POMMS AND SST MATRIX

LAPO SANTI

Contents

1. Posets and Dags

Definition 1. *[Poset]To define poset, a partially ordered set we start from D, a set of elements. The binary relation $\prec$ on D is said to be a partial order if:*

(1) *For any $x \in D, x \prec x$(reflexivity)*

(2) *For any $x, y, z \in D, x \prec y$ and $y \prec z \implies x \prec z$(transitivity)*

(3) *For any $x, y \in D, x \prec y$ and $y \prec x \implies x = y$(antisymmetry).*

Then we call $(D, \prec)$ a partially ordered set, or a poset.

A *finite poset* $(D, \prec)$ is a poset where D has a finite number of distinct elements.

Example: let D be the finite set defined by representing the $M \times N$ array of probabilities. Let (u, v) and (q,r) be any two elements of D and define the binary relation on D by

$$(q, r) \prec (u, v) \iff q \prec u \text{ and } r \prec v \tag{4}$$

There exists a correspondence between posets and directed cyclic graphs.

Let (D, F) be a directed acyclic graph, where $D = \{y_1, \ldots, y_n\}$, a finite set. To construct a poset to which this digraph corresponds, we define the binary relation $\prec$ on D by

(5) $y_i \prec y_i$ for $i = 1 \ldots n$

(6) $y_i \prec y_j$ if there exists a directed path from y_i to $y_j \in (D, F)$

We saw above that the correspondence is many-to-one. Given a finite poset, one may construct a class of directed acyclic graphs; the correspondence described above is in a sense the minimal directed acyclic graph since it has the smallest possible directed edge set Pomms definitions

Definition 2 (Cone)**.** *For any $y \in D$, the cone of y is the set*

$$cone\ y = \{x \in D : x \prec y; x \neq y\}$$

Definition 3 (Adjl)**.** *For any $y \in D$ the adjacent lower neighbourhood of y is the set*

$$adjl\ y = \{x \in D : (x, y) is\ a\ directed\ edge\ in (D, F)\}$$

Definition 4 (Dilation)**.** *For any $y \in D$, the dilation of y is the set*

$$dil\ y = \bigcup \{\overline{adjl}\ x : y \in \overline{adjl}\ \}$$

Definition 5 (Excluded dilation)**.** *For any $y \in D$, the excluded dilation of y is the set*

$$dil\ ^* y = dil\ y \setminus \{y\} \tag{7}$$

Definition 6 (Minimal element)**.** *In general, an element $y \in D$ is called minimal element if there is no other element x satisfying $x \prec y$ where adjl s is the set of adjacent lower neighbors of $s \in D$.*

Definition 7 (Cover of a Subset)**.** *The cover of a subset B is a set of all elements x in D such that x is adjacent to an element in B and x is not in B. Formally, the cover of B is defined as follows:*

$$covr\ B = x \in D : adjl\ x \subset B \text{ and } x \notin B$$

where adjl x is the set of all adjacent elements of x in D.

Intuitively, the cover of a subset B represents all the elements in D that are outside of B but are adjacent to at least one element in B. In other words, the cover of B captures the neighborhood of B in D.

Definition 8 (Level Sets)**.** *The level sets of a poset D are a sequence of nonempty cover sets defined recursively as follows:*

$$L^0 = D_{min}; \quad L^i = covr\left(\bigcup_{k=0}^{i-1} L^k\right)$$

where D_{min} is the set of all minimal elements in D.

The first level set L^0 is simply the set of all minimal elements in D. The subsequent level sets are defined by taking the union of all the previous level sets and taking the cover of this union. Intuitively, each level set captures the neighborhood of the previous level sets in D.

1.1. **Partially ordered Markov models:** Consider a finite set of random variables $\{Z(s_1), \ldots, Z(s_n)\}$ indexed by location or "points"

$$D = \{s_1, \ldots, s_n\} : n \in \{1, 2, \ldots\}$$

That is, we assume the existence of a directed acyclic graph (D, F) and its corresponding poset $(D, \prec)$. Let (D, F) be a finite, directed acyclic graph and its corresponding poset $(D, \prec)$. Consider $s \in D$ and recall the definition of cone s. Also, let the quantity U_s denote any subsets of points not related to s. Formally:

$$U_s \subset \{u \in D : u \text{ and } s \text{ are not related } \}$$

Definition 9 (POMM)**.** *Then $\{Z(s) : s \in D\}$ is said to be a partially ordered Markov model (POMM) if , for all $s \in D$ and any U_s*

$$P(Z(s)|Z(\ cone\ s), Z(U_s)) = P(Z(s)|Z(\ adjl\ s) \tag{8}$$

Proposition 1. *[Joint Distribution] Let (D, F) be a directed acyclic graph with no singleton points and let $(D, \prec)$, be its associated poset. Suppose that $\{Z(s) : s \in D\}$ is a POMM. Then*

$$P(Z(D)) = P(Z(L^0)) \prod_{k=1}^{m} \prod \{P(Z(u))|Z(\ adjl\ u) : u \in L^k\} \tag{9}$$

$$= P(Z(L^0)) \prod \{P(Z(u))|Z(\ adjl\ u) : u \in D \setminus L^0\} \tag{10}$$

where $L^0, L^1, \ldots L^m$ are the level sets as defined previously.

Result 1 relates the probability of a random variable defined on a poset to the probabilities of its restrictions to the lower level sets of the poset.

The result states that the probability of Z on the entire poset D can be expressed as a product of the probabilities of Z restricted to the level sets $L^0, L^1, \ldots, L^m$ of the poset, where $L^0 = D_{\min}$ is the set of minimal elements of D, and L^k is the set of elements of D that are not in any of the previous level sets $L^0, L^1, \ldots, L^{k-1}$ and whose immediate predecessors are all in the union of the previous level sets $\bigcup_{i=0}^{k-1} L^i$.

