BNQDflow: A Bayesian non-parametric quasi-experimental design library using GPflow

Melle Starke
Faculty of Social Sciences
Radboud University
Nijmegen, Netherlands

Contents

Abstract

In this thesis we investigate the performance of a re-implementation of BNQD: a library for Bayesian Non-parametric Quasi-experimental Designs. BNQD functions as a Bayesian framework for quasi-experiments, and is capable of inferring causality under different assumptions than the ones required by a randomised controlled trial. We will compare the performance of this new implementation to the previous one, as well as examine the effectiveness of the features that were previously unavailable.

Keywords: Causal Inference, Quasi-experimental Designs, Bayesian Non-parametric Quasi-experimental Designs (BNQD), Regression Discontinuity Design, Gaussian Processes, Bayesian Model Comparison, GPflow, TensorFlow

1. Introduction

The ability to infer causality between multiple phenomena is paramount within all fields of science. And the most well known and common method for causal inference, the randomised controlled trial (RCT), has been proven to be a very useful tool. However, this method relies on a rigid experimental procedure to yield reliable results. Namely, one needs to divide the units or participants completely randomly into two groups. This random separation in tandem with the application of some treatment or condition to only one of the two groups makes it possible to accredit any differences between the two groups to the treatment (given that both groups are large enough to even out any confounding variables). These two requirements (i.e. random group assignment and sufficient group size) make it impossible to apply an RCT to certain situations. For instance, it might be unrealistic to obtain enough units, such as when investigating trends between different countries or continents. Additionally, it might be unethical or impossible to randomly divide units between the control and intervention group. For example, it is impossible to investigate the effect of a lock-down on the spread of COVID-19 via an RCT, since one cannot randomly divide citizens into a group that is affected by the lock-down and another group that is not. However, under certain other assumptions, it is possible to apply several quasi-experimental designs (QEDs) to infer causality even when an RCT cannot be applied (Campbell and Stanley, 2015; Marinescu et al., 2018). Recently, interest in QEDs as alternatives to RCTs has been renewed (Moscoe et al., 2015; Marinescu et al., 2018), and with the increase of time-series data in the current pandemic, this interest is likely to grow[1]. The most common type of QED is the regression discontinuity design (RDD) (Imbens and Lemieux, 2008b; Lee and Lemieux, 2010). RDDs can be applied to situations where units only receive the

1. See, for instance, the daily updated "Coronavirus Source Data" on ourworldindata.org: `https://ourworldindata.org/coronavirus-source-data`

treatment if they exceed some threshold value. In our example of investigating the effect of a lock-down on the spread of COVID-19, this threshold value would be the point in time when the lock-down was instantiated (do note that since the controlled variable is temporal, we would apply an Interrupted Time Series (ITS) analysis, which is slightly different from RDD in its assumptions (Wagner et al., 2002; Kontopantelis et al., 2015)). Although an RDD can estimate an effect in such situations, whether or not this effect is actually present and not simply due to noise in the data can only be determined with a statistical p-value test. This approach is only able to quantify evidence in favour of the alternative model, meaning that we can never know if the absence of a discontinuity is more likely than a presence. Alternatively, if we can apply Bayesian methods, we would be able to quantify the evidence in favour of either a continuity or a discontinuity (Wagenmakers, 2007). This idea was realised by Hinne et al. (2019), who developed the Python library "BNQD ", capable of performing RDD within a Bayesian framework. This implementation performs regression non-parametrically by virtue of Gaussian processes (GPs). In addition to their Bayesian nature, GPs are able to perform widely different regressions via their kernel method. This decision, in tandem with the specification of a continuous and a discontinuous model, allows us to reformulate the the problem of RDD into Bayesian model comparison (Raftery, 1995; Hinne et al., 2019). Consequently, this enables us to quantify evidence in favour of either model, as well as define the effect of the treatment as a probability distribution, as opposed to a single value with a significance test. However, this implementation is rather limited in its functionality, only offering a small number of benefits that are possible with this Bayesian reformulation. Additionally, the implementation depends on GPy, which is a GP library for Python that is not supported anymore (GPy, since 2012). In this thesis, we will cover a novel implementation of BNQD, which offers more Bayesian functionality than its predecessor, in addition to being built on a more efficient and currently supported GP library. Ultimately, the purpose of this library is to be an alternative to contemporary QED libraries. However, due to the limited time span of this project, the current end goal is to provide a library that implements the theory of BNQD, while providing several features that were unavailable in the previous implementation. These features should leverage this Bayesian approach to provide more representative conclusions regarding the effect of a treatment (sections 3.3, 3.7). Additionally, we will also address known weaknesses of the previous implementation (sections 3.5 and 3.6).

2. Theory

2.1 Regression Discontinuity Design

Through the application of an RDD, one attempts to find the causal effect of a treatment on some outcome variable y (Imbens and Lemieux, 2008a). This effect is known as the "treatment effect", "effect size", or "discontinuity" and will be indicated by τ. As opposed to in an RCT, treatment assignment in RDD is not randomised and instead dependent on some variable x called the "treatment variable" or "assignment variable" (Bloom, 2012; Lee and Lemieux, 2010). In sharp RDD, this dependency takes the shape of a simple threshold indicator (also known as the "intervention point"), which we will indicate by c (Imbens and Lemieux, 2008a). A unit is subsequently exposed to the treatment only if their assignment variable exceeds the intervention point. For the sake of simplicity, we also define a binary treatment indicator w, which is 0 if the unit precedes c 1 if the unit exceeds c.

In order for an RDD to be deemed valid, a number of assumptions must hold. In this paper, we will adopt the following assumptions. Though be mindful that the exact definitions of these assumptions vary slightly across different fields (Lee and Lemieux, 2010; Geneletti et al., 2015; Skovron and Titiunik, 2015; Hahn et al., 2001; Imbens and Lemieux, 2008b).

(A1) *Imprecise Control.*
Units are assumed to have, at best, imprecise control over the assignment variable. Meaning that units cannot deterministically influence their assignment group. In the classic example of Thistlethwaite and Campbell (1960), students are given merit awards depending on their scores on a certain test. And although test scores can most definitely be influenced by studying, one does not have precise control over their own test score. Note that this assumption can be tested by comparing the density of the data around the intervention point (Lee and Lemieux, 2010).

(A2) *Continuity.*
y is assumed to be continuous in x at c, conditioned on w.
This assumption results in any subsequent discontinuity in y being contributed to w. i.e. any difference in y at c is due to the effect of the treatment.

(A3) *Stationarity.*
For some stochastic process, the cumulative probability of some sequence of events is similar to the cumulative probability of another sequence with the same length, but at a different location (Park and Park, 2018):

$$F_X(x_{t_1}, \ldots, x_{t_n}) = F_X(x_{t_1+\tau}, \ldots, x_{t_n+\tau}) \quad \forall \tau, t_1, \ldots, t_n \in \mathbb{R} \wedge \forall n \in \mathbb{N} \tag{1}$$

Where $F_X(S)$ is the cumulative probability over the sequence S, x_{t_n} is the data point at time point t_n, and τ is some offset in the time domain. This essentially entails that the parameters of the latent function responsible for generating the data do not change over the span of the assignment variable. Therefore, we should theoretically be able to find the parameters of this latent function, if given enough data.

(A4) *Global Effect Size.*
Since we are only able to investigate discontinuities around the intervention point, we must assume that this effect is global over the range of the assignment variable (van Leeuwen et al., 2016). Otherwise, the results are not applicable to units far away from the intervention point.

Assumptions A1 and A2 have a particularly desirable consequence on the properties of the data. Due to the imprecise control of the assignment variable, units just below the intervention point are practically indistinguishable from units just above the intervention point. In tandem with the absence of a discontinuity in other variables around the intervention point, this results in an essentially random distribution of the data points around the intervention point.

Within RDD literature, it is commonly assumed that there exist two separate latent functions. One for units that received the treatment, and another for units that did not (Lee and Lemieux, 2010; Imbens and Lemieux, 2008a; Skovron and Titiunik, 2015). However, since we are not able to observe values from both functions simultaneously, we generally examine the average effect over the sub-populations and extrapolate the results at the intervention point. The average effect is captured by a single regression with the following shape (Imbens and Lemieux, 2008b):

$$f(x) = (1 - w) \cdot f_0(x) + w \cdot f_1(x) \tag{2}$$

Where f_0 is the regression over units that did not receive the treatment, and f_1 over the ones that did. This allows one to define the regression as two independent functions, where one is used before the intervention point, and the other after. The effect size is then defined as:

$$\tau = f_1(c) - f_0(c) \tag{3}$$

Though typically one applies a linear regression (Lee and Lemieux, 2010):

$$f(x) = \alpha + w\tau + \beta x + \epsilon \tag{4}$$

Where α is the bias, β is the slope, and ϵ is the error term. One fits the regression to the data by optimising parameters α, β and τ to minimise the error. The significance of the effect size can then be determined via a p-value null-hypothesis test.

2.2 Gaussian Process Regression

Of course, simple linear regression with an error term is merely one approach. Gaussian Process regression is one of the many alternatives. Though what makes GP regression so enticing is its flexible functional shape depending on the choice of kernel(s). Similarly to how probability distributions are distributions over vectors, a GP is a distribution over functions (Rasmussen, 2003):

$$f(x) \sim \mathcal{GP}\left(\mu(x), k(x, x')\right) \tag{5}$$

Where $\mu(x)$ is the *mean function* of the GP, and $k(x, x')$ is the *covariance function*, which takes the shape of a kernel method. This means that $f(x)$ is distributed according to some mean and some covariance, relative to x. This generalised definition allows one to define regressions with varying behaviours. Particularly through the selection of kernel methods, which also contain some number of hyper-parameters (such as the variance and the lengthscale) that specify the shape of the covariance function. (Duvenaud, 2014)[2].

Unlike in regular regression, a GP regression depends heavily on the data it describes.

2. Examples of kernel methods can also be found in the "Kernel Cookbook" by David Duvenaud `https://www.cs.toronto.edu/~duvenaud/cookbook/`

By defining the mean and covariance function of the GP, one effectively obtains a prior useable in Bayesian inference (Benavoli and Mangili, 2015; Rasmussen, 2003). This prior is subsequently updated with data via matrix transformations, resulting in the posterior GP (see appendix A).

