Data Min Knowl Disc (2017) 31:1580–1621
DOI 10.1007/s10618-017-0510-5

Regimes in baseball players' career data

Marcus Bendtsen[1]

Received: 28 February 2016 / Accepted: 12 May 2017 / Published online: 27 May 2017

Abstract In this paper we investigate how we can use gated Bayesian networks, a type of probabilistic graphical model, to represent *regimes* in baseball players' career data. We find that baseball players do indeed go through different regimes throughout their career, where each regime can be associated with a certain level of performance. We show that some of the transitions between regimes happen in conjunction with major events in the players' career, such as being traded or injured, but that some transitions cannot be explained by such events. The resulting model is a tool for managers and coaches that can be used to identify where transitions have occurred, as well as an online monitoring tool to detect which regime the player currently is in.

1 Introduction

Baseball has been the de facto national sport of the United States since the late nineteenth century, and its popularity has spread to Central and South America as well as the Caribbean and East Asia. It has inspired numerous movies and other works of arts, and has also influenced the English language with expressions such as *"hitting it out of the park"* and *"covering one's bases"*, as well as being the origin of the concept of a *rain-check*. As we all are aware, the omnipresent baseball cap is worn across the world. Excluding the postseason, there are approximately 2430 games of major league baseball in a season, equating to 162 games per team played over 6 months.[1] It speaks to the popularity of the game when the team with the *lowest* average attendance per

[1] While it may seem surprising, teams play virtually every day throughout the season, with only a few rest days scheduled.

✉ Marcus Bendtsen
marcus.bendtsen@liu.se

[1] Linköping University, Linköping, Sweden

game during the 2015 season had an average attendance of 21,796 (the highest was 40,502).

Apart from being an extremely popular pastime, there is also a business side to baseball. A winning team tends to draw bigger crowds, thus ticket and merchandise sales increase. From a managerial perspective it is therefore necessary to analyse players to identify beneficial trades and strategies, in order to create the conditions for a successful team. Furthermore, with the growth of sports betting and fantasy leagues, this analysis is not only of interest for decisions makers within the teams, but from fans and spectators as well.

Bill James was the first to coin the term *sabermetrics*, referring to the empirical analysis of baseball statistics. SABR is an acronym for The Society for American Baseball Research, which was founded in 1971 and was a lynch pin for modern day baseball analysis, thus *saber*metrics. Bill introduced the concept of *ageing curves*, a representation of how the ability of a certain player increases until it peaks at a certain age, and then falls off on the other side. This was an important insight, as it explained some of the variance in the summary statistics used at the time. Another of Bill's ideas was to compare current players with past players, creating a similarity score used to gain insight into how players performed relative to each other. It was later that Nate Silver combined these ideas to create the famous PECOTA system, which essentially used a nearest neighbour approach to create clusters of players based on their ageing curves (Silver 2012). While the system is used to create predictions of how players will perform over the coming season, it can also give insight into the worst-, best-, and most-likely scenario that will happen, as the ageing curves in the cluster of a player are also informative.

While owners and managers have always to some degree incorporated statistics as part of their strategy to trade players and win games, it was the release of Michael Lewis's book *Moneyball* (Lewis 2003) that unveiled to the general public the extent to which sabermetrics was used by Billy Beane when he managed the Oakland Athletics with great success, despite financial constraints. Today the use of sabermetrics is widespread across the league, but still creates headlines when teams like the Huston Astros employ former NASA engineers (McTaggart 2012), or when breaches akin to corporate espionage are detected (Langosch 2015).

Naturally, one of the goals of sabermetrics is to produce predictions of players' future performance. There is, as one might suspect, a rich history of approaches for this task, and an untold number of books, magazines, news letters and forums are dedicated in furthering the performance of this endeavour (Baseball prospectus, www.baseballprospectus.com; Bill James online, www.billjamesonline.com; Baseball think factory, www.baseballthinkfactory.org). From these efforts several well known systems have grown, such as Steamer (www.steamerprojections.com), Marcel (www.tangotiger.net/marcel), and the aforementioned PECOTA system (Baseball prospectus, www.baseballprospectus.com). However, sabermetricians are equally concerned with the process that leads to the prediction (Bill James' similarity scores and Nate Silver's outcome scenarios are examples of this). While we recognise the importance of making predictions, and acknowledge the difficulties in doing so, our intentions in this paper is in line with understanding the data. We are specifically interested in learning if there are *regimes* in the career data, and if so, how these regimes transition

into one another. A regime in this case is defined as a steady state of the relationships between the variables in the data, but where these relationships may change between regimes. We will describe this in further detail in Sect. 2.1, for now it suffices to think of regimes as segments of the data for which a baseball player's performance is different enough that it warrants estimating a new model.

The reason why a baseball player would transition into a new regime may not be recorded or even directly observable, but may be a complex combination of increased skill and experience, ageing, strategic decisions, etc. For some regime changes it may be more evident what could have caused the shift in the player's performance, e.g. most of us would expect any sportsperson to play differently after an injury. What attracts our attention in this paper is the possibility of detecting regime transitions, and then identifying which of them are not directly explainable from publicly available data. The motivation for why this is of value lies in the fact that while coaches and managers make many decisions throughout a season, they may not be aware that a combination of these decisions may transition players to and from regimes in which they have been performing extraordinarily or subpar. Detecting regime changes in the data would allow decision makers to analyse the decisions they made during the time of the transition to see if they can recreate (or stay away from) the conditions that transitioned the player to the extraordinary (or subpar) regime.

We will approach this problem by using a combination of Markov Chain Monte Carlo (MCMC) and Bayesian optimisation to learn a gated Bayesian network (GBN). A GBN combines several Bayesian networks (BNs) in order to describe the relationships amongst the variables during different regimes in the data. The BNs can be either active or inactive, and are combined using so called *gates* which control when the states' of the contained BNs change. The GBN can be depicted as a directed graph, in which the nodes are either BNs or gates, and from the graph we can tell how the different regimes can transition into one another. We will introduce the GBN model in Sect. 2.3.

The rest of the paper is disposed as follows. In Sect. 2 we will introduce regimes more formally, as well as giving an introduction to BNs and GBNs. In Sect. 3 we will contrast our approach with related work. In Sect. 4 we will explain in detail how we aim to learn a GBN to model a set of variables that exhibit regimes. We will in Sect. 5 return to the principal interest of this paper, using the proposed procedure on baseball players' career data, where we will investigate the regime changes that may occur. In Sect. 5 we will also offer a brief introduction to the game of baseball. Finally, we will offer some concluding remarks and thoughts about future work in Sect. 6.

2 Regime changes and gated Bayesian networks

In this section, and the following two, we will depart from the world of baseball. We will do so in order to give a description of the general regime identification problem, as well as the model that we aim to apply. In Sect. 2.1 we offer the problem definition, which consists of subgoals, namely representing and detecting regimes. With the aim of using a single model that incorporates both of these subgoals, we suggest using a generalisation of GBNs (Bendtsen and Peña 2016, 2013), described in Sect. 2.3. In Sect. 4 we offer an algorithm for learning such a model.

2.1 Problem definition

When we observe a probabilistic system over time, it is natural that the observations we make differ from each other. We expect this to be the case due to the randomness of the variables within the system. However, as long as the relationships between the variables, and the distributions of the individual variables are unchanged, the observations we make of the system can be judged as coming from the same joint probability distribution.

Let X, Y and Z be three random variables, and let there exist a joint probability distribution $p(X, Y, Z)$. We can factorise this joint distribution using the chain rule to get $p(X, Y, Z) = p(X|Y, Z)p(Y|Z)p(Z)$. If we also know that the relationship $X \perp Y|Z$ holds, then the factorisation can be reduced to $p(X, Y, Z) = p(X|Z)p(Y|Z)p(Z)$, where Y has been dropped from the conditioning set in the distribution of X, since they are independent given Z.

However, over time the relationships between the variables may change. Assume that after some time $X \perp Y|Z$ no longer holds. This implies that the factorisation cannot be reduced, and therefore $p(X, Y, Z)$ prior to the change and after the change are not the same. Observations before the change and after the change should no longer be judged to be samples from the same joint probability distribution.

Formally, let $\mathcal{S}$ be a system of random variables $\mathbf{X}$, and let the system $\mathcal{S}$ have regimes labeled $R_1, R_2, \ldots, R_m$. For each regime R_k there exists an independence model M_k over the variables $\mathbf{X}$ such that the joint probability distribution $p_k(\mathbf{X})$ is positive. Furthermore, assume that $M_i \neq M_j \; \forall \; i \neq j$. Let $\mathcal{D}$ be a dataset with complete observations of $\mathbf{X}$, where d_i is the i:th observation in $\mathcal{D}$.

By $d_i \sim R_k$ we mean that d_i is an observation of the variables $\mathbf{X}$ when $\mathcal{S}$ is in regime R_k, thus d_i is a sample from $p_k(\mathbf{X})$. We will for brevity say that d_i comes from regime R_k. Then $\mathcal{D} = \{d_1 \sim R_1, d_2 \sim R_1, d_3 \sim R_1, d_4 \sim R_2, d_5 \sim R_2\}$ means that the first three observations came from regime R_1 while the last two observations came from regime R_2. When there is no ambiguity we will shorten $\mathcal{D} = \{d_1 \sim R_1, d_2 \sim R_1, d_3 \sim R_1, d_4 \sim R_2, d_5 \sim R_2\}$ to $\mathcal{D} = \{R_1, R_1, R_1, R_2, R_2\}$.

Given $\mathcal{D} = \{R_1, R_1, R_1, R_2, R_2\}$ it is possible to directly identify which regimes that can transition into each other, i.e. in this case $\mathcal{S}$ can transition from R_1 to R_2. We call this the *regime transition structure* of $\mathcal{S}$, which can be drawn as a graph or denoted with $R_1 \rightarrow R_2$. Had the observations been different, such that $\mathcal{D} = \{R_1, R_1, R_1, R_2, R_2, R_1, R_1\}$, then this would have identified a different structure where R_1 can transition to R_2, and R_2 can transition to R_1 ($R_1 \leftrightarrows R_2$). It is necessary to assume a Markovian property of the regime transitions, such that knowing that $\mathcal{S}$ is in regime R_i is the only necessary information in order to identify which regimes it can transition into. We will defer the reasoning and consequences of this Markovian assumption to Sect. 4.2.2.

We say that $\mathcal{D}$ is a dataset for $\mathcal{S}$ if it identifies the *true* regime structure of $\mathcal{S}$, i.e. $\mathcal{D}$ is a valid sample of $\mathcal{S}$. For instance, if the true regime structure of $\mathcal{S}$ is $R_1 \rightarrow R_2$ then $\mathcal{D} = \{R_1, R_2\}$ is a dataset for $\mathcal{S}$, while $\mathcal{D} = \{R_1\}, \mathcal{D} = \{R_2\}, \mathcal{D} = \{R_2, R_1\}, \mathcal{D} = \{R_1, R_2, R_1\}$ are not. It is implied that datasets are made available to us completely and immediately. However, we will also make use of observations that are given to us one by one, as $\mathcal{S}$ is observed over time. We call this a *stream* of data. The size of a

stream $\mathcal{O}$ depends on how many observations we have made thus far: at time $t = 1$ the stream only contains a single observation, e.g. $\mathcal{O} = \{d_1 \sim R_1\}$, and at time $t = j$ it contains j observations, e.g. $\mathcal{O} = \{d_1 \sim R_1, \ldots, d_j \sim R_k\}$.

Given a dataset $\mathcal{D} = \{d_1, \ldots, d_n\}$ for $\mathcal{S}$, where it is not known from which regime the individual observations came from, and it is not known how many regimes $\mathcal{S}$ exhibits, the primary aim is to learn a model of $\mathcal{S}$ from $\mathcal{D}$. In order to do so we must:

- Identify where in $\mathcal{D}$ there are regime changes.
- Identify the regimes $R_1, \ldots, R_m$ of $\mathcal{S}$, and their corresponding independence models $M_1, \ldots, M_m$ and joint distributions $p_1(\mathbf{X}), \ldots, p_m(\mathbf{X})$.
- Identify the regime transition structure of $\mathcal{S}$.

Once such a model has been defined, the secondary aim is to take a stream of data $\mathcal{O}$ and correctly identify which regime $\mathcal{S}$ currently is in, given every new observation in $\mathcal{O}$. In order to do so we must extend our model to allow it to detect regime changes in $\mathcal{O}$. This will result in a model that can be used to reason about the current regime of $\mathcal{S}$, and also to identify which M_k and $p_k(\mathbf{X})$ should be used for inference purposes.

We will meet the primary and secondary aim by learning a GBN, which is a model that builds on BNs. We will therefore in the next section introduce BNs, followed by an explanation of the GBN model.

2.2 Bayesian networks

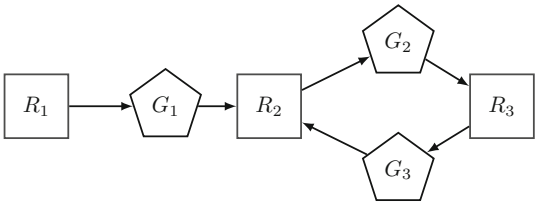
Introduced by Pearl (1988), BNs consists of two major components: a qualitative representation of independencies amongst random variables through a directed acyclic graph (DAG), and a quantification of certain marginal and conditional probability distributions, so as to define a full joint probability distribution. A feature of BNs, known as the local Markov property, implies that a variable is independent of all other non-descendant variables given its parent variables, where the relationships are defined with respect to the DAG of the BN. Let $\mathbf{X}$ be a set of random variables in a BN, and let $\Pi(X_i)$ be the set of variables that consists of the parents of variable $X_i \in \mathbf{X}$, then the local Markov property allows us to factorise the joint probability distribution according to Eq. 1.

$$p(\mathbf{X}) = \prod_{X_i \in \mathbf{X}} p(X_i | \Pi(X_i)) \tag{1}$$

From Eq. 1, it is evident that the independencies represented by the DAG allow for a representation of the full joint distribution via smaller marginal and conditional probability distributions, thus making it easier to elicit the necessary parameters, and allowing for efficient computation of posterior probabilities. For a full treatment of BNs please see Pearl (1988), Korb and Nicholson (2011) and Jensen and Nielsen (2007).

While a BN has advantages when representing a single independence model, it lacks the ability to represent several independence models simultaneously, i.e. it lacks the ability to model several regimes. Therefore, we suggest using a GBN to represent systems that have several regimes, where each regime is represented by a BN. GBNs were originally defined to operate on the posterior distribution of certain variables in the

Fig. 1 Example GBN



BNs (Bendtsen and Peña 2016, 2013), and to be used to model processes with distinct phases, such as algorithmic trading (Bendtsen and Peña 2016, 2014; Bendtsen 2015). However, in the coming section we will generalise the definition of GBNs, allowing them to operate on entire BNs (rather than a set of variables), while at the same time adding certain restrictions on how the GBNs can be constructed.