The first part of the result states that the probability of Z on D is equal to the product of the probability of Z on L^0 and the conditional probabilities of Z on the elements of each subsequent level set L^k, given the values of Z on their immediate predecessors. This can be seen as a form of the chain rule of probability, where the joint probability of Z on D is decomposed into a product of conditional probabilities.

The second part of the result simplifies the product by noting that the conditional probabilities of Z on the elements of $D \setminus L^0$ are determined by the values of Z on their immediate predecessors, which are all in L^0 or $D \setminus L^0$. Therefore, the product can be simplified to the product of the probability of Z on L^0 and the conditional probabilities of Z on the elements of $D \setminus L^0$ given the values of Z on their immediate predecessors in $D \setminus L^0$. This simplification reduces the number of terms in the product and makes the computation of the joint probability of Z on D more efficient.

2. Application to the SST matrix

In section, I first start by describing a matrix exhibiting the SST property. Then, I make the connection between the POMMs and see how to fit those into context. Bear in mind that the final goal is to estimate the probability of victory between players in a tournament. These probabilities should exhibit a strong stochastic transitivity (SST), which is a monotonicity property constraining the probability. Practically speaking, we would like to represent the fact that if Djoković is stronger than Medvedev and Medvedev is stronger than Kyrgios, then Djoković must be stronger than Kyrgios. The main content of this section will be, starting from this transitivity property, to define also the reflexivity and antisymmetry ones, to build a Poset from the matrix P.

The matrix under consideration, that is P, is a collection of victory probabilities among K entities, which could represent players or also groups of players. The matrix is denoted by $\underset{K\times K}{P}$, where K is the number of players/ group of players taken into consideration.

$$P = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,K} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,K} \\ \vdots & \vdots & \vdots & \vdots \\ p_{K,1} & p_{K,2} & \cdots & p_{K,K} \end{pmatrix}$$

where the entry $p_{i,j}$ is the probability of player i beating player j. In the Tennis game, ties are not allowed. Either you win or you lose. Therefore, $p_{i,j} + p_{j,i} = 1$ for them to be probabilities. It follows that $p_{j,i} = 1 - p_{i,j}$. So, the lower triangular part of P is deterministic given the upper triangular matrix P. From now on, I focus just on modelling the upper part of the matrix. I require that the elements in the upper triangular part must always be greater than 0.5, otherwise the assumption on the monotonicity in the probabilities would be violated. For example, if $p_{1,2} = 0.4 \leq 0.5 \implies p_{2,1} = 0.6$, which would violate the fact that player 1 is stronger than player 2. Finally, the main diagonal is set equal to 0.5 in the case of teams and to 0 in the case of players.

Finally, to impose the SST property, these entries should be ordered. Without loss of generality, we can assume that player/team 1 is the strongest, while player/team K is the weakest. From that, the probabilities should be constrained as follows:

- The probabilities must increase monotonically as the index of the columns j increases;
- The probabilities must decrease monotonically as the index of the rows i increases.

The matrix P with the described modification will look like this:

$$\begin{pmatrix} 0.5 & \leq & p_{1,2} & \leq & \ldots & \leq & p_{1,K} \\ & & \vee| & & \ldots & & \vee| \\ 1-p_{1,2} & \leq & 0.5 & \leq & \ldots & \leq & p_{2,K} \\ \vdots & \vdots & \vdots & \vdots & \ldots & \vdots & \vdots \\ 1-p_{1,K} & \leq & 1-p_{2,K} & \leq & \ldots & \leq & 0.5 \end{pmatrix}$$

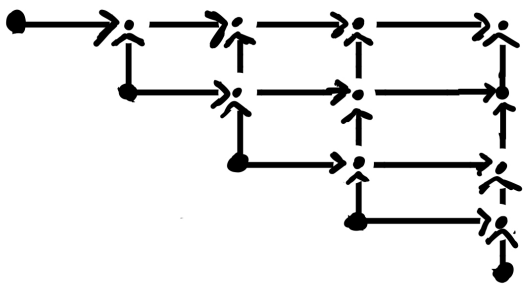

FIGURE 1. Dag representation of the Poset imposed onto the SST matrix

2.1. **Defining a Poset over P.** Having defined P, now we want to define a finite Poset $(P, \leq)$. Let (i, j) and (p, q) be two elements of P and define the binary relation $\leq$ on P by

$$(p, q) \leq (i, j) \iff p \leq i \text{ and } q \leq j \tag{11}$$

Now, $(P, \leq)$ is clearly a Poset since it satisfies the three properties of Definition (**??**)def:Poset). We represent the corresponding directed acyclic graph (P, F), where F is the set of directed edges between vertices in Figure (1) by using the definition of Adjacent Lower Neighborhood that we introduced before.