This interaction between the mean function, covariance function, and the data means that posterior GP regressions can vary widely simply due to different data. However, the parameters of the functions distributed by the Gaussian process are determined by the GP's hyper-parameters, meaning that the hyper-parameters still very much affect the regression. Therefore, we aim to optimise the hyper-parameters when fitting the regression on the data.

2.3 Bayesian Non-parametric Quasi-experimental Design

In BNQD, the problem of RDD is reformulated into a Bayesian model comparison problem with GP regressions (Hinne et al., 2019). This has a number of consequences.

2.3.1 Model Specification

Since we wish to perform Bayesian model comparison as an alternative to a significance test, we need to define two models that describe the data, instead of a single regression with a "jump". We define a continuous model $\mathcal{M}_0$, which assumes no discontinuity, and a discontinuous model $\mathcal{M}_1$, which assumes a discontinuity at c. The continuous model takes the shape of a single GP. Whereas the discontinuous model consists of two GPs: the control regression $\mathcal{M}_{1_C}$ and the intervention regression $\mathcal{M}_{1_I}$. We define the complete data set as $\mathcal{D}$, and split it into two subsets: $\mathcal{D}_C$ for $\mathcal{M}_{1_C}$, and $\mathcal{D}_I$ for the $\mathcal{M}_{1_I}$, for which $\mathcal{D}_C \cup \mathcal{D}_I = \mathcal{D}$ and $\mathcal{D}_C \cap \mathcal{D}_I = \emptyset$. Note that data points can theoretically be exactly on $x = c$, making them no more part of $\mathcal{D}_C$ than of $\mathcal{D}_I$. However, if we were to include such data in both $\mathcal{D}_C$ and $\mathcal{D}_I$, $\mathcal{M}_1$ would contain duplicate data points and would not describe the same data as $\mathcal{M}_0$, making model comparison impossible. Alternatively, including such data in neither subset also results in $\mathcal{M}_0$ and $\mathcal{M}_1$ describing different data. In this re-implementation, we have simply chosen to only include such data in $\mathcal{D}_C$.

2.3.2 Effect Size in BNQD

Since we define $\mathcal{M}_1$ as two GP regressions that span the entirety of the assignment variable, we can extrapolate both the control and intervention regressions at the intervention point, resulting in two probability distributions over y (typically Gaussian distributions). By subtracting these distributions, we obtain a distribution over the effect size estimated by the discontinuous model:

$$p(\tau|\mathcal{M}_1) = p(y|\mathcal{M}_{1_I}, x = c) - p(y|\mathcal{M}_{1_C}, x = c) \tag{6}$$

Additionally, we are able to quantify the likelihood of the data under a certain model by multiplying the probability density of each data point under that model (where θ are the hyper-parameters of the model):

$$p(\mathcal{D}|\mathcal{M}, \theta) = \prod_{(x,y)\in\mathcal{D}} p(y|\mathcal{M}, x, \theta) \tag{7}$$

We can subsequently perform Bayesian model comparison between $\mathcal{M}_0$ and $\mathcal{M}_1$ (if we can obtain an accurate estimate of the marginal likelihood). This allows for the quantification of evidence in favour of *either* model, as opposed to a p-value null-hypothesis test which can only show evidence in favour of the alternative model (Wagenmakers, 2007). The evidence is quantified as the Bayes factor (BF), which shows evidence in favour of $\mathcal{M}_0$ if the below 1, and in favour of $\mathcal{M}_1$ if above 1 (Raftery, 1995):

$$\mathrm{BF}_{10} = \frac{p(\mathcal{D}|\mathcal{M}_1)}{p(\mathcal{D}|\mathcal{M}_0)} \tag{8}$$

By combining the estimated discontinuities of both models with the Bayes factor, we can obtain a more representative estimate of the effect size. This is done by computing the Bayesian model average (BMA) of the effect size given $\mathcal{M}_0$ and $\mathcal{M}_1$, which is simply the estimated effect size of each model weighed by their respective posterior probabilities (Hoeting et al., 1999).

$$p(\tau|\mathcal{D}) = \sum_{i \in \{0,1\}} p(\tau|\mathcal{D}, \mathcal{M}_i) p(\mathcal{M}_i|\mathcal{D}) \tag{9}$$

Where the estimated effect size under $\mathcal{M}_0$ (i.e. $p(\tau|\mathcal{D}, \mathcal{M}_0)$) is a spike at $\tau = 0$. However, for us to apply these methods, we need to somehow quantify the marginal likelihood of the data given the model. Theoretically, this is obtained by integrating over all possible values for hyper-parameters θ (Gelman et al., 2013). But since the hyper-parameters are continuous variables, this is nearly always analytically intractable (Rasmussen, 2003). Instead, we choose to approximate the marginal likelihood, which can be done with several different methods. One such method is the Bayesian Information Criterion (BIC), which was also used in the original BNQD implementation (Hinne et al., 2019). This method assumes that the number of data points is much larger than the number of hyper-parameters, that the hyper-parameters are optimised (indicated by $\hat{\theta}$), and that θ are distributed normally around $\hat{\theta}$:

$$\log p(\mathcal{D}|\mathcal{M}) \approx \mathrm{BIC}(\mathcal{D}, \mathcal{M}, \hat{\theta}) = \log p(\mathcal{D}|\mathcal{M}, \hat{\theta}) - \frac{l}{2} \log n \tag{10}$$

Where l is the number of hyper-parameters and n is the number of data points.
Although this method is computationally efficient, it can be a poor approximation of the evidence, since it does not capture our uncertainty in the parameters of the latent function. Instead, it simply subtracts a penalty term based on the number of parameters. Other methods do acknowledge this uncertainty by sampling the posterior distribution of the hyper-parameters θ, such as Hamiltonian Monte Carlo (HMC), and allow for another approach to approximating the evidence (Girolami and Calderhead, 2011; Heinonen et al., 2016). Due to the benefits that posterior hyper-parameter sampling can offer, it was made available in this library, and its implementation and performance can be found in sections 3.7 and 4.3 respectively.

2.3.3 Model Averaging over Multiple Kernels

Since we are able to approximate the evidence of both models under a certain kernel, we can combine these to obtain an approximation of the evidence of the kernel itself (Hinne et al., 2019). Subsequently, we can obtain a posterior distribution over multiple kernels K given the data:

$$p(k|\mathcal{D}) = \frac{p(\mathcal{D}|k)p(k)}{\sum\limits_{k \in K} p(\mathcal{D}|k)p(k)} \tag{11}$$

$$\text{Where: } p(\mathcal{D}|k) = \sum_{i \in \{0,1\}} p(\mathcal{D}|\mathcal{M}_i, k)p(\mathcal{M}_i|k) \tag{12}$$

We can then obtain a BMA effect size estimate over all kernels K by weighing their individual estimates by their posterior probability and summing the results. This can also be done for the discontinuous effect size specifically by calculating $p(k|\mathcal{D}, \mathcal{M}_1)$ instead of $p(k|\mathcal{D})$.

2.3.4 Properties of Gaussian Process Regression for QED

The possibility to apply these Bayesian methods is facilitated by our choice to use Gaussian processes, which, as mentioned previously, allow for very flexible functional shapes. Additionally, we are able to define a prior distribution over the hyper-parameters of a GP to quantify our initial belief in their approximate value, further expanding on this Bayesian framework (Rasmussen, 2003). This limits the scope of possible values for the hyper-parameters, allowing us to somewhat control the shape of the regression. Specifying priors is particularly useful when there are little data, but we have some knowledge regarding the latent function, for instance in the case of predicting seasonal animal populations.

2.3.5 Fuzzy Regression Discontinuity

Thus far we have discussed RDDs where only the units above the threshold are exposed to the treatment, but in some cases units might not comply with this rule. For example, some general practitioners might prescribe a cholesterol lowering drug to patients who have a cardiovascular disease risk score above 20% (the intervention point), while not prescribing it to patients with a risk score under 20% (Geneletti et al., 2015). Additionally, patients that are given a prescription may choose to simply not take the medication. RDD on such data is known as *fuzzy* RDD, and requires additional data manipulation and assumptions if one wants to conclude an effect size. Lee and Lemieux (2010) mention two distinct methods for recovering an effect size in fuzzy RDD:

- The first and most obvious method is to simply disregard all non-complying units. The regression is subsequently fitted on the remaining data:

  $$\mathcal{D}_{new} = \{u \mid u \in \mathcal{D},\ w_u = 1 \leftrightarrow u > c\} \tag{13}$$

  However, this results in the discard of potentially useful data.

- The second approach is to fit the regression normally and inflate the perceived effect size. This is achieved by fitting a sigmoid or polynomial on the treatment indicators of the data points, with a discontinuity at the intervention point. This results in an estimate of the treatment probability. If the process that generated the data is fully sharp (i.e. non-fuzzy), then the treatment probability is essentially a step-function, giving a discontinuity of 1 at c. But as it becomes more fuzzy, this discontinuity approaches 0. The underlying assumption is that non-compliant units reduce the perceived effect size by counteracting the effect of the compliant units. An estimate of the true effect size can subsequently be recovered by dividing the perceived effect size by the discontinuity in treatment probability. However, this method relies on the treatment probability regression being properly fitted on the data.

These methods are necessary for retrieving an effect size in fuzzy RDD, but only due to the fact that classic RDD applies a single regression to the data. Conversely, BNQD's alternative model $\mathcal{M}_1$ consist of two sub-regressions which both span the entirety of the assignment variable. Therefore, we can simply fit $\mathcal{M}_{1_I}$ on the units that received the treatment and $\mathcal{M}_{1_C}$ on the units that did not. This way, units influence their respective sub-regression just as much as they would if the data were sharp, without needing to perform additional data manipulation. However, one must be aware that a strong relation between the assignment variable and the treatment probability must remain for assumption A2 to hold, otherwise we cannot assume a random distribution of data around the intervention point.

3. Implementation

BNQDflow differs from the 2019 implementation of BNQD in a number of ways. In this section we discuss these differences, as well as the general implementation of BNQDflow and how to interact with it. The full implementation can be found on `https://github.com/MelleStarke/BNQD`.

