Latent Branching Trees

A Class of Semi-Parametric Time Series Models

Theo Kypraios
http://www.maths.nottingham.ac.uk/∼tk

Joint work with:

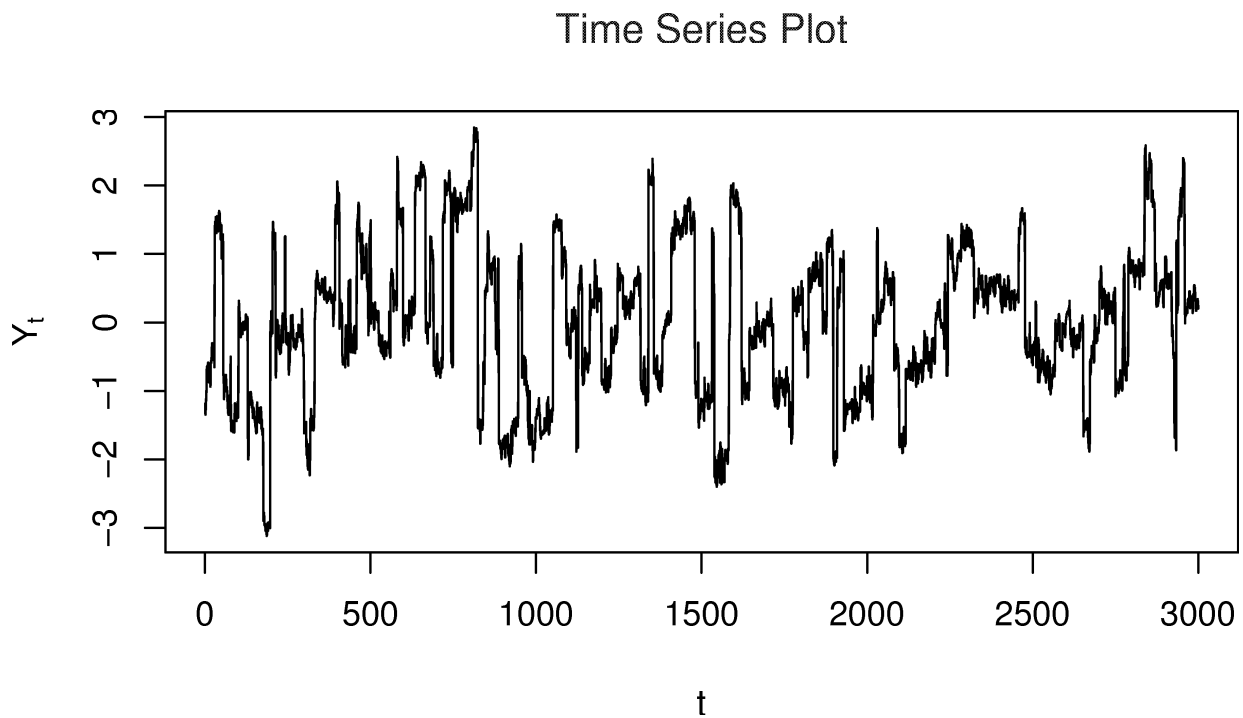
Gareth O. Roberts @ University of Warwick
Paul Fearnhead @ Lancaster University

Outline

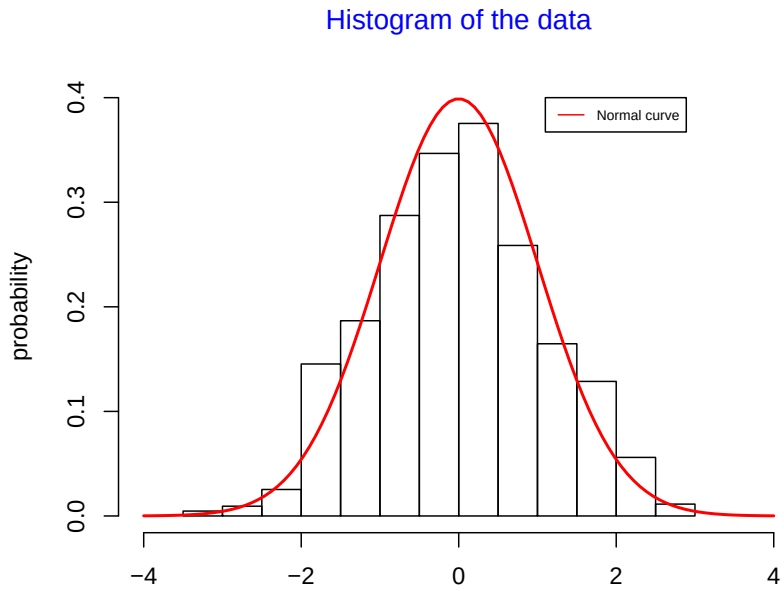
- Part I
  - Motivation
  - Construction
- Part II
  - Bayesian Inference
  - MCMC
- Part III
  - Applications
    1. Genome Scheme Data
    2. Internet Traffic

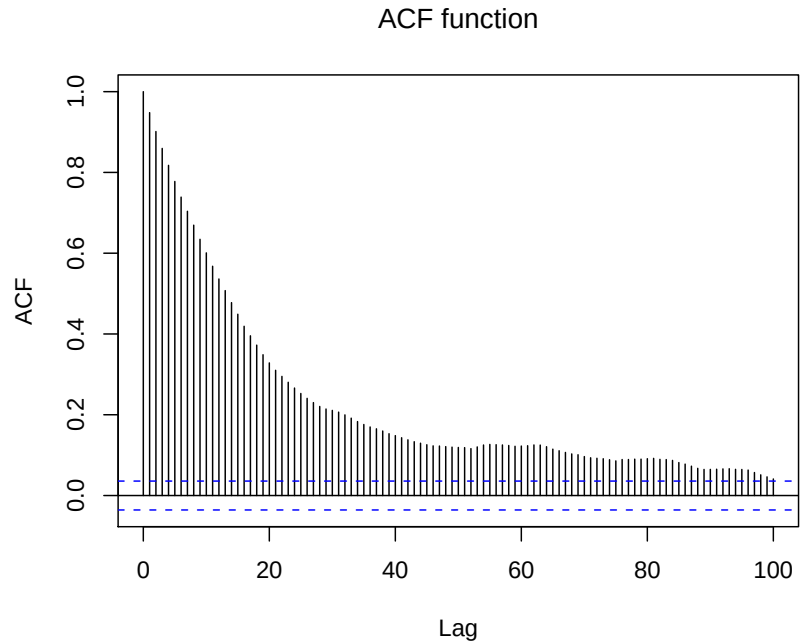
Motivation

It is often the case, where observations have been collected over a time interval and is easy to specify their marginal distribution but not that easy to say anything about their correlation structure.



Motivation (ctd)





Objectives

- We are interested in constructing a class of semi-parametric time series models such that:
  - we would be able to specify easily the marginal distribution of the observations and then,
  - build their dependence structure around them.

- We define a stochastic process which we call, a *Latent Branching Tree* (LBT) and will allow us to build such models with a rich class of covariance structure.

- Given observed data we wish to draw inference for the model parameters.

Motivated originally by Radford Neal's Dirichlet Diffusion Trees which have been generalised by Knowles and Ghahramani to the Pitman-Yor Diffusion Trees.

Construction

A Latent Branching Tree: Construction

Suppose, for now, that we would like to construct a time series which will consist of n observations which follow a Gaussian distribution with zero mean and unit variance.

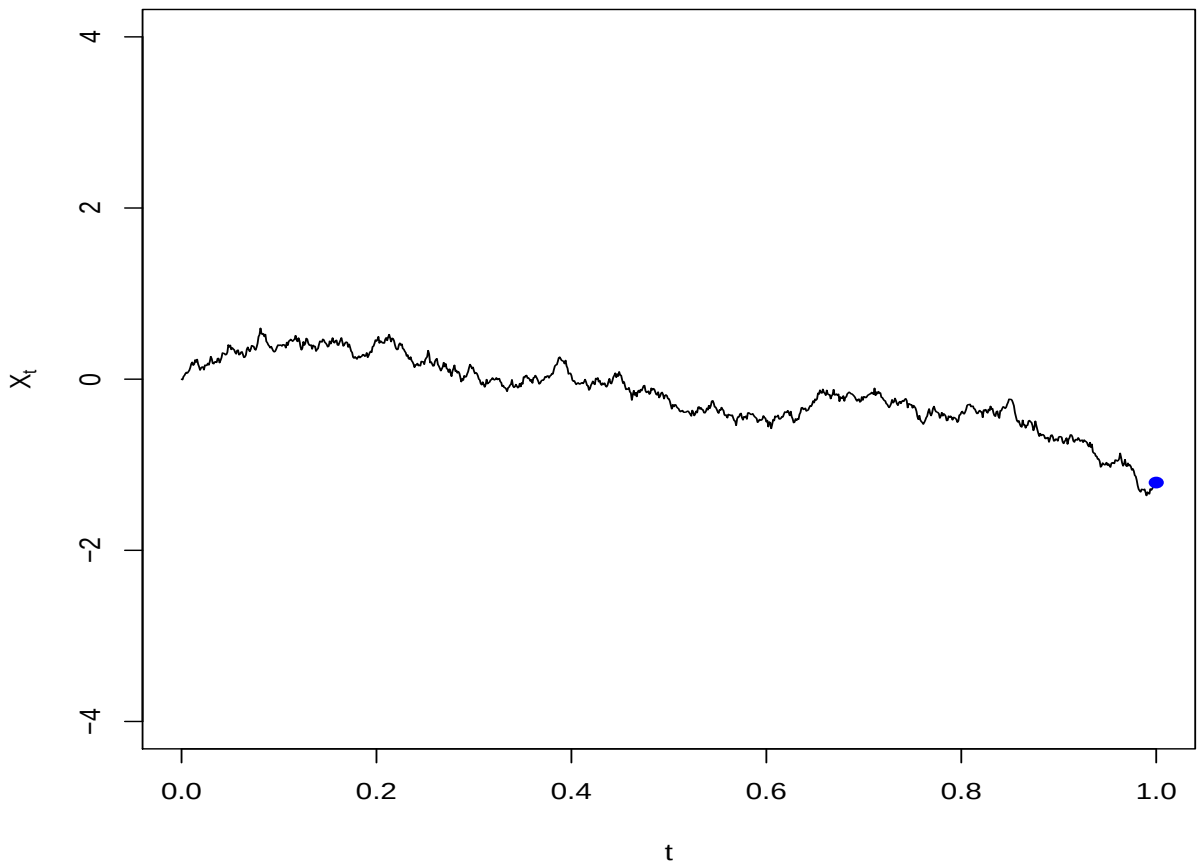
The first data point, denoted by Y_1, is drawn by a diffusion which starts from a fixed origin and operates for a fixed length of time.

Note that:

- We fix, the starting point the end point to be $t = 0$ and $t = 1$ respectively, i.e. $X(t), 0 \leq t \leq 1$.
- Other diffusions can be used instead.