Definition 10 (Cone over P)**.** *In this case, for any $(i, j) \in D$ the cone of (i, j) is the set:*

$$cone\ (i, j) = \{(i, j-1), \ldots (i, i), (i+1, j), \ldots, (K, j)\} \tag{12}$$

Definition 11 ($\overline{\text{cone}}(i, j)$)**.** *In this case, for any $(i, j) \in P$ the closure of the cone of (i, j) is the set:*

$$\overline{cone}\,(i, j) = \{(i, j), \ldots (i, i), (i, j), \ldots, (K, j)\} \tag{13}$$

Definition 12 (The adjacent lower neighborhood of (i, j))**.** *In this case, for any $(i, j) \in P$ the adjacent lower neighborhood of (i, j) is the set:*

$$adjl\ (i, j) = \{(i, j-i), (i+1, j)\} \tag{14}$$

Definition 13 (closure of adjl (i, j))**.** *In this case, for any $(i, j) \in P$ the closure of adjl (i, j) is the set:*

$$\overline{adjl}\,(i, j) = \{(i, j-i), (i+1, j), (i, j)\} \tag{15}$$

Definition 14 (D^P_{min})**.** *In this case, the minimal element denoted by D_{min} is such that*

$$P_{min} = \{(i, j) \in P : i = j\} \tag{16}$$

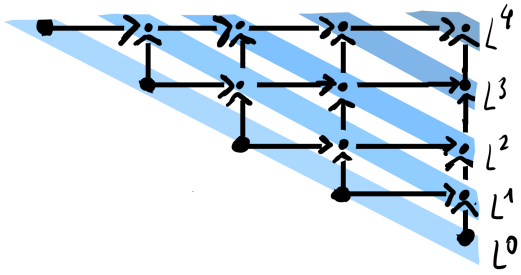

FIGURE 2. Visual Display of the level sets $L^k : k = 0, \ldots, 4$

namely the main diagonal of the P matrix.

The *level sets* of P, in this case, corresponds to the diagonals above the main one Given $L^0 \equiv D_{min}$. Then $L^1 \equiv L^0$.

2.2. **Partially ordered Markov Models applied to P.** Considering the finite set of random variables $\{P_{1,1}, \ldots, P_{K,K}\}$ indexed by locations where

$$D \equiv \{(1,1), \ldots (K,K)\}$$

Having showed the existence of a directly acyclic graph (P, F) and its corresponding poset $(P, \leq)$, we can write down

$$P(P_{ij}|Z(\text{cone } i,j)) = P(P_{ij}|P(\text{adl } i,j)) \tag{17}$$

$$P(P_{ij}|P_{i+1,j}, P_{i,j-1}) \tag{18}$$

Now, exploiting Proposition (1), we can write:

$$P(P(D)) = \prod_{i=1}^{K} \prod_{j=i}^{K} P(P_{ij}|P_{i+1,j}, P_{i,j-1}) \tag{19}$$

3. Stick-Breaking prior process

In this section, we define a generative process for the SST matrix using the POMM framework. We consider the above-defined level sets $L^1, \ldots, L^K$ for a $K-$ dimensional matrix P and we specify a hyperprior distribution process over the POMM one. In particular, we will resort to a modified version of the stick-breaking prior process to create matrices which satisfy the SST property.

3.1. General definition of the Stick-Breaking prior process.

A Stick-Breaking prior process, conceptually, involves repeatedly breaking off and discarding a random fraction (sampled from a Beta distribution) of a "stick" that is by default of length 1. The distribution drawn is discrete with probability 1. In the stick-breaking process view, we explicitly use the discreteness and give the probability mass function of this (random) discrete distribution as:

$$f(\theta) = \sum_{k=1}^{\infty} \beta_k \cdot \delta_{\theta_k}(\theta)$$

where δ_{θ_k} is the indicator function which evaluates to zero everywhere, except for $\delta_{\theta_k}(\theta_k) = 1$. Since this distribution is random itself, its mass function is parameterized by two sets of random variables: the locations $\{\theta_k\}_{k=1}^{\infty}$ and the corresponding probabilities $\{\beta_k\}_{k=1}^{\infty}$.

The probabilities β_k are given by a procedure resembling the breaking of a unit-length stick (hence the name).

$$\beta_k = \beta'_k \cdot \prod_{i=1}^{k-1} \left(1 - \beta'_i\right) \tag{20}$$

where $\beta'_k$are independent random variables with the beta distribution Beta$(1, c)$. The resemblance to 'stick-breaking' can be seen by considering β_k as the length of a piece of a stick. We start with a unit-length stick and in each step, we break off a portion of the remaining stick according to β'_k and assign this broken-off piece to β_k. The formula can be understood by noting that after the first $k-1$ values have their portions assigned, the length of the remainder of the stick is $\prod_{i=1}^{k-1}(1-\beta'_i)$ and this piece is broken according to β'_k¿ and gets assigned to β_k

The smaller c is, the less of the stick will be left for subsequent values (on average), yielding more concentrated distributions.

3.2. Application to the SST case.

In the context of the specific case, the level sets of the POMM are defined as the locations θ_k, and a stick-breaking prior is imposed on them through truncated Beta distributions. The first step is to define the Beta$(1, c)$ distributed random variables, namely b^i: we do that in Equation (21). Then, p_i values are generated according to the stick-breking process as in Equation (22). The step in equation (23) maps these proportions from the $[0, 1]$ interval to the $[0.5, \beta^0_{\max}]$ one, via the function $f^{\beta^0_{\max}}$,

which is nothing but a linear transformation of the values. Moreover, since we want to induce an increasing behaviour in the boundaries for the p_{ij} we take the cumulative sum of the simulated p_i. Finally, in Equation (24) we simulate the entries of the POMM from a Beta$(1,1)$, where we have different truncations depending on the level set, meaning those entries $\{p_{ij} \in P : (j-i) = k\}$ for $k = 1, \ldots, K$.

Here is the full model specification of a POMM$(c, \beta^0_{\max}, K)$ process:

$$b_i \sim \text{Beta}(1, c) \qquad \text{for } i = 1, \ldots, K \tag{21}$$

$$p_i \sim b_i \prod_{j=1}^{i-1}(1 - b_j) \qquad \text{for } i = 1, \ldots, K \tag{22}$$

$$\beta^0_i = f_{[0,1]\to[0.5,\beta^0_{\max}]}\left(\sum_{j=1}^{i} p_j\right) \qquad \text{for } i = 1, \ldots, K \tag{23}$$

$$p_{ij} \sim \text{Beta}(1,1)\mathbb{I}(\beta^0_{i-1} < L^k < \beta^0_{i+i}) \qquad \text{for } (j-i) = k, \quad \text{for } k = 1, \ldots, K \tag{24}$$

where c is the concentration parameter, $\beta^0_1 = 0.5$ and $\beta^0_{\max}$ is the maximum value that the entries p_{ij} can take.

To better understand the role of c, we have represented in Figure 3 the evolution of the process β^0_i for $i = 1, ..., K$ according to different c values. We have done the same for the hyper-parameter $\beta^0_{\max}$ in Figure 4. Finally, the resulting stick-breaking prior can be visualized in Figure 5 an example of the simulated matrix given $K = 10, c = 3, \beta^0_{\max} = .7$.

Overall, the Stick-Breaking prior process provides a rigorous and coherent method for generating discrete distributions with a flexible and controllable parametrization

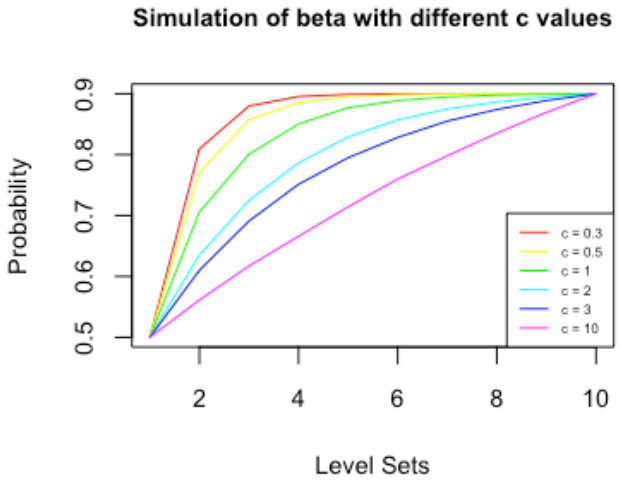

FIGURE 3. The plot shows the evolution of different β prior processes. Every line corresponds to a different concentration parameter c. The plot shows that for small c values we obtain a very abrupt increase in the probabilities, and then, after reaching the plateau, the values are basically similar to the others. Instead, for larger c values, we observe a more smooth increase in the probabilities.

4. Full model specification

The following Bayesian model is used to describe and analyze pairwise data, with the specific aim to identify clusters of points with similar connectivity patterns. Specifically, the model assumes that the nodes in the network belong to a fixed number of clusters or blocks and that the connectivity between nodes is determined by the probability of edge formation between nodes within the same block.

The model uses a Poisson distribution to model the number of blocks, a Dirichlet-multinomial distribution to model the distribution of node assignments across blocks, and a binomial distribution to model the distribution of edges within blocks. Additionally, the model includes a POMM process to model the probability of edge formation between nodes within blocks.

The goal of the model is to estimate the number of blocks, the distribution of nodes across blocks, and the probability of edge formation between nodes within blocks, given observed network data. The Bayesian approach allows for uncertainty in these estimates and provides a framework for incorporating prior knowledge and updating beliefs as new data becomes available.

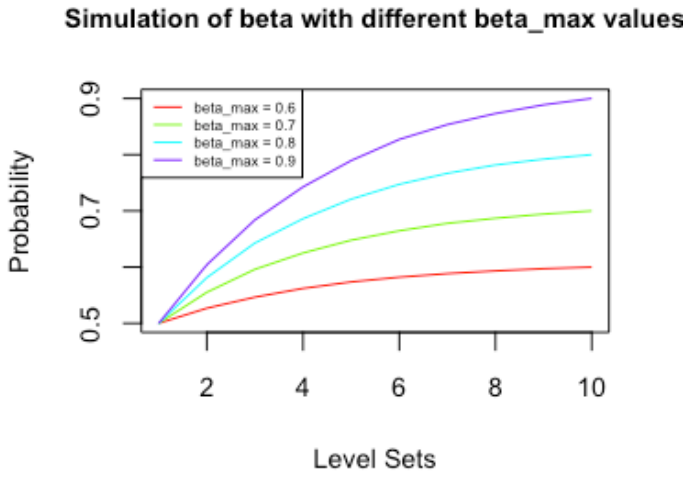

FIGURE 4. The plot shows the evolution of different β^0 prior processes. Every line corresponds to a different β_{max} value, given a concentration parameter $c = 3$. The plot shows that while the shape of each line is quite similar, the β_{max} parameter effectively modifies the range of the probabilities through the different level sets.

4.1. **Model specification.** We denote the nodes in this setting as $1, \ldots, n$. We define the model as follows:

- Let K follow a Poisson distribution with mean 1, subject to the condition $K > 0$.
- Let $\theta_1, \ldots, \theta_K$ be drawn from a Dirichlet distribution with parameter vector γ.
- Let z_i be independently and identically drawn from a multinomial distribution with one trial and probability vector $(\theta_1, \ldots, \theta_K)$.
- Let $P = \{p_{i,j}\} \sim \text{POMM}(c, \beta^0_{\max}, K)$.
- Let $y_{i,j}$ follow a binomial distribution with parameters n_{ij} and p_{z_i,z_j}, subject to the condition $n_{ij} > 0$.

Here, z_i takes values in $1, \ldots, K$. The likelihood function can be specified as:

$$p(\mathbf{y}|p, \theta, z, \gamma) = \prod_{i<j}^{n-1} \binom{n_{ij}}{y_{ij}} p_{z_i,z_j}^{y_{ij}} (1 - p_{z_i,z_j})^{n_{ij}-y_{ij}} \tag{25}$$

To specify the prior distribution, we start with the prior on z following a multinomial distribution. We have n players to assign to K different labels, by replacing the extracted labels after each draw. We denote the variable which is the extracted label $i \in \{1, ..., K\}$ as Z_i, and denote as θ_i the probability that a given extraction will be of label i. The joint

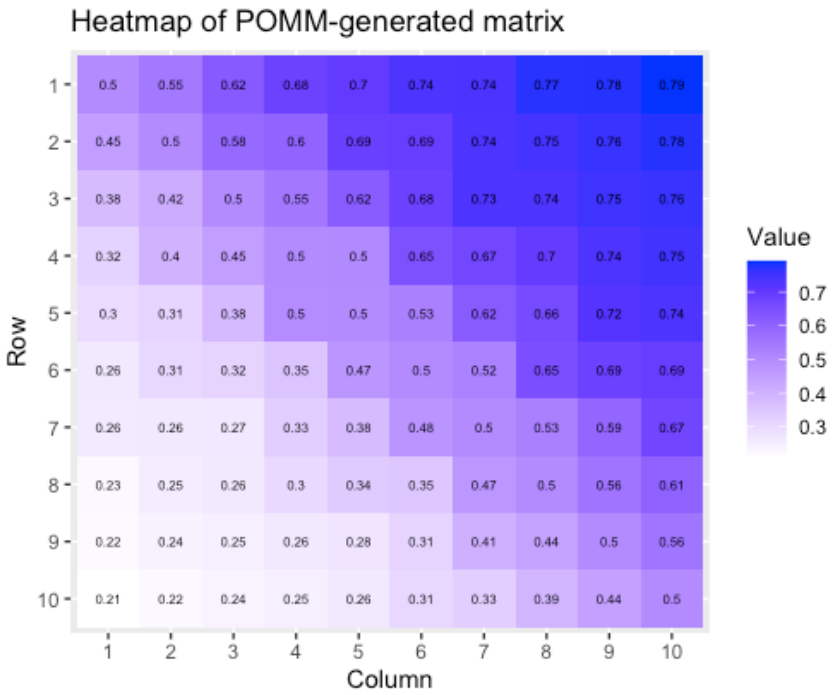

FIGURE 5. The plot shows a POMM matrix generated with parameters $K = 10, c = 3, \beta_{max} = 0.8$. The higher value is 0.79, which means that we are able to enforce the truncation as wanted. The parameter $c = 3$ translates into a moderately concentrated matrix, where we do not have big differences in players' winning probability

probability of $Z_1, \ldots, Z_n$ will be:

$$f(z_1, \ldots, z_K; n, \theta_1, \ldots, \theta_k) = \Pr(Z_1 = z_1, \ldots, Z_K = z_K) \tag{26}$$

$$= \begin{cases} \dfrac{K!}{z_1! \cdots z_K!} \theta_1^{z_1} \times \cdots \times \theta_k^{z_k}, & \text{when } \sum_{i=1}^{K} z_i = n \\ 0 & \text{otherwise,} \end{cases} \tag{27}$$

for non-negative integers $z_1, ..., z_K$.

The probability mass function can be expressed using the gamma function as:

$$p(z_1, \ldots, z_k | \theta_1, \ldots, \theta_k) = \frac{\Gamma(\sum_i z_i + 1)}{\prod_i \Gamma(z_i + 1)} \prod_{i=1}^{K} \theta_i^{z_i} \tag{28}$$

Here, n_i represents the number of nodes/players allocated to block i. On θ, the Dirichlet prior has the following form:

$$f(\theta_1, \ldots, \theta_K; \gamma_1, \ldots, \gamma_K) = \frac{1}{\mathrm{B}(\boldsymbol{\gamma})} \prod_{i=1}^{K} \theta_i^{\gamma_i - 1} \tag{29}$$

By marginalizing out θ, and setting the hyperparameter $\boldsymbol{\gamma}/K = 1$, we can express the marginal distribution of z as:

$$p(z|K) = \frac{\Gamma(K)}{\Gamma(N+K)} \prod_{i=1}^{K} \Gamma(1+n_i) \tag{30}$$

For the distribution of P we refer to the previous section in which we have detailed the specification.

Finally, we specify the prior distribution for the parameter K as:

$$p(K|\lambda = 1) = \frac{1^K e^{-1}}{K!} = \frac{1}{e \cdot K!} \propto \frac{1}{K!} \tag{31}$$

5. Simulating synthetic data: code description

The code `simulating_tournament_test` simulates a tournament with multiple matches between players who belong to different clusters. The number of clusters is simulated from a truncated Poisson distribution, and each player within a cluster is assigned to one of the possible clusters using a Dirichlet multinomial distribution. Then, for each cluster pair, the probability of a player from one cluster winning against a player from another cluster is calculated using POMM process. Finally, a given number of matches between random pairs of players is simulated, where the number of victories of each player is also simulated based on a binomial distribution.

More specifically, the function takes the following arguments:

- `n`: the total number of players in the tournament. alpha: a parameter used to simulate the SST matrix, controlling the weight assigned to the spike (i.e., the probability of a draw). beta: a parameter used to simulate the SST matrix, controlling the weight assigned to the slab (i.e., the probability of one player winning).
- `min_clust` and `max_clust`: the minimum and maximum number of clusters that can be created using a truncated Poisson distribution.
- `M`: the number of matches to be simulated.
- `n_ij_max`: the maximum number of games that can be played between any two players the tournament.

5.1. Simulating K.

As we already said, K is simulated according to a truncated Poisson, i.e.,

$$K \sim Poisson(\lambda = 1)\mathbb{I}(\texttt{min\_clust} < K < \texttt{max\_clust}) \tag{32}$$

distribution with parameter. As for `min_clust`, `max_clust`, they are input under the choice of the user, to better control the number of clusters. In order to sample from the truncated poisson we use inverse sampling:

$$\begin{aligned} U &\leftarrow Unif(0,1) \\ K &\leftarrow F^{-1}(((F(b,\lambda) - F(a,\lambda)) * u + F(a,\lambda)),\lambda)) \end{aligned} \tag{33}$$

where F is the CDF of a Poisson distribution with parameter λ and a, b are respectively the lower and the upper truncations. In our specific case we have $\lambda = 1, a = \texttt{min\_clust}, \texttt{max\_clust}$ Also in the following parts, when we will need to simulate a truncated Beta distribution, we will use the same method.

5.2. Simulating $z|K$.

After obtaining K, we assign each player in the tournament to a cluster using a Dirichlet multinomial distribution with a concentration parameter called γ as follows:

$$p(z_1, \ldots, z_k | \gamma_1, \ldots, \gamma_k) = \frac{\Gamma(\sum_i z_i + 1)}{\prod_i \Gamma(z_i + 1)} \prod_{i=1}^{K} \gamma_i^{z_i} \tag{34}$$

We are working under the assumption that cluster 1 is the strongest and cluster K is the weakest. Given that, we have multiple choices for the γ parameter, and here we report three cases that are not exhaustive but explain the kind of reasoning behind γ. Indeed, γ controls the expected size of the clusters, namely how many nodes are assigned to each label. Here are the three cases:

- $\gamma = 1 \quad \forall k \in \{1, \dots, K\}$, so that the expected size of each cluster is the same.
- $\gamma = [1, \dots, K]$, so that the expected size of the clusters is inversely proportional to the probability of victory;
- $\gamma = [K, \dots, 1]$, so that the expected size of the clusters is directly proportional to the probability of victory.

Now, given γ, we simulate

$$p \leftarrow \texttt{rgamma}(K, \gamma) \tag{35}$$

which is an indirect way to sample from the Dirichlet distribution. Then, we sample the labels from the Multinomial

$$z \leftarrow \texttt{sample(1:K,N,prob=p/sum(p),rep=TRUE)} \tag{36}$$

where z is $N-$dimensional vector in which rows index stand for the players' IDs and each entry corresponds to a label of the assigned cluster.

5.3. **Simulating** POMM$(c, \beta^0_{\max}, K)$**.** The code `simulating_POMM_stick_breaking` implements the stick-breaking process as follows:

- It generates K beta-distributed random variables `b_1`, ..., `b_K` with a concentration parameter c using the function `rbeta(K, 1, c)`.
- It computes the stick-breaking proportions `p_1, ..., p_K` using the formula

```
p_i = b[i] * prod(1 - b[1:(i-1)])
```

  .
- It sums over the proportions, `p_sum` $\leftarrow$ `cumsum(p)` to ensure an increasing behaviour and maps them to a desired scale using the formula

```
beta_0 = (beta_max - 0.5) * (p_sum - min(p_sum))/(max(p_sum) - min(p_sum)) + 0.5
```

  where `beta_max` is a user-defined parameter that sets the maximum value of β, and `p_sum` is the cumulative sum of the proportions.
- It generates the β values using the proportions as truncations with the function `sample_beta_trunc(n_items,1,1,beta_0[K-i], beta_0[K-i+1])`, where `n_items` is the number of β values to generate and $K - i$ and $K - i + 1$ are the indices of the two corresponding proportions in $beta\_0$.
- It puts everything together in a matrix and returns the result.

5.4. **Sampling the matches.** The data frame is called `z_players`, and it consists of two columns: `id` and `z`. The `id` column contains the values 1 to n (where n is the number of players), which are the players' IDs, and the `z` column contains the cluster assignment for each player, which is determined by the previous step in which we have simulated the parameter z.

This code generates a data frame (`df`) with the columns `player_1, player_2, n_ij, y_ij`, and `p_ij`. The variables `player_1` and `player_2` represent the players who compete against each other in a match, `n_ij` is the number of games they play, `y_ij` is the number of games won by `player_1`, and `p_ij` is the probability of `player_1` winning a game against `player_2`.

The code uses a while loop that runs `M` times, where `M` is the user-specified number of matches. In each iteration of the loop, it samples two players (`pl_1_i` and `pl_2_i`) from a pool of players (`z_players$id`) based on their probability of being chosen (`aux2$p`). `pl_2_i` is chosen from the remaining players who have not been chosen as `pl_1_i`.

In this specific case, we set `aux2$p` to be inversely proportional to the size of the cluster each player belongs to, and instead proportional to the winning p_ij, to take into account the fact that stronger players play more, due to the knockout mechanisms which are prevailing in almost every tennis tournament. In another application, this probability could be uniform, or inversely proportional to p_ij

So, by saying that the probability of playing is inversely related to the cluster size, we are effectively associating to each player, not only with its cluster but also with the probability of being assigned to that cluster, which we discussed in the Dirichlet-Multinomial step. This probability can be found at (`??`), namely `prob=p/sum(p)`. With this association, we build `aux2$p`, a $n \times 1$ vector which provides the probabilities for matches formation.

A new data frame (`matches_i`) is then created with the selected players, and their respective blocks (`z_1` and `z_2`) are added to the data frame. The number of games they play (`n_ij`) is sampled from a truncated Poisson distribution with a maximum value of `n_ij_max`. The probability of `player_1` winning a game against `player_2` (`p_ij`) is calculated based on their blocks (`z_1` and `z_2`) using the pre-simulated matrix `p`.

The number of games won by `player_1` (`y_ij`) is then sampled from a binomial distribution with parameters `n_ij` and `p_ij`.

Finally, the information for the current match is added to the df data frame using the `rbind()` function. The loop continues until it has run M times. The full code is presented in Algorithm (1).

6. ATP tennis dataset

Our raw data are in a repository which can be found at

`'https://pkgstore.datahub.io/sports-data/atp-world-tour-tennis-data'`

. If two players have played multiple times, they compare on different rows as we can see in table(1).

To begin with, I select all rows with the first pair of players, regardless of the order in which they apper. Then, I construct these two new variables:

Algorithm 1 Building Matches

Initialize data frame `df` with columns `player_1`, `player_2`, `n_ij`, `y_ij`, `p_ij`, where each column is filled with `NA` values.
Initialize i to 0.
while $i < M$ **do**
Sample player `pl_1_i` from `z_players$id` with probability `aux2$p`.
Sample player `pl_2_i` from the remaining players in `z_players$id` with probability `aux2$p[setdiff(z_players$id,pl_1_i)]`.
Create a new data frame `matches_i` with columns `pl_1_i, pl_2_i, z_1, z_2, n_ij, y_ij`, where `z_1` and `z_2` are the blocks of the players.
Set `matches_i$z_1` to the block of `pl_1_i` from `z_players`.
Set `matches_i$z_2` to the block of `pl_2_i` from `z_players`.
Sample the number of games `n_ij` from a truncated Poisson distribution with maximum value `n_ij_max`.
Calculate the probability of `pl_1_i` winning a game against `pl_2_i` as `p_ij` based on their blocks `z_1` and `z_2` using a matrix `p`.
Sample the number of games won by `pl_1_i` as `y_ij` from a binomial distribution with parameters `n_ij` and `p_ij`.
Add the current match information to *df* using the `rbind()` function to create a new row with values `pl_1_i`, `pl_2_i`, `n_ij`, `y_ij`, `p_ij`.
Increment i by 1.
end while

Winner	**Loser**
Djockovic	Medvedev
⋮	⋮
Djockovic	Medvedev
Djockovic	Medvedev
Medvedev	Djockovic
⋮	⋮
Medvedev	Djockovic
Medvedev	Djockovic

TABLE 1. Raw data

- n_{games}, e.g. total number of games between Djockovic and Medvedev = total number of victories of Djockovic vs Medvedev + total number of victories of Medvedev vs Djockovic. Basically one counts the number of times these 2 names appear on the same row, irrespective of their order

Player1	**Player2**	n_{games}	$n_{victories}$
Djockovic	Medvedev	5	3
Djockovic	Nadal	8	6
Nadal	Tsonga	4	0

TABLE 2. Final dataset

- $n_{victories}$, e.g. total number of victories of Djockovic against Medvedev = total number of victories of Djockovic vs Medvedev. Basically one counts the number of times Djockovic and Medvedev appear exactly in this order on the same row.

Then, I retain just the first pair observation, e.g. Djockovic & Medvedev, and I disregards all the others. Then, I repeat the same process for all unique unordered pairs of players. In this way, I obtain a dataset storing, for each pair of players who have played at least one, all the relevant informations in one single entry.

What about the observations in which Medvedev won against Djockovic? These will no longer be an entries of the dataframe.However, this data will still be indirectly available by computing total number of games between Djockovic and Medvedev (available in column n_{games}, at the correspective row) - total number of victories of Djockovic vs Medvedev (available in $n_{victories}$,at the correspective row).

The final dataframe will look like as in table(2)

7. Descriptive statistics comparing simulated vs ATP data

We start by choosing a particular configuration of parameters to simulate a given tournament to compare it with the ATP data.

- `N`=537 since in the ATP dataset we have exactly 537 players
- `M`=3389 since in the ATP dataset this is the total number of matches
- `max_number_games` = 4 since this is the max value we observe in the ATP data.

So, for these three initial parameters, we are replicating the features of the real tennis data. With respect to the more "statistical" ones there is no benchmark, so we choose to follow the intuition:

- `max_clust`=3
- `min_clust`=3
- `c` = 1.7
- `beta_max` = .8

After generating the data, we compare the simulated tournament and the true ATP data by computing some summary statistics and some meaningful plots shown in Figures (6), (7), (8), (9), (10). The scatterplot shows a similar pattern of strong correlation between y_{ij} and n_{ij}. However, in ATP there is a higher points density close to the origin, while in the simulated tournament we have more density close to the mean. It means that real data are more dispersed than simulated ones. We can reach the same conclusion by looking at the number of games distribution and the number of victories. In particular, by looking at Figures (9) and (10) the discrepancy seems more enhanced for the number of victories y_{ij}.

The interesting aspect of this methodology is that we could compute the discrepancy between the two distributions, for example using a Kullback-Leibler divergence, and find those parameters that minimize it.

8. Possible applications

The POMM model can have various applications in fields where pairwise comparisons are made. Some examples of applications are:

- Sports Analytics: The POMM model can be used to rank sports teams based on their pairwise comparison results. It can also be used to predict the probability of a team winning a match against another team.
- Marketing: The POMM model can be used to rank products based on their pairwise comparison results in surveys. It can also be used to estimate the probability of a customer preferring one product over another.
- Decision Making: The POMM model can be used to rank options based on their pairwise comparison results. It can also be used to estimate the probability of one option being preferred over another in a decision-making process.
- Social Science: The POMM model can be used to study social preferences by asking individuals to compare two different options. For example, it can be used to understand people's preferences for different political candidates, policies, or social norms.

