Time of first goal analysis

Enea Skendomemaj

Introduction

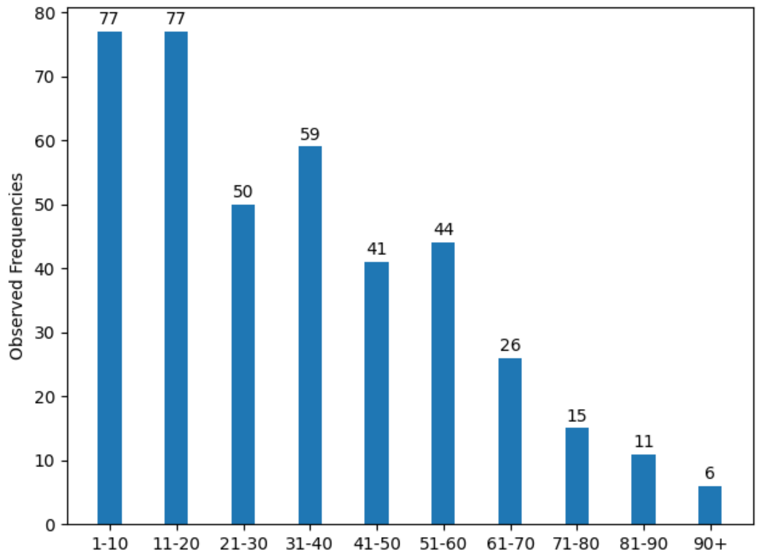
The purpose of this script is to study the time of first goal in the Greek Superleague. This can be seen as a time-to-event analysis allowing the use of some well established tools from survival analysis. Data has been collected from the 2019/2020 and 2020/2021 seasons up to May 2nd, 2021. Older seasons were not considered as the tournament format has changed. This script will be under frequent revision as the 2020/2021 season progresses.

1 Methodology

Data was collected from a popular sports app and stored in a *csv* file. To avoid any issues with halftime stoppage time the **total time** is considered. The total time is defined as the time elapsed from the beginning of the match until either a goal is scored or the match ends, including any stoppage time. For example, if the first goal was scored in the 62nd minute and there were 2 minutes of halftime stoppage time then the total time recorded is 64 minutes. If a match ends in a goalless draw and there were 5 minutes of total stoppage time then the total time recorded is 95minutes. Data analysis was done in *Python* using the *scipy* [1], *pandas* [2], *numpy* [3], *matplotlib* [4] and *lifelines* [5] libraries.

1.1 A first look at the data

Currently there is a total of 459 matches in the dataset. In this dataset the issue of censored observations is prevalent. First, the data is *right censored* since matches can end in a goalless draw. Second, it is also *interval censored*. When a goal is said to have been scored in the 62nd minute it is essentially meant that the goal was scored sometime in the $[61, 62)$ interval but the exact time is unknown. A *matplotlib* histogram of the observed first goal times can help make things clearer and possibly reveal a recognizable pattern. The horizontal axis represents the total time intervals. Again, total time includes any additional stoppage time if the goal was scored in the second half.



It is clear that there is a downward trend as the match progresses which is to be expected. First goals are scored in the first half more often than not. Although the observed goals have a multimodal distribution, an exponential distribution will be used to model the first goal time.

1.2 Estimation

1.2.1 Maximum likelihood estimation

Even though the data is incomplete it is still possible to produce an estimate through the maximum likelihood (ML) method [6]. The survival function of the exponential distribution is $S(t) = P(T > t) = e^{-\mu t}$ where μ is the rate parameter. In the context of this script, $S(t)$ is the probability that the match will be goalless by the beginning of the t-th minute. Thus, the probability of a goal being scored in the t-th minute is $S(t-1) - S(t) = e^{-\mu(t-1)}(1 - e^{-\mu})$. Assuming that each match is independent of any other one (a fairly realistic assumption) the likelihood function is given by

$$L = \prod_{i \in G} [S(t_i - 1) - S(t_i)] \prod_{i \in NG} S(t_i) = e^{-\mu \sum_{i=1}^{n} t_i + \mu n_1} (1 - e^{-\mu})^{n_1}$$

where: - G is the set of matches containing a goal, any t_i belonging to this set is a first goal time.
- NG is the set of goalless matches, any t_i belonging to this set is the total duration of the match.
- n_1 is the cardinality of G, the number of games containing a goal.

The log-likelihood is simply the logarithm of L:

$$\mathcal{L} = lnL = -\mu \sum_{i=1}^{n} t_i + \mu n_1 + n_1 ln(1 - e^{-\mu})$$

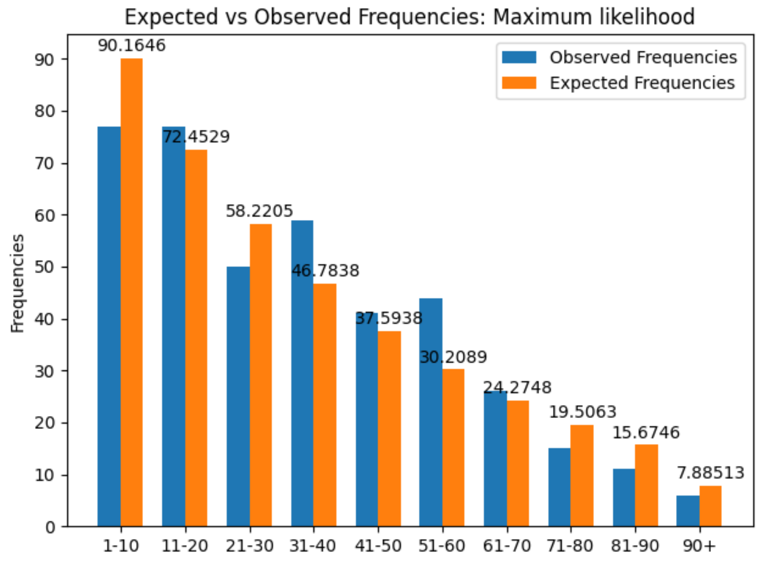
Setting the first derivative equal to zero gives a closed form solution for μ:

$$\hat{\mu}_{ML} = ln\left(\frac{\sum\limits_{i=1}^{n} t_i}{\sum\limits_{i=1}^{n} t_i - n_1}\right)$$

Nevertheless, using the *minimize* function from *scipy.optimize* on $-\mathcal{L}$ yields $\hat{\mu}_{ML} = 0.02186997985839844$. With an estimate for μ available it is now possible to estimate the expected frequencies so a chi-squared goodness of fit test can be used. The formula of expected frequencies is

$$E_i = n[S(i) - S(i+10)] = ne^{-\mu i}(1 - e^{-10\mu})$$

for $i \in \{0, 10, 20, ..., 80\}$. The last bin will be $[90, 96]$ since currently no first goal has been scored after the 96th minute. Here is a histogram comparing the observed and expected frequencies. Each E_i is estimated using $\hat{\mu}$.



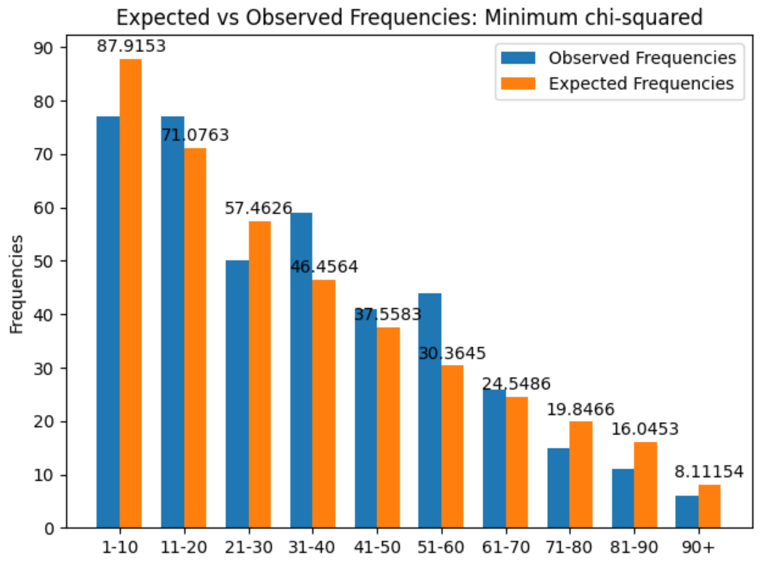
The chi-squared test loses a degree of freedom ($df = 8$) since μ was estimated. Using the *chisquare* function from *scipy.stats* yields a p-value of 0.03999657941575173.

1.2.2 Minimum chi-squared estimation

The chi-squared goodness of fit statistic is defined as:

$$X^2 = \sum_i \frac{(O_i - E_i)^2}{E_i}$$

where O_i denotes the observed frequencies and E_i is the same as before. The idea here is to minimize X^2 to obtain the $\hat{\mu}$ that best fits the observed data [6]. This is a very complicated expression and no closed form solution exists, solution through numerical means is imperative. Using the *minimize* function from *scipy.optimize* on X^2 yields $\hat{\mu} = 0.02126199555397034$. Comparing histograms again it can be seen that this method gives a slightly better fit.



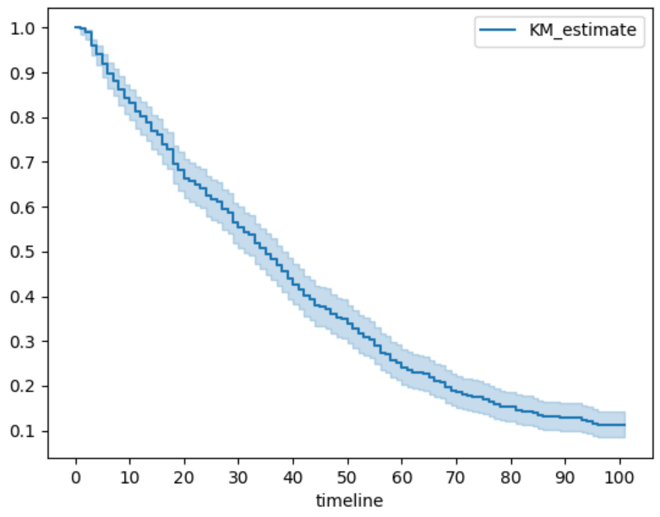
The difference however is not great, using the *chisquare* function from *scipy.stats* the p-value is calculated as 0.04168461612719509. Only marginally higher than before. Note that the minimum X^2 method is dependent on how the data is binned, in this script the bins are created in such way that ensures all expected frequencies are greater than 5 thus allowing the use of the chi-squared test.

1.2.3 Kaplan-Meier estimation

Also known as the **Product Limit** estimator, this method is non parametric. It is defined as

$$\hat{S}(t) = \prod_{i:t_i \leq t} \left(1 - \frac{d_i}{n_i}\right)$$

where t_i is a time when at least a goal was scored, d_i is the number of goals scored at t_i and n_i is the number of goalless matches up to time t_i. Using *KaplanMeierFitter* from *lifelines* a Kaplan-Meier (KM) curve can be obtained:



The lighter blue lines encapsulate the 95% confidence interval. The KM curve seems to decay exponentially further strengthening the belief of an underlying exponential distribution. The median time of first goal is 35 minutes.

2 Results

Although the p-values were marginally significant it can be seen that the time of first goal can loosely be modeled by the exponential distribution using survival analysis tools. Both ML and min X^2 methods seem to produce decent estimates with ML being more preferable because of its simplicity, existence of a closed form solution and well known asymptotic properties.

References

[1] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: `10.1038/s41592-019-0686-2`.

[2] Wes McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: `10.25080/Majora-92bf1922-00a`.

[3] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585 (2020), pp. 357–362. DOI: `10.1038/s41586-020-2649-2`.

[4] John D. Hunter. “Matplotlib: A 2D Graphics Environment”. In: *Computing in Science Engineering* 9.3 (2007), pp. 90–95. DOI: `10.1109/MCSE.2007.55`.

[5] Cameron Davidson-Pilon. “lifelines: survival analysis in Python”. In: *Journal of Open Source Software* 4.40 (2019), p. 1317.

[6] Dick London. *Survival models and their estimation.* ACTEX publications, 1997.