.3428  -1.508   0.1314    
(Intercept):4  -0.6880     0.3451  -1.993   0.0462 *  
(Intercept):5  -0.8037     0.3476  -2.312   0.0208 *  
(Intercept):6  -4.2693     0.7454  -5.728 1.02e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors:  6 

Names of linear predictors: loglink(alpha2), loglink(alpha3), loglink(alpha4), 
loglink(alpha5), loglink(alpha6), loglink(alpha0)

Log-likelihood: -130.9393 on 0 degrees of freedom

Number of Fisher scoring iterations: 6 
```

```
Warning: Hauck-Donner effect detected in the following estimate(s):
'(Intercept):6'
```