3.1 GPflow and TensorFlow

BNQDflow uses GPflow as its Gaussian process library instead of GPy (Matthews et al., 2017; GPy, since 2012). GPflow is built on TensorFlow, which is able to efficiently perform calculations due to calculations being represented as graphs and evaluated as needed (Abadi et al., 2015). Computation can be sped up further if one has access to a graphics card with Nvidia CUDA support[3].

3.2 The GPMContainer Class

GPflow inherently offers several models that can perform GP regression, prediction, and the calculation of several Bayesian metrics. Such classes include the `GPR` (GP regression model), `SVGP` (sparse variational model), and `GPMC` (multi-chain Monte Carlo model) classes, but all are extensions of the `GPModel` class. BNQDflow's `GPMContainer` class wraps a number of `GPModel` objects and provides an interface between the user and the existing methods of the `GPModel` class, as well as some additional methods. By allowing the user to wrap any number of GP regression models, a `GPMContainer` can function as $\mathcal{M}_0$ by wrapping a single model, and as $\mathcal{M}_1$ by wrapping two.

3. `https://developer.nvidia.com/about-cuda`

```
class GPMContainer:

        def __init__(self,
                regression_source: Union[Kernel, List[GPModel]],
                data_list: Optional[List[RegressionData]] = None,
                intervention_points: Optional[List[Tensor]] = [],
                gpm_type: Optional[str] = 'gpr',
                likelihood: Optional[Likelihood] = Gaussian()
                share_params: Optional[bool] = True,
                ind_var_ratio: Optional[float] = 0.01
        )
```

The user can either pass a `gpflow.kernels.Kernel` object or a list of `gpflow.models.GPModel` objects as the `regression_source`. If a kernel is given, BNQDflow will generate a list of `GPModel` objects according to the `gpm_type` argument. The number of models generated depends on the length of the `data_list`, which contains one entry of `RegressionData` per model. Currently, the user can generate three types of GP regression models with the `gpm_type` argument: `GPR`, `SVGP`, and `GPMC`. Although within the BNQD framework we use at most a model with two regressions, a `GPMContainer` can contain any number of `GPModel` objects, and by extension any number of regressions. The list of `intervention_points` specifies the border(s) between each model, and the `ind_var_ratio` specifies the proportion of inducing points to real points if one is using sparse GP models.

3.3 Hyper-parameter Priors

Unlike the 2019 implementation, BNQDflow allows users to specify a prior distribution over the values of the hyper-parameters. This subsequently affects the shape of the regression and the possible solutions considered during optimisation. Specifying priors is handled by GPflow, and requires one to use a distribution from `tensorflow_probability` as well as cast floats to the proper type:

```
from tensorflow_probability.distributions import Gamma

gpf_flt = gpflow.utilities.to_default_float

model = GPMContainer(...)

model.kernel.variance.prior = Gamma(gpf_flt(4), gpf_flt(2))
model.kernel.lengthscales.prior = Gamma(gpf_flt(4), gpf_flt(2))
model.likelihood.variance.prior = Gamma(gpf_flt(4), gpf_flt(2))
```

3.4 Optimisation

The old implementation only allowed for optimisation of the hyper-parameters by maximising the maximum likelihood (ML) estimate via the SciPy optimiser. But since GPflow allows one to define priors over hyper-parameters, we can instead maximise the maximum a posteriori (MAP) estimate. Additionally, since BNQDflow is built on GPflow (and by extension on TensorFlow), it has access to and support for all TensorFlow optimisers. If one wants to use the `Adam` optimiser from TensorFlow, for instance, they would specify it as such:

```
from tensorflow.keras.optimizers import Adam

model = GPMContainer(...)

opt = Adam(learning_rate=0.01)

model.train(
    opt,
    max_iter = 1000,  # max nr. of training iterations
    loss_variance_goal = 0.02  # maximum variance in the past N loss values
                               # for training to stop early
)
```

If one chooses to use the `gpflow.optimizers.Scipy` optimiser, they can specify what method should be used via the `scipy_method` key argument of the `train` method.

Optimisation is performed by minimising the `GPModel._training_loss` method, which is defined as such:

$$L = -\left(\sum_{m\in M} \ln p(\mathcal{D}_m|m,\theta)\right) - \ln p(\theta) \tag{14}$$

Where M is the set of `GPModel` objects contained in the `GPMContainer`, and $\mathcal{D}_m$ are the data (i.e. `RegressionData`) described by model m. This definition is able to compute the training loss of a `GPMContainer` regardless of how many models it contains. The hyper-parameters θ are obtained via `GPMContainer.trainable_variables`, which recursively finds all `tensorflow.Variable` objects set as trainable.

3.5 Sharing Hyper-parameters

Obviously, by choosing to include more regressions in a `GPMContainer`, one increases the complexity of the model. For example, a container with two regressions (i.e. $\mathcal{M}_1$) will have a higher complexity than a container with one regression (i.e. $\mathcal{M}_0$) since it allows for a discontinuity, but also because the hyper-parameters of the first regression ($\mathcal{M}_{1_C}$) can be different from the ones of the second regression ($\mathcal{M}_{1_I}$). If we wish to investigate the possibility of a discontinuity, it is desirable for $\mathcal{M}_0$ and $\mathcal{M}_1$ to *only* differ in their allowance of a discontinuity. This means that we need to ensure that any and all regressions in the container use the same kernel and likelihood, and by extension the same hyper-parameters. By doing this, both $\mathcal{M}_0$ and $\mathcal{M}_1$ will have the same amount of hyper-parameters, while also ensuring that reported discontinuities are not due to non-stationarities (Hinne et al., 2019). However, the old implementation achieved this in a rather naive way: it trained both regressions of $\mathcal{M}_1$ separately and averaged their resulting hyper-parameters. BNQDflow addresses this issue by specifying a likelihood function for $\mathcal{M}_1$ used in optimisation:

$$p(\mathcal{D}|\mathcal{M}_1) = p(\mathcal{D}_C|\mathcal{M}_{1_C}) \cdot p(\mathcal{D}_I|\mathcal{M}_{1_I}) \tag{15}$$

This is a valid definition of the likelihood since we assume $\mathcal{D}_C$ and $\mathcal{D}_I$ to be independent (due to the assumption of a discontinuity), meaning that we can apply $P(A, B) = P(A) \cdot P(B)$.

Subsequently, BNQDflow forces both GPs to utilise the same hyper-parameters *before* the optimisation step by assigning the pointers to the hyper-parameters of all regression models to the hyper-parameters of the first one. This is automatically done with the `GPMContainer._ensure_same_params` method, which can also be called manually:

```
def _ensure_same_params(self, params: List[str]) -> None:
        for p in params:
            p = p.lower()

            if p in ['k', 'kern', 'kernel']:
                for i in range(1, self.n_models):
                    self.models[i].kernel = self.models[0].kernel

            elif p in ['l', 'lik', 'likelihood']:
                for i in range(1, self.n_models):
                    self.models[i].likelihood = self.models[0].likelihood

            elif p in ['mf', 'mean', 'mean function', 'mean_function']:
                for i in range(1, self.n_models):
                    self.models[i].mean_function = self.models[0]
                                                       .mean_function
```

3.6 Sparse Gaussian Process Support

Although Gaussian processes offer many benefits, they are infamous for their poor scalability with data set size (Cao et al., 2013; Snelson and Ghahramani, 2006). BNQDflow addresses this concern by allowing for sparse Gaussian process regression. Instead of performing regression on the entire data set, sparse Gaussian processes perform regression on a small subset of the data. This subset essentially functions as a summary of the data as a whole. Specifying that one wants to use sparse GPs in BNQDflow is done via the `gpm_type` argument of the `GPMContainer`. However, since sparse GPs perform regression on only the inducing points, we cannot directly calculate the likelihood. Instead, we use variational inference to calculate the evidence lower bound (ELBO) as an estimate (Hoffman and Johnson, 2016).

3.7 Posterior Hyper-parameter Sampling

Since we are able to calculate the likelihood *and* specify priors over the hyper-parameters, BNQDflow is able to sample the posterior distribution of the hyper-parameters via Hamiltonian Monte Carlo (HMC) sampling (Heinonen et al., 2016; Girolami and Calderhead, 2011). In contrast to ML and MAP optimisation, which only provide a single estimation of the value of a hyper-parameter, HMC sampling provides samples of the joint distribution over all hyper-parameters. This allows us to compare different estimates of the hyper-parameter values to each other. Additionally, we can use the hyper-parameter samples to more accurately estimate the posterior GP regression. Lastly, we can record the MAP values during the sampling process, and take the mean as an estimate of the evidence.

Do note that, although this approach to evidence approximation explicitly acknowledges our uncertainty in the values of the hyper-parameters (unlike the MAP or BIC approaches (MacKay, 1999)), it is well documented that this approach is far from perfect. For instance, the approximation is very sensitive to changes in the prior (Friel and Wyse, 2012; Robert and Wraith, 2009)[4]. Additionally, the evidence estimated by this approach generally has a very high variance, since there tends to be little overlap between the distributions of the prior and the likelihood, thus requiring a very large amount of samples to provide a proper estimate (Cheung and Beck, 2010).

3.8 Performing a BNQD Analysis

The `analyses.SimpleAnalysis` class provides an interface for comparing two `GPMContainer` classes, where one contains a single regression ($\mathcal{M}_0$) and the other contains two ($\mathcal{M}_1$). In addition to the methods of the `GMPContainer` class, the `SimpleAnalysis` class offers the following methods:

1. `log_bayes_factor`, which returns the log of the Bayes factor of $\mathcal{M}_1$ over $\mathcal{M}_0$ ($\log \mathrm{BF}_{10}$).
2. `posterior_model_probabilities`, which provides a categorical probability of the models given the data.
3. `get_effect_size`, which returns metrics of the perceived effect size. Assuming the class has access to an `EffectSizeMeasure` object.

3.9 Multi-kernel Analysis

In section 2.3.3 we discussed the possibility of fitting regressions with different kernels and combining the results. This is realised by the `analyses.MultiKernelAnalysis` class. This class contains a list of `SimpleAnalysis` objects, and offers additional methods:

1. `total_effect_size`, which gives the discontinuous and BMA estimates of the effect size, marginalised over the kernels. The actual evaluation of this is handled by the appropriate `EffectSizeMeasure` object.