2.3 Gated Bayesian networks

GBNs use *gates* that connect with BNs using directed edges, where the parent/child relationship is given by the direction of the edge. Each gate has exactly one parent and one child BN.[2] For instance, in Fig. 1 the gate G_1 has BN R_1 as its parent and BN R_2 as its child. As is shown in the figure, a BN can be both a child and a parent for different gates (R_2 and R_3), and while it is not evident from the figure, a BN can be a parent or a child of several gates.

A feature of GBNs is that they keep the contained BNs in an *active* or *inactive* state. Assuming that all BNs are inactive in Fig. 1 except for R_1, any probabilistic or causal queries should be answered using R_1. However, the GBN also defines when the active/inactive state of the BNs change, thus one of the primary tasks of a GBN is to, for each new observation from a stream of data, re-evaluate which BN should be active.

To control this, the gates of a GBN are programmed with predefined logical expressions, known as the gates' *trigger logic*. The trigger logic is an expression regarding the gate's parent and child BNs.[3] If the trigger logic is satisfied, then the gate is said to *trigger* and the gate's parent BN is deactivated while the child BN is activated.

Let there be a function $f(\mathcal{B}, \mathcal{T}) \in \mathbb{R}$, where $\mathcal{B}$ is a BN, and $\mathcal{T}$ is a dataset of observations over the variables in $\mathcal{B}$. The trigger logic of a gate is defined as a comparison between the value of f given the parent and the value of f given the child, given some dataset $\mathcal{T}$. For instance, the trigger logic for each gate in Fig. 1 is presented in Eq. 2, where we require that the child's value divided by the parent's value be greater than some threshold θ in order for the gate to trigger. For each new observation in a stream of data, we re-evaluate the trigger logic of all gates that have an active parent BN.

[2] This restriction does not apply in the original definition of GBNs (Bendtsen and Peña 2013, 2016).

[3] This is a distinction between the generalised GBN presented here and the original GBN definition (Bendtsen and Peña 2013, 2016), where the trigger logic is defined over the posterior probability of a certain variable in the parent BN.

$$
\begin{aligned}
TL(G_1) &:= \frac{f(R_2, \mathcal{T})}{f(R_1, \mathcal{T})} > \theta_1 \\
TL(G_2) &:= \frac{f(R_3, \mathcal{T})}{f(R_2, \mathcal{T})} > \theta_2 \\
TL(G_3) &:= \frac{f(R_2, \mathcal{T})}{f(R_3, \mathcal{T})} > \theta_3
\end{aligned} \tag{2}
$$

By using a GBN to represent a system that has several regimes, we can model each regime using a BN, and then connect them with gates according to the regime transition structure. We can also define the trigger logic in such a way that the model is capable of, given a stream of observations, accurately keeping the BN active that represents the current regime. Therefore, a GBN is a model that can incorporate both aims presented in Sect. 2.1, and to this end we will suggest an algorithm in Sect. 4 to learn a GBN from data. Before we introduce this algorithm, we shall contrast our approach to related work.

3 Related work

Refining or updating the structure and conditional distributions of a BN in response to new data has been studied for some time (Buntine 1991; Lam and Bacchus 1994; Friedman and Goldszmidt 1997; Lam 1998). However, these approaches assume that data is received from a stationary distribution, i.e. a system that does not undergo regime changes.

Nielsen and Nielsen (2008) approach the problem of having a stream of observations which they say is *piecewise stationary*, i.e. observations within a section of the stream come from the same distribution, but changes may occur into a new stationary section. Their goal is to incrementally learn the structure of a BN, adapting the structure as new observations are made available. They achieve this by monitoring local changes among the variables in the current network, and when a conflict occurs between what is currently learnt and what is observed, they refine the structure of the BN. Focusing only on local changes allows them to reuse all previous observations for parts of the BN that have not changed, resulting in a procedure that is less wasteful and more computationally feasible. Since their goal is to adapt the structure of a single BN, their aim is not to identify reoccurring steady state regimes, but rather to adapt to what is known as *concept drift*. Our goal is to learn a BN for each regime and to find the regime transition structure of the underlying system. We also intend to use our model to decide which regime is current given a new stream of data, preserving the learnt BNs for each regime. We do not make the assumption of local change between regimes, but relearn the entire BN, which allows for less assumptions about changes, but as Nielsen and Nielsen (2008) point out, can be wasteful of data and computation time.

Robinson and Hartemink (2010) assume that a complete dataset exists, and use MCMC to identify change points and to revise a non-stationary dynamic BN (nsDBN). They have several potential moves that they can take within the MCMC iterations, for

instance they can shift, split or merge change points, or add/remove an edge in the nsDBN at a change point, etc. By doing so they make the structural learning of the nsDBN an integral part of the MCMC proposals, while our approach plugs in any existing structure learning algorithm. Their aim is to segment the dataset and identify which structure was most likely at the different segments, and thereby discovering non-stationary relationships. Our approach is closely related to the approach of Robinson and Hartemink, in the sense that it is the BNs retained after MCMC that represent the model. However, we aim at identifying changes between static regimes, and not to capture the dynamics between timesteps, thus the GBN is not a DBN. Furthermore, Robinson and Hartemink do not approach the problem of trying to identify if regimes reoccur (each segment is considered unique). Therefore the regime transition structure is not captured, and this entails that they do not attempt to use the model on a new data stream in order to predict which of the already learnt structures should be used. Our approach addresses these two latter points.

The nsDBN approach sets itself apart from frameworks such as hidden Markov models (HMMs) and switching state-space models (Ghahramani and Hinton 2000) because, as Robinson and Hartemink state, a nsDBN "has no hidden variables, only observed variables". In other words, the nsDBN approach does not assume that there is a hidden random variable representing the regime at each time point. For this reason, modelling the transition probabilities between such random variables, as HMMs and switching state-space models do, does not make sense. Our GBN approach sets itself apart from HMMs and switching state-space models for the same reasons as the nsDBN approach.

Guo et al. (2007) and Xuan and Murphy (2007) consider graphs with only undirected edges, i.e. Markov networks rather than BNs. The structure of the Markov network is considered latent by Guo et al. (2007), thus their proposed model is a HMM where the states of the latent variable is the entire structure. In order to identify segmentations in time series, Xuan and Murphy (2007) score segmentations by computing the marginal posterior over segmentations given all possible decomposable Markov networks. Thus the goal of Xuan and Murphy (2007) is to identify the segmentation, rather than the graph. As was pointed out earlier, the GBN that we learn does not contain any hidden variables, and the structure within each regime is explicitly learned. Furthermore, we intend to identify reoccurring regimes and, given new data, predict which regime the system currently is in.

For more on adaptation, we refer the interested reader to a recent survey by Gama et al. (2014), and a comparison between GBNs and a broader range of models can be found in our previous publication (Bendtsen and Peña 2016).

The concept of regimes is not only confined to the realm of probabilistic graphical models. For instance, Markov regime-switching regression models have been employed in economical studies (Goldfeld and Quandt 1973; Hamilton 1989), where one switches the regression coefficients depending on the current regime. This model has also been applied for detecting regimes when baseball player's wages were put in relation to their recent performance (Haupert and Murray 2012), showing abrupt changes in the relationship between performance and salary (during the period of 1911 through 1973).

In our previous publications (Bendtsen and Peña 2016, 2014), we proposed an algorithm for learning GBNs that can be used in decision making processes, specifically in trading financial assets. The GBNs learnt use posterior probabilities in the trigger logic of their gates to decide when it is an opportune time to buy and sell stock shares (the two decisions may use different BNs, thus a GBN is created). The algorithm uses a set of predefined BNs and gates and finds the candidate that achieves the highest reward. However, the algorithm does not learn the structure of these BNs from data, but they are rather supplied by experts. Such a library of BNs was possible to create since there is a wealth of information of how the variables under consideration interact [in the case of Bendtsen and Peña (2016), Bendtsen and Peña (2014) we considered financial indicators as variables]. However, such information is not always available, for instance in the case of the current baseball setting. Furthermore, no regime identification step is taken as part of the learning algorithm, thus the resulting GBN is not the one that best describes the data (in terms of data likelihood), but rather the GBN that performs best according to some score function.

4 Learning algorithm

In this section we will describe the algorithm that we propose in order to learn a GBN, conforming to the definition in Sect. 2.3, that fulfils the primary and secondary aim of the problem definition in Sect. 2.1. The pseudocode for the learning algorithm can be found in "Appendix A". There are three major steps involved, which we will briefly outline here, and then discuss in the rest of this section:

1. The GBN that we are learning consists of a BN for each regime. We shall partition a dataset $\mathcal{D}$ into subsets, treating each subset as observations from a regime, and learn the structure and parameters of a BN for each subset. The first step of the algorithm is to find the partitioning that represents the regime changes that occurred during the collection of $\mathcal{D}$. We will to this end assume that observations within each subset are independent and identically distributed, and that we can calculate the likelihood of the entire model by the product of the likelihoods of the contained BNs.
2. Since we only detect regime changes in the first step, the resulting regime transition structure is a chain of regimes, i.e. no regimes reoccur. However we are also interested in identifying reoccurring regimes, and we will therefore hypothesise mergers of the identified subsets, to find the mergers that lead to the most likely regime transition structure.
3. The final step of the learning algorithm is to introduce gates between the identified regimes, and to optimise the required parameters of these gates. The goal is to be able to use the GBN on a stream of observations, and for each new observation decide which regime the system currently is in.

The rest of this section describes these three steps in detail, in Sect. 5 we will return to the world of baseball, applying the proposed learning algorithm on both synthetic and real-world data.

4.1 Identifying regime changes in the dataset

In order to identify where regime changes have occurred, in a given dataset, we use Metropolis-Hastings MCMC (MH). A typical Bayesian feature selection method is employed, where we have k splits $\delta_1, \ldots, \delta_k$, each defined by their position in the dataset $\beta_1, \ldots, \beta_k$, and an indicator variable $I_1, \ldots, I_k$, that is either 0 or 1. By defining $\delta_i = I_i \beta_i$ the splits can move along the dataset by changing the corresponding β, and be turned on and off by the corresponding I. For a certain configuration of βs and Is we specify a model by learning a BN for each subset of the data defined by the δs that are nonzero. We want to estimate the values of the δs, given the available data, and therefore we are interested in the posterior distribution over the βs and Is given a dataset $\mathcal{D}$ with n observations. That is, we need samples from the posterior distribution in Eq. 3.

$$\begin{aligned} p(\beta_1, \ldots, \beta_k, I_1, \ldots, I_k|\mathcal{D}) &\propto p(\mathcal{D}|\beta_1, \ldots, \beta_k, I_1, \ldots, I_k) \\ U(\beta_1; 0, \beta_2) U(\beta_2; \beta_1, \beta_3) \cdots U(\beta_k; \beta_{k-1}, n+1) p(I_1) \cdots p(I_k) \end{aligned} \tag{3}$$

As is evident from Eq. 3, we will a priori assume that Is are independent, and that βs are distributed according to a discrete uniform distribution, where $U(a; b, c)$ represents a discrete uniform distribution over a with the range (b, c). A Bernoulli distribution is used as prior for each I, which is parameterised with $p = 0.5$ to represent how likely each split is a priori.

4.1.1 Likelihood derivation

In order to sample from the posterior in Eq. 3, we must be able to calculate the marginal likelihood of the data, $p(\mathcal{D}|\beta_1, \ldots, \beta_k, I_1, \ldots, I_k)$. To do this we use the following restructuring:

- Let $\{\gamma_1, \ldots, \gamma_{k'}\}$ represent the subset of $\{\delta_1, \ldots, \delta_k\}$ for which the corresponding $I_1, \ldots, I_k$ are equal to one, i.e. $\{\gamma_1, \ldots, \gamma_{k'}\}$ represents the nonzero δs. We will first assume that $k' > 1$, the two cases of $k' = 0$ and $k' = 1$ will be addressed subsequently.
- Let $\mathcal{D}_{\gamma_i}^{\gamma_j - 1}$ represent observations $d_{\gamma_i}, \ldots, d_{\gamma_j - 1}$ where $i < j$.
- We can then write the marginal likelihood expression as:

$$p(\mathcal{D}|\beta_1, \ldots, \beta_k, I_1, \ldots, I_k) = p\left(\left\{\mathcal{D}_1^{\gamma_1 - 1}, \mathcal{D}_{\gamma_1}^{\gamma_2 - 1}, \ldots, \mathcal{D}_{\gamma_{k'}}^{n}\right\}|\gamma_1, \ldots, \gamma_{k'}\right) \tag{4}$$

- Since we have assumed that a different model holds for each subset, and that we can calculate the marginal likelihood of the entire model by the product of the contained models, we can further rewrite the expression from Eq. 4:

$$\begin{aligned} &p\left(\left\{\mathcal{D}_1^{\gamma_1 - 1}, \mathcal{D}_{\gamma_1}^{\gamma_2 - 1}, \ldots, \mathcal{D}_{\gamma_{k'}}^{n}\right\}|\gamma_1, \ldots, \gamma_{k'}\right) \\ &\quad = p\left(\mathcal{D}_1^{\gamma_1 - 1}|\gamma_1\right) p\left(\mathcal{D}_{\gamma_1}^{\gamma_2 - 1}|\gamma_1, \gamma_2\right) \cdots p\left(\mathcal{D}_{\gamma_{k'}}^{n}|\gamma_{k'}\right) \end{aligned} \tag{5}$$

- We shall use a BN for each subset, thus the full Bayesian approach of evaluating each factor on the right hand side in Eq. 5 would be to sum over all BNs, e.g. $p(\mathcal{D}_1^{\gamma_1-1}|\gamma_1) = \sum_{BN_i \in \mathbf{BN}} p(\mathcal{D}_1^{\gamma_1-1}|BN_i)p(BN_i|\gamma_1)$. However, this is impractical and therefore we shall approximate each factor by using the maximum a posteriori (MAP) BN, obtained by using a learning algorithm L. Let $L(\gamma_i, \gamma_j - 1)$ represent the BN learnt using algorithm L and dataset $\mathcal{D}_{\gamma_i}^{\gamma_j-1}$. Then the MAP approximation[4] to the full Bayesian approach is given by:

$$\begin{aligned} &p(\mathcal{D}|\beta_1, \ldots, \beta_k, I_1, \ldots, I_k) = \\ &p\left(\mathcal{D}_1^{\gamma_1-1}|L(1, \gamma_1 - 1)\right) p\left(\mathcal{D}_{\gamma_1}^{\gamma_2-1}|L(\gamma_1, \gamma_2 - 1)\right) \cdots p\left(\mathcal{D}_{\gamma_{k'}}^{n}|L(\gamma_{k'}, n)\right) \end{aligned} \tag{6}$$

In the special case when $k' = 1$, only the first and last factor of Eq. 6 applies, and when $k' = 0$ we have $p(\mathcal{D}|\beta_1, \ldots, \beta_k, I_1, \ldots, I_k) = p(\mathcal{D}|L(1, n))$.