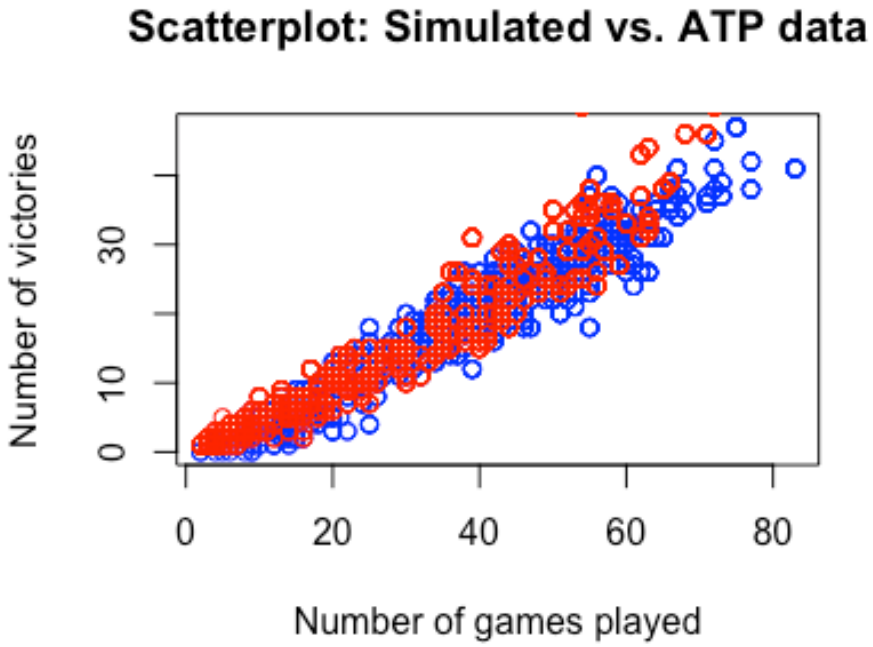

FIGURE 6. Scatterplot of n_{ij}, the number of games played between any two pair of players, and y_{ij}, the number of victories of player i vs player j. We show the values both for Simulated (blue) and ATP data (red)

- Biology: The POMM model can be used to study the relative fitness of different genotypes in evolutionary biology or the preferences of animals for different stimuli in behavioral ecology.

Overall, the POMM model can be applied in any field where pairwise comparisons are made and where the goal is to rank or estimate the probabilities of different options.

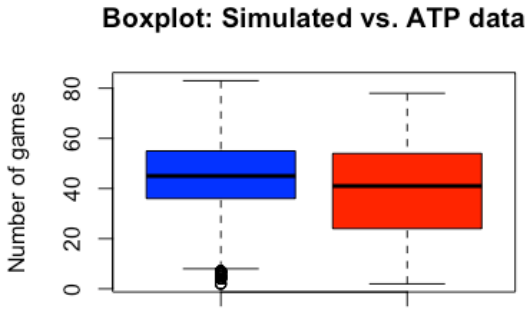

FIGURE 7. Boxplot of n_{ij} values both for Simulated (blue) and ATP data (red)

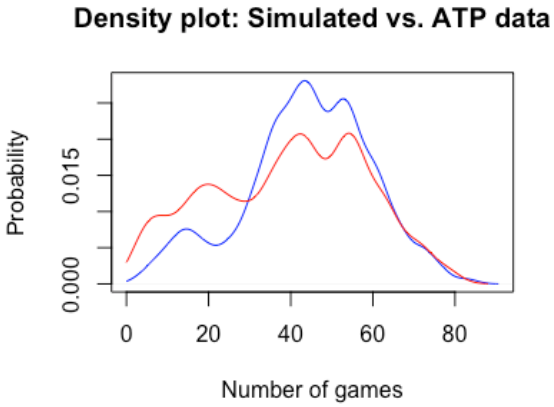

FIGURE 8. Density Plot for the n_{ij} values both for Simulated (blue) and ATP data (red)

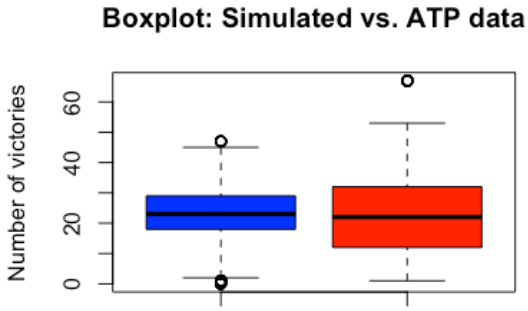

FIGURE 9. Boxplot of y_{ij} values both for Simulated (blue) and ATP data (red)

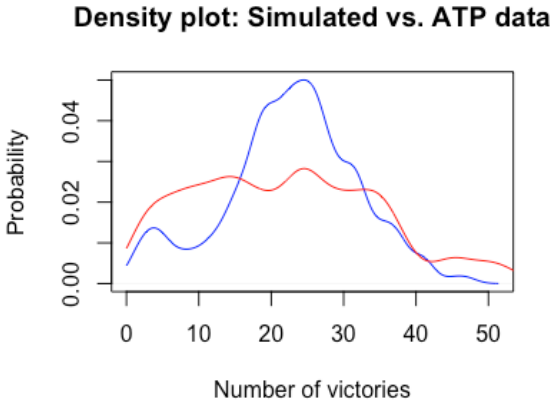

FIGURE 10. Density Plot for the n_{ij} values both for Simulated (blue) and ATP data (red)