Poisson Ratings

https://penaltyblog.readthedocs.io/en/latest/ratings/massey_ratings.html (https://penaltyblog.readthedocs.io/en/latest/ratings/massey_ratings.html)

In [27]:
```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

In [28]:
```
import os
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import penaltyblog as pb
```

In [29]:
```
DATA_DIR = os.path.join(os.getcwd(), 'data/')
CHART_DIR = os.path.join(os.getcwd(), 'charts/')
```

In [30]:
```
#data_file = './data/FMF_TA_2021.csv'
data_file = './data/lmf-ac-2021-22.csv'
df = pd.read_csv(data_file, index_col=0)
df.head()
```

Out[30]:

Home \ Away	AMÉ	ATL	ASL	CAZ	GUA	JUÁ	LEÓ	MAZ	MON	NEC	PAC	PUE	QUE	SA
América	—	0–2	2–3	0–0	0–0	3–0	2–0	2–0	0–0	2–1	1–3	2–0	1–1	2–
Atlas	0–1	—	1–0	0–0	1–1	2–0	2–0	1–2	2–1	2–1	0–1	0–1	2–0	2–
Atlético San Luis	0–1	2–6	—	0–0	2–2	0–1	2–0	1–0	1–1	0–2	0–2	2–1	1–1	1–
Cruz Azul	2–1	1–0	0–1	—	0–1	1–0	0–1	0–2	1–1	1–2	1–1	1–3	2–0	1–
Guadalajara	0–0	0–1	1–2	1–1	—	2–2	0–3	3–0	1–3	2–1	1–0	2–3	1–1	1–

In [31]:
```
df.index = df.columns
rows = []
for i in df.index:
    for c in df.columns:
        if i == c: continue
        score = df.loc[i, c]
        if score == '–': continue
        ssplit = score.split('–')
        rows.append([i, c, int(ssplit[0]), int(ssplit[1])])
df = pd.DataFrame(rows, columns = ['HomeTeam', 'AwayTeam','FTHG',
                                   'FTAG'])
df.head()
```

Out[31]:

	HomeTeam	AwayTeam	FTHG	FTAG
0	AMÉ	ATL	0	2
1	AMÉ	ASL	2	3
2	AMÉ	CAZ	0	0
3	AMÉ	GUA	0	0
4	AMÉ	JUÁ	3	0

In [6]:
```
df.home_score = df['FTHG'].astype('int')
df.away_score = df['FTAG'].astype('int')
```

In [32]:
```
df.dtypes
```

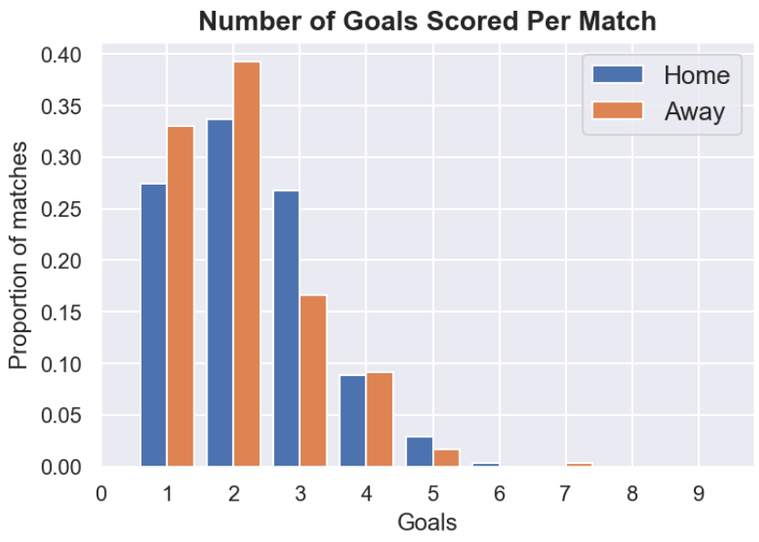
Out[32]:
```
HomeTeam    object
AwayTeam    object
FTHG         int64
FTAG         int64
dtype: object
```

In [8]:
```
df[["FTHG", "FTAG"]].mean()
```

Out[8]:
```
FTHG    1.271242
FTAG    1.084967
dtype: float64
```

In [36]:
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()

max_goals = 10
plt.hist(
    df[["FTHG", "FTAG"]].values, range(max_goals),
                 label=["Home", "Away"], density=True
)
plt.xticks([i - 0.5 for i in range(max_goals)],
           [i for i in range(max_goals)])
plt.xlabel("Goals")
plt.ylabel("Proportion of matches")
plt.legend(loc="upper right", fontsize=13)
plt.title("Number of Goals Scored Per Match", size=14,
          fontweight="bold")
```