From Eq. 6 we can deduce that to calculate the marginal likelihood in Eq. 3, we must calculate the marginal likelihood of the subset of data used to learn the corresponding BN. For discrete BNs there exists a closed form expression to exactly calculate the marginal likelihood of a BN structure via the Bayesian–Dirichlet equivalent (BDe) score (Heckerman et al. 1995) (where the parameters of the marginal and conditional distributions have been marginalised out, assuming a conjugate Dirichlet prior, and it is assumed that observations are independent and identically distributed).

4.1.2 Proposal distributions

The MH simulation is initialised by setting all βs so that the δs are evenly spaced out across the dataset, and all Is are set to 1. Proposing new values for the indicator variables I is done via a Bernoulli distribution with $p = 0.5$, so that it is equally likely that the I will change or that it will stay the same. The β values require a bit more work, as there are constraints on how we want them to be proposed.

First of all, the βs have to be proposed within the bounds of the dataset, i.e. between 1 and n. Secondly, we want $\beta_i < \beta_j$ for all $i < j$, so that two βs never collide or jump over each other. Finally, we want the βs to have a positive probability of taking on any value between their respective upper and lower bounds. Since the βs are positions in a dataset they are discrete values, thus we can create a proposal distribution for each β according to Eqs. 7 through 11. In the equations, β_j is the current value while β_j^* is the proposed value. In Eqs. 7 and 8 we define the upper and lower bound for β_j^*, in such a way that β_j^* always is proposed within the dataset, and so that β_j^* never can be proposed as the same value as any of the other βs. In Eqs. 9 and 10, each allowed value i of β_j^* is given a probability that is proportional to its distance from the current value β_j. Z is a normalisation constant defined in Eq. 11.

[4] In the current setting, using the MAP approximation is rather convenient, since it gives us a BN for each regime, and it is these BNs that combine into the full GBN.

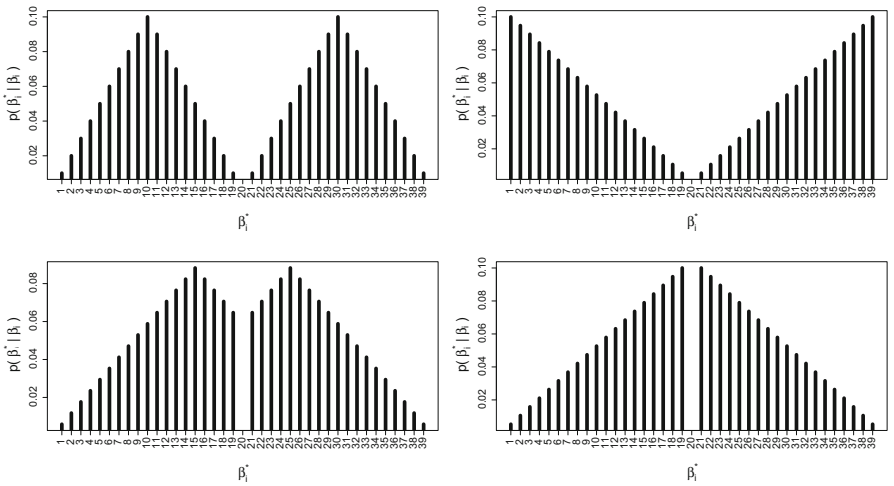


Fig. 2 Example proposal distributions for βs

$$lb(\beta_j^*) = \begin{cases} 1 & \text{if } j = 1 \\ \beta_j - \left\lfloor \frac{1}{2}(\beta_j - \beta_{j-1}) \right\rfloor - 1 & \text{otherwise} \end{cases} \tag{7}$$

$$ub(\beta_j^*) = \begin{cases} n & \text{if } j = k \\ \beta_j + \left\lfloor \frac{1}{2}(\beta_{j+1} - \beta_j) \right\rfloor - 1 & \text{otherwise} \end{cases} \tag{8}$$

$$\kappa = \max(\beta_j - lb(\beta_j^*), ub(\beta_j^*) - \beta_j) \tag{9}$$

$$p(\beta_j^* = i|\beta_j) = \begin{cases} \frac{1}{Z}(1 + i - \beta_j + \kappa) & \text{for } lb(\beta_j^*) \le i \le \beta_j \\ \frac{1}{Z}(1 - i + \beta_j + \kappa) & \text{for } \beta_j < i \le ub(\beta_j^*) \end{cases} \tag{10}$$

$$Z = \sum_{i=lb(\beta_j^*)}^{\beta_j} (1 + i - \beta_j + \kappa) + \sum_{i=\beta_j+1}^{ub(\beta_j^*)} (1 - i + \beta_j + \kappa) \tag{11}$$

To visualise this, assume that there are two βs and $n = 39$. In the top left plot in Fig. 2 the current values are $\beta_1 = 10$ and $\beta_2 = 30$. As can be seen, β_1^* and β_2^* can be proposed in both directions within their respective bounds. In the top right plot the two βs are positioned at 1 and 39 respectively, and are now constrained to move in only direction due to the dataset bounds. In the bottom left plot the two βs are positioned at 15 and 25, and points that they have in common in their range are removed from both proposals. The bottom right plot shows the case where the βs are positioned at 19 and 21, and there is no probability of the βs moving closer to each other.

4.1.3 Iterations and samples

While it is possible to monitor the MH iterations to decide when to stop sampling, we run the simulation for a fixed number of iterations and throw away the first half of the

samples (treating them as burn-in samples). We use the mean of the marginal of each scalar parameter β and I and round to integer values, i.e. if the marginal mean of I_j is 0.4 we will round this to 0 and if it is 0.6 we will round it to 1. The resulting nonzero δs then identify where in the dataset regime changes have occurred.

4.2 Identifying regimes and structure

Having identified nonzero δs where regime changes have occurred in a dataset $\mathcal{D}$, the next step is to identify the individual regimes, as well as the regime transition structure of the underlying system. Naïvely assuming that each change identifies a new regime would lead to a chain of regimes, i.e. if there were two nonzero δs identified we would have $R_1 \rightarrow R_2 \rightarrow R_3$. While it may be entirely possible for a system to exhibit this type of regime structure, it is also possible that R_2 transitioned back to R_1, and not into a new regime R_3. Therefore we must hypothesise each possible recursive merging of nonadjacent subsets, as defined by the nonzero δs, and score a new model based on these new subsets. Follows does an explanation and example of this merging.

4.2.1 Merging subsets

In the example in Fig. 3 we have identified three nonzero δs, resulting in four subsets of the data (labeled 1, 2, 3 and 4). The first hypothesis requires no merging at all; it suggests that each subset identifies a new regime (depicted to the left) and the regime transition structure is therefore a chain of four regimes (depicted to the right). From the first hypothesis we cannot merge subsets 1 and 2, since they are adjacent and we would not have a nonzero δ here if the two subsets belonged to the same regime. A new hypothesis can however be constructed by merging subsets 1 and 3, resulting in the second hypothesis in the figure. Note that we have now labelled subset 3 with R_1 and subset 4 with R_3, as we now only have three regimes rather than four in the previous hypothesis. Should this hypothesis be true, then the regime transition structure must also reflect that from R_1 it is possible to transition to R_3, and from R_2 it is possible to transition to R_1. Thus when two subsets are merged, we also merge the parent and child sets in the regime transition structure. From the second hypothesis the only

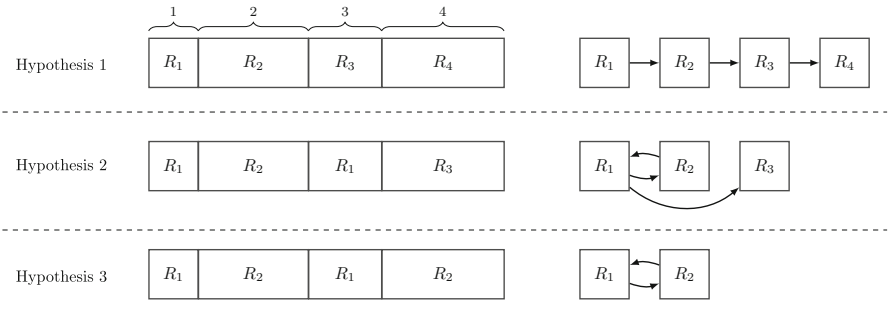


Fig. 3 Example of merging subsets

merging that can be done is to merge subsets 2 and 4, resulting in the third hypothesis. Note that the example is not complete, as we should go back to the first hypothesis and start the recursive procedure again, but this time by merging subsets 1 and 4 (and similarly so for 2 and 4).

In order to identify the regime structure of a system $\mathcal{S}$, we start by hypothesising the chain of regimes that is given directly from the nonzero δs, and then continue to hypothesise each possible recursive merging. For each hypothesis, we merge the subsets of $\mathcal{D}$ accordingly, and learn a new set of BNs using these merged subsets. The learnt BNs are scored according to their marginal likelihood (a simple rephrasing of Eq. 6). The hypothesis with the highest marginal likelihood, among all the hypotheses, represents the identified regime structure of $\mathcal{S}$. While merging, previously scored hypotheses may be created, so the number of hypotheses to score can be substantially pruned. To see this we can continue the merging example from before. This time we start by merging subsets 2 and 4, resulting in subset labelling R_1, R_2, R_3, R_2, and then we merge subsets 1 and 3, resulting in R_1, R_2, R_1, R_2, which is the same as the last hypothesis when we started by merging 1 and 3.

The number of hypotheses to score is directly connected to the number of nonzero δs identified. If there are none, or only one, nonzero δ then there is no merging possible, resulting in a single hypothesis. If there are two nonzero δs there are two hypotheses to score, and with three nonzero δs there are five hypotheses to score. If there are k' nonzero δs, then the number of hypotheses to score equals the number of partitions of $\{0, \ldots, k'\}$ into subsets of nonconsecutive integers (including the partition where each element belongs to its own subset, which we would interpret as a chain of regimes). Following from Theorem 2.1 in Munagi (2005), the number of hypotheses to score therefore follow the well known Bell numbers (A000110 in OEIS), where the first ten numbers are: 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147. By starting with the partition where each element belongs to its own subset, and then recursively merging two subsets (without violating the constraint that the subsets must not contain consecutive numbers), we will visit each allowed partition, thus each possible hypothesis will be scored. While at some point the computational effort required to score hypotheses may exceed current limitations, regime changes are infrequent in the current setting, keeping the effort within the feasible range.

4.2.2 Markovian assumption

Merging the parent and child sets of the regime transition structure is only valid if we assume that the transitions are first order Markovian. For instance, in the second hypothesis in Fig. 3 we assume that the system can transition from R_1 to R_3, however without the Markovian assumption the data suggests that the system can transition from R_1 to R_3 only if it has transitioned to R_2 previously. If the system only allowed transitions from R_1 to R_3 after it had passed through R_2, then the system would be somehow different depending on if it had gone through R_2 or not. It would then be arguable that R_1 prior to R_2 would not be the same as R_1 after R_2, and therefore they would not truly be identical regimes. Hence, it is reasonable to assume this Markovian

property of the regime transition structure, allowing us to identify different regime transition structures depending on the most likely hypothesis.

4.3 Constructing a GBN

Having identified the regime transition structure, and the BNs that represent each regime, the final step to complete the model of the system is to construct a GBN and configure the gates. The goal is to accept a stream of observations, and for each new observation decide which regime the system currently is in. In each gate we define a comparison between the parent and the child BN in such a way that if the child is preferred over the parent, then the gate triggers, thereby deactivating the parent and activating the child. This comparison is a likelihood ratio between the parent and child using a moving window of the most recent τ datapoints. If this ratio is above some threshold θ, then the gate triggers. Let $\mathcal{O}$ be a stream of observations and $\mathcal{O}_\tau$ represent the τ most recent observations, then $p(\mathcal{O}_\tau|R_1)$ is the likelihood of the most recent τ observations given regime R_1. Each gate is then configured to trigger when $p(\mathcal{O}_\tau|R_{child})/p(\mathcal{O}_\tau|R_{parent}) > \theta$.

For instance, assuming that we have identified the regime structure $R_1 \rightarrow R_2 \leftrightarrows R_3$, we construct the GBN in Fig. 1, where the square nodes represents BNs for each regime. For this GBN we define the trigger logic in Eq. 12, where we use the same τ and θ for all gates.

The remaining task, which we will describe in Sect. 4.3.1, is to choose appropriate values for τ and θ. The task requires the optimisation of a score function, and in the current setting there are two major factors that we need to consider. First, the function that we wish to optimise is a *black box*, and second, evaluating the function is expensive. As we cannot compute gradients analytically we would have to sample them, which would require several evaluations of the function during each iteration (which we would like to avoid). The Bayesian approach is to go from a prior to a posterior utilising observations that we collect, hopefully identifying the global maximum of the function. However, this implies being able to compute posteriors from priors, but since we cannot describe our function in closed form, we cannot produce an expression for the posterior. A non-parametric approach (i.e. one that does not depend on the shape of the function) is to use a Gaussian process (GP) to act as a proxy for the function. This derivative free technique, commonly known as Bayesian optimisation (Brochu et al. 2009), has become a popular way of optimising expensive black box functions, and we will now turn our attention to how we shall adopt this framework for the task of choosing τ and θ.

$$\begin{aligned} TL(G_1) &:= \frac{p(\mathcal{O}_\tau|R_2)}{p(\mathcal{O}_\tau|R_1)} > \theta \\ TL(G_2) &:= \frac{p(\mathcal{O}_\tau|R_3)}{p(\mathcal{O}_\tau|R_2)} > \theta \\ TL(G_3) &:= \frac{p(\mathcal{O}_\tau|R_2)}{p(\mathcal{O}_\tau|R_3)} > \theta \end{aligned} \tag{12}$$

4.3.1 Gaussian processes and Bayesian optimisation

Ultimately, we would like to find the parameter pair $\Lambda = \{\tau, \theta\} \in \Lambda$ that maximises an objective function f. In the current setting, f is the accuracy of the predictions of the current regime (we will give more attention to the details of f in Sect. 4.3.2). While we can evaluate f at a given point Λ, we do not have a closed form expression for f, thus we cannot solve the problem analytically. Instead, we will introduce a random variable g to act as a proxy for f, and place on g a prior over all possible functions. This is done by using a GP, which is defined as a distribution over an infinite number of variables, such that any finite subset of these variables are jointly multivariate Gaussian. Thus, for each Λ we treat $g(\Lambda)$ as a Gaussian random variable, where $f(\Lambda)$ is a realisation of the random variable $g(\Lambda)$, and treat a set of such random variables as jointly multivariate Gaussian.

Formally, the GP is parameterised by a mean function $\mu(\Lambda)$, and a covariance kernel $k(\Lambda, \Lambda')$, which are defined over all input pairs $\Lambda, \Lambda' \in \Lambda$. The mean function $\mu(\Lambda)$ represents the expected value of $g(\Lambda)$, and it is commonly assumed to be zero for all $\Lambda \in \Lambda$, although this is not necessary if prior information exists to suggest otherwise. The covariance kernel $k(\Lambda, \Lambda')$ represents how much the random variables $g(\Lambda)$ and $g(\Lambda')$ covary, thus it defines how smooth the function f is thought to be, and can therefore be tuned to ones prior belief about f. For instance, by using the *radial basis kernel* in Eq. 13, we can tune c to fit our prior belief about smoothness.

$$k(\Lambda, \Lambda') = \exp(-c||\Lambda - \Lambda'||^2) \tag{13}$$

For points close to each other, Eq. 13 will result in values close to 1, while points further away will be given values closer to 0. The GP prior will obtain the same smoothness properties, as the covariance matrix is completely defined by k. To visualise the smoothness achieved by tuning c, Figure 4 shows the decreasing covariance as distance grows with three different settings of c (0.5, 1.0 and 2.0). As can be seen, as c increases the decrease is faster, thus less smoothness is assumed.