A Latent Branching Tree: Construction (ctd)

The value of the Bm at time $t = 1$ is considered as the first data point, i.e. $Y_1 = X_1(1)$



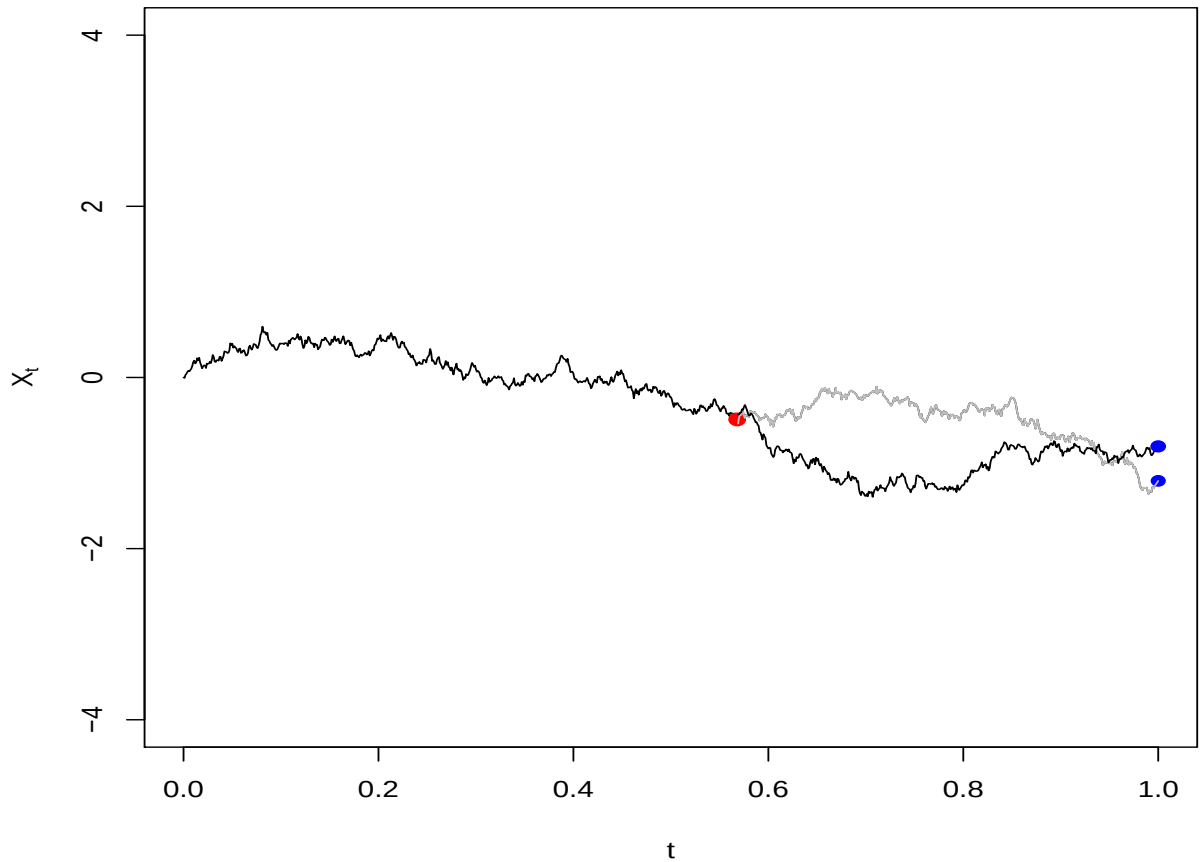
A Latent Branching Tree: Construction (ctd)

Consider now the following Bm:

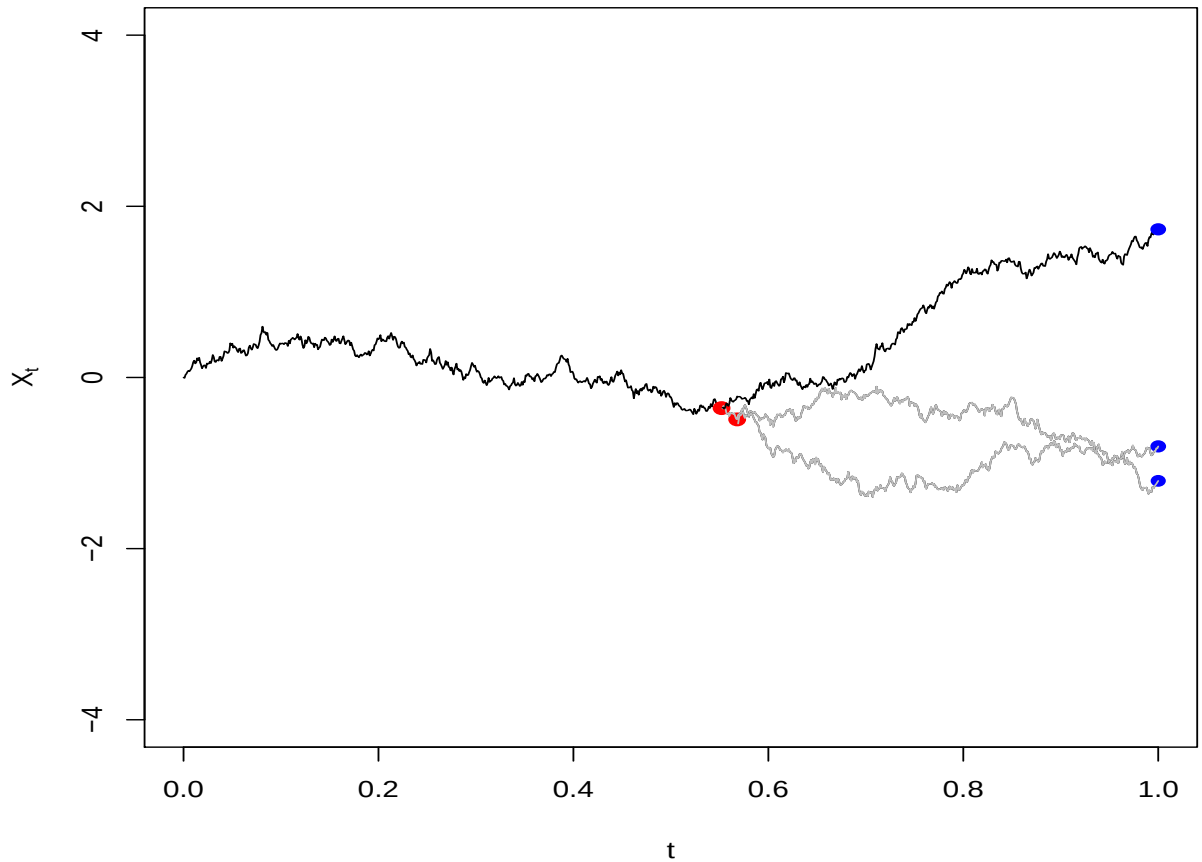
$X_2(t)$ follows exactly the same path as $X_1(t)$ up to time τ_1 and then traverses to another path independent of $X_1(t), t \geq \tau_1$.

Then, the second data point is the value of the second Brownian motion at $t = 1$, i.e. $Y_2 = X_2(1)$

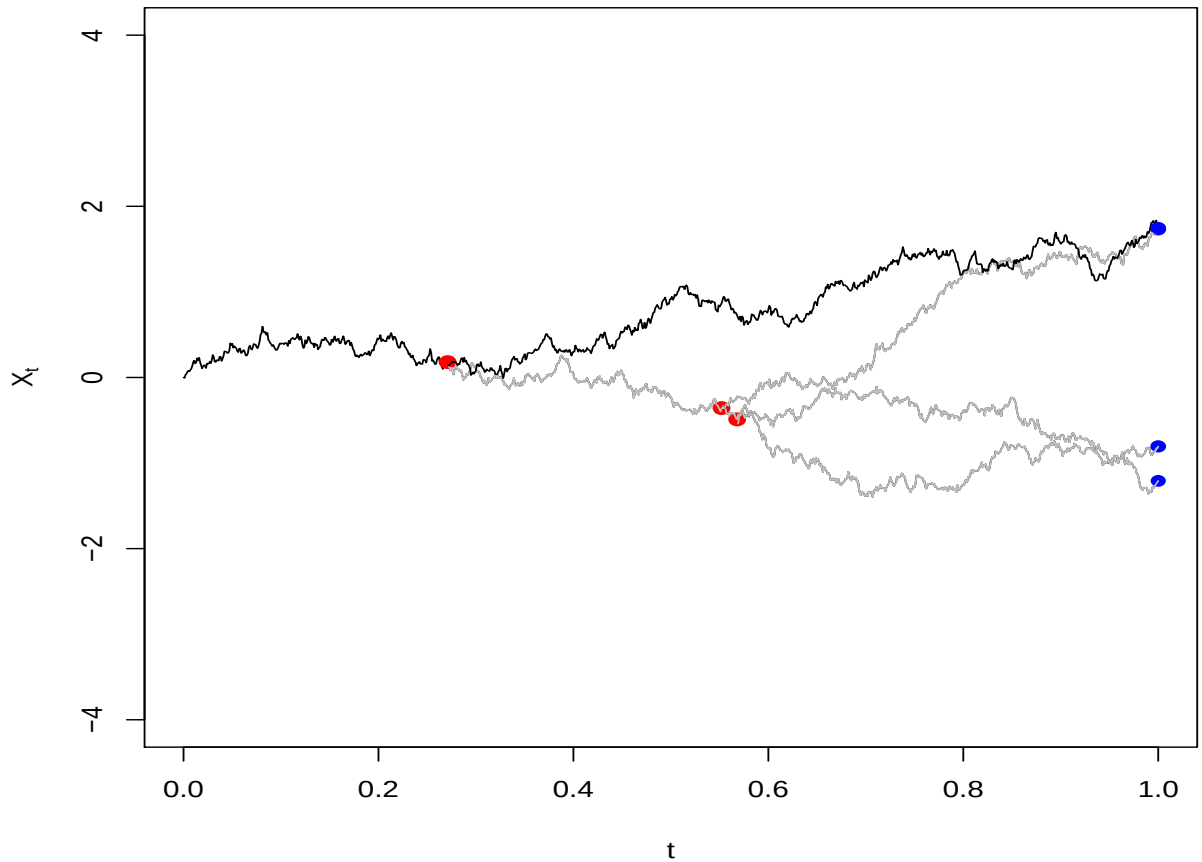
A Latent Branching Tree: Construction (ctd)