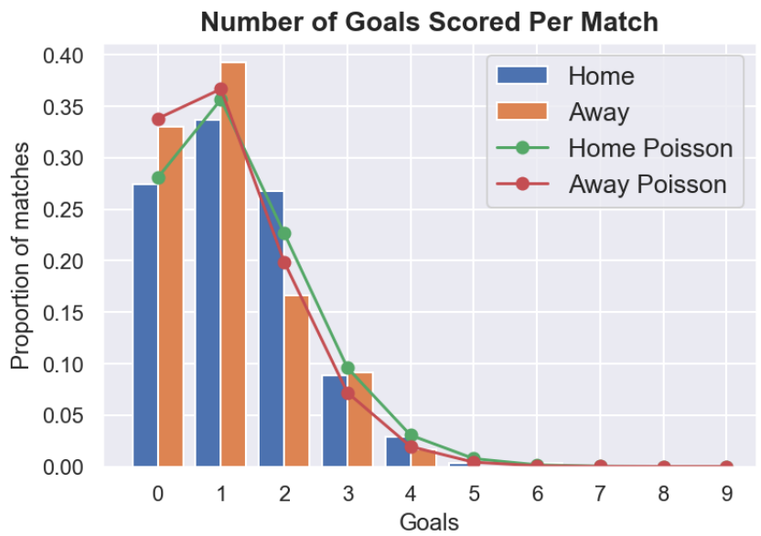
Out[36]: Text(0.5, 1.0, 'Number of Goals Scored Per Match')



In [10]:

```
import numpy as np
from scipy.stats import poisson

home_poisson = poisson.pmf(range(10), df["FTHG"].mean())
away_poisson = poisson.pmf(range(10), df["FTAG"].mean())

max_goals = 10
plt.hist(
    df[["FTHG", "FTAG"]].values, range(max_goals),
                label=["Home", "Away"], density=True
)

plt.plot(
    [i - 0.5 for i in range(1, max_goals + 1)],
    home_poisson,
    linestyle="-",
    marker="o",
    label="Home Poisson",
)

plt.plot(
    [i - 0.5 for i in range(1, max_goals + 1)],
    away_poisson,
    linestyle="-",
    marker="o",
    label="Away Poisson",
)


plt.xticks([i - 0.5 for i in range(1, max_goals + 1)],
           [i for i in range(max_goals)])
plt.xlabel("Goals")
plt.ylabel("Proportion of matches")
plt.legend(loc="upper right", fontsize=13)
plt.title("Number of Goals Scored Per Match", size=14, fontweight="bol
```

Out[10]: Text(0.5, 1.0, 'Number of Goals Scored Per Match')



In [33]:
```
def log_likelihood(
    goals_home_observed,
    goals_away_observed,
    home_attack,
    home_defence,
    away_attack,
    away_defence,
    home_advantage,
):
    goal_expectation_home = np.exp(home_attack +
                                   away_defence + home_advantage)
    goal_expectation_away = np.exp(away_attack + home_defence)

    if goal_expectation_home < 0 or goal_expectation_away < 0:
        return 10000

    home_llk = poisson.pmf(goals_home_observed, goal_expectation_home)
    away_llk = poisson.pmf(goals_away_observed, goal_expectation_away)

    log_llk = np.log(home_llk) + np.log(away_llk)

    return -log_llk
```

In [34]:
```
from scipy.optimize import minimize


def fit_poisson_model():
    teams = np.sort(np.unique(np.concatenate([df["HomeTeam"],
                                              df["AwayTeam"]])))
    n_teams = len(teams)

    params = np.concatenate(
        (
            np.random.uniform(0.5, 1.5, (n_teams)),  # attack strength
            np.random.uniform(0, -1, (n_teams)),  # defence strength
            [0.25],  # home advantage
        )
    )

    def _fit(params, df, teams):
        attack_params = dict(zip(teams, params[:n_teams]))
        defence_params = dict(zip(teams, params[n_teams :
                                                (2 * n_teams)]))
        home_advantage = params[-1]

        llk = list()
        for idx, row in df.iterrows():
            tmp = log_likelihood(
                row["FTHG"],
                row["FTAG"],
                attack_params[row["HomeTeam"]],
                defence_params[row["HomeTeam"]],
                attack_params[row["AwayTeam"]],
                defence_params[row["AwayTeam"]],
                home_advantage,
            )
            llk.append(tmp)

        return np.sum(llk)

    options = {
        "maxiter": 100,
        "disp": False,
    }
```

```
    constraints = [{"type": "eq", "fun": lambda x:
                    sum(x[:n_teams]) - n_teams}]

    res = minimize(
        _fit,
        params,
        args=(df, teams),
        constraints=constraints,
        options=options,
    )

    model_params = dict(
        zip(
            ["attack_" + team for team in teams]
            + ["defence_" + team for team in teams]
            + ["home_adv"],
            res["x"],
        )
    )

    return model_params

model_params = fit_poisson_model()
```