Let $\Lambda_{1:i}$ represent i different Λs, without any specific ordering, and let $f_{1:i}$ represent the value of f at these Λs, i.e. $f_j = f(\Lambda_j)$. Similarly, let g_j represent the random

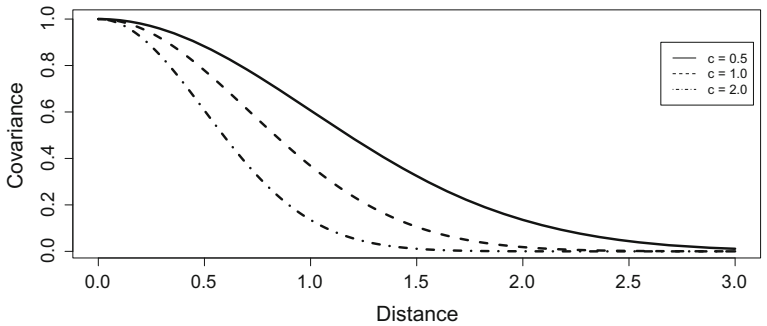


Fig. 4 Covariance decreases by distance

variable $g(\Lambda_j)$. Having collected observations $\{\Lambda_{1:i}, f_{1:i}\}$, we can calculate the posterior distribution of g for some new input Λ_{i+1}, where both the prior smoothness and the observed data have been considered. A closed form expression exists for this calculation as described in Eq. 14. Notice that Eq. 14 is the usual equation to compute the conditional density function of X given an observation for Y, when X and Y are jointly Gaussian. For more on GPs, please see Rasmussen and Williams (2006).

$$
\begin{aligned}
&\begin{bmatrix} g_{1:i} \\ g_{i+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_*^{\mathbf{T}} \\ \mathbf{K}_* & \mathbf{K}_{**} \end{bmatrix}\right) \\
&\mathbf{K} = \begin{bmatrix} k(\Lambda_1, \Lambda_1) & \cdots & k(\Lambda_1, \Lambda_i) \\ \vdots & \ddots & \vdots \\ k(\Lambda_i, \Lambda_1) & \cdots & k(\Lambda_i, \Lambda_i) \end{bmatrix} \\
&\mathbf{K}_* = \begin{bmatrix} k(\Lambda_{i+1}, \Lambda_1) & \cdots & k(\Lambda_{i+1}, \Lambda_i) \end{bmatrix} \\
&\mathbf{K}_{**} = k(\Lambda_{i+1}, \Lambda_{i+1}) \\
&p(g_{i+1} | \{\Lambda_{1:i}, f_{1:i}\}) = \mathcal{N}(\mu_i(\Lambda_{i+1}), \sigma_i^2(\Lambda_{i+1})) \\
&\mu_i(\Lambda_{i+1}) = \mathbf{K}_* \mathbf{K}^{-1} f_{1:i} \\
&\sigma_i^2(\Lambda_{i+1}) = \mathbf{K}_{**} - \mathbf{K}_* \mathbf{K}^{-1} \mathbf{K}_*{}^T
\end{aligned}
\tag{14}
$$

Since g is acting as a proxy for the objective function f, using a GP allows us to encode prior beliefs about the objective function, and sampling the objective function allows us to update the posterior over objective functions. In the framework of Bayesian optimisation (Brochu et al. 2009), we utilise the posterior over objective functions to inform us of how to iterate between sampling and updating, in order to find the Λ that maximises the objective function f. The next Λ for which to evaluate f is the Λ that maximises an *acquisition* function. Several acquisition functions have been suggested, however the goal is to trade off exploring areas where the posterior uncertainty is high, while exploiting points that have a high posterior mean. We will use the *upper confidence bound criterion*, which is expressed as $UCB(\Lambda) = \mu(\Lambda) + \eta\sigma(\Lambda)$, where $\mu(\Lambda)$ and $\sigma(\Lambda)$ represent the posterior mean and standard deviation of $g(\Lambda)$, and η is a tuning parameter to allow for more exploration (as η is increased) or more exploitation (as η is decreased). Once a predetermined number of iterations m have passed, the Λ_i for which f_i is greatest in $\{\Lambda_{1:m}, f_{1:m}\}$ is the set of parameters that maximises the objective function.

4.3.2 Evaluating accuracy

In our case, the objective of Bayesian optimisation is to find the pair τ and θ that maximises the *accuracy* of the GBN. We say that the GBN is correct if, after processing a new observation in a stream of data (see Sect. 2.3), the active BN corresponds to the true BN that generated the observation. Thus, if an observation in a stream truly came from R_1 and the GBN has R_1 active after processing the observation, then this is considered correct, while any other situation is considered incorrect.

Since we only have access to the training dataset during the elicitation process, there is a risk that we may overfit this data. In order to reduce this risk we resample the observations in each identified subset of the original data, and concatenate the resamples to create a new dataset of the same size as the original one. Doing so several times we synthetically create new datasets from the original one. As a result of the resampling we can calculate the average accuracy over all resampled datasets for a pair of τ and θ. Thus, the objective function f that we aim to maximise is therefore the average accuracy of a pair τ and θ over all the resamples. Notice that we do not use the true regime labels from the training data, since we assume that these are not known to us, but rather add regime labels based on the regime transition structure identified during subset merging.

Given a GBN with all gates parameterised with some values τ and θ, and a stream of data $\mathcal{O}$, where each observation is associated with a regime label, the accuracy of the GBN is calculated as follows:

- Activate the BN that represents R_1, deactivate all others.
- For the first $\tau - 1$ observations from $\mathcal{O}$ do not process any observations, the current regime according to the GBN will be R_1 for all of these observations. Having less than τ observations does not allow us to compute the values in Eq. 12.
- Process each observation beyond the first $\tau - 1$ observations, the active BN for that observation represents the current regime according to the GBN.
- When no more observations are available, count the number of observations for which the GBN had the correct BN active.
- The accuracy of the GBN is the number of times the GBN was correct divided by the number of observations in the stream $\mathcal{O}$.

5 Regimes in baseball players' career data

Having defined the problem and introduced the intended model in Sect. 2, as well as introduced a learning procedure for this model in Sect. 4, we now turn our attention back to baseball. Our goal is to gain insight into what a GBN may tell us regarding a baseball player's career, specifically via the regime transition structure. As was mentioned in Sect. 1, the cause of a regime transition may be evident, for which public information is available, such as injuries or trades. However, if the data suggests a regime transition at a point where there is no obvious reason to why the player is playing differently, then this may be of value for the managers and coaches. Consider a player that was performing extraordinarily well during a past regime, but has now transitioned back to a more normal level of performance. It may be of great value for the coaches to identify when the transitions in and out of the extraordinary regime happened, and then go back and analyse what may have caused them. Managers and coaches have access to data that is not made public, such as practice, exercise and strategic decisions. From this data they may be able to identify what it was that caused the player to transition to the extraordinary regime, and may be able to arrange for the conditions for this to happen again.

We are therefore interested in discovering whether or not there are regimes in the data, and if there are regimes, then how many of these can we accredit to evident

events, and how many of these are not explained by publicly available data. We will provide a summarised view of the analysis of 30 players, followed by a more in-depth view of two of these. We will however begin this section with a brief explanation of the game of baseball, followed by a description of the available data and procedure configuration, and then end this section by reporting and discussing our results.

5.1 The game of baseball

The official major league baseball rules are documented in a publication consisting of 282 pages, and is revised yearly for clarifications, removals and amendments. Here we will attempt to give only the most basic understanding of how the game of baseball is played, there are many more finer details and exceptions that we will not cover. We will on occasion refer to the stylised representation of a baseball field offered in Fig. 5, the figure is an abstract representation and not to scale.

The game of baseball is played between two teams over nine *innings*, where in each inning the teams are given the opportunity to play both offense and defense. The team playing offense is able to score *runs*, and the team with the most runs at the end of the nine innings is considered the winner. Extra innings are added if both teams have the same number of runs at the end of regulation play, there are no draws.

The main component of the game is a *pitcher* (positioned at 1 in the figure) that pitches (i.e. throws) the ball to the *catcher* (positioned at 2), both are on the defensive team. The *batter*, part of the offensive team, stands between the pitcher and catcher (at position 3) and tries to hit the ball. The remaining defensive players are positioned strategically on the field. The defensive team needs to get three batters *out* in order to end their defensive part of the inning.

There are several outcomes that may take place when the ball is pitched, we describe the relevant outcomes here:

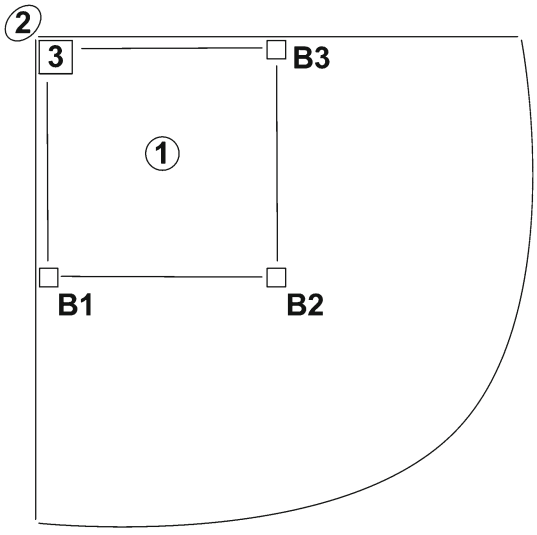


Fig. 5 Stylised representation of a baseball field

- Assuming that the batter does not swing the bat at all, a ball that is pitched to the catcher can count as either a *strike* or a *ball*. If the baseball passes the batter within the *strike zone* then the pitch is counted as a strike, if it is outside the strike zone then it is counted as a ball. The strike zone is an imaginary volume of space that extends from the batters knees to the midpoint of the batters torso. Simply put, it is a zone in which the batter has a fair chance of hitting the baseball. Whether or not the baseball is within the strike zone is decided by an umpire positioned behind the catcher.
- If the batter swings at the baseball and misses then it counts as a strike, even if the pitch was outside the strike zone.
- If the strike count goes to three, then the batter is out. If the ball count goes to four, then the batter can freely move to first base (positioned at B1 in the figure), this is known as a *walk*.
- If the batter hits the baseball then there are two main outcomes. The ball can either fly into the air and be caught by a defensive player before hitting the ground, at which point the batter is *out*. Alternatively, the ball can hit the ground in which case the batter becomes a runner, and needs to reach first base before a player in the defensive team touches first base while holding the baseball. The runner (previously known as the batter) is also allowed to attempt to advance to second or third base (2B and 3B in the figure), but will be out if a player on the defensive team manages to touch the runner while holding the ball. If caught then the runner is out, if safe at a base then the play is over and the next player in the offensive team becomes the batter. Each time an offensive player is allowed an attempt to bat it is counted as a *plate appearance*. If there was an offensive player already on a base when the batter hit the ball, then they can advance further (sometimes they are forced to advance).
- Each time the offensive team gets a runner all the way around the bases, i.e. running through B1, B2, B3 and back to where the batter is positioned at 3 in the figure, then the offensive team scores a run.
- If the batter hits the baseball outside the field, e.g. into the stands or even outside the arena itself, then it is called a *home run* and the batter can freely run through all bases and come back home, thereby scoring a run. Any other offensive players that were already safe at one of the bases can also run home, thus scoring a run for each player that comes home.
- There are several other reasons why a batter may freely move to first base, for instance if the pitcher hits the batter with the baseball or the defensive team makes a rule error.

There is a myriad of statistics developed that aim at summarising a players performance. We will use a statistic developed by Bill James, known as the somewhat cumbersome *on-base plus slugging* (OPS) statistic which is calculated by Eq. 15. When a batter hits the ball and makes it to first base, it counts as a *single*, if the batter makes it to second base it is called a *double*, a *triple* represents making it to third base and home runs are home runs. As is evident from the calculation of *slugging* (SLG) there is a weighting on how much the outcomes are worth, and the linear combination is divided by the number of *at-bat*s. While the number of plate appearances does

Table 1 Discretisation of OPS statistics

Label	Description	OPS range
A	Great	$[0.9000, +\infty)$
B	Very good	$[0.8333, 0.9000)$
C	Above average	$[0.7666, 0.8333)$
D	Average	$[0.7000, 0.7666)$
E	Below average	$[0.6333, 0.7000)$
F	Poor	$[0.5666, 0.6333)$
G	Atrocious	$[0, 0.5666)$

not equate exactly to the number of at-bats, it can be thought of as the same for the purposes of this paper. If a player hits either a single, double, triple or home run it counts as a *hit*, which is used in the calculation of *on-base percentage* (OBP). A walk is as described earlier, and *hit by pitcher* happens when the pitcher hits the batter with the baseball. A *sacrifice fly* is a strategic option where the batter on purpose hits the ball far and high, making it easy for the defensive team to catch it before it hits the ground thus giving them an out, but an offensive player safe on a base may make it back to score a run (thus the batter is sacrifised). The OPS statistic aims to represent an offensive player's skill, however does not take into account the player's running abilities (fast players are not credited by an increased OPS). Bill James also suggested a discretisation of OPS, presented in Table 1.

Apart from OPS we will also make use of the less complex *runners batted in* (RBI) statistic, which is a count of how many runs a batter is responsible for, e.g. if the offensive team has one runner safe on third base and the batter hits the ball allowing the runner to make it back home, then it counts as a run for the offensive team and a RBI for the batter.

$$\begin{aligned} SLG &= \frac{single + 2 * double + 3 * triple + 4 * home\ run}{at\text{-}bat} \\ OBP &= \frac{hit + walk + hit\ by\ pitcher}{at\text{-}bat + walk + sacrifice\ fly + hit\ by\ pitcher} \\ OPS &= OBP + SLG \end{aligned} \tag{15}$$

Following the above definitions, we here present the variables that we extracted for our purposes:

- Outcome—The outcome of the event, with states: single, double, triple, home-run, walk, out or reaching first base for another reason.
- OPS—A 30 day rolling window calculation of OPS, discretised according to Table 1.
- RBI—A 30 day rolling window calculation of RBI, discretised into three equal width bins.
- B1—Whether or not there is a runner on first base (true or false).
- B2—Whether or not there is a runner on second base (true or false).
- B3—Whether or not there is a runner on third base (true or false).

- Home—Whether or not the offensive team is the home team (true or false).
- Arm—Whether or not the pitcher throws with the left or right arm (left or right).
- Outs—The number of outs in the inning (0, 1 or 2).
- Balls—The ball count (0, 1, 2 or 3),
- Strikes—The strike count (0, 1 or 2).

5.2 Datasets

Retrosheet is an organisation founded in 1989 (www.retrosheet.org) that digitally stores events of major league baseball games. Each event describes the outcome of a plate appearance, and the conditions under which the event happened. There are over 8900 players in the Retrosheet database, for which a complete analysis was not within our aims of this paper. Instead we defined a subpopulation of all these players, and then sampled from this subpopulation. We filtered the event data in such a way that we were left with players that had their major league debut during the 2005 season or later, and with the criteria that they had to have at least 2000 event data entries. The range of number of events in this subpopulation was 2001 to 6614. This subpopulation consisted of 150 players, from which we uniformly sampled 30 players.

Since we do not know exactly when regime transitions have occurred in the career data of the 30 sampled players, we cannot use this data to show how well the proposed learning algorithm identifies the location of regime transitions. Therefore we also created synthetic datasets representing fictional careers, for which we knew when the transitions occurred. We created a BN, using the variables introduced in Sect. 5.1, and then manipulated this BN slightly to create another BN, representing a new regime. We continued this manipulation until we had five different BNs, representing five different regimes in a fictional baseball player's career. We then arranged these BNs into three different regime transition structures, and sampled data from these structures. For each structure, one sample was used as learning data, and 100 samples was held out as testing data. The testing data was used to estimate the accuracy of the learnt GBN, i.e. measuring whether or not the GBN had the correct BN activated given the testing data, as described in Sect. 4.3.2. For full details of the BNs and the transition structures, please see "Appendix B".

5.3 Method

During MH we used $k = 8$ splits, thus we are not assuming any knowledge about how many splits there truly are in the data, only that there are a maximum of eight. We ran 100,000 iterations, throwing away the first 50,000 samples as burn-in. A greedy thick-thinning algorithm for learning the BN structures was used (Heckerman 1995), where the log marginal likelihood was the target to improve.

We ran 250 iterations of Bayesian optimisation using the radial basis kernel with $c = 2.0$, and the upper confidence bound criterion acquisition function using $\eta = 5$. The objective function was the average accuracy over 1000 resamples.

5.3.1 Synthetic data

For the synthetic data we used the generated learning datasets to learn a GBN for each transition structure, and then used the 100 testing datasets to compute the accuracy of the learnt GBNs, following the description in Sect. 4.3.2. We shall report the location of the identified regime transitions, and compare them with the true locations, as well as the average accuracy of the GBNs over the test datasets. For comparative reasons, we also learnt a standard HMM using each one of the training datasets, and calculated the state probabilities given all data. Since we knew how many regimes there were in the synthetic datasets, we fixed the number of states during the learning of the HMM, rather than employing a cross-validation scheme to also learn this number. We however emphasise that no such advantage was given to the learning of GBNs.

5.3.2 Real world data

In order to count the number of regime transitions for which there is an evident cause, we created the following three criteria. If a regime transitioned happened in conjunction with an event for which any of the criteria are satisfied, then the cause was considered evident:

1. An injury to the player—We expect a player to play differently a period after an injury, thus if a regime transition coincides with an injury we count it as a transition where the cause is known.[5]
2. The player being traded—When a player moves to a new team there is a good chance that the player needs to learn a new system, new coaching and may need time to adjust. While the trade itself may not be the cause, a transition in conjunction with a trade is to some degree explained by the trade[5].
3. The beginning of a new season—A new season usually brings new players to a team, sometimes new managers and coaches, new strategies are put in place, etc. A regime transition early in the season is therefore considered to be explained by these changes.

From these criteria it should be understood that transitions not counted towards the evident reasons are those where a transition occurs mid season, without a major injury or trade. The reason for such a transition may be clear for the managers and coaches, since they have access to data that the public has not, however they may also be unaware of these transitions, and identifying the transitions allows for analysis of what may have caused them. It is precisely here that our procedure adds value to the decision making by managers and coaches, discovering transitions that may have been unintentional or the result of multiple events and decisions.

5.4 Results and discussion

We will divide this section into four parts. In the first part we will account for the performance of the GBN learning using the synthetic datasets. In the second part we

[5] Information about players' injuries and trades were sourced from their respective Wikipedia pages.

will give a summary view of the GBNs that were learnt for the 30 sampled players, and then immediately discuss our findings in relation to this sample. Thereafter we will look at two players in more detail. This detailed analysis allows us to show the GBNs that were learnt, as well as discuss the regime transitions which we can explain and which we cannot. Finally, we will end this section with a discussion regarding the representational power of GBNs versus BNs.

5.4.1 Synthetic data

In Table 2 we present the true regime transition locations in the synthetic data, the locations identified using the proposed learning algorithm, and the difference between the two. The three structures are the ones presented in "Appendix B". As is evident from the table, the proposed learning algorithm identifies well the true locations, as the differences are relatively small. The merging procedure of the learning algorithm was also able to correctly identify the regime transition structures in all three cases. The standard HMM that we also learnt using the synthetic data was however not as successful, the results for which we defer to "Appendix C".

Turning our attention to Table 3, we present the optimised τ and θ values for each structure, along with the average and minimum accuracy achieved over the 100 test datasets. As is evident, the average accuracy is high, however since the minimum for Structure 2 was lower, we also counted the number of times the accuracy was below 0.9 and 0.8. From the table we can tell that such occurrences were few.

All in all, these results show that the proposed learning algorithm is capable of correctly identifying regime transition structures, and that the resulting GBN can be used to accurately identify the current regime. This is consistent with a more extensive evaluation on synthetic data that we have completed (Bendtsen 2017).

Table 2 Location of identified regime transitions during GBN learning using synthetic data

	True locations				Identified locations				Difference			
Structure 1	590	977	1360	1933	574	990	1337	1931	16	13	23	2
Structure 2	364	823	1321	1683	355	809	1322	1711	9	14	1	28
Structure 3	351	884	1223	1757	337	886	1212	1743	14	2	11	14

Table 3 Accuracy of learnt GBNs

	τ	θ	Average	Minimum	# <0.9	# <0.8
Structure 1	10	1.08	0.98	0.98	0	0
Structure 2	13	1.18	0.94	0.77	14	1
Structure 3	10	1.07	0.97	0.82	7	0

Table 4 Summary view of the GBNs learnt for a sample of baseball players

Player	$\delta s \neq 0$	BNs	Cycles	Explained/unexplained	τ	θ
Ike Davis	5	4	T	2/3	26	1.10
Nate McLouth	5	4	T	2/3	18	1.04
Chris Denorfia	3	3	T	1/2	19	1.07
Jose Altuve	5	5	T	2/3	10	1.13
Ian Desmond	4	3	T	1/3	17	1.11
Desmond Jennings	6	5	T	1/5	16	1.08
Nyjer Morgan	6	4	T	3/3	15	1.15
Matt Wieters	3	3	T	1/3	32	1.05
Kendrys Morales	5	4	T	2/3	45	1.02
Melky Cabrera	6	5	T	4/2	50	1.02
Carlos Quentin	5	4	T	4/1	47	1.01
Freddie Freeman	5	4	T	1/4	11	1.08
Nick Hundley	3	3	T	2/1	24	1.04
Alberto Callaspo	5	4	T	2/3	17	1.03
Jason Kipnis	5	4	T	1/4	11	1.08
Buster Posey	4	4	T	1/3	43	1.04
Ryan Doumit	3	4	F	1/2	27	1.06
Jason Heyward	6	4	T	3/3	10	1.07
Michael Brantley	3	4	F	2/1	10	1.14
Troy Tulowitzki	4	3	T	3/1	25	1.02
Kyle Seager	5	5	T	1/4	10	1.10
Mark Teahen	5	5	T	2/3	37	1.85
Will Venable	4	3	T	1/3	10	1.06
Daric Barton	4	4	T	1/3	18	1.09
Gerardo Parra	4	4	T	1/3	10	1.08
Gregor Blanco	4	4	T	1/3	15	1.07
Rajai Davis	2	2	T	1/1	10	1.06
Alex Avila	4	4	T	2/2	43	1.00
Adam Lind	5	4	T	4/1	45	1.06
Kevin Kouzmanoff	3	3	T	1/2	16	1.04

5.4.2 Summary view

For each baseball player analysed we present a summary of the GBN learnt in Table 4. In the table we first present the player's name, followed by the number of nonzero δs identified, i.e. the number of transitions. The table continues with the number of BNs contained in the GBN, and a Boolean value representing whether or not there were reoccurring regimes in the GBN, i.e. if the GBN contains directed cycles. The third to last column is the number of regime transitions for which we could

find an evident cause, and the number for which we could not. The last two columns are the τ and θ values determined during optimisation of the gate parameters.

The first observation we make is the fact that there certainly are regimes in the data, using a single BN to represent a player's career is not preferred for any of the players. What is further interesting is that, for all but two players, the regimes are not structured in a chain, but the regimes rather reoccur. While we expect any sportsperson to change the way they play their respective game throughout their career, be it due to increasing skill or age, it is not necessarily obvious why we find that some players revert back to previous regimes. Though highly speculative at this point, transitioning into one regime and then transitioning back may be due to strategic decisions by the manager or coach, or due to injury, since players may be playing more cautiously while receiving treatment.

There certainly is an overweight towards regime transitions which we could not explain from our superficial investigation. While we are not ruling out that there may be publicly available sources of information that may be helpful in order to explain the remaining transitions, it is noteworthy that there is reason for further investigation. As we shall see in Sect. 5.4.3, some of these transitions may be highly sought after (e.g. transitioning into a regime where a player is playing extraordinarily well), thus for the managers and coaches it can be essential to look at the decisions that were made at the time of the transition.

The last thing we wish to comment on regarding this summary view are the τ and θ values. In the synthetic experiments, reported in Sect. 5.4.1, we saw relatively low values for τ in all three cases. However, in a previous publication we ran the learning algorithm on a more extensive set of synthetic datasets (Bendtsen 2017). These experiments showed that the algorithm produced GBNs with high accuracy, and the average τ value was 18.2 (with a range of [10,39]). As is evident from the last two columns in Table 4, the τ values for the GBNs learnt in these experiments are mostly within this range. What this tells us is, that if the GBNs were to be used on new data, for instance the 2016 season, they would not require an unusually large window (τ) to detect regime changes, and the required threshold (θ) is comparable to what we have shown had good accuracy on synthetic data. An unusually large window could imply that the GBN would be detecting regime changes much later than they occurred, thus the use of the GBN as a detection model would have been limited. So while we do not explore this opportunity further in this paper, the learnt GBN is a model that can be used to detect if the player has transitioned to any of the identified regimes. We imagine that a coach could learn a GBN using data from previous seasons, from this GBN identify what the causes were for a favourable regime transition, put these causes in place and then monitor the player by entering the new data that the player is generating into the GBN. If the GBN does transition into the intended regime, then the coach has succeeded.

5.4.3 Detailed analysis

In order to gain further understanding of how the regime changes in the players' data can be used, we picked out two players that served as good examples for how a manager or coach could do a further analysis. For these players, Nyjer Morgan and Kendrys

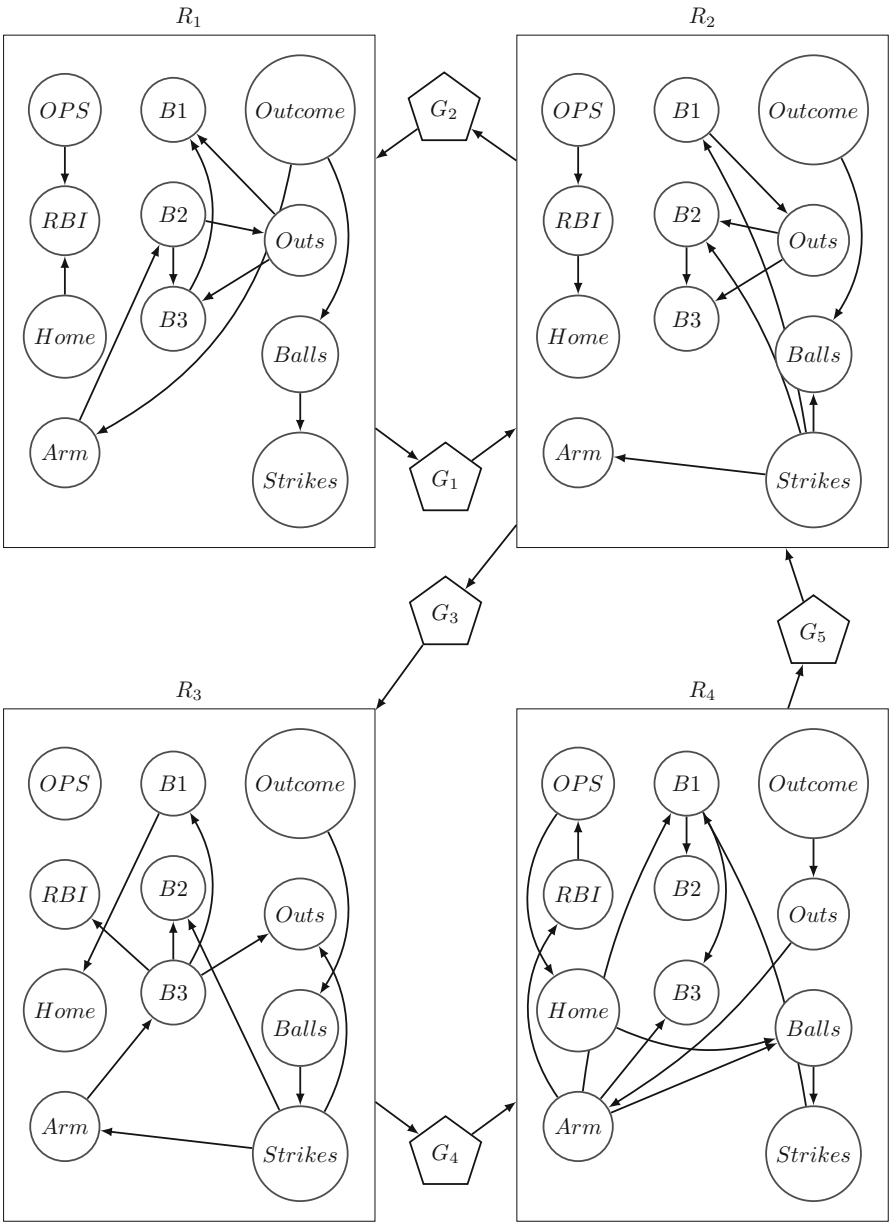


Fig. 6 GBN learnt for Nyjer Morgan

Morales, we present the GBNs learnt in Figs. 6 and 8. In Figs. 7 and 9 we present representations of the subsetting of the data, similar to the ones given in the example in Fig. 3. In these representations we have marked to which regime each subset belongs, as well as the OPS statistic and label according to the discretisation in Table 1 for each particular subset. We have also added the date on which the transitions occurred.

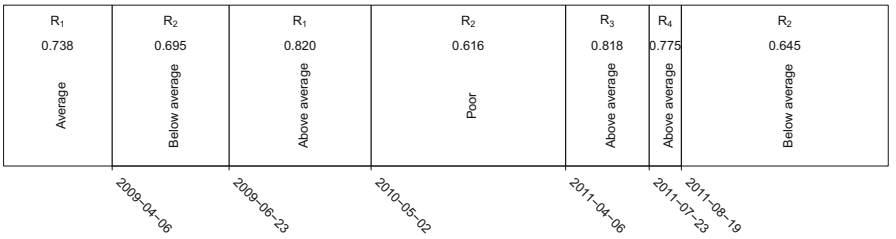


Fig. 7 Regime subsets with OPS statistic and label for Nyjer Morgan

We will begin our analysis with Nyjer Morgan (Figs. 6, 7). We first remark on the OPS labels given to the regimes. The subsets 2, 4 and 7 all belong to regime R_2, which have the OPS labels *below average*, *poor* and *below average*, thus we conclude that the R_2 regime represents a low performance regime for Nyjer (with respect to OPS). The subsets 1 and 3 belong to R_1 and have been given *average* and *above average*, and we will therefore refer to regime R_1 as a high performance regime. Subset number 5 is the only subset that belongs to regime R_3 with the label *above average*, a regime that also represents high performance, but not the same as R_1. Subset number 6 is the sole subset for regime R_4, and given the OPS label *above average* it also seems to correspond to a high level of performance, however we note that the OPS value of 0.775 in subset 6 is lower compared to 0.818 in subset 5.

Turning our attention to the regime transitions, we will offer a narrative that may explain why the transitions happened at these specific dates. Nyjer's major league career started with the Pittsburgh Pirates, where he was given the starting role in left field in the beginning of the 2009 season (this was based on a positive second half of 2008). We see a transition from the high performance regime R_1 to the low performance regime R_2 on the 6th of April (the 2009 season started on the 5th of April, but the Pirates first game was on the 6th). At the beginning of any season there are a lot of changes to the team and tactics, including giving Nyjer the starting role in left field. Perhaps due to the performance of Nyjer, he was traded on the 30th of June 2009 to the Washington Nationals. We see a regime change on the 23rd of June to the high performance regime R_1 at approximately the same time. The next regime change on the 2nd of May is however not as easy to pinpoint. It is another move from R_1 to R_2 (high to low performance), which is not necessarily tied to a new season, trade or injury. However, Nyjer was involved in some headline worthy mistakes and unsportsmanlike actions during the second half of 2010, that may have affected the way he played. In any case, this is a type of transition where a manager or coach could delve deeper into their data to find the causes of this regime transition. On the 27th of March 2011 Nyjer was traded to the Milwaukee Brewers (this happened during the offseason), and in the beginning of the 2011 season, on the 6th of April 2011, a new regime change occurred (Milwaukees first game of the 2011 season was on the 31st of March, but Nyjer did not play full games the first days of the season). During this last transition Nyjer moved into R_3, a high performing regime which lasts until the 23rd of July 2011, when Nyjer transitions to R_4 (also a high performance regime, but as we noted with a lower OPS), and then to the low performance regime R_1 on the

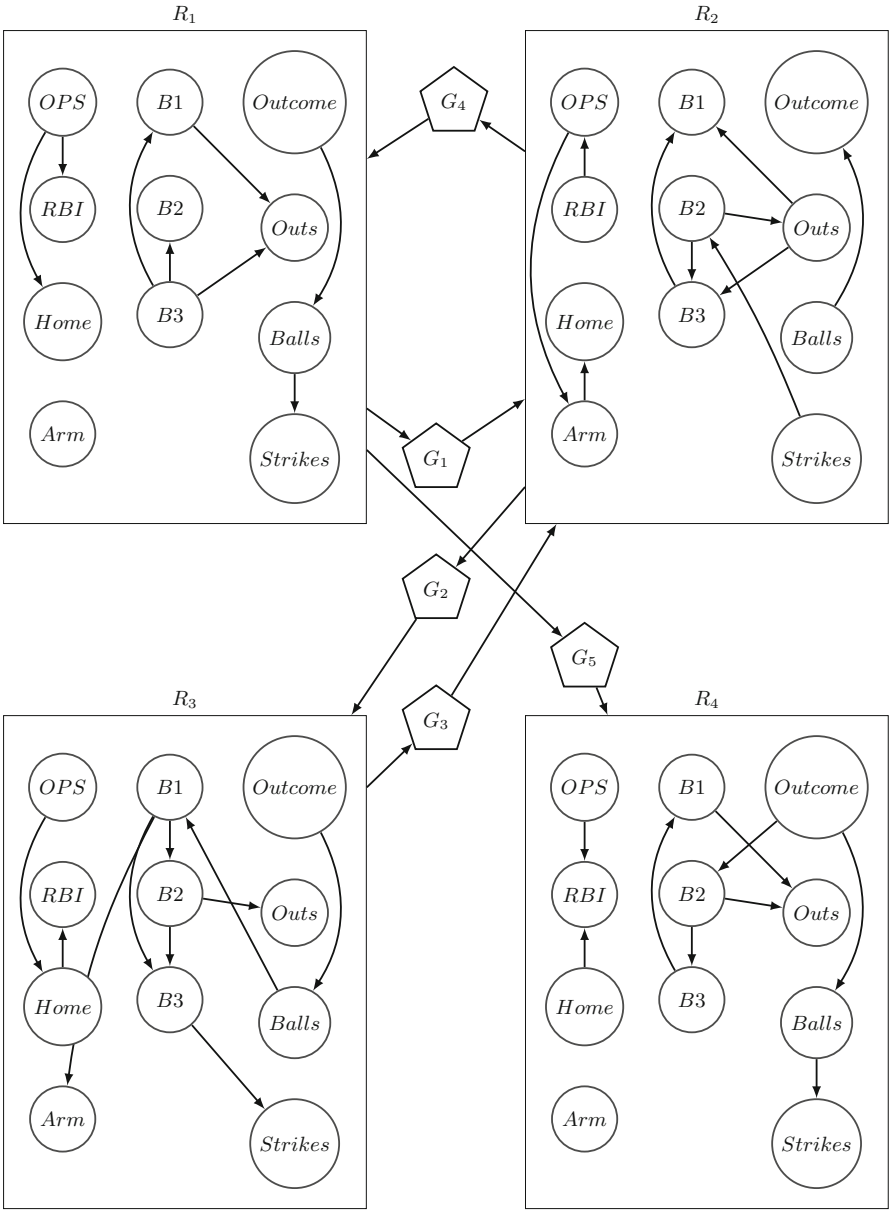


Fig. 8 GBN learnt for Kendrys Morales

19th of August 2011. These last two regimes cannot be directly accredited to a new season, trade or injury, and are examples of transitions needing further investigation by managers and coaches. What can be said is that the regime transitions R_3 to R_4 and finally R_2 seem to indicate a dwindling performance for Nyjer, with respect to the OPS values. He would later take contracts for other major league teams, as well

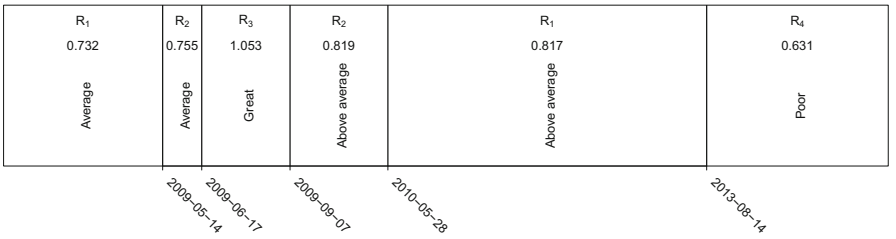


Fig. 9 Regime subsets with OPS statistic and label for Kendrys Morales

as stints in both Japan and Korea. However Nyjer has not returned to his former high level performance R_1 or R_3 regimes.

The second player we will analyse is Kendrys Morales, for whom we display the GBN and subset representation in Figs. 8 and 9. For Kendrys we could just as well assign a high performance label to both R_1 and R_2 as they have one subset with *average* and *above average* respectively. The narrative we soon will offer may clear up the distinction, however it serves as a reminder that summary statistics such as OPS does not tell the full story. Regime R_3 represents exceptional performance, as the subset has been given the label *great*. The last regime R_4 will be referred to as a low performance regime due to the OPS label *poor*.

As we did for Nyjer, we will offer an explanation for the transitions we found in Kendrys' data. At the beginning of the 2009 season Kendrys was promoted to starting first baseman.[6] He had a fantastic season, playing well in the beginning and even better in the second half of the season. He was given the Player of the Month award in August 2009. The first transition on the 14th of May from R_1 to R_2 came early in the season (the 2009 season started on the 5th of April), and while the transition is not exactly on the first day of the season, it is reasonable to assume that Kendrys was adjusting into his new position. There is no evident reason for the transition from R_2 to R_3 on the 17th of June, however it is clear that Kendrys was playing better, thus there is certainly reason for further investigation into why there was a transition on this date. The transition from R_3 to R_2 happened on the 7th of September 2009, again a transition for which no evident reason can be given. However, looking at the OPS values rather than the labels, we see that the transitions $R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_2$ coincide with an increase, peak and decline of the OPS value. A question, to which we have no direct answer, is to ask whether this is simply due to Kendrys going on a hot-streak and then cooling off, or if there were actions taken at these times that caused the transitions through these regimes. If the latter is the answer to the question, then it would be extremely valuable for managers and coaches to investigate these regime changes in order to find these causes so that they may be put in place again. In an unfortunate turn of events, Kendrys was injured on the 29th of May 2010 (coinciding with the transition from R_2 to R_1). He was sidelined for the rest of the 2010 season

[6] Baseball players usually occupy a specific position on the field when playing defense, thus players are often referenced to as first baseman, second baseman, shortstop, etc. However, these players still bat and play offense, but since everybody is either a batter or runner when playing offense it is the defensive position that is referenced.

and the entire 2011 season. In 2012 Kendrys came back, and it seems that he was able to play at a reasonable level as before (R_1), but did not come back to his high and extraordinary performance regimes R_2 and R_3. Kendrys was traded to the Seattle Mariners prior to the 2013 season, however the transition from R_1 to R_4, the low performance regime, did not come until mid season. While Kendrys was traded, the data does not suggest that this change caused any disruption in Kendrys performance, but rather something that happened later.

5.4.4 GBNs versus BNs

From a modelling perspective, regardless of application domain, it is interesting to note that the BNs that are learnt do in fact represent the data better for their subsets than do the other BNs. This is shown in Tables 5 and 6, where the log marginal likelihood for each combination of subset and BN is presented, as well as for a single BN that was learnt using all the data (using data for Nyjer Morgan and Kendrys Morales). Note that these are log-terms, so differences may look small in some cases when they are in fact orders of magnitude.

It is clear then that if we use a single BN to represent our data, we will not be able to capture and take advantage of the structural changes in the data. However, while treating a weakness of BNs we are also exposing one of GBNs. Looking at Figs. 6 and 8 it is definitely much harder to now grasp what the underlying structure of the data is. What we gain in expressive power, we lose in clarity. On the positive side, if one takes

Table 5 Log marginal likelihood of data subsets given individual BNs (Nyjer Morgan)

	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5	Subset 6	Subset 7
R_1	**−2416.16**	−2763.77	**−3228.26**	−4252.85	−2007.11	−797.71	−4693.33
R_2	−2427.00	**−2756.20**	−3236.63	**−4251.72**	−2007.46	−801.51	**−4689.58**
R_3	−2437.66	−2856.39	−3288.64	−4304.85	**−1987.94**	−819.07	−4719.25
R_4	−2464.09	−2831.43	−3348.70	−4355.55	−2027.88	**−773.21**	−4791.58
Single	−2446.08	−2770.41	−3238.49	−4270.29	−2019.44	−809.21	−4698.99

The bold value indicates which regime model gives the highest log marginal likelihood for the different subsets of the data

Table 6 Log marginal likelihood of data subsets given individual BNs (Kendrys Morales)

	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5	Subset 6
R_1	**−4929.38**	−1266.78	−2690.71	−3206.58	**−9897.58**	−5472.86
R_2	−4945.14	**−1257.95**	−2696.66	**−3200.29**	−9945.11	−5487.72
R_3	−5027.57	−1266.08	**−2667.84**	−3258.13	−10078.60	−5532.56
R_4	−4931.26	−1260.78	−2695.41	−3217.07	−9900.53	**−5460.90**
Single	−4959.28	−1285.23	−2730.32	−3235.33	−9943.27	−5496.74

The bold value indicates which regime model gives the highest log marginal likelihood for the different subsets of the data

the time to analyse the BN structures in the GBNs, valuable pieces of information may be extracted. For instance, in Fig. 8 we see that only in R_2 does the pitchers throwing arm stand in relation with Kendrys playing at home and his recent OPS value. This may be due to the opposing teams' coaches choosing a favourable matchup (it is generally accepted that a right handed pitcher has an advantage over a right handed batter, and the same goes for left on left, however the batter has the advantage when they are opposite). Adding to the complexity is the fact that Kendrys himself is a *switch hitter*, meaning that he will change his batting hand to always have the advantage. Perhaps this is why the relationships are only brief, the opposing coaches giving up on trying this tactic. Whatever the reason for these brief relationships, they would vanish if one learnt a single BN using all the data, thus this type of extraction would not be possible.

6 Conclusions

All sports seem to demand a certain degree of data collection, it would be hard to imagine any type of competitive market for trading players in any sport without some statistical description of the players. For baseball however, it seems to have been taken to a level of healthy obsession. Retrosheet is certainly a remarkable endeavour, not only attempting, but doing very well in achieving its goal to computerise every single plate appearance of every single major league game since 1871. We have only picked out a handful of the variables that are represented in the database. It is however possible that this obsession also creates a certain bias, as those looking at the data may become falsely assured that usable information is extractable simply due to the sheer amount of data available. Nonetheless, the penetration of sabermetrics into major league teams is undeniable.

In this paper we have investigated the use of GBNs for modelling baseball players' careers, specifically in order to identify regimes in the data. It is clear from our experiments that such regimes do exist, and that these regimes can reoccur. We have also shown how a deeper analysis of a GBN allows us to label the regimes depending on the player's performance, thereby offering a tool to managers and coaches that would allow them to investigate further the causes for the regime changes. Since we have seen that regimes reoccur, we cannot rule out the fact that it is possible for the causes to be put in place again, attempting to force a regime transition. Although only mentioned briefly (see Sect. 5.4.2), we also explained how the GBNs that were learnt can be used on future data in order to monitor whether or not the decisions of the coaches are transitioning their players to favourable regimes.

In a more general context, when collecting data over time, we must keep in mind that regime transitions can occur. In ecology, biology, finance etc. regime changes are common (e.g. periods of exuberance and panic in financial markets). We have seen in Sect. 5.4.4 that learning a single model when regimes do occur results in a model that is worse at explaining the data, than are the individual models for each regime. It happens to be the case that we are analysing career data of baseball players in the current setting, however regardless of the context it is important to acknowledge that regime transitions may occur.