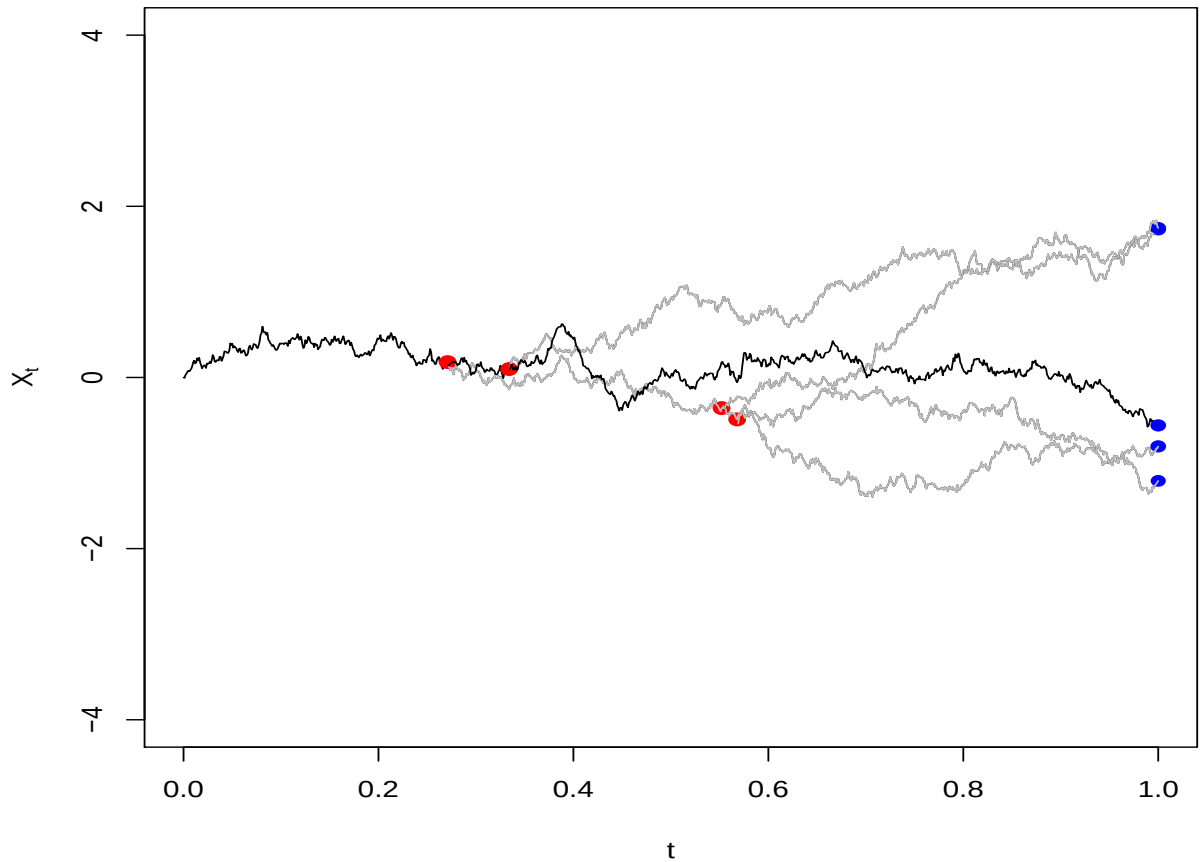
A Latent Branching Tree: Construction (ctd)



A Latent Branching Tree: Construction (ctd)



A Latent Branching Tree: Construction (ctd)



A Latent Branching Tree: Construction (ctd)

In general the i_{th} data point is the value of the i_{th} Brownian motion, i.e. $Y_i = X_i(1), i = 0, \ldots, n$ where:

$$\begin{aligned} X_i(t) &= X_{i-1}(t), \text{ for } 0 \leq t \leq \tau_{i-1} \\ X_i(t) - X_i(\tau_{i-1}) &\perp X_{i-1}(t) - X_{i-1}(\tau_{i-1}), \text{ for } \tau_{i-1} < t \leq 1 \end{aligned}$$

- Easy to see that in order to generate n data points, n diffusions and $n-1$ divergence points, $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_{n-1})^T$ are required.
- Note, that there is always only **one** "alive" Bm and that is a fundamental difference to DDTs and POYTs.

show video!

(Loading Branching Tree Movie)

Properties of the LBT

Recall the generation of the first two data points Y_1, Y_2 and denote by T the time where the 2nd Bm diverged. The covariance between Y_1 and Y_2 is:

$$\text{cov}(Y_1, Y_2) = \mathbb{E}[Y_1 Y_2] - \mathbb{E}[Y_1]\mathbb{E}[Y_2] = \mathbb{E}_T[\mathbb{E}[Y_1 Y_2 | T]]$$

But:

$$\begin{aligned} \mathbb{E}_T[Y_1 Y_2 | T] &= \mathbb{E}_T[(X_T + Y_1 - X_T) \cdot (X_T + Y_2 - X_T) | T] \\ &= \mathbb{E}_T[X_T^2] \\ &= \text{var}(X_T) \\ &= T \end{aligned} \tag{1}$$

i.e. the closer the time which the 2nd Bm diverged to 1 is, the more correlated the two observations are!

A Theorem

It is easy to prove the following theorem:

Theorem 1

Let $Y_1, Y_2, \ldots, Y_n$ the data points generated by a a latent branching tree and denote by $\tau_1, \ldots, \tau_{n-1}$ the divergence time points, where $\tau_i \sim f(\cdot)$. The covariance of Y_1 and Y_n is equal to the expected value of the minimum of the τ_i's,s i.e.

$$cov(Y_1, Y_n) = \mathbb{E}\left[\min(\tau_1 \ldots, \tau_{n-1})\right]$$

Key point: It is easy to see that the choice of the distribution with density f characterizes the dependence structure of the observations.

Covariance Structure

The distribution that the divergence points $\tau = (\tau_1, \ldots, \tau_{n-1})^T$ follow is called *"jump distribution"*.

- Different choice of the *"jump distribution"* will give us a different covariance (correlation) structure of the observations, $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^T$. Therefore, by appropriate choices we can get the desired covariance structure.
- The "jump distribution" could any distribution which is (or can be) truncated between 0 and 1, e.g. Uniform, Beta.

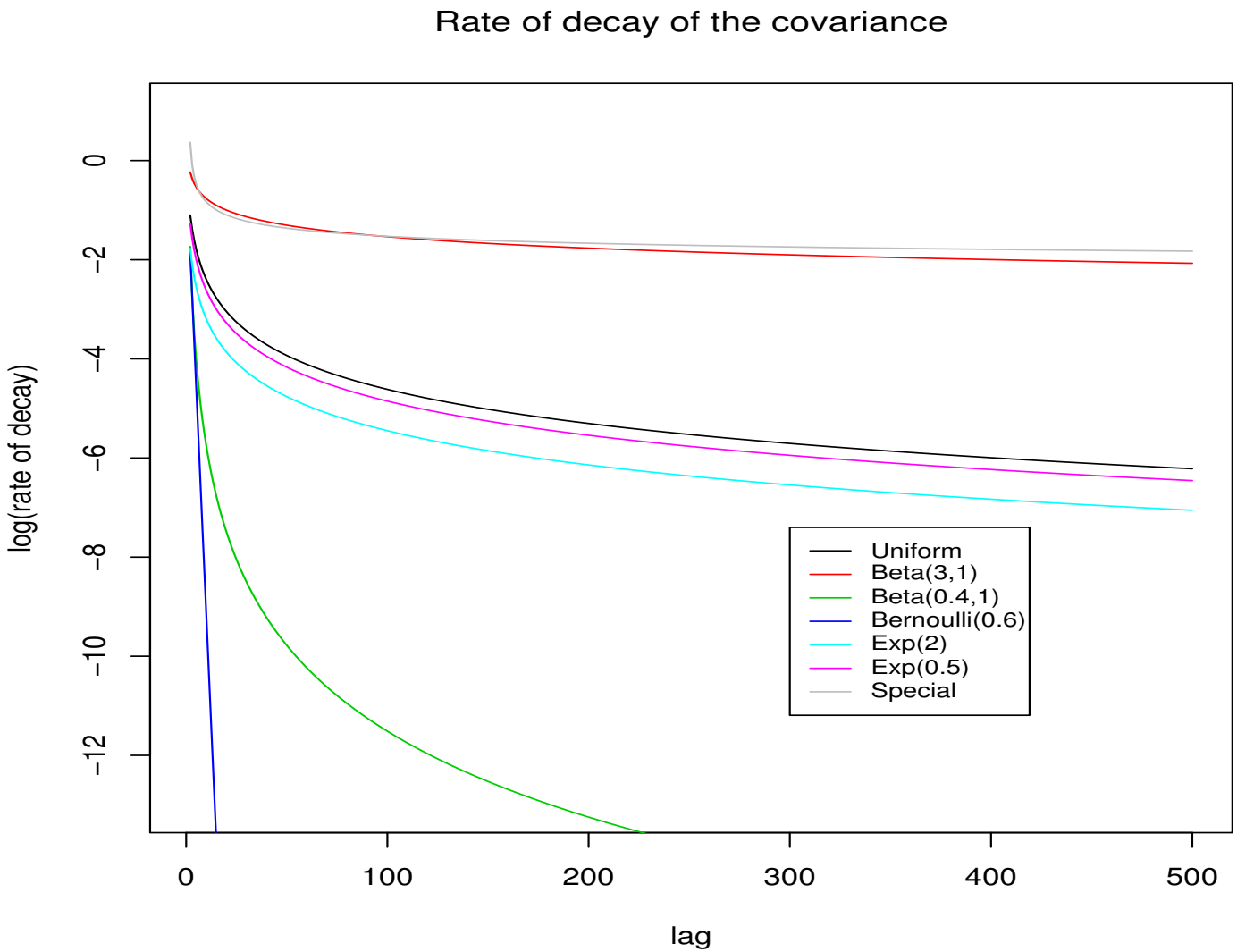
Covariance Structure (ctd)

Table 1 summarizes the rates of decay for each of the distributions. The different decays are due to the different amount of probability mass around zero.

Table: Rate of decays $O(\cdot)$

"Jump Distribution"	**Density** $f_T(\tau)$	**Rate of decay** $O(\cdot)$
Uniform, $U(0,1)$	1	$\propto \frac{1}{n}$
Beta$(a,1)$	$a \cdot \tau^{a-1}$	$\propto \frac{1}{n^{1/a}}$
Bernoulli(p)	$p^{\tau} \cdot (1-p)^{1-\tau}$	$\propto (1-p)^n$
Truncated Exponential(λ)	$\frac{\lambda}{1-e^{-\lambda}} \cdot e^{-\lambda\tau}$	$\propto -\frac{1}{\lambda}\frac{n+e^{-\lambda}}{n+1}$
Frechet	$\tau^{-2}e^{1-\frac{1}{\tau}}$	$\propto \frac{1}{\log n}$

Covariance Structure (cont.)



How about using a mixture of distributions as the "jump distribution"?

Consider the following "jump distribution":

$$\tau_i = \begin{cases} Beta(1, 100) & \text{with probbility 1-p} \\ Beta(100, 1) & \text{with probbility p} \end{cases}$$

Suppose we choose $p = 0.95$.

What sort of pattern should we expect?