4. Be mindful that the authors of these papers discuss the *harmonic* mean of the posterior densities, as opposed to simply the mean.

2. `posterior_kernel_probabilities`, which provides a categorical probability distribution of the kernels given the data (equation 11). This information is used to calculate the total effect size, as described above.

3. `plot_kernel_effect_sizes`, which plots the estimated effect sizes of all kernels.

4. `plot_kernel_probabilities`, which shows a bar plot of the posterior probabilities of the kernels given the data (equation 11). Within these bars, a distinction is made between the contribution by $\mathcal{M}_0$ and by $\mathcal{M}_1$. This allows one to see how likely these models are under a certain kernel.

3.10 Effect Size Measures

BNQDflow calculates the effect size of a treatment in the same fashion as the old implementation. However, BNQDflow implements this in a disjointed fashion, namely via a visitor pattern. This allows one to define new effect size measures that calculate the effect size differently for different `bnqdflow.analyses.Analysis` objects, while still being able to share fields and methods.

The only effect size measure currently implemented in BNQDflow is the `effect_size_measures.Sharp` class, though it works for both the `SimpleAnalysis` and the `MultiKernelAnalysis` class. The `Sharp` effect size measure quantifies the effect size the exact same way as the 2019 implementation did.

4. Simulations

In this section, we will perform several experiments on simulated data: two data sets generated the same way as by Hinne et al. (2019), one small data set with an underlying trend, and one very large data set. Additionally, we will examine the performance of posterior hyper-parameter sampling on the second data set.

4.1 Discontinuity in a Linear Model

4.1.1 Method

The data will be generated the exact same way as was done in the experiment by Hinne et al. (2019):

$$\begin{aligned} x_i &\sim \mathcal{U}(-3, 3) \\ \epsilon_i &\sim \mathcal{N}(0, \sigma^2) \\ y_i &\sim 0.3x_i + \tau_{true}[x_i > c] + \epsilon_i, \quad i = 1, \ldots, N \end{aligned} \tag{16}$$

Where $\sigma = 1$, $c = 0$, $N = 100$, and $\tau_{true} \in \{2^{-2}, 2^{-1}, \ldots, 2^2\}$ is the true discontinuity at c. The experiment will be run 100 times with different random seeds. We will measure the log Bayes factors and expected effect size estimates (both discontinuous and BMA) under different kernels: linear, exponential, squared exponential (RBF), and Matern32.

For all kernels, we assign priors to the kernel variances (s_k), kernel lengthscales (l_k), and likelihood variances (s_l) as such:

$$\begin{aligned} p(s_k) &= \Gamma(1.5, 0.5) \\ p(l_k) &= \Gamma(1.5, 0.5) \\ p(s_l) &= \Gamma(1.5, 0.5) \end{aligned} \tag{17}$$

Note that priors are only necessary in section 4.3, but by assigning them at earlier experiments, we can obtain more comparable results.

We expect that the result will be similar to the one in the 2019 paper, since the theory is fundamentally the same.

4.1.2 Result

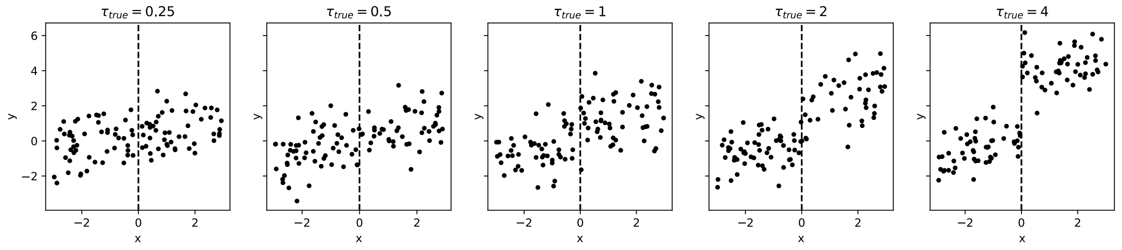

Figure 1: Example linear data sets with different discontinuities (τ_{true}), used in the linear model experiment (section 4.1).

When looking at figure 2 and comparing the results to the ones of the 2019 paper, we can immediately see a difference in the log Bayes factor plot. In the 2019 paper, the means of the log Bayes factors only surpass 0 after a true discontinuity of 1. However, in our results, this already happens at roughly $\tau_{true} = 0.5$. Since all conditions have some discontinuity present, one could argue that these results show the ability of the implementation to detect the presence of a discontinuity, even under high noise levels. However, one can also argue that it should be very difficult to detect a discontinuity in data with such a low signal to noise ratio, implying that the model is incorrect in its detection of a discontinuity. Since both implementations apply the same theory, the difference in outcomes more than likely results from a difference in implementation. The most likely explanation is that the old implementation assigns too little evidence to the discontinuous model due to its naive approach to optimising the hyper-parameters of $\mathcal{M}_1$ (see section 3.5), subsequently resulting in a lower overall Bayes factor.

The expected discontinuity plot also shows a stark difference to the one in the 2019 paper. Here, we see that the difference in expected effect sizes between the discontinuous

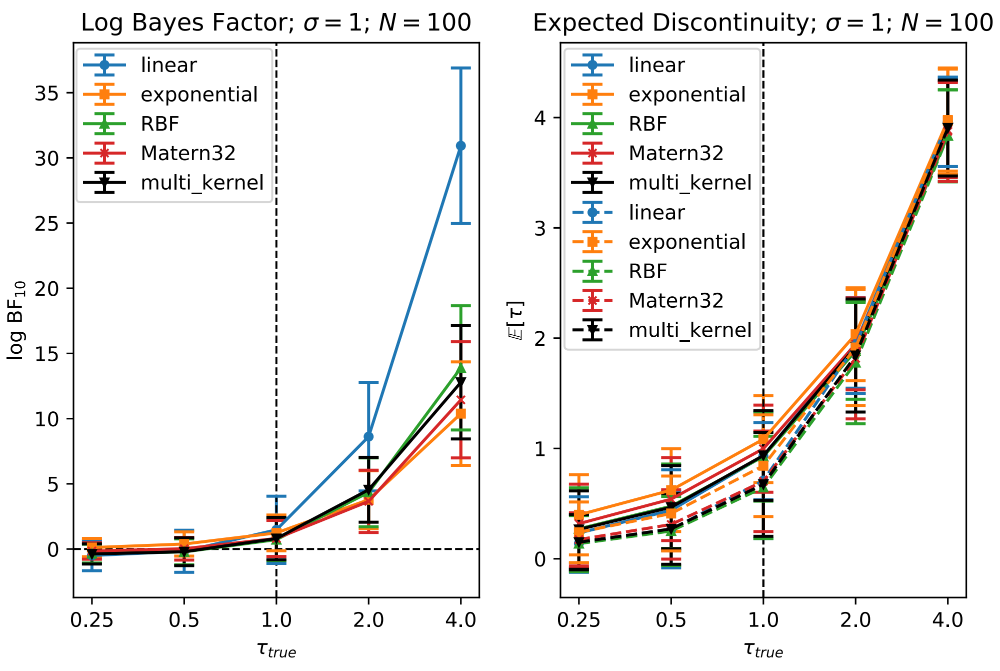

Figure 2: Results of the linear model experiment (section 4.1). The error bars show the standard deviation over the results. **Left:** the log Bayes factors across different discontinuities and under different kernels. The "multi_kernel" bars show the combined log Bayes factors of all kernels weighed by their posterior probabilities. **Right:** the expected effect size ($\mathbb{E}[\tau]$) across different discontinuities and under different kernels. The filled lines show the predictions under the discontinuous models, and the dotted lines show the BMA estimates.

models and BMAs is much smaller than in the 2019 paper. In the 2019 paper, the BMA predictions consistently underestimate the effect size, and only converge with the discontinuous estimates at $\tau_{true} = 4$. In figure 2, we see can see that the BMA predictions still underestimate the true discontinuities, but much less so. This is expected when considering the difference in Bayes factors, resulting in the predictions of the discontinuous models being weighed heavier than in the 2019 paper.

However, aside from these two differences, the results are very similar to the ones in the 2019 paper.

4.2 Discontinuity in a Nonlinear Model

4.2.1 Method

Again, the data will be generated the exact same way as was done in the experiment by Hinne et al. (2019):

$$\begin{aligned} x_i &\sim \mathcal{U}(-3,3) \\ \epsilon_i &\sim \mathcal{N}(0,\sigma^2) \\ y_i &\sim 1.2 + 2\sin\left(\frac{\pi}{3}x_i\right) + \tau_{true}[x_i > c] + \epsilon_i, \quad i = 1,\dots,N \end{aligned} \tag{18}$$

Where $\sigma = 1$, $c = 0$, and $\tau_{true} \in \{2^{-2}, 2^{-1}, \dots, 2^2\}$ is the true discontinuity at c. Our approach and expectations are identical to the ones of the linear model experiment (section 4.1).

4.2.2 Result

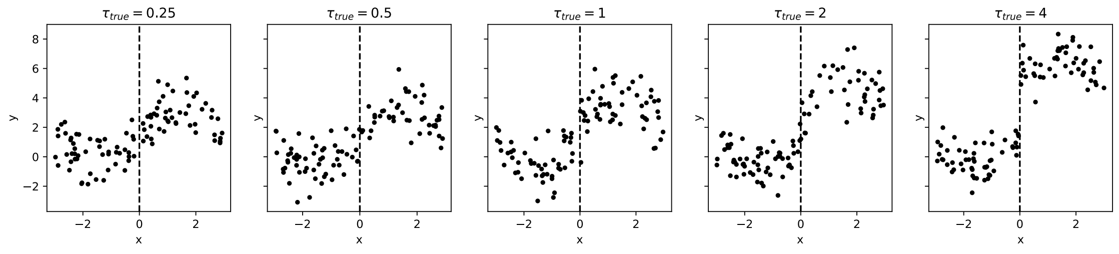

Figure 3: Example non-linear data sets with different discontinuities (τ_{true}), used in the non-linear model experiment (section 4.2).