Moving forward there are a few things that we would like to address. From a modelling perspective we are interested in collapsing the entire GBN into something that is easier to comprehend, for instance by representing all BNs at the same time using a single chain graph (Sonntag and Peña 2015). There would naturally be a loss of granularity, however a chain graph could potentially give a brief overview of a full GBN. We are also interested in merging our work with the ideas of Bill James and Nate Silver. It is possible that we could cluster players based on the GBN structure, and then attempt to predict which regime structure a new player will take (similar to clustering ageing curves). Finally, when we collect data over time, we should be aware that structural changes may occur. Thus for any player of any sport we could potentially find regimes, and make the same conclusions as we have for the baseball players in this study.

Appendix A: Pseudocode for the proposed learning algorithm

In this appendix we shall present pseudocode for the learning algorithm introduced and discussed in Sect. 4. As we did in Sect. 4, we will divide the algorithm into three distinct parts: identifying regime changes, learning the transition structure, and creating a GBN.

Appendix A.1: Identifying regime changes in the dataset

The pseudocode for the initial identification of regime transitions can be found in Fig. 10. The function $IDENTIFY$ is called using a dataset $\mathcal{D}$, the maximum number of transitions k, and the number of MCMC iterations as input. The algorithm begins by initialising the current parameters. To distinguish between current and proposed parameters, we use a superscript c for the current parameters, and superscript p for the proposed parameters.

On line 9 the MCMC iterations begin. Proposed parameters are sampled according to their proposal distributions that are described in detail in Sect. 4.1.2. Two sets of nonzero δs are calculated, one for the current parameters Γ^c and one for the proposed parameters Γ^p. On line 18, the dataset supplied to the algorithm is split according to the nonzero δs (again, this is done for both the current and the proposed parameters). For each resulting subset of $\mathcal{D}$ a BN is learnt, resulting in a set of BNs for the current parameters, and a set of BNs for the proposed parameters.

On lines 23 through 26 we calculate the likelihood of the data given the current BNs and the proposed BNs, as well as the posterior probability of the current and proposed parameters. Note that the calculations are described in more detail in Eqs. 3 and 6.

The rest of the algorithm decides if the current parameters should be replaced with the proposed parameters. First calculate the probability of moving from the current parameters to the proposed (and vice versa). Then calculate the acceptance ratio r and

function IDENTIFY($\mathcal{D}, k, iterations$)
  $n \leftarrow$ number of data points in $\mathcal{D}$
  $\beta_1^c = \frac{n}{k+1}$, $\beta_2^c = \frac{2n}{k+1}$, ..., $\beta_k^c = \frac{kn}{k+1}$ ▷ Space out βs evenly
  $I_1^c = 1, ..., I_k^c = 1$ ▷ Initially set all splits to 1
  $\delta_1^c = \beta_1^c I_1^c$, ..., $\delta_k^c = \beta_k^c I_k^c$
  $iteration = 0$
  $\mathbf{I} \leftarrow \{\}$ ▷ Container for posterior samples
  $\Delta \leftarrow \{\}$ ▷ Container for posterior samples
  repeat
    $iteration = iteration + 1$
    $\beta_1^p, ..., \beta_k^p \leftarrow$ propose new βs according to Equations 7 through 11
    $I_1^p = Bernoulli(0.5), ..., I_k^p = Bernoulli(0.5)$ ▷ Propose new indicator values
    $\delta_1^p = \beta_1^p I_1^p$, ..., $\delta_k^p = \beta_k^p I_k^p$

    $\Gamma^c = \{\delta_i \in \{\delta_1^c, ..., \delta_k^c\} : \delta_i \neq 0\}$ ▷ Select current nonzero δs
    $\Gamma^p = \{\delta_i \in \{\delta_1^p, ..., \delta_k^p\} : \delta_i \neq 0\}$ ▷ Select proposed nonzero δs

    $\mathbf{D}^c \leftarrow$ split $\mathcal{D}$ into subsets according to Γ^c
    $\mathbf{BN}^c = \{D_i^c \in \mathbf{D}^c$: Learn a BN using $D_i^c\}$ ▷ Learn a BN for each subset
    $\mathbf{D}^p \leftarrow$ split $\mathcal{D}$ into subsets according to Γ^p
    $\mathbf{BN}^p = \{D_i^p \in \mathbf{D}^p$: Learn a BN using $D_i^p\}$ ▷ Learn a BN for each subset

    $p(\mathcal{D}|\Gamma^c) = p(D_1^c|BN_1^c)...p(D_{k'}^c|BN_{k'}^c)$ ▷ k' is the number of elements in Γ^c
    $p(\beta_1^c, ..., \beta_k^c, I_1^c, ..., I_k^c|\mathcal{D}) = p(\mathcal{D}|\Gamma^c)U(\beta_1^c; 0, \beta_2^c)...U(\beta_k^c; \beta_{k-1}^c, n+1)p(I_1^c)...p(I_k^c)$
    $p(\mathcal{D}|\Gamma^p) = p(D_1^p|BN_1^p)...p(D_{k'}^p|BN_{k'}^p)$ ▷ k' is the number of elements in Γ^p
    $p(\beta_1^p, ..., \beta_k^p, I_1^p, ..., I_k^p|\mathcal{D}) = p(\mathcal{D}|\Gamma^p)U(\beta_1^p; 0, \beta_2^p)...U(\beta_k^p; \beta_{k-1}^p, n+1)p(I_1^p)...p(I_k^p)$

    $J^p = p(\beta_1^p, ..., \beta_k^p, I_1^p, ..., I_k^p|\beta_1^c, ..., \beta_k^c, I_1^c, ..., I_k^c)$ ▷ Probability going from c to p
    $J^c = p(\beta_1^c, ..., \beta_k^c, I_1^c, ..., I_k^c|\beta_1^p, ..., \beta_k^p, I_1^p, ..., I_k^p)$ ▷ Probability going from p to c

    $r = \frac{p(\beta_1^p,...,\beta_k^p,I_1^p,...,I_k^p|\mathcal{D})/J^p}{p(\beta_1^c,...,\beta_k^c,I_1^c,...,I_k^c|\mathcal{D})/J^c}$
    $u = U(0, 1)$ ▷ Sample a value at uniform between 0 and 1
    if r > u **then**
      $\beta_1^c = \beta_1^p$,...,$\beta_k^c = \beta_k^p$
      $I_1^c = I_1^p$,...,$I_k^c = I_k^p$
      $\delta_1^c = \beta_1^c I_1^c$, ..., $\delta_k^c = \beta_k^c I_k^c$
    end if

    if $iteration > iterations/2$ **then**
      $\mathbf{I} = \mathbf{I} \cup \{I_1^c, ..., I_k^c\}$
      $\Delta = \Delta \cup \{\delta_1^c, ..., \delta_k^c\}$
    end if
  until $iteration == iterations$

  $nonzero\ \delta s = \{i \in 1, ..., k : Mean_\Delta(\delta_k)\ if\ Mean_\mathbf{I}(I_k) > 0.5\}$
  return $nonzero\ \delta s$.
end function

Fig. 10 Pseudocode for identification of nonzero δs

accept the proposed parameters with this probability (if $r > 1$ then always accept). If the proposed parameters are accepted, then we set the current parameters to the proposed parameters.

Since we use a burn-in period to throw away samples at the beginning of the MCMC iterations, on line 39 we only store posterior samples if the current iteration is more than half the total iterations (i.e. throw away first half of samples).

Once all MCMC iterations are complete, we identify which δs are nonzero on line 45. Given the posterior samples $\mathbf{I}$, we calculate the marginal mean for each indicator I, and if the mean is greater than 0.5, then the corresponding δ is considered nonzero. The value for the δ parameter is the marginal mean given the posterior samples Δ.

Finally, we return the nonzero δs identified in the dataset.

```
1: function MERGE(𝒟, nonzero δs)
2:     D ← split 𝒟 into subsets {D_1, .., D_k} according to nonzero δs
3:     R ← {R_1 = {D_1}, R_2 = {D_2}, ..., R_{k−1} = {D_{k−1}}, R_k = {D_k}}
4:     C ← {C_1 = {2}, C_2 = {3}, ..., C_{k−1} = {k}, C_k = {}}
5:
6:     ℛ𝒞 ← {{R, C}}                          ▷ Container for all possible structures
7:     COLLAPSE(R, C, ℛ𝒞)      ▷ Collapse initial structure into all possible structures
8:
9:     return  argmax_{{R_k, C_k} ∈ ℛ𝒞}  p(𝒟|R_k, C_k)        ▷ Computed using Equation 6
10: end function
11:
12: function COLLAPSE(R, C, ℛ𝒞)
13:     k′ ← number of elements in R
14:     for i ∈ {1, .., k′} do
15:         for j ∈ {i + 1, .., k′} do
16:             if j ∉ C_i ∧ i ∉ C_j then      ▷ We only allow non-adjacent subsets to merge
17:                 R^new, C^new ← COMBINE(R, C, i, j)
18:                 ℛ𝒞 ← ℛ𝒞 ∪ {R^new, C^new}
19:                 COLLAPSE(R^new, C^new, ℛ𝒞)                          ▷ Recursion
20:             end if
21:         end for
22:     end for
23: end function
24:
25: function COMBINE(R, C, i, j)                          ▷ Combine subsets i and j
26:     R^new ← R
27:     R_i^new ← R_i^new ∪ R_j^new                  ▷ All j's subsets are now i's subsets
28:     Remove element j from R^new and reindex
29:
30:     C^new ← C
31:     C_i^new ← C_i^new ∪ C_j^new                 ▷ All j's children are now i's children
32:     for all C_k^new ∈ C^new do
33:         if j ∈ C_k^new then
34:             C_k^new ← C_k^new \ {j}                                ▷ j no longer exists
35:             C_k^new ← C_k^new ∪ {i}                  ▷ Add i as a child instead of j
36:         end if
37:     end for
38:     Remove element j from C^new and reindex
39:     return R^new, C^new
40: end function
```

Fig. 11 Pseudocode for regime transition structure learning

Appendix A.2: Merging subsets

The pseudocode for the transition structure identification part of the overall learning algorithm is presented in Fig. 11. The function $MERGE$ takes as input a dataset $\mathcal{D}$ and the nonzero δs identified by the $IDENTIFY$ function described in Sects. 4.1 and "Identifying regime changes in the dataset" section in "Appendix A".

The first three operations first split the supplied dataset into subsets, according to the supplied nonzero δs. Then two mappings are created that together define a regime transition structure. First, $\mathbf{R}$ is a mapping between regime labels, e.g. R_1 and R_2, to a subset of $\mathbf{D}$. That is, it defines which subsets of $\mathcal{D}$ belongs to which regimes. The second mapping, $\mathbf{C}$, defines the children of each regime, e.g. $C_1 = \{2\}$ means that regime 2 is a child of regime 1. If $C_1 = \{2, 3\}$ this would mean that both regime 2 and 3 are children of regime 1. A regime transition structure can therefore be defined by a pair $\mathbf{R}$ and $\mathbf{C}$.

Since we wish to test all possible regime transition structures, we create a container $\mathcal{RC}$ to hold them, before we call the $COLLAPSE$ function on line 7. Before discussing the $COLLAPSE$ function, we note that the $MERGE$ function ends by computing the marginal likelihood of the data $\mathcal{D}$ for each pair $\{\mathbf{R}_k, \mathbf{C}_k\}$ in $\mathcal{RC}$ and returns the pair (i.e. the transition structure) that has the highest marginal likelihood. Computing the marginal likelihood can be done using a simple restructuring of Eq. 6.

Inside the $COLLAPSE$ function we create a nested loop to identify non-adjacent regimes. If two regimes are adjacent then we do not want to merge them (see the discussion in Sect. 4.2.1). Therefore, on line 16, we ignore cases where either regime j is a child of regime i, or regime i is a child of regime j. If two regimes are non-adjacent then we call the function $COMBINE$. This will result in a new regime transition structure (i.e. a new pair $\mathbf{R}^{new}$, $\mathbf{C}^{new}$), which we store in $\mathcal{RC}$ and use as input to $COLLAPSE$, thus creating a recursion. Once the recursion is complete, $\mathcal{RC}$ will contain all regime transition structures.

In $COMBINE$, the goal is to take a regime transition structure and create a new one by merging two non-adjacent regimes. The algorithm starts by copying $\mathbf{R}$ into $\mathbf{R}^{new}$ and then adding all data subsets that used to belong to regime j to regime i in $\mathbf{R}^{new}$ (thus we have merged the subsets). We then remove element j from $\mathbf{R}^{new}$ and reindex the mapping (that is, any index that was higher than j is reduced by one so as to avoid gaps in the regime labelling). For the mapping of child regimes, we must take a bit more caution. First, on line 31, children of regime j are now children of regime i. However, all regimes that used to have j as a child should now have i as a child instead. The loop therefore goes through all child mappings and removes j if it exists and adds i. Once the loop terminates, we remove element j from $\mathbf{C}^{new}$ and reindex (again, to avoid gaps in the regime labels). Finally, we return the new regime transition structure.

Appendix A.3: Constructing a GBN

The final step of the learning algorithm is to construct a GBN. In Fig. 12 we present the pseudocode for this part of the algorithm. As was the case in "Merging subsets" section

```
1:  function CONSTRUCT(D, R, C)
2:      GBN ← use R to learn a BN for each regime, connect using gates according to C
3:      τ = 25                                  ▷ Initial choice for τ, the lookback window in gates
4:      θ = 1.5                                 ▷ Initial choice for θ, the threshold in gates
5:      number of resamples = 1000              ▷ Number of resamples to create
6:      iterations = 250                        ▷ Number of Bayesian optimisation iterations
7:      η = 5.0                                 ▷ Exploration/exploitation tradeoff coefficient
8:
9:      resamples ← {}
10:     for iteration ∈ {1, ..., number of resamples} do
11:         resamples ← resamples ∪ resample D preserving regimes according to R
12:     end for
13:     g ∼ Gaussian process                    ▷ g follows a Gaussian process
14:     Λ ← {}                                  ▷ Container for parameters
15:     f ← {}                                  ▷ Container for accuracies
16:     for iteration ∈ {1, .., iterations} do
17:         accuracy = ACCURACY(GBN, resamples, R, τ, θ)
18:         Λ ← Λ ∪ {τ, θ}
19:         f ← f ∪ {accuracy}
20:         τ, θ ← arg max_{τ_i,θ_i} μ_g(τ_i, θ_i) + ησ_g(τ_i, θ_i)   ▷ Section 4.3 and Equation 14
21:     end for
22:
23:     Parameterise gates in GBN with Λ_i for which f_i is greatest
24:     return GBN
25: end function
26:
27: function ACCURACY(GBN, resamples, R, τ, θ)
28:     accuracies ← {}
29:     for resample ∈ resamples do
30:         n ← number of data points in resample
31:         Activate the BN that represents the first regime in GBN
32:         for t ∈ {1, ..., n} do
33:             Process resample_{1:t} in GBN if t > τ − 1          ▷ Section 2.3
34:             if active BN in GBN == true regime according to R then
35:                 accuracies ← accuracies ∪ {1}
36:             else
37:                 accuracies ← accuracies ∪ {0}
38:             end if
39:         end for
40:     end for
41:     return sum(accuracies)/n
42: end function
```

Fig. 12 Pseudocode for construction of a GBN

in "Appendix A", **R** maps each regime to subsets of $\mathcal{D}$. The algorithm begins by learning a BN for each regime, and connects these BNs using gates according to the child relationships contained in **C**. Thereafter, initial values for the parameters of the gates (τ and θ) are chosen. The number of resamples, iterations and the exploration/exploitation coefficient η is set (please see Sect. 4.3.1 for details).