Intuitively, if we draw 100 data points (and therefore 99 divergence times), then roughly speaking, about 5 of them will be drawn from this branch of the mixture:

$$Beta(1, 100)$$

which will result to values of τ's very close to zero.

Clustering Time Series.

(Loading Branching Tree Movie)

Margins outside the Gaussian context

The choice of a Brownian motion which operates from $t = 0$ to $t = 1$ lead to the generation of observations which follow a Normal$(0, 1)$ distribution.

For particular applications this maybe unrealistic and therefore we would be interested in marginal distributions outside the Gaussian context.

Within our framework this can be done in two ways:

- Choose a different diffusion according to the desired marginal distribution, i.e. Gamma process for Gamma marginals.
- Apply appropriate transformations to a $N(0, 1)$ random variable, e.g. if $Z \sim N(0, 1)$ then $Z^2 \sim X_1^2$.

Simulation

Simulation of a LBT

There are two algorithms for generating n data points. An *approximate* and an *exact*.

The **approximate algorithm**:

– is fairly easy to implement. Use Euler approximation to simulate each path of the Brownian motions.

However,

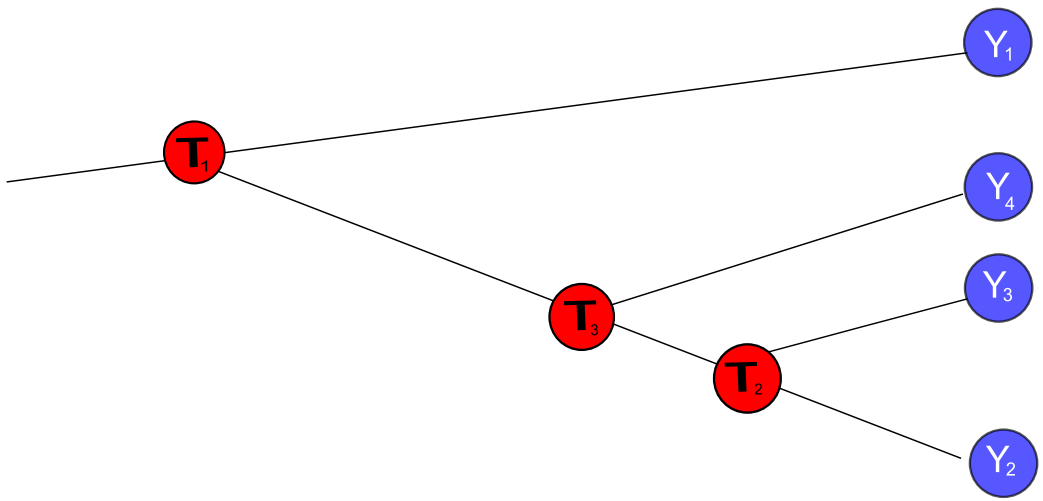
–at each step of generation of a data point we need to store the full path of the Bm.

Exact or approximate?

In fact, we do not need the full path of the Bm ...

... but just the value of the Bm at each of the divergence points, i.e. the τ and X_τ

Therefore, the exact algorithm is based on the idea of the-so-called technique *retrospective sampling* which avoid any Euler approximation.



The Exact Algorithm

1. Draw the $n-1$ divergence points from $t \sim F$;
2. Generate the first data point by simulating from a Normal distribution: $Y_1 \sim N(0,1)$;
3. Draw $X(\tau_1)$ by simulating from a Brownian Bridge between 0 and 1;
4. Generate the second data point (Y_2) by simulating from a Normal distribution: $Y_2 \sim N(X(\tau_1), 1-\tau_1)$;

The Exact Algorithm

5. Set $i = 2$, $\tau_0 = 0$ and $\tau_n = 1$;
6. While $(i < n)$ {
   Define the sets $\boldsymbol{L}$ and $\boldsymbol{U}$ as follows:
   - $\boldsymbol{L} = \{j \in [0, i-1] : \tau_j < \tau_i\}$ and $l = \max(\boldsymbol{L})$
   - $\boldsymbol{U} = \{j \in [l+1, i-1] : \tau_j > \tau_l \ \& \ \tau_j > \tau_i\}$. Let $\tau_u = \max(\tau_j)$, $j \in \boldsymbol{U}$.

   i. Simulate the value of the Brownian motion at the i_{th} divergence time point τ_i, $X(\tau_i)$ by drawing from a Brownian bridge between τ_l and τ_u.
   ii. Generate the the i_{th} data point by drawing from a Normal distribution: $Y_i \sim N(X(\tau_{i-1}), 1 - \tau_{i-1})$;
   iii. $i = i + 1$;

   }

Bayesian Inference

Inference

We are interested in modelling the covariance structure of a set of observations with fixed margins.

Assuming that the observed data were generated by an LBT mechanism we would like to draw inference about the structure of the tree and any parameters associated with it.

For instance, suppose that

$$\tau_i \sim \text{Beta}(\alpha, 1), i = 1, \ldots, N-1.$$

We are interested in inferring about α, as well as for the divergence times $\boldsymbol{\tau}$ which characterize the dependence structure of the observations.

Complete data: Likelihood

It is relatively easy to write down the likelihood of an LBT given that

– the divergence points $(\tau_i, i = 1, \ldots, n-1)$

– their locations $X(\tau_i)$, i.e. the state of the process at each divergence time

– the data points $Y_i, i = 1, \ldots, n$ are known and

– the the order of these events as well.

Complete data: Likelihood

The likelihood can be expressed as product of two factors, one describing the tree (L_1) and the other the data (L_2).

The first factor is the probability of obtaining the (ordered) divergence points (τ_i) and the second is the probability of obtaining the tree given the locations of the divergence points ($X(\tau_i)$) and the final data points (Y_i) given the divergence times.

Complete data: Likelihood

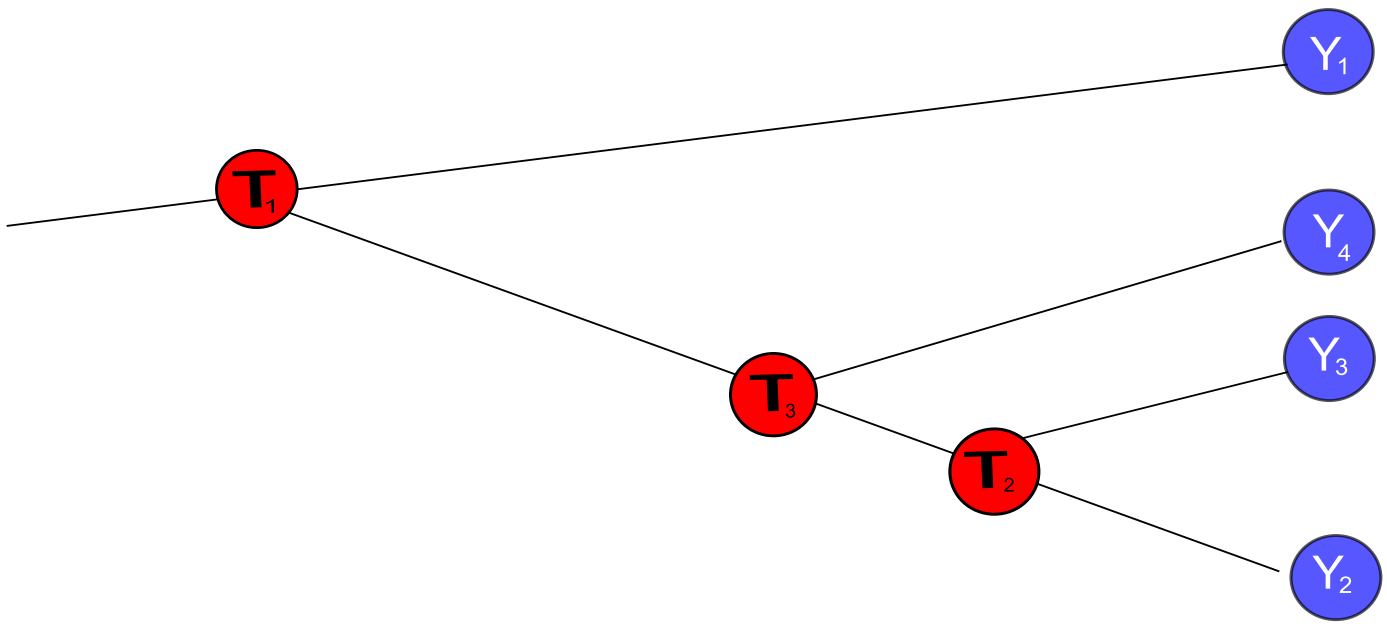
The likelihood $f(\boldsymbol{Y}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)$ can be written as follows:

$$
\begin{aligned}
f(\boldsymbol{Y}; \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3) &= \prod_{i=1}^{n} f(\tau_i) \times \phi(Y_1, 0, 1) \\
&\times \prod_{i=1}^{n-1} \left(\phi^{BB} \left(X(\tau_i), \tau_i, \tau_i^-, \tau_i^+, X(\tau_i^-), X(\tau_i^+) \right) \right) \\
&\times \prod_{i=1}^{n-1} \left(\phi(Y_{i+1}, X(\tau_i), 1 - \tau_i) \right)
\end{aligned}
$$

where $\phi^{BB}(x_b, t_b, t_a, t_c, x_{t_a}, x_{t_c})$ be the conditional density of a Brownian bridge in the interval (t_a, t_c).

Complete data: Likelihood

Alternatively, the probability of obtaining the tree given the locations of the divergence points and the final data points can be written as the product of Normal segments:



Data Augmentation

Unfortunately, in practice we only observe the data points: $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$.

Therefore we augment the observed dataset $\boldsymbol{Y}$ with whole structure of the tree, i.e. divergence times, locations and the order of theses events.

We can then employ a Markov Chain Monte Carlo algorithm to draw samples from the posterior distribution of interest:

$$\text{target distribution: } \pi(\boldsymbol{\tau}, \boldsymbol{X}(\tau), \alpha | \boldsymbol{Y})$$

A Sketch of the MCMC Algorithm

1. Initialisation;
2. Choose one of the divergence parameters j, $1 \leq j \leq n-1$ and update τ_j (individually) using Metropolis Hastings;
3. Update each of the location parameters $X(\tau_j)$, $1 \leq j \leq n-1$ using Gibbs sampler;
4. Update $\boldsymbol{\theta}$ using either Metropolis Hastings or Gibbs sampler depending on the parameter.

- Repetition of Step 2 improves mixing;

Some Remarks (I)

- Each of the $X(\tau_i)|\boldsymbol{X}(-\tau_i), \boldsymbol{\theta}_1, \boldsymbol{\theta}_3$, $i = 1, \ldots, n-1$ is Normally distributed due to the form of products of the Normal densities in L_2 (second part of the likelihood).
- A Uniform $U(0, 1)$ could be used as a proposal for updating the divergence times.
- In the case, where we assumed a Beta "jump distribution", i.e. $Be(\alpha, 1)$ by assigning a conjugate Gamma prior to α, we can then employ a Gibbs sampler to update this parameter.

Some Remarks (II)

- The standard MCMC algorithm works well for some cases, e.g. Uniform prior for divergence times, small data sets etc ...
- ... nevertheless, there are situations where such standard algorithms do not have a great efficiency.
- Efficient MCMC algorithms have been developed which involve
  - Integrating out the location parameters;
  - Block update of the location parameters;
  - Efficient Non-Centered Reparameterisations;
  - Approximate Bayesian Computation (ABC);

MCMC Strategies: Integrate out $X(\tau)$

Target distribution:

$$\pi(\boldsymbol{\tau}, \boldsymbol{X}(\tau), \theta_3 | \boldsymbol{y})$$

Idea:

- It easy to write down the **joint** distribution of $(\boldsymbol{X}(\tau), \boldsymbol{Y})|(\boldsymbol{\tau}, \theta_3)$ – A Multivariate Normal distribution.
- Make use of standard results to integrate $\boldsymbol{X}(\tau)$ out and end up with:

$$\pi(\boldsymbol{Y}|\boldsymbol{\tau}, \theta_3) \sim N_d(\cdot, \cdot)$$

- Combine this with the prior distribution on $\boldsymbol{\tau}$ and $\boldsymbol{\theta}_3$.
- Effectively, we are sampling from $\pi(\boldsymbol{\tau}, \boldsymbol{\theta}_3 | \boldsymbol{Y})$.

Unfortunately, no free lunch! – inversion of **large** but **sparse** matrices is required.

MCMC Strategies: Block Update $X(\tau)$

- Each of the full conditionals of $[\boldsymbol{X}(\tau_i)|\boldsymbol{X}(\tau_{-i}), \boldsymbol{Y}, \boldsymbol{\theta}_3]$ is Normal with some mean and some variance.
- Again, by making use of the Normality, we can write down

$$\pi(\boldsymbol{X}(\tau), \boldsymbol{Y}|\boldsymbol{\tau}) \sim N_d \cdot (0, Q^{-1})$$

which allows us as to derive the joint conditional distribution of the location parameters as a multivariate Gaussian, i.e.

$$\pi(\boldsymbol{X}(\tau)|\boldsymbol{Y}, \boldsymbol{\tau}) \sim N_d(\cdot, \cdot)$$

No free lunch! This procedure involves the inversion of **large** but **sparse** matrices.

MCMC Strategies: Efficient Non-Centered Parameterisations

Suppose that *a-priori*

$$\tau_i \sim \text{Beta}(\alpha, 1) \qquad i.i.d.$$

where α is an unknown parameter of interest.

It is obvious that already, this parameterisation introduces some dependence *a priori* which is then "carried" *a posteriori*.

The idea behind a non-centered reparameterisation is to break this dependence by introducing a change in variables

$$(\boldsymbol{\tau}, \boldsymbol{X}(\tau), \alpha) \to (\boldsymbol{U}, \boldsymbol{X}(\tau), \alpha)$$

Free lunch? Efficiency is gained with minimal extra computational cost compared to the standard algorithm?

Approximate Bayesian Computation (ABC)

- The aforementioned MCMC strategies are computationally intensive, i.e. takes some time to compute the augmented likelihood etc ...
- ... on the other hand, simulating (exactly) the data is **very fast** even for vey large datasets.
- ABC seems a natural framework for this problem if we are only interested in the parameters of the "jump distribution".

Approximate Bayesian Computation (ABC) (II)

The following simple algorithm can be implemented

1. Draw a value for the parameter from the prior, i.e. $\boldsymbol{\theta}_3 \sim \pi(\boldsymbol{\theta}_3)$
2. Simulate an LBT, i.e. $\boldsymbol{Y}_{sim}|\boldsymbol{\theta}_3$
3. Accept $\boldsymbol{\theta}_3$ if

$$d(\boldsymbol{Y}_{sim}, \boldsymbol{Y}_{obs}) < \epsilon$$

A natural choice for the distance metric could be

$$d(\boldsymbol{Y}_{sim}, \boldsymbol{Y}_{obs}) = \sum_{i=1}^{k} (\rho^i_{sim} - \rho^i_{obs})^2$$

where ρ^i denotes the sample correlation and the values for k and ϵ to be determined by the user.

If the prior is very informative this algorithm will be quite inefficient $\rightarrow$ use ABC-SMC instead.

A Simulation Study

For illustration, we are simulating a set of observations, generated by an LBT where the various "jump distribution" is:

$$\text{Beta}(\alpha, 1)$$

where $\alpha = 3$.

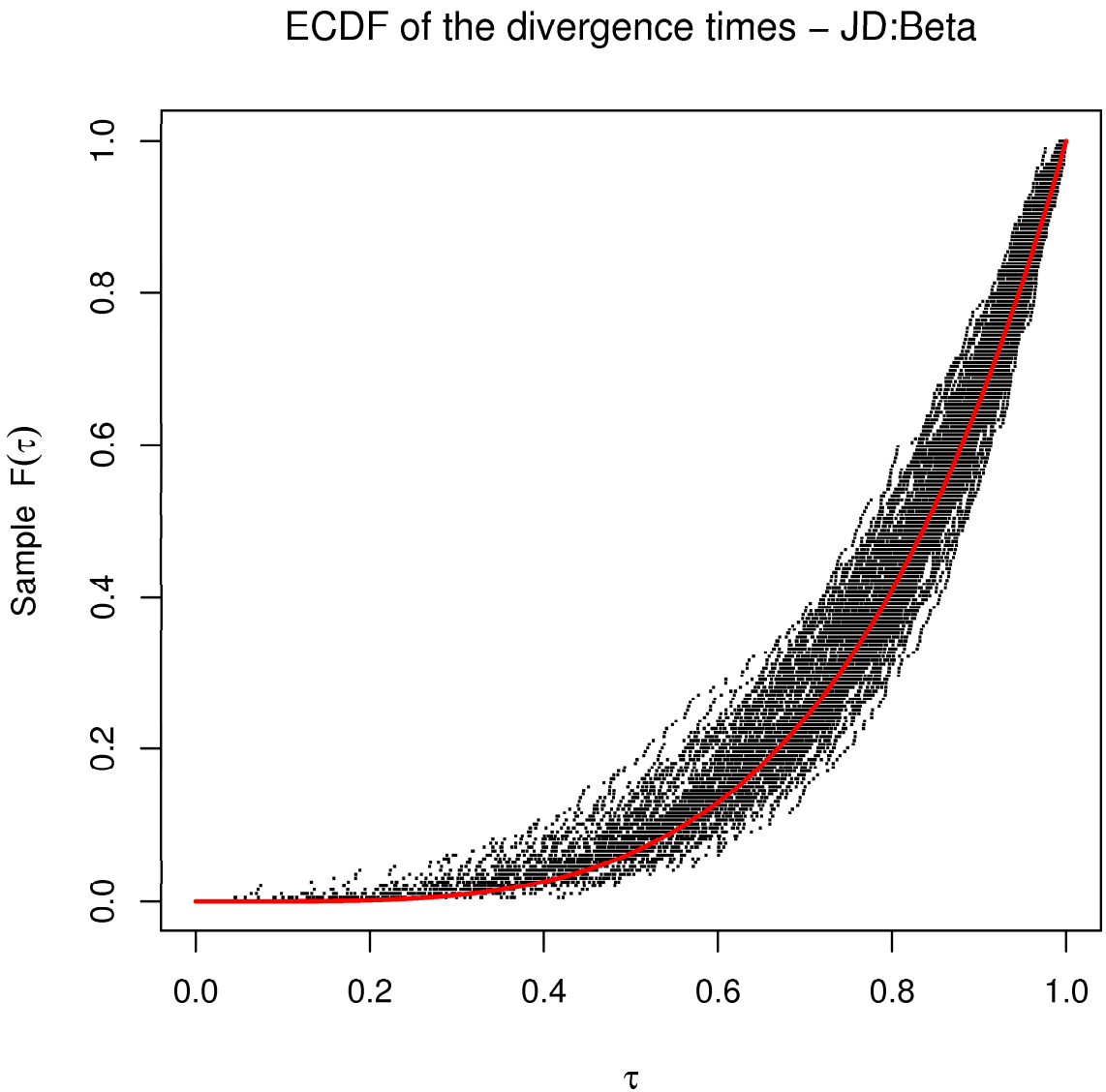
We assume that we have only observed the data points ($\boldsymbol{Y}$) and we would like to "re construct" the latent branching tree, i.e. obtain the posterior distributions of interest:

divergence times: $\pi(\boldsymbol{\tau}|\boldsymbol{Y})$

"jump" paramter: $\pi(\alpha|\boldsymbol{Y})$

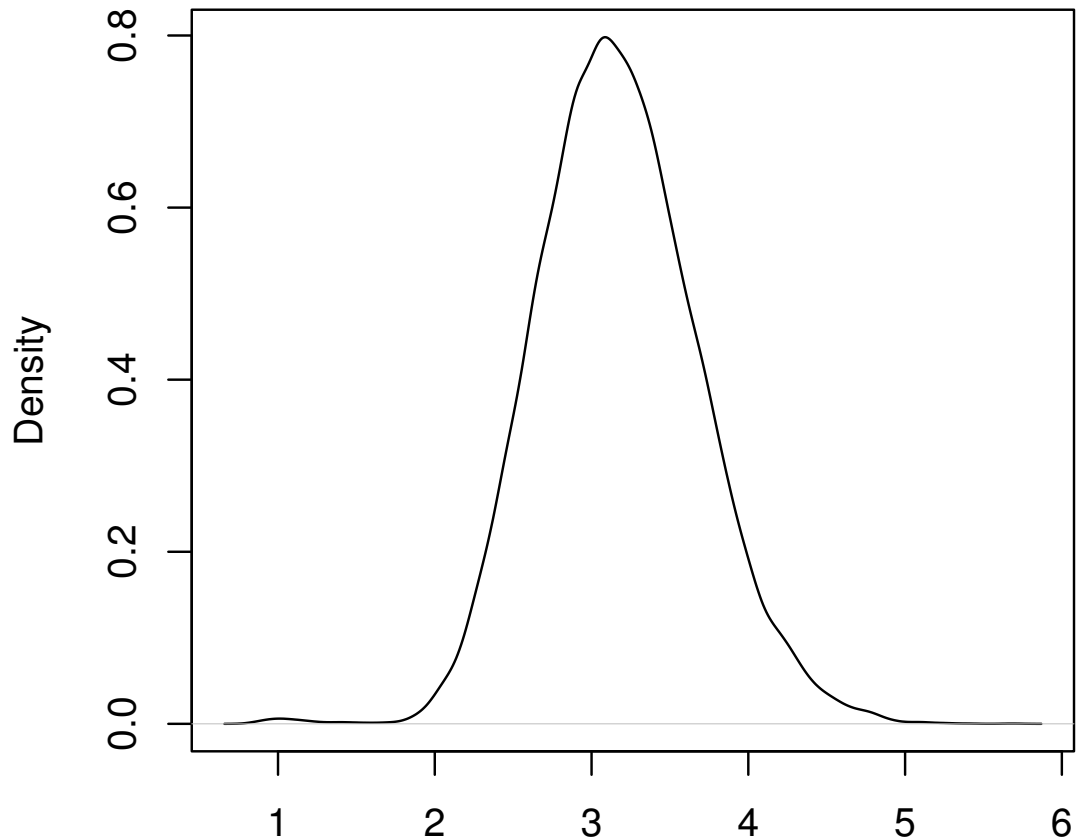
"Jump Distribution:":Beta$(\alpha, 1)$ (ctd)

Figure: Empirical Cumulative Distribution Function of the divergence times. Red line shows the "true" ecdf ($F_T(\tau) = \tau^\alpha$)



"Jump Distribution:":Beta$(\alpha, 1)$ (ctd)

Figure: Density plot of the *"jump parameter"* α



An Application on Genome Scheme Data

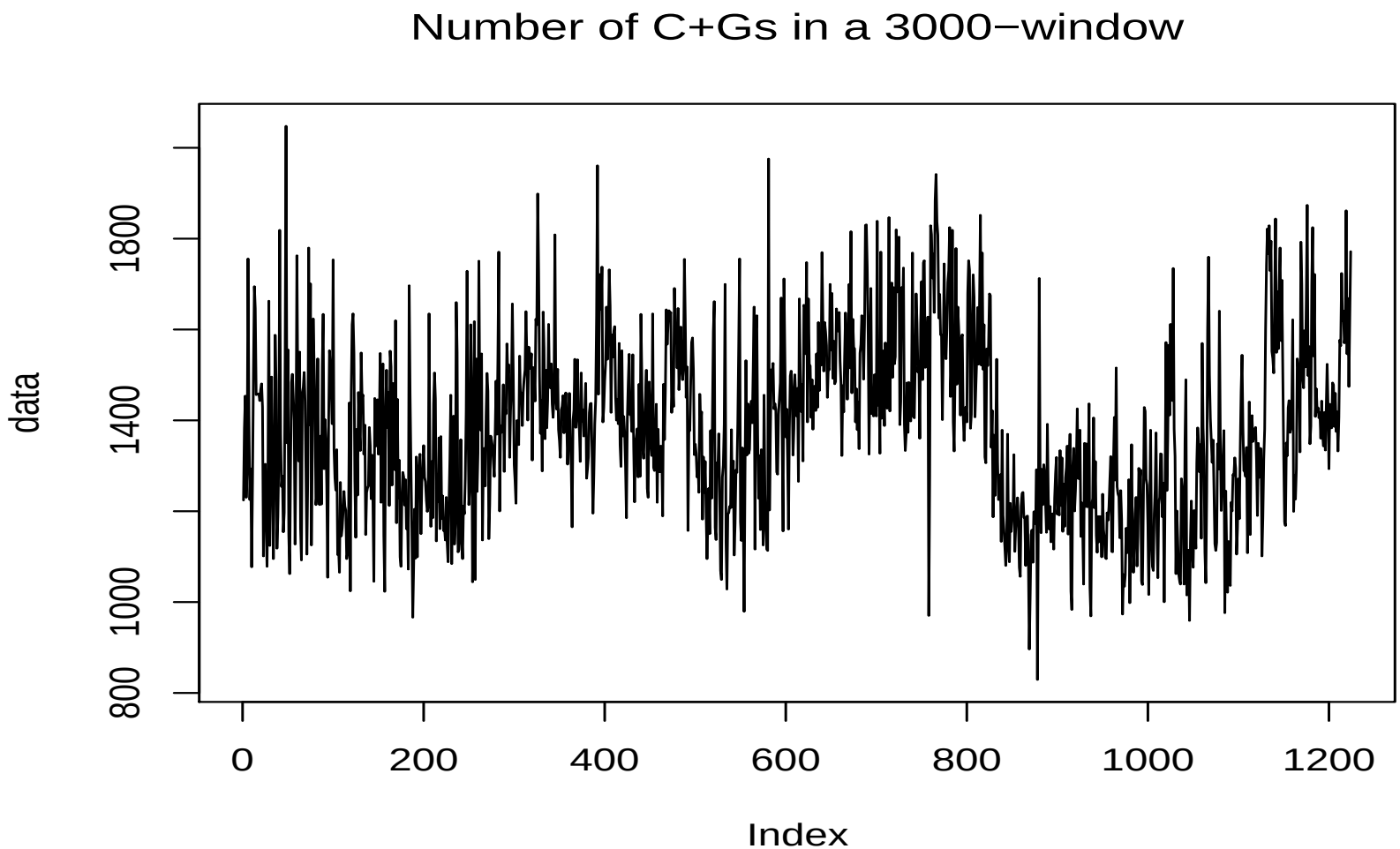
Isochores

The human genome is a mosaic of *isochores*, which are long DNA segments which are relatively homogeneous in G+C when compared to the pronounced heterogeneity throughout the entire genome.

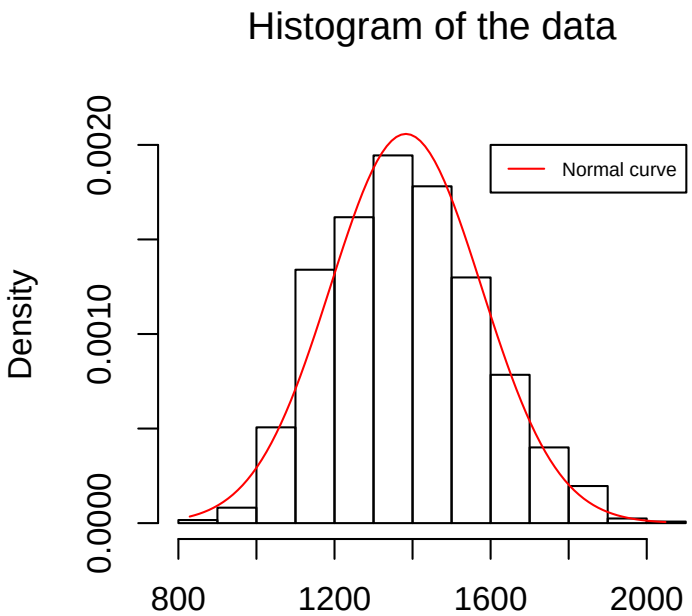
We are given a DNA-sequence of length $3,675,000$. Windows of length $3,000$ were chosen the number of C+Gs is counted (see Figure).

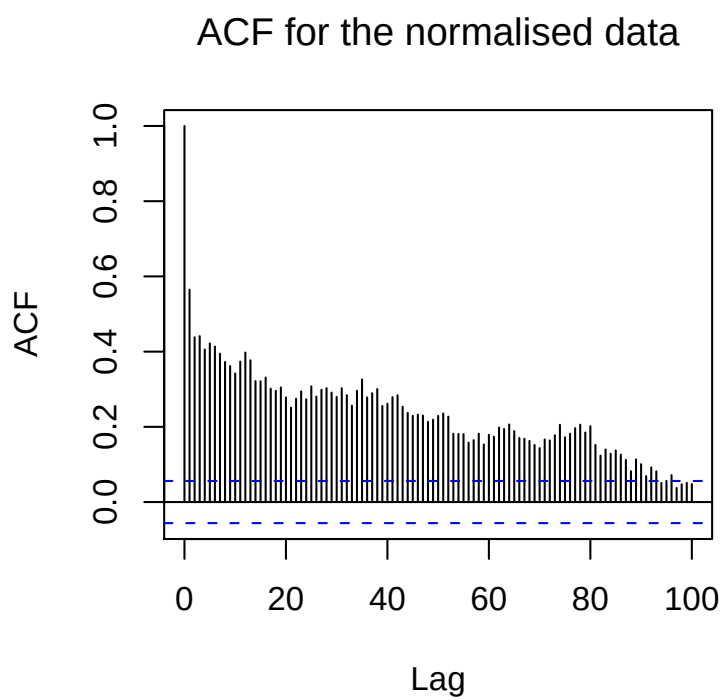
Our objective is to identify the *isochores* which are of particular biological interest.

Real Data


Number of C+Gs in a 3000−window
data
800
1000
1400
1800
0
200
400
600
800
1000
1200
Index

Marginal Distributions and Correlation





Our Approach

Our approach is to fit a Latent Branching Tree to this dataset and try to identify the isochores.

The are various ways to do this, simply by placing different prior distributions on the divergence times;

- Uniform ;
- a mixture of Beta distributions;

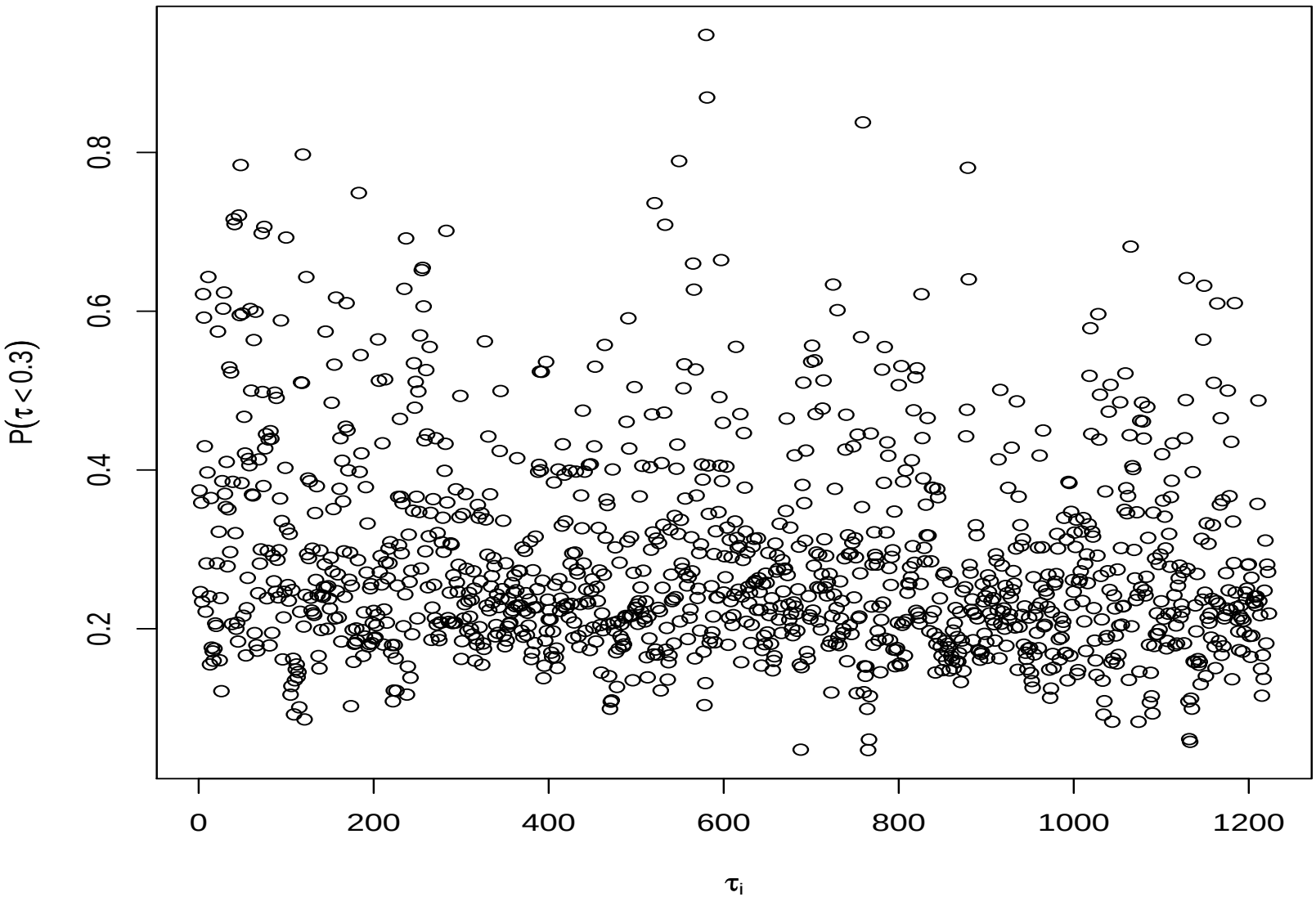
Recall that during the construction of an LBT, values of the divergence times close to 0 tend to give us "clusters" in the time series ...

... therefore, someone maybe interested in identifying such divergence times. We propose to do that by computing the following probabilities:

$$P(\tau < k | \boldsymbol{Y})$$

for some small for k.

$\mathbb{P}(\tau_i < 0.3|\boldsymbol{Y})$ under a Uniform prior

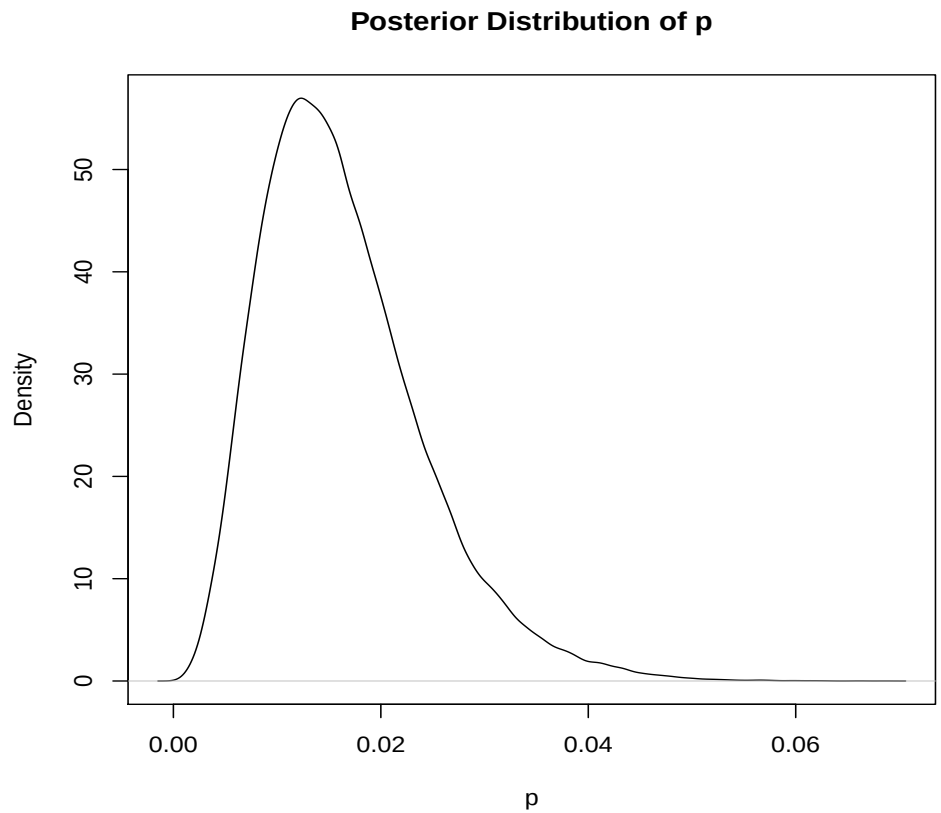


A mixture of Beta distributions

A-priori

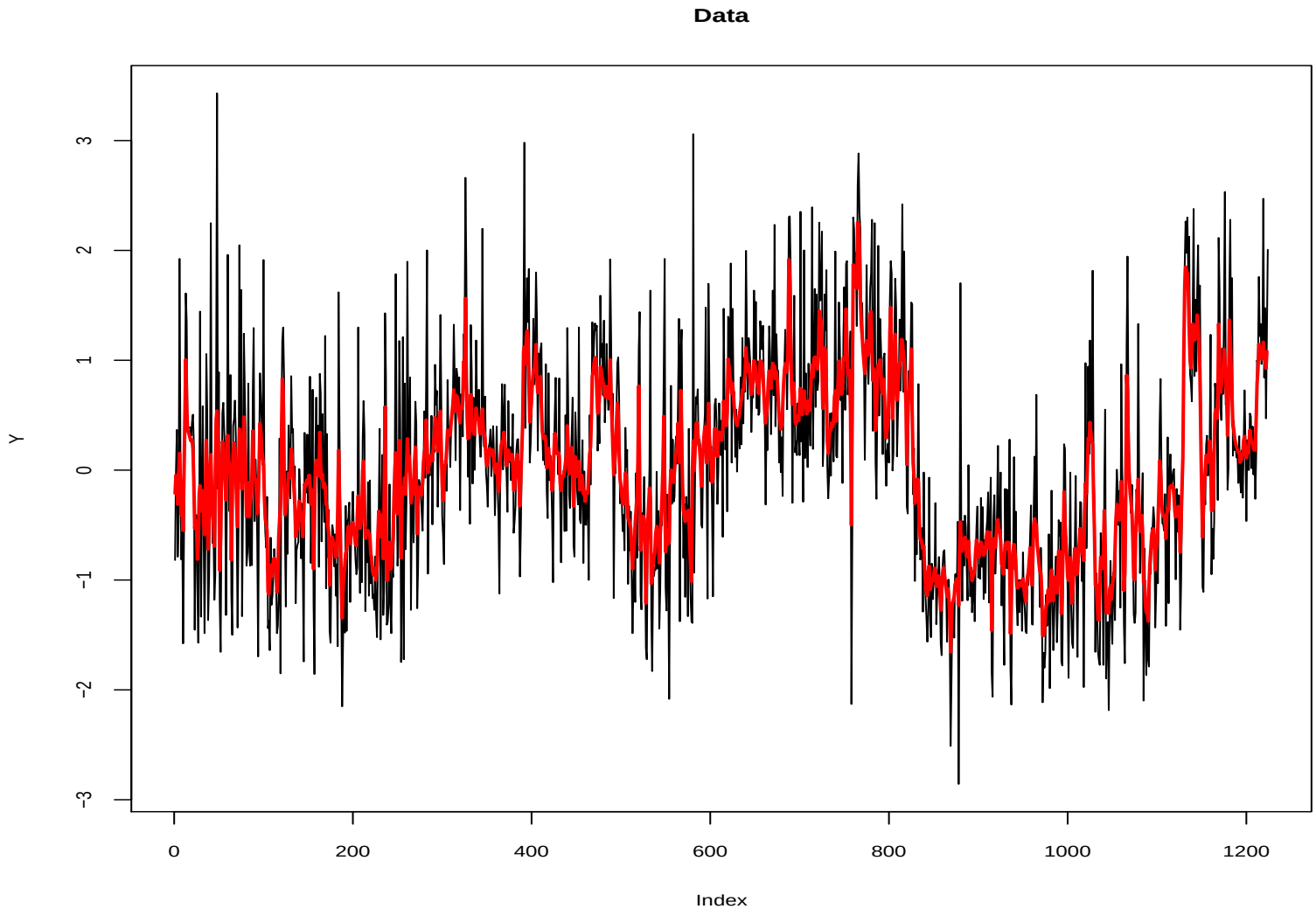
$$\tau_i = \begin{cases} Beta(a, b) & \text{with probbility p} \\ Beta(c, d) & \text{with probbility 1-p} \end{cases}$$

Interested in p:



A mixture of Beta distributions (II)

A "smoothed" signal of the data can also be obtained by looking at posterior distribution of e of $\boldsymbol{X}(\tau)|\boldsymbol{Y}$:

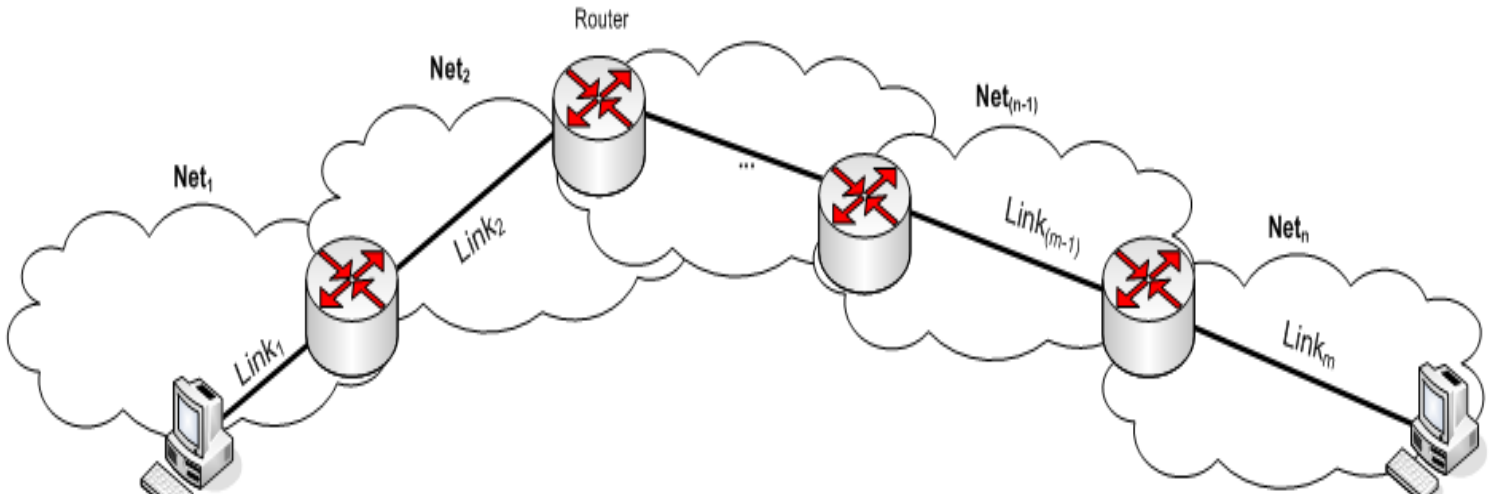


Modelling Internet Traffic

Joint work with Dimitris Pezaros

Modelling Internet Traffic

- Modelling Internet traffic is a huge field with researchers from (very) different research background (Computer Scientists, Physicists, Mathematicians, Statisticians etc).
- Highly complex problem $\rightarrow$ Very diverse enviroment (e.g. different protocols)
- Interested in Internet performance, and in particular delay:
  - **One-way delay**
  - **Round-Trip-delay**



Internet Performance Analysis

There has been a vast amount of literature on modelling internet traffic:

- Self-Similarity and Long-Range-Dependence are widely accepted facets of network traffic.
- Forward-Simulation Modelling.
- Markov Modulated Poisson Processes (MMPP).