Just as in section 4.1, we can see that most of the log Bayes factors in figure 4 are higher here than in the 2019 paper. However, the log Bayes factors under the linear kernel are substantially lower than in the 2019 paper. The fact that the linear kernel results in a higher log Bayes factor than the other kernels is expected, since having two separate regressions provides a large increase to the flexibility of a linear regression, one that it desperately needs on these data from a non-linear latent function. However, the linear log Bayes factors being so much smaller than in the 2019 paper is not expected. As theorised in 4.1, the new approach to optimising $\mathcal{M}_1$ should result in a better fit, and subsequently a higher log Bayes factor. Why this is not the case for the linear kernel is unclear. Regardless, the log Bayes factors under the multi-kernel approach show that the results of the linear kernel are barely considered, indicating that the posterior probability of the linear kernel is extremely low. Therefore, this difference with the 2019 paper may be of little significance.

In the expected discontinuity plot, we again see that the difference between discontinuous and BMA estimates is lower than in the 2019 paper. Additionally, all kernels show

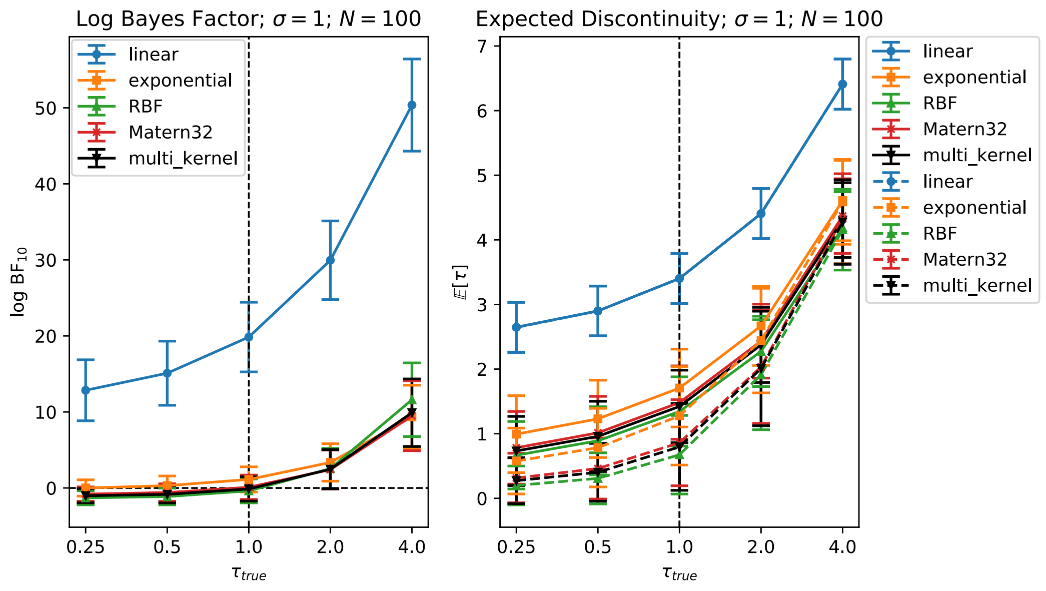

Figure 4: Results of the non-linear model experiment (section 4.2). The error bars show the standard deviation over the results. **Left:** the log Bayes factor across different discontinuities and under different kernels. The "multi_kernel" bars show the combined log Bayes factors of all kernels weighed by their posterior probabilities. **Right:** the expected effect size ($\mathbb{E}[\tau]$) across different discontinuities and under different kernels. The filled lines show the predictions under the discontinuous models, and the dotted lines show the BMA estimates.

higher standard deviations in their effect size estimates than in the 2019 paper, indicating less certainty of the expected effect size. We can also see that the discontinuous predictions overestimate the true effect size, which does not appear to be the case in the 2019 paper. Lastly, the predictions of the discontinuous model under the linear kernel are significantly higher than in the 2019 paper. However, the multi-kernel results show that the contribution of the linear kernel is, again, negligible.

4.3 Posterior Hyper-parameter Sampling

4.3.1 Method

The data used for this experiment will be generated identically to the data in the non-linear model experiment (see equation 18). But in addition to optimising the model using the MAP estimate, we sample the posterior distribution of the hyper-parameters and use these results in our calculation of the Bayes factor and effect size. We also report three different estimates of the model evidence:

1. The MAP approach (i.e. posterior density, or $p(\mathcal{D}|\mathcal{M}, \theta) \cdot p(\theta|\mathcal{M})$).

2. The BIC approach.
3. The HMC sampling approach (where we take the average of all MAP values obtained during the sampling process as an estimate of the evidence).

Unfortunately, the current implementation seems to only be able to perform posterior hyper-parameter sampling under the RBF kernel. Therefore, the experiment will only be performed with this kernel.

The results are averaged over 50 iterations and with 50 data points, in order to reduce run time. Additionally, 500 HMC samples are drawn per iteration. The performance of the HMC approach will be compared to the BIC estimates (trained on the same data, with the same number of iterations).

We expect that the estimated HMC effect size will be closer to the true effect size, but will likely have a higher variance, as explained in section 3.7. We also expect that the MAP approach results in the highest estimate of the evidence, as this approach tends to overestimate its certainty in the approximation (MacKay, 1999; Friel and Wyse, 2012).

4.3.2 Result

In the left plot of figure 5, we can immediately see a high standard deviation of the log Bayes factor estimated by the HMC samples. As discussed in section 3.7, the HMC approach tends to have a high standard deviation in its approximation of the evidence, extending to a high standard deviation in the Bayes factor. Additionally, the HMC approach explicitly defines our uncertainty in the values of the hyper-parameters, as opposed to the MAP and BIC approaches. This uncertainty is likely to translate into a higher uncertainty in the evidence, and by extension, the Bayes factor. Of note is that the standard deviation in the Bayes factor at $\tau_{true} = 0.25$ and $\tau = 0.5$ is particularly high, though exactly why the true discontinuity has such a drastic effect on the standard deviation of the log Bayes factor is unclear. Lastly, we can also see that the HMC approach results in lower log Bayes factors after $\tau_{true} = 1$.

In the middle plot, we see that the HMC approach does indeed result in a larger standard deviation of the estimated effect size. Though, curiously, this spread is significantly smaller than the spread of the log Bayes factors, even though the log Bayes factor is included in the calculation of the BMA effect size. Lastly, we can see that the HMC approach is not in fact closer to the true effect size, as we expected. Instead, it under estimates the effect size compared to both other approaches, starting from $\tau_{true} = 1$. This is more than likely due to the lower log Bayes factor, which assigns more weight to the continuous model than in the MAP optimisation approach.

In the right plot, we can see that it is in fact the HMC sampling approach that results in the highest overall evidence estimates, not the MAP approach. However, since we do not have access to the true value of $p(\mathcal{D})$, we do not have a base-line to compare the estimates to, and therefore cannot be certain which one is closer to the true evidence.

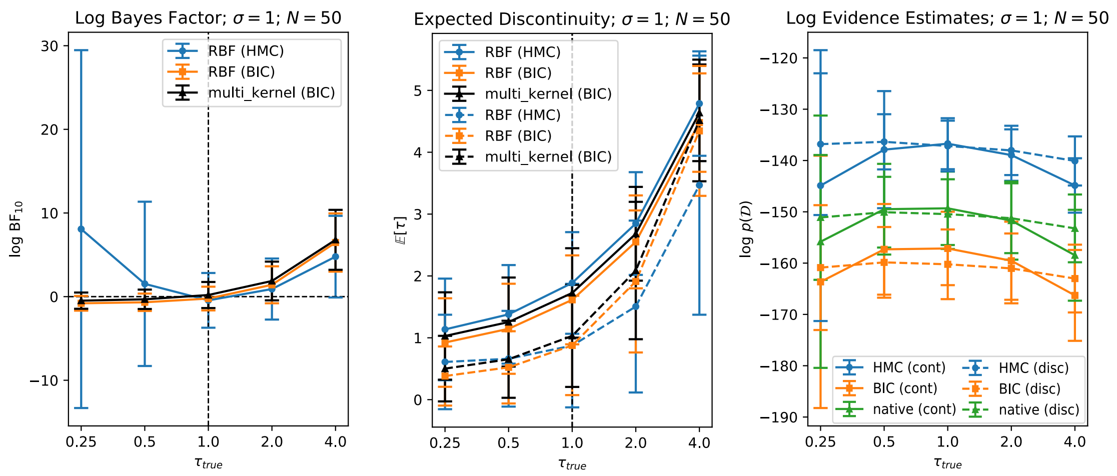

Figure 5: Results of the posterior hyper-parameter sampling experiment (section 4.3). The error bars show the standard deviation over the results. **Left:** Log Bayes factors across different discontinuities. The black error bars show the multi-kernel results of the linear, exponential, RBF, and Matern32 kernels. The blue error bars show the log Bayes factors obtained from the posterior hyper-parameter samples. **Middle:** Expected effect sizes across different discontinuities. The blue error bars show the estimated effect sizes obtained via the posterior hyper-parameter samples. **Right:** Evidence estimations across different discontinuities, using different estimation methods. Both the estimated evidences of the continuous and discontinuous models are shown.

N.B.: Sometime after running this experiment, the HMC sampling functionality broke down. The results above were fortunately recovered before it completely broke down, but keep in mind that this might impugn their validity.

4.4 Hyper-parameter Priors

4.4.1 Method

Our data will be generated similarly to the other experiments, but with the following definition of y_i:

$$y_i \sim 1.7\sin(0.7\pi x) + \tau_{true}[x_i > c] + \epsilon_i \tag{19}$$

With $\sigma = 0.7$ and $\tau_{true} = 1$.
The prior distribution over the lengthscales is defined as such:

$$\begin{aligned} p(k_l) &= \mathcal{N}(\mu, 0.1) \\ \mu &\in \{0.2, 0.5, \ldots, 2.0\} \end{aligned} \tag{20}$$

The small standard deviation in the lengthscale prior was chosen to minimise the probability density under negative lengthscale values, while still allowing a low prior mean. Another method to avoid this is to use a different distribution (such as a Gamma distribution), but since GPflow puts priors in a constrained space, and the location parameter of a Gamma distribution is harder to interpret, a Gaussian distribution seemed more appropriate.