To reduce the risk of overfitting the test data, on line 10 we resample $\mathcal{D}$ while preserving the regimes according to **R**. This is described in more detail in Sect. 4.3.2,

however the idea is that we take the subsets that belong to the first regime and resample these, placing them in the new sample such that the regime transition structure is intact, and then continue to the second regime, and so on. We then create our proxy random variable g (that follows a GP) and containers to store parameters and function values (i.e. the accuracies).

On line 16 the main Bayesian optimisation loop begins. We calculate the average accuracy of the GBN parameterised with τ and θ over the resamples. We store the parameters and the accuracy, and then identify the next pair τ and θ by maximising the UCB (*upper confidence bound criterion* discussed in Sect. 4.3.1). Once a predefined number of iterations of Bayesian optimisation has completed, we parameterise the GBN with the parameters that gave the maximum accuracy, and return the parameterised GBN (thus ending the learning algorithm).

In the $ACCURACY$ function we calculate the accuracy of a pair of parameters θ and τ over all resamples. The steps taken are those described in Sect. 4.3.2. Essentially, we treat the resample as a stream of data, increasing its size with each iteration, and process it in the GBN. We compare which BN is active in the GBN to the true regime according to $\mathbf{R}$, and give a score of one if they match. To clarify, if data point with index 250 belongs to a subset of $\mathcal{D}$ that in turn belongs to regime R_3 (which we can read off from $\mathbf{R}$), then if the BN that represents R_3 is active after the stream that ends with index 250 has been processed, then we say that the GBN is correct, all other cases are incorrect. Note that in order to calculate the ratios inside the gates we must have at least τ data points, therefore no processing of data is done until we reach this number.

Appendix B: Synthetic data generation

In this appendix we present the BNs and transition structures used when generating data for the synthetic experiments. The five BNs used are shown in Fig. 13, and the three transition structures in Fig. 14. We created one learning dataset for each of the three transition structures, and 100 testing datasets for estimating accuracy. The datasets were created by sampling a random number of samples (uniform between 250 and 600) from each regime, and then concatenating to create a dataset. For instance, for Structure 2 in Fig. 14, we first sampled a random number of samples from R_1 in Fig. 13, then another random number of samples from R_2, then from R_1 again, and finally from R_3 and R_4.

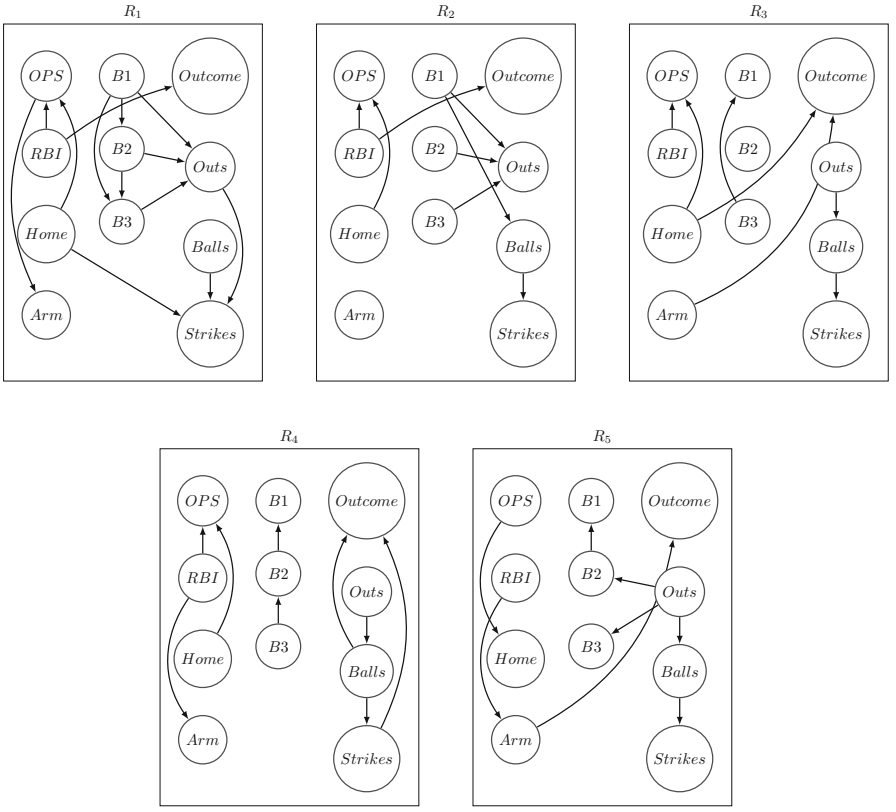


Fig. 13 BNs created to generate synthetic career data

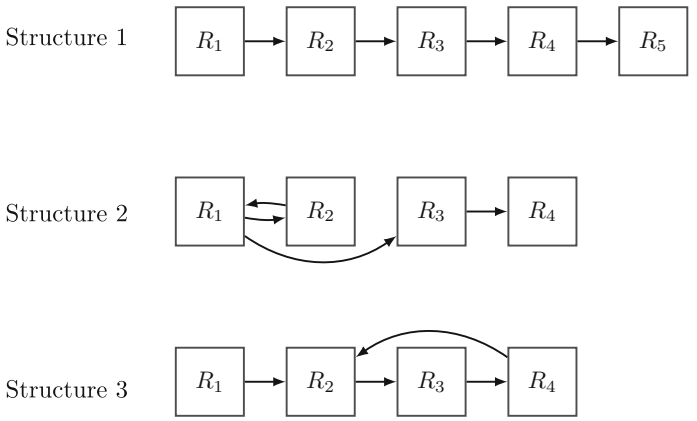


Fig. 14 Transition structures used in synthetic experiments

Appendix C: Results of HMM learning on synthetic data

In this appendix we present the results from learning HMMs[7] using the synthetic datasets described in "Appendix B". We shall do so graphically, by comparing the true regimes in the learning data, with the inferred regimes by the HMMs (at each datapoint we pick the regime with the highest smoothed probability, i.e. the state with the highest probability given all data). The results for the three regime transition structures are presented in Figs. 15, 16 and 17, where the black line represents the true regime, and the grey line represents the inferred regime.

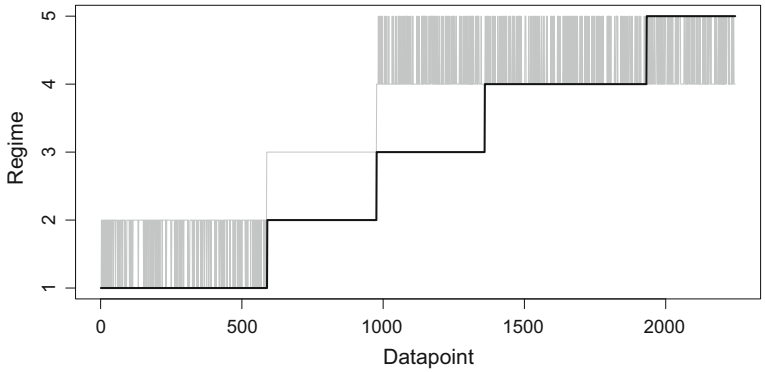


Fig. 15 Structure 1—True regime and regime given the HMM

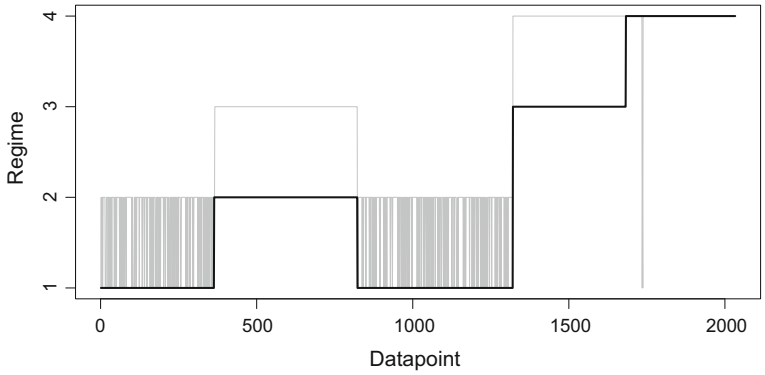


Fig. 16 Structure 2—True regime and regime given the HMM

[7] The R package depmixS4 (Visser and Speekenbrink 2010) was used for learning and inference.

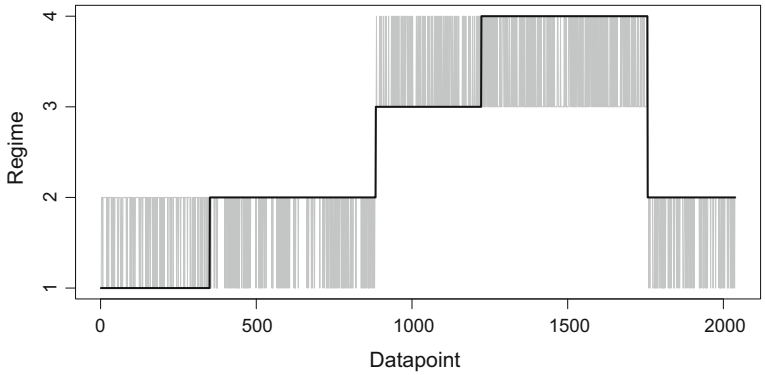


Fig. 17 Structure 3—True regime and regime given the HMM

A pattern emerges when comparing the three figures. For Structure 1 (Fig. 15), the HMM cannot distinguish between the first and second regime for the duration of the initial regime, however when the data truly transitions to the second regime, the HMM identifies this change into a new third regime. When the transition into the third regime does occur, the HMM transitions into an oscillation between the fourth and fifth regime, and continues to oscillate for the rest of the dataset. For Structure 2 (Fig. 16), we again see that when the first regime is in place, the HMM oscillates between regimes one and two, and when the true regime is three and four, it identifies this as the fourth regime. For Structure 3 (Fig. 17), the HMM is again not capable of distinguishing between the first and second regime, and similarly so for regimes three and four.

It seems that the HMMs that we have learnt are not capable of correctly identifying the current regime, in contrast to the GBNs that we learnt and evaluated in the main results (see Sect. 5.4.1, specifically Table 2). Furthermore, the regime transition structures are not correctly captured by the HMMs, for instance, for Structure 1 the HMM implies that it is possible to transition from the second regime to the first regime, however the structure that generated the data was a chain of regimes (see Fig. 14).

References

Baseball prospectus. www.baseballprospectus.com
Baseball think factory. www.baseballthinkfactory.org
Bendtsen M (2015) Bayesian optimisation of gated Bayesian networks for algorithmic trading. In: Proceedings of the twelfth Bayesian modeling applications workshop
Bendtsen M (2017) Gated Bayesian networks. Doctoral thesis, Linköping University.
Bendtsen M, Peña JM (2013) Gated Bayesian networks. In: Proceedings of the twelfth Scandinavian conference on artificial intelligence, pp 35–44
Bendtsen M, Peña JM (2014) Learning gated Bayesian networks for algorithmic trading. In: Proceedings of the seventh European workshop on probabilistic graphical models, pp 49–64
Bendtsen M, Peña JM (2016) Gated Bayesian networks for algorithmic trading. Int J Approx Reason 69(1):58–80
Bill James online. www.billjamesonline.com

Brochu E, Cora VM, de Freitas N (2009) A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Tech. Rep. UBC TR-2009-023 and arXiv:1012.2599
Buntine W (1991) Theory refinement on Bayesian networks. In: Proceedings of the seventh conference on uncertainty in artificial intelligence, pp 52–60
Friedman N, Goldszmidt M (1997) Sequential update of Bayesian network structure. In: Proceedings of the thirteenth conference on uncertainty in artificial intelligence, pp 165–174
Gama J, žliobaitė I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. ACM Comput Surv 46(4):44:1–44:37
Ghahramani Z, Hinton GE (2000) Variational learning for switching state-space models. Neural Comput 12(4):831–864
Goldfeld SM, Quandt RE (1973) A Markov model for switching regressions. J Econom 1(1):3–15
Guo F, Hanneke S, Fu W, Xing EP (2007) Recovering temporally rewiring networks: a model-based approach. In: Proceedings of the 24th international conference on machine learning, pp 321–328
Hamilton JD (1989) A new approach to the economic analysis of nonstationary time series and the business cycle. Econometrica 57(2):357–384
Haupert M, Murray J (2012) Regime switching and wages in major league baseball under the reserve clause. Cliometrica 6(2):143–162
Heckerman D (1995) A tutorial on learning with Bayesian networks. Tech. Rep. MSR-TR-95-06, Microsoft Research
Heckerman D, Geiger D, Chickering DM (1995) Learning Bayesian networks: the combination of knowledge and statistical data. Machine Learning 20(3):197–243
Jensen FV, Nielsen TD (2007) Bayesian networks and decision graphs. Springer, New York
Korb KB, Nicholson AE (2011) Bayesian artificial intelligence. Taylor and Francis Group, Boca Raton
Lam W (1998) Bayesian network refinement via machine learning approach. IEEE Trans Pattern Anal Mach Intell 20(3):240–251
Lam W, Bacchus F (1994) Using new data to refine a Bayesian network. In: Proceedings of the tenth international conference on uncertainty in artificial intelligence, pp 383–390
Langosch J (2015) FBI concludes investigation on Astros breach. www.mlb.com
Lewis M (2003) Moneyball. W.W. Norton, New York
Marcel. www.tangotiger.net/marcel
McTaggart B (2012) Analyze this: Astros' Mejdal takes on unique role. www.mlb.com
Munagi AO (2005) Set partitions with successions and separations. Int J Math Math Sci 2005(3):451–463
Nielsen SH, Nielsen TD (2008) Adapting Bayes network structures to non-stationary domains. Int J Approx Reason 49(2):379–397
Pearl J (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers, San Francisco
Rasmussen CE, Williams CKI (2006) Gaussian processes for machine learning. MIT Press, Cambridge
Retrosheet. www.retrosheet.org
Robinson JW, Hartemink AJ (2010) Learning non-stationary dynamic Bayesian networks. J Mach Learn Res 11(1):3647–3680
Silver N (2012) The signal and the noise. Penguin, London
Sonntag D, Peña JM (2015) Chain graph interpretations and their relations revisited. Int J Approx Reason 58(1):39–56
Steamer. www.steamerprojections.com
Visser I, Speekenbrink M (2010) depmixS4: an R package for hidden Markov models. J Stat Softw 36(7):1–21
Xuan X, Murphy K (2007) Modeling changing dependency structure in multivariate time series. In: Proceedings of the 24th international conference on machine learning, pp 1055–1062