Models based on Latent Branching Trees could be useful in identifying particular correlation structures.

Fitting Latent Branching Tree Models

Have looked at two datasets describing the one-way-delay of packets for two different wireless technologies:

- GPRS
- WLAN

We are interested in comparing the dependence structure of the observations for each of the two different datasets.

Within an LBT framework once we derive the posterior distribution of the divergence times for each dataset

$$\pi(\boldsymbol{\tau}_{gprs}|\boldsymbol{Y}_{gprs}) \text{ and } \pi(\boldsymbol{\tau}_{wlan}|\boldsymbol{Y}_{wlan})$$

and compare them appropriately.

For example, one can look at the probability mass around zero for each distribution etc.

End-to-End Transmission

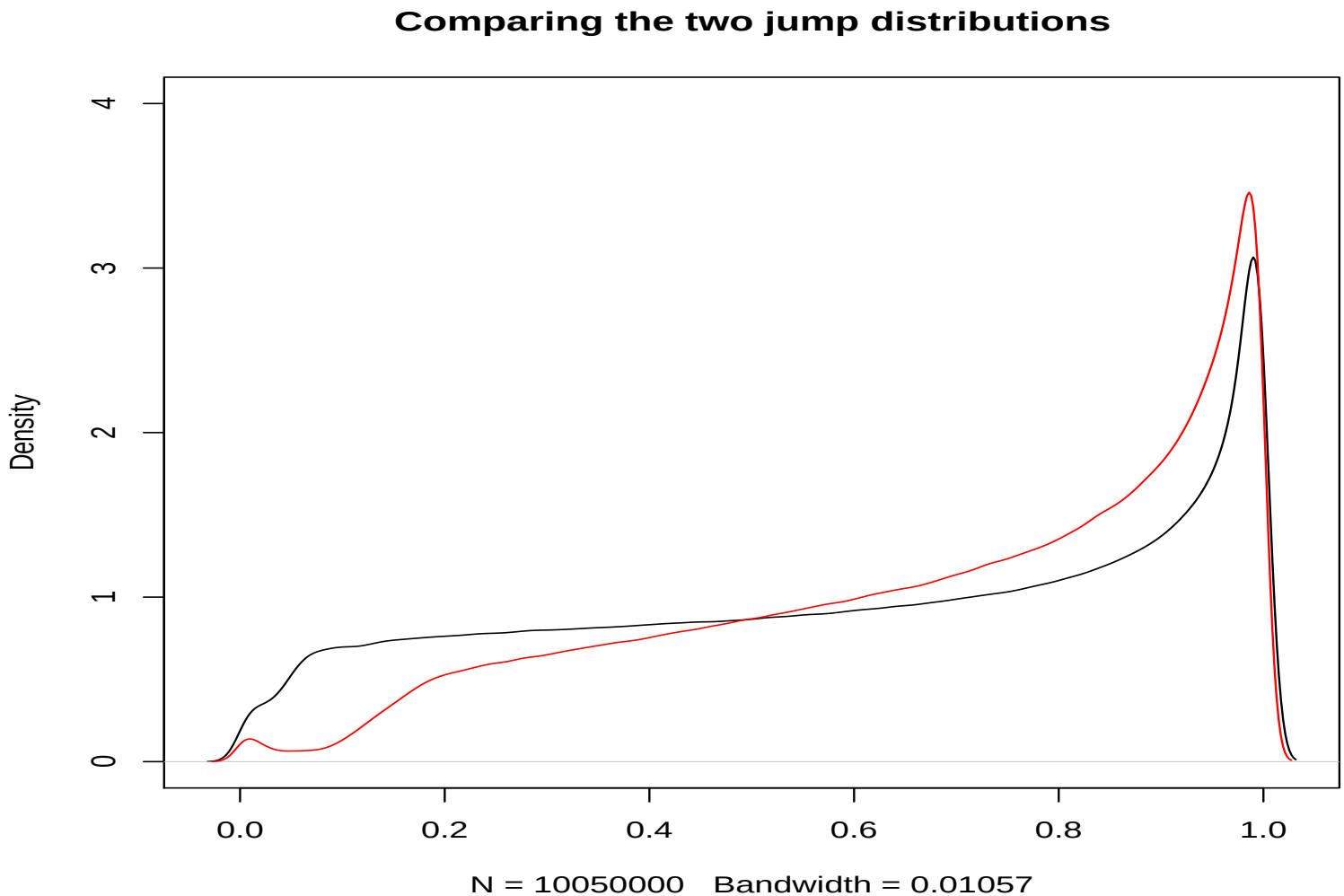


Figure: Jump distributions for the GPRS and WLAN datasets

End-to-End Transmission (2)

- An alternative way to compare the dependence structure between the two datasets is to derive a Monte Carlo estimate of the autocorrelation function (ACFs).
- That is very easy to do within this framework since

$$\mathrm{cov}(Y_1, Y_n) = \mathbb{E}[\min(\tau_1, \ldots, \tau_{n-1})]$$

and we have already drawn samples from $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_{n-1})$ and therefore, the covariance can be computed for any lag n.

An Alternative Model

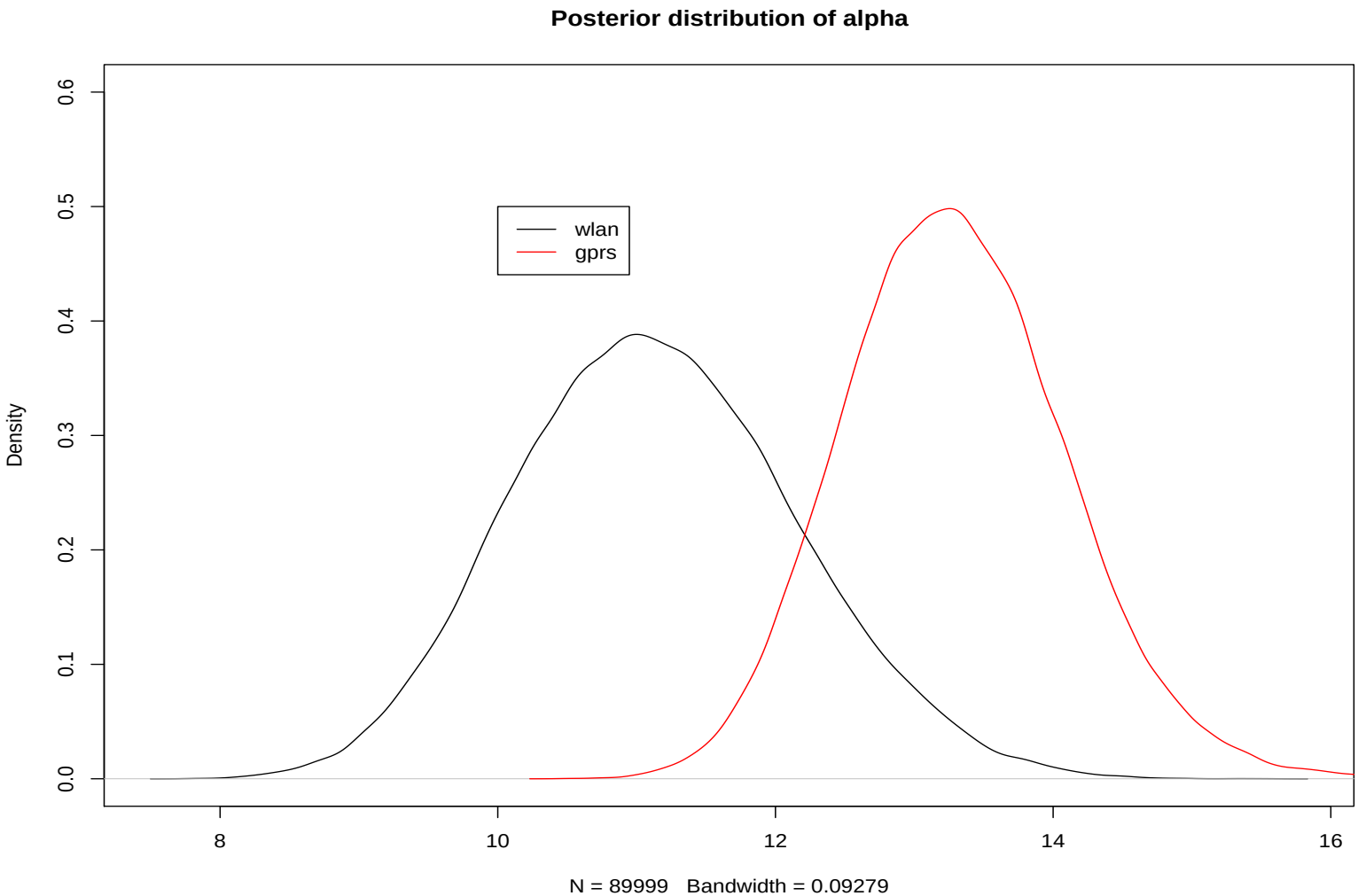
- In this example, so far, we have assumed *a priori* that the joint distribution of the divergence times (τ) follow a $U(0,1)$ distribution.
- Nevertheless, we can assume a different distribution, e.g. a Beta$(\alpha, 1)$ where α will be estimated from the observed data.‘
- Recall that in this case,

$$\text{cov}(Y_1, Y_n) \propto \frac{1}{n^{1/\alpha}}$$

  i.e. the rate of decay of the covariance at lag n decreases with α.

- Comparison of the estimates of α for the two different datasets will indicate the difference in their dependence structure.

Results from an Alternative Model: "Jump parameter" α



Conclusions & Further Work

Conclusions

A semi-parametric time series models has been introduced by using a *Latent Branching Tree* (LBT).

The LBT construction gives a very flexible way of defining a very rich covariance structure of observations.

Bayesian inference on an LBT can also be drawn by using MCMC.

There exist ways of drawing efficiently samples from the full posterior distribution by applying block-updates and/or Non−Centered Parameterisations.

Ongoing & Further Work

- **Methodology**
  - Margins outside the Gaussian context
  - Prediction
  - Multivariate setting
  - Bayesian Non-Parametric
  - Connection with change-point models
  - Algorithm's efficiency.
- **Applications**
  - Network traffic
  - Gene expression data - "jump distribution" as mixture.
  - ?