In [13]:
```
from pprint import pprint

pprint(model_params)
```

```
{'attack_AMÉ': 1.1157517663080405,
 'attack_ASL': 1.0240686854858334,
 'attack_ATL': 1.0437058723135053,
 'attack_CAZ': 1.0326076523690522,
 'attack_GUA': 0.9564234025677263,
 'attack_JUÁ': 0.5233965966782209,
 'attack_LEÓ': 0.9325857018722608,
 'attack_MAZ': 0.977015992538689,
 'attack_MON': 1.0064041723741655,
 'attack_NEC': 0.942826312219134,
 'attack_PAC': 1.2144175242981365,
 'attack_PUE': 1.034074534793523,
 'attack_QUE': 0.6941494716223073,
 'attack_SAN': 1.2011716000067907,
 'attack_TIJ': 0.7472095511456406,
 'attack_TOL': 1.1163311488819572,
 'attack_UNL': 1.3647959307101474,
 'attack_UNM': 1.0730640838148688,
 'defence_AMÉ': -1.3241171754067405,
 'defence_ASL': -0.8190132857496649,
 'defence_ATL': -1.4056199682372583,
 'defence_CAZ': -1.098804169356003,
 'defence_GUA': -1.103175929522568,
 'defence_JUÁ': -0.6784506355121515,
 'defence_LEÓ': -1.0473403657705984,
 'defence_MAZ': -0.7572025086818273,
 'defence_MON': -1.1301947159174432,
 'defence_NEC': -0.8690752810728818,
 'defence_PAC': -1.0296582006650643,
 'defence_PUE': -1.0697286367677188,
 'defence_QUE': -0.9532952008820649,
 'defence_SAN': -0.9005343207327874,
 'defence_TIJ': -0.6696275415366564,
 'defence_TOL': -0.5594789930655382,
 'defence_UNL': -1.0459738441179975,
 'defence_UNM': -0.816033013787446,
 'home_adv': 0.15844294647628168}
```

In [14]:
```
def predict(home_team, away_team, params, max_goals=10):
    home_attack = params["attack_" + home_team]
    home_defence = params["defence_" + home_team]
    away_attack = params["attack_" + away_team]
    away_defence = params["defence_" + away_team]
    home_advantage = params["home_adv"]

    home_goal_expectation = np.exp(home_attack + away_defence
                                   + home_advantage)
    away_goal_expectation = np.exp(away_attack + home_defence)

    home_probs = poisson.pmf(list(range(max_goals + 1)),
                             home_goal_expectation)
    away_probs = poisson.pmf(range(max_goals + 1),
                             away_goal_expectation)

    probability_matrix = np.outer(home_probs, away_probs)

    return probability_matrix
```

In [15]:
```
EL = 'CAZ'
EV = 'UNL'
```

In [16]:
```
probs = predict(EL, EV, model_params, 4)
pprint(probs)
```

```
array([[0.08536214, 0.11137406, 0.07265622, 0.03159878, 0.01030692],
       [0.09868958, 0.1287627 , 0.08399991, 0.03653224, 0.01191613],
       [0.05704891, 0.0744331 , 0.04855734, 0.02111798, 0.00688829],
       [0.02198529, 0.02868474, 0.01871284, 0.00813836, 0.00265458],
       [0.00635445, 0.00829081, 0.00540861, 0.00235225, 0.00076726]])
```

In [17]:
```
# draw
print(f'Probabilidad que empaten {EL} vs {EV} es =')
np.sum(np.diag(probs))
```

```
Probabilidad que empaten CAZ vs UNL es =
```

Out[17]:
```
0.2715878017217973
```

In [18]:
```
# home win
print(f'Probabilidad que gane {EL} vs {EV} es =')
np.sum(np.tril(probs, -1))
```

```
Probabilidad que gane CAZ vs UNL es =
```

Out[18]:
```
0.3219605825772931
```

In [19]:
```
# away win
print(f'Probabilidad que pierda {EL} vs {EV} es =')
np.sum(np.triu(probs, 1))
```

```
Probabilidad que pierda CAZ vs UNL es =
```

Out[19]:
```
0.3890451086760247
```

In [20]:
```
probs = predict(EV, EL, model_params, 4)
pprint(probs)
```

```
array([[0.08082663, 0.07975348, 0.03934729, 0.01294162, 0.00319245],
       [0.12356178, 0.12192121, 0.06015122, 0.01978419, 0.00488038],
       [0.09444605, 0.09319207, 0.04597737, 0.0151223 , 0.00373038],
       [0.04812738, 0.04748838, 0.02342893, 0.00770595, 0.00190091],
       [0.01839339, 0.01814918, 0.0089541 , 0.00294507, 0.00072649]])
```

In [21]:
```
# draw
print(f'Probabilidad que empaten {EV} vs {EL} es =')
np.sum(np.diag(probs))
```

```
Probabilidad que empaten UNL vs CAZ es =
```

Out[21]:
```
0.2571576597894537
```

In [22]:
```
# home win
print(f'Probabilidad que gane {EV} vs {EL} es =')
np.sum(np.tril(probs, -1))
```

```
Probabilidad que gane UNL vs CAZ es =
```

Out[22]:
```
0.4786863289687837
```

In [23]:
```
# away win
print(f'Probabilidad que pierda {EV} vs {EL} es =')
np.sum(np.triu(probs, 1))
```

```
Probabilidad que pierda UNL vs CAZ es =
```

Out[23]:
```
0.2408042117069925
```

In [ ]:

In [ ]: