KOREA UNIVERSITY

BAYESIAN RATING MODELS

STARCRAFT II

Introduction to Bayesian Statistics, Korea University

Adam Frederik Ingwersen Linnemann, 2018952670

28. 12. 2018

Abstract

This report investigates the power of bayesian rating models in predicting professional Starcraft II league matches. Prediction on outcomes for the 2017 Korean championships are being considered; a variety of distributional relations are proposed, but it is found that the modelling of attacking and defensive parameters are suitable - and should follow a normal distribution. The proposed model rightly predicts the winner of the 2017 championships - but only some of the runner-ups are predicted correctly. More complicated models are proposed to alleviate this drawback of the base model.

Contents

	match_date	player_1	player_1_match_status	score	player_2	player_2_match_status	player_1_race	player_2_race	addon	tournament_type
0	09/19/2016	MC	[loser]	0–2	Stats	[winner]	P	P	LotV	online
1	09/19/2016	MC	[winner]	2–1	NaTuRal	[loser]	P	T	LotV	online
2	09/19/2016	MC	[loser]	1–2	Dark	[winner]	P	Z	LotV	online
3	09/13/2016	MC	[loser]	0–2	INnoVation	[winner]	P	T	LotV	online

Figure 1: Raw data

1 Scope and Purpose

The aim of this report is to investigate the performance and interpretability of bayesian rating models, as proposed by Baio & Blangiardo (2009), on Starcraft II match data.

2 Data

Data on Starcraft II matches from 2014 to 2017 is obtained from Kaggle. The data contains 187.397 match results played between 1.856 distinct players (professional and non-professional).

In order to achieve a sensible amount of data for MCMC simulation on a laptop, the data has to be subsetted. This is done by looking at match-dates after *2016-01-01*. Furthermore, the number of players are limited to 8, by grabbing the top 8 players in the **2017 Starcraft II World Championship** - where the top 8 constitutes the finalists of that tournament.

```
xpath = "//tr/td[@align = 'left']/a/text()"
url = "https://liquipedia.net/starcraft2/2017_StarCraft_II_World_Championship_Series_Korea/S
page = requests.get(url)
tree = html.fromstring(page.content)
top  = tree.xpath(xpath)
top8 = top[:8]
```

Further data preprocessing steps are documented in the attached `report.ipynb`.
The players' defensive and offensive performance is represented in the figure below for the two considered seasons.

In addition, the proportionality should ideally be somewhat equivalent between players. It's apparent that **Rogue** and **TY** are slightly underrepresented in the data.

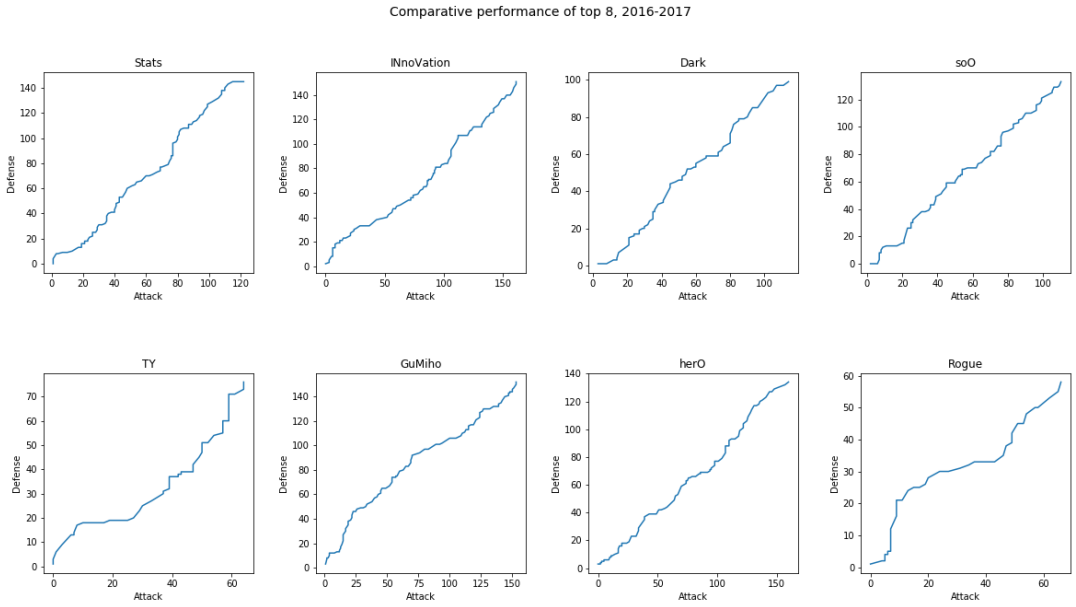

Figure 2: Relative player def/off performace

Table 1: Player proportion of total dataset

players	proportion
Stats	0.1497731
INnoVation	0.1527988
Dark	0.1180030
soO	0.1240545
TY	0.0816944
GuMiho	0.1543116
herO	0.1512859
Rogue	0.0680787

3 Modelling

In attempting to model the relations present in a Starcraft II tournament and capture the inhrerent dynamics that apply to such competitions, three distinct models are proposed. Each of which are to be explained in sections below. All models rely on having a set of proxy-distributions that represent the offensive and defensive attributes of any particular player. The model statistics and related plots are presented in the attached `ipython-notebook`.

3.1 Baseline Model

This model serves as the basline to which other models are evaluated. It's a stripped down version of the model presented by *Baio & Blangiardo (2010)*, that takes a fixed zero-mean in offensive/defensive attributes, and only the parameter variance is modeled explicitly.

3.1.1 Specification

```
tau_att    = pm.Gamma('tau_att', 0.1, 0.1)
tau_def    = pm.Gamma('tau_def', 0.1, 0.1)
atts_star = pm.Normal("atts_star", mu = 0, sd = tau_att, shape = n_players)
defs_star = pm.Normal("defs_star", mu = 0, sd = tau_def, shape = n_players)
atts = pm.Deterministic('atts', atts_star - tt.mean(atts_star))
defs = pm.Deterministic('defs', defs_star - tt.mean(defs_star))
p1_theta = tt.exp(atts[player_1] + defs[player_2])
p2_theta = tt.exp(atts[player_2] + defs[player_1])
p1_points = pm.Poisson('p1_points', mu = p1_theta, observed = observed_player1_score)
p2_points = pm.Poisson('p2_points', mu = p2_theta, observed = observed_player2_score)
```

3.1.2 Results

With NUTS-sampling, the model converges quickly and achieves excellent **Rubin-Gelman** statistics and **BFMI (Bayesian Fraction of Missing Information)**, which indicates that the model both converged and is well-specified, respectively.

3.2 Intercept Model

This model is a re-write of the model presented in *Baio & Blangiardo (2010)*, with a tunable intercept to indicate a home-court advantage to one of the teams. This intuitively should not provide any additional information in a Starcraft II context.

3.2.1 Specification

```
tau_att    = pm.Gamma('tau_att', 0.1, 0.1)
tau_def    = pm.Gamma('tau_def', 0.1, 0.1)
intercept  = pm.Flat('intercept')
atts_star = pm.Normal("atts_star", mu = 0, sd = tau_att, shape = n_players)
defs_star = pm.Normal("defs_star", mu = 0, sd = tau_def, shape = n_players)
atts = pm.Deterministic('atts', atts_star - tt.mean(atts_star))
defs = pm.Deterministic('defs', defs_star - tt.mean(defs_star))
p1_theta = tt.exp(intercept + atts[player_1] + defs[player_2])
p2_theta = tt.exp(intercept + atts[player_2] + defs[player_1])
p1_points = pm.Poisson('p1_points', mu = p1_theta, observed = observed_player1_score)
p2_points = pm.Poisson('p2_points', mu = p2_theta, observed = observed_player2_score)
```

3.2.2 Results

This specification has been run several times under the exact same conditions as the baseline model, but does not appear to converge - and may be misspecified as well. This suggests, that the inclusion of the intercept as a trainable parameter only makes MCMC more complicated, and does not provide any additional information. Thus, the homecourt-advantage is not present in the observed Starcraft II mathces, and perheaps in Starcraft II in general.

3.3 Tunable Mean Model wo. Intercept

This model is, again, inspired by the one suggested by *Baio & Blangiardo (2010)*, but disregards the intercept, and uses a trainable mean for both the offensive and defensive stats of the players. This seems to be a more sensible approach in the Starcraft II context.

3.3.1 Specification

```
mu_att     = pm.Normal('mu_att', 0)
mu_def     = pm.Normal('mu_def', 0)
tau_att    = pm.Gamma('tau_att', 0.1, 0.1)
```

```
tau_def    = pm.Gamma('tau_def', 0.1, 0.1)
atts_star = pm.Normal("atts_star", mu = mu_att, sd = tau_att, shape = n_players)
defs_star = pm.Normal("defs_star", mu = mu_def, sd = tau_def, shape = n_players)
atts = pm.Deterministic('atts', atts_star - tt.mean(atts_star))
defs = pm.Deterministic('defs', defs_star - tt.mean(defs_star))
p1_theta = tt.exp(atts[player_1] + defs[player_2])
p2_theta = tt.exp(atts[player_2] + defs[player_1])
p1_points = pm.Poisson('p1_points', mu = p1_theta, observed = observed_player1_score)
p2_points = pm.Poisson('p2_points', mu = p2_theta, observed = observed_player2_score)
```

3.3.2 Results

This model performs rather similar to the baseline model, but does yield a lot of divergences in the sampling process, and takes longer to converge - thus it should be discarded in favor of the baseline model.

4 Discussion of major findings

4.1 Picking the best model

4.2 Starcraft v. Soccer

5 Implications

6 Concluding Remarks