We will generate two separate data sets: one with $N = 100$ ($\mathcal{D}_{100}$), and one with $N = 20$ ($\mathcal{D}_{20}$). We will fit regressions on the data with differently defined priors over the lengthscales of the kernels, and then compare the differences between the data sets. We expect that the log Bayes factors and effect size estimates remain the same across different lengthscale priors for $\mathcal{D}_{100}$, since the data will likely outweigh the priors. We also expect that, at a certain prior, effect size estimates for $\mathcal{D}_{20}$ are similar to the ones for $\mathcal{D}_{100}$. This expectation cannot necessarily be applied to the log Bayes factors, however. This is because a larger amount of data generally results in more extreme Bayes factors, since there is more information available to compare the models on.
Again, we average the results across 100 trials. However, we will only apply the RBF, exponential, and Matern32 kernels. Since the linear kernel does not have a lengthscale parameter.

Note that we simply use the log posterior density instead of the BIC score, since the BIC score requires a high ratio of data to parameters to be a proper estimate.

4.4.2 Result

In figure 7, we can indeed see some differences between $N = 100$ and $N = 20$. Firstly, the standard deviations in the log Bayes factor plots are generally larger for $N = 100$. This effect is particularly noticeable for lower lengthscale means. However, when compared to the error bars in figure 3 for $\tau_{true} = 1$, the differences are minimal, with both having a range of roughly 5. The smaller error bars for $N = 20$ are unsurprising, since less data points result in less evidence in favour of either the continuous or discontinuous model. We can also see that the log Bayes factor for $N = 20$ increases as the lengthscale prior increases. This is more than likely due to the continuous model being flexible enough to fit the data nearly as well as the discontinuous model when given smaller lengthscales. Curiously, we can see that for both $N = 100$ and $N = 20$, the RBF kernel shows a much higher mean and standard deviation in the Bayes factor starting from roughly $\mu(p(l_k)) = 1.4$, which is not replicated by the other kernels. Though the multi-kernel results show that the RBF kernel does not fit the data very well. This sudden increase in the log Bayes factor is likely due to $\mathcal{M}_0$ fitting the data poorly with such a high lengthscale, while $\mathcal{M}_1$ can fit the data much better thanks to its discontinuity at c (not unlike the results of the linear kernel in section 4.2). Normally, such effects of the prior would be overshadowed by the likelihood if $N = 100$. But in this instance, the prior over the lengthscale has such a small standard deviation that its influence persists even when $N = 100$.

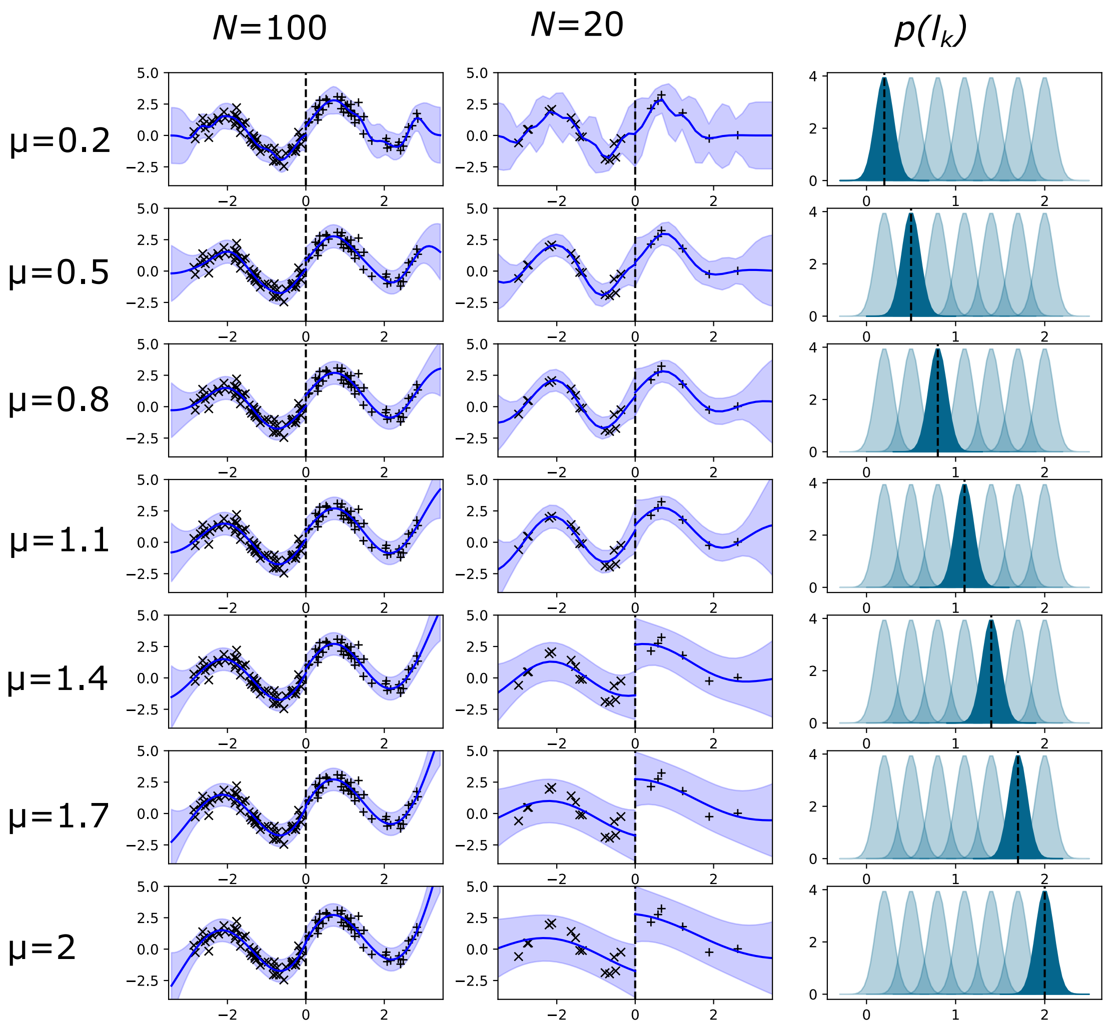

Figure 6: Discontinuous regressions fitted on an example data set with different lengthscales. Related to the hyper-parameter prior experiment in section 4.4. All models were given a lenthscale equal to the mean of the prior distribution over the lengthscale (i.e. $\mu(p(l_k))$) prior to training. The lengthscale was set to non-trainable, and the models were subsequently optimised using the MAP estimate. **Left:** example $\mathcal{M}_1$ fits on $\mathcal{D}_{100}$ for different lengthscale values. **Middle:** example $\mathcal{M}_1$ fits on $\mathcal{D}_{20}$ for different lengthscale values. **Right:** the relevant prior distribution over the lengthscales. The dotted black line shows the value of $\mu(p(l_k))$.

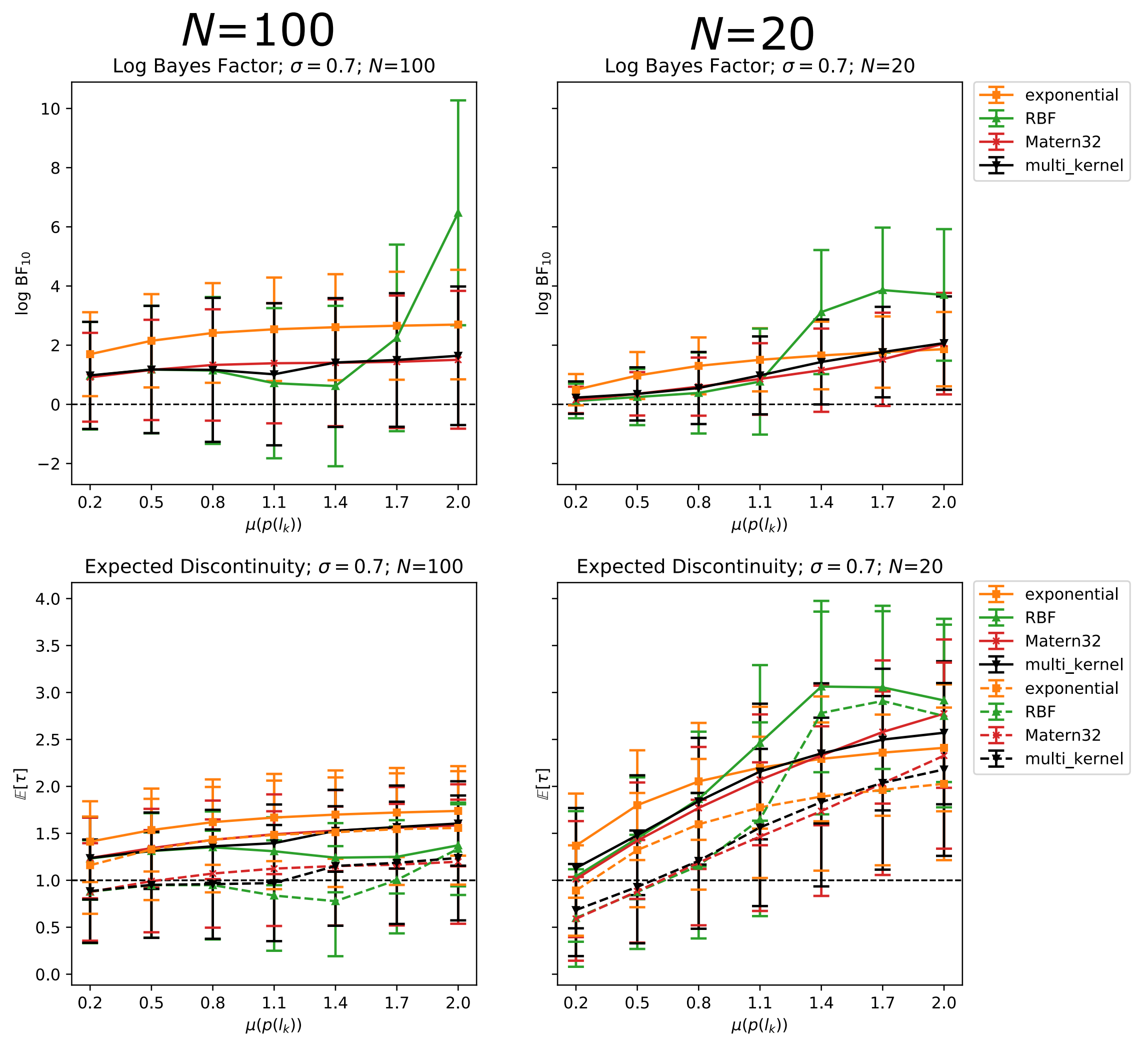

Figure 7: Results of the hyper-parameter prior experiment (section 4.3). The error bars show the standard deviation over the results. The left column shows the results for $\mathcal{D}_{100}$, and the right for $\mathcal{D}_{20}$. The first row shows the log Bayes factors, and the second the expected discontinuities. All results are plotted across different means of the lengthscale prior $(\mu(p(l_k)))$. The filled lines in the expected discontinuity plots are the predictions of the discontinuous model, and the dotted lines are the BMA estimates. The thin, horizontal, dotted line in the expected discontinuity plots represents τ_{true}.

Secondly, when looking at the expected discontinuity plots, we can see that with $N = 20$, the discontinuity predicted by the BMA multi-kernel approach crosses τ_{true} between $\mu(p(l_k)) = 0.5$ and $\mu(p(l_k)) = 0.8$. However, as the prior mean increases, the BMA multi-kernel prediction approaches $\mathbb{E}[\tau] = 2$. This overestimation of the effect size is likely due to c being at the steepest part of the latent function. And without sufficient data points, the GPs are not likely to properly fit the slope, and instead attribute the differences to a 0th-order discontinuity. However, although BNQDflow approximates τ_{true} at some definition of the lengthscale prior, being able to determine a proper definition of the prior ahead of time remains a difficult task, if not simply guesswork. If one wishes to fit BNQDflow on data with a known, underlying period, it is recommended to try different priors, and select the ones that result in a regression that picks up this trend.

Thirdly, the log Bayes factors and expected discontinuities for $N = 100$ do indeed barely change with the lengthscale prior (as we predicted). However, we do see a gradual increase in the standard deviation of the log Bayes factor, as well as a gradual increase in the expected discontinuity.

Lastly, although a minor remark, we can see that under high lengthscale prior means and $N = 20$, the RBF kernel seems to suddenly overestimate the expected discontinuity. This is likely due to the sub-regressions having such high lengthscales that they nearly fit a straight line through the data. This is corroborated by figure 6, where the plots in the middle column start to overestimate the discontinuity starting from $\mu = 1.4$.

4.5 Sparse Gaussian Processes

4.5.1 Method

We will generate data according to equation (19), but with $N = 1000$, $\tau_{true} = 2$, and $\sigma = 1$. We will the apply sparse GPs, while varying the number of inducing points: $N_{ind} = \{1000 \cdot r \mid r \in \{10^{-2}, 10^{-3/2}, 10^{-1}, 10^{-1/2}, 10^{0}\}\}$ (or $N_{ind} = \{10, 31, 100, 316, 1000\}$). We will then compare the log Bayes factors (computed using the ELBO scores), effect size estimates, and run times of the conditions.

We expect that the effect size and log Bayes factor will converge starting from $n_{ind} = 100$, and that the run time grows exponentially as n_{ind} increases.

Note that in this experiment we average over 50 trials as opposed to 100, in order to reduce the total run time of the experiment.

4.5.2 Result

In figure 9, we can see that the log Bayes factor means stay consistent across different values of n_{ind}. The linear kernel seems to give particularly consistent results. Such results might raise suspicion regarding the validity of the experiment, but it may be possible that the simplicity of the linear kernel, combined with the consistency of the external data across different condition, outweigh the effects that n_{ind} has on the regression.

Similarly, the expected discontinuity of the linear kernel remains very consistent across different values of n_{ind}. Again, this may be due to a theoretical technicality of applying a linear kernel to a sparse GP. Unlike as we predicted, the expected discontinuities do not converge at $n_{ind} = 100$, but instead at $n_{ind} = 316$. Curiously, we can also see a dip in the

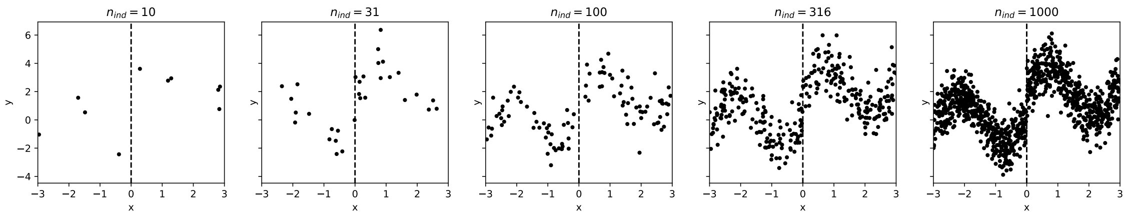

Figure 8: Example data sets of different inducing points. Note that all models will have the same amount of external data (i.e. 1000 data points), while only varying in inducing points. Related to section 4.5.

BMA predictions at $n_{ind} = 100$. Exactly why this happens is not clear, but it may be due to some bias in the generated data, or due to a technicality with MAP optimisation.

We hypothesised that there would be an exponential relation between n_{ind} and run time, and the training durations plot does seem to show such a relation. However, since the complexity of fitting a GP regression is $O(n^3)$, this relation is more than likely cubically (Rasmussen, 2003)). We can also see slight differences in error bars between the conditions of the training durations plot. These are likely due to inconsistent usages of the PC while the experiments were run. The results for $n_{ind} = 100$ were calculated overnight, resulting in much smaller error bars. Therefore, any conditions with larger error bars can be interpreted as having their true run times lower than the mean.

5. Applications

In this paper, we put an emphasis on evaluating the performance of the implementation on simulated data, with the aim of retrieving a known effect. However, we still need to examine whether or not this performance generalises well to real data. Naturally, we would like to compare the performance of BNQDflow to the 2019 implementation in regards to real data. However, since the French municipality data set violates assumption A1, and the Dutch 2017 parliament election data requires a 2-dimensional GP input space (which is currently not inherently supported in BNQDflow), we will only apply BNQDflow to the Sicilian smoking ban data.

5.1 Effects of the Sicilian Smoking Ban

In January 2005, Italy installed a national smoking ban in public places, with the aim of reducing adverse effects on health due to second hand inhalation Richiardi et al. (2009). Several studies have investigated the effects of this measure, but the differences in conclusions have prevented the formation of a clear consensus (Barone-Adesi et al., 2011). The data consist of recorded hospital admissions for acute coronary events (ACEs) across different months, ranging from January 2002 to November 2006. Hinne et al. (2019) found a log Bayes factor of −2.9 under both the linear and RBF kernels, and a log Bayes factor

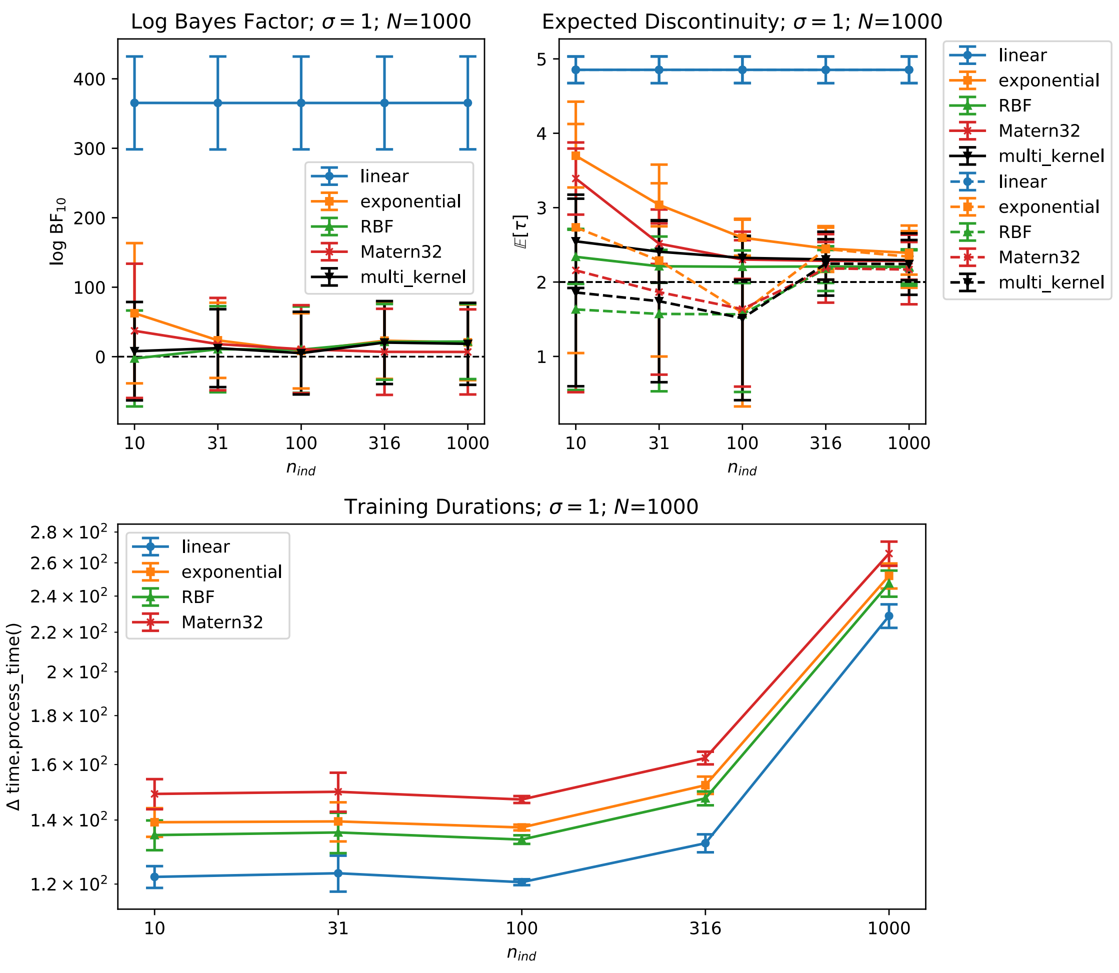

Figure 9: Results of the sparse GP experiment (section 4.5). The error bars show the standard deviation over the results. **Top left:** log Bayes factors for different kernels and different amounts of inducing points. **Top right:** expected discontinuities for different kernels and different amounts of inducing points. The thin, horizontal, dotted line represents τ_{true}. **Bottom:** run times of the training step (done via MAP optimisation) for different kernels and different amounts of inducing points. The run times are measured as the time difference of the `process_time()` function of the `time` Python module, which is the sum of the system and user CPU time. The run times are plotted on a logarithmic scale, since n_{ind} increases exponentially, and we are interested in the possible exponential relation between n_{ind} and run time.

of -3.3 under the multi-kernel approach. This shows ”moderate to substantial evidence in favour of the continuous model” (Hinne et al., 2019). Consequently, the BMA effect size plots under the linear and RBF kernels both showed a high probability density for $\tau = 0$. However, we expect that the log Bayes factors will be higher in our experiment, as we have previously seen to be the case for the simulations. Again, this may be due to the different approach to sharing hyper-parameters in the discontinuous model, or perhaps some other implementational difference.

5.1.1 Method

As was done in the 2019 paper, we will apply both a linear and an RBF kernel to the data. However, we will also apply a Matern32 and exponential kernel, and use the `MultiKernelAnalysis` class to determine the most likely outcome, marginalised over all kernels. We will report the log Bayes factors and effect size plots of all kernels (both discontinuous and BMA), as well as the effect size plot marginalised over the kernels. Additionally, we will report the posterior probability distribution over the kernels to show the applicability of each one.

5.1.2 Result

The kernels were not all able to properly fit the seasonal trend by themselves, so a Gamma prior was placed over the lengthscale with *loc* = 1.5 and *scale* = 2. This prior is still very general, but disallows high lengthscales.

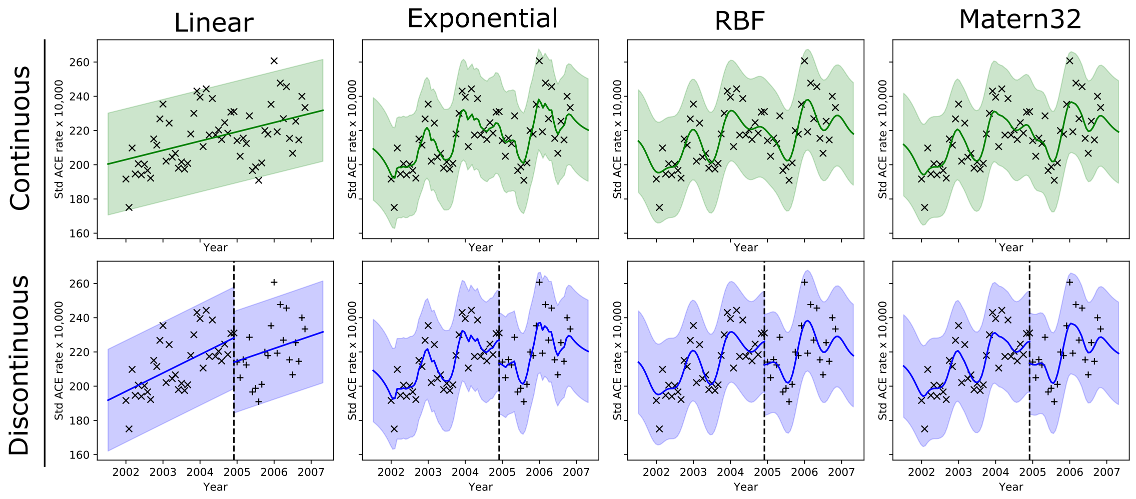

Figure 10: Regressions over the Sicilian smoking ban data, for different kernels.

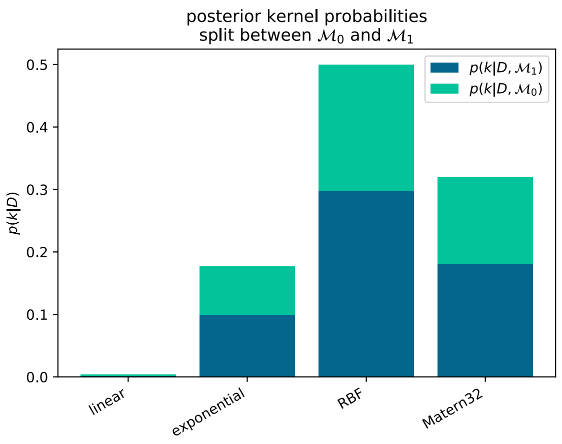

Figure 11: Posterior probabilities of the kernels given the Sicilian smoking ban data. Each bar is split into two parts, indicating the contribution of $\mathcal{M}_0$ and $\mathcal{M}_1$ to the kernel's probability.

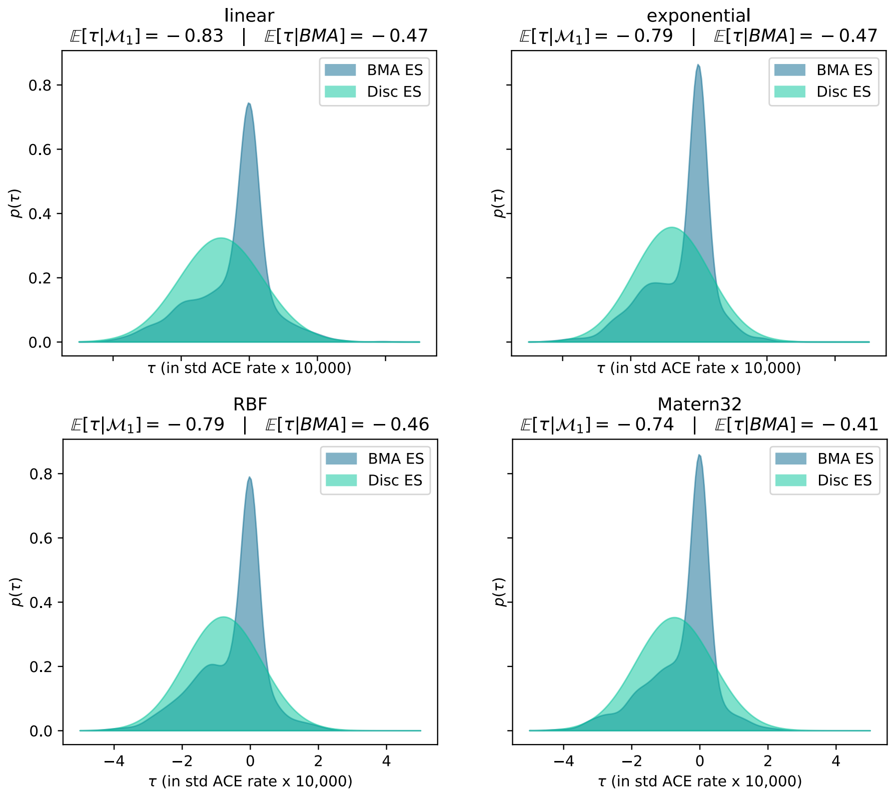

Figure 12: Effect size plots per kernel. For each kernel, the expected discontinuity under the discontinuous model as well as the BMA estimate are included in the title.

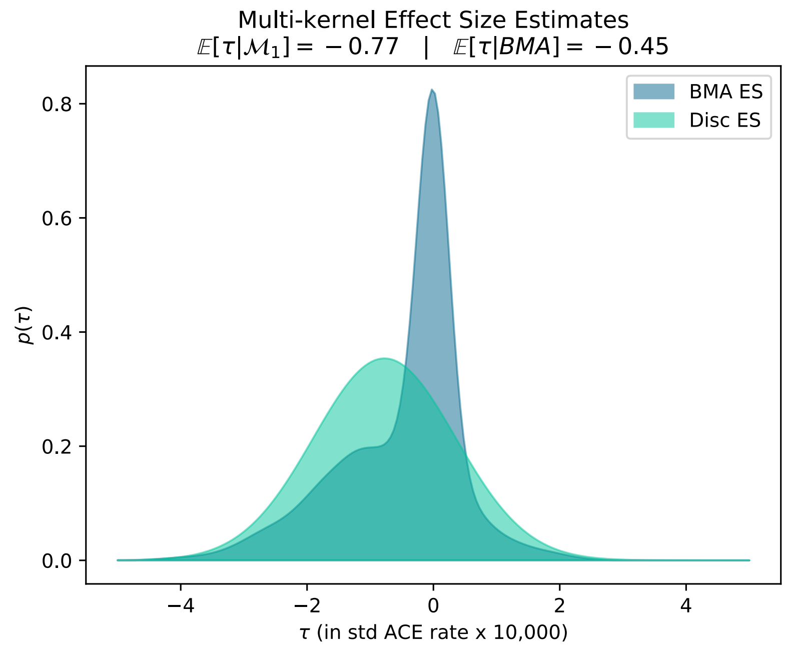

Figure 13: Effect size plots according to the `MultiKernelAnalysis`. Obtained by weighing each plot by the respective posterior probability of their kernels.

Kernel	$\log \mathrm{BF}_{10}$	$\mathbb{E}[\tau\|\mathcal{M}_1]$	$\mathbb{E}[\tau\|BMA]$	Kernel prob.	$\mathcal{M}_0$ prob.	$\mathcal{M}_1$ prob.
Linear	0.29	-0.83	-0.47	0.0040	0.43	0.57
Exponential	0.26	-0.79	-0.47	0.18	0.44	0.56
RBF	0.39	-0.79	-0.46	0.50	0.40	0.60
Matern32	0.27	-0.74	-0.41	0.32	0.43	0.57
Multi-kernel	-0.0042	-0.77	-0.45	-	0.50	0.50

As can be seen in the table above, we get quite different results in this experiment than Hinne et al. (2019) showed in their paper. All log Bayes factors are significantly higher than in the 2019 paper, with the multi-kernel log Bayes factor showing no favour towards either model. When inspecting the effect size plots of all kernels (figure 12) and those of the multi-kernel approach (figure 13), we can see that the BMA effect size estimates are significantly more spread out than in the 2019 paper, where they were essentially spike densities at $\tau = 0$. The higher overall Bayes factors make this difference unsurprising. When consulting the total log Bayes factor, it is likely to assume that there simply is not enough information in the data to conclude either a presence or absence of a discontinuity. This difference in log Bayes factors with the 2019 paper is likely due to one (or some combination) of three reasons: 1.) The optimisation step resulted in such different hyper-parameters that the results are affected. 2.) The novel approach to sharing hyper-parameters results in a better fit of the discontinuous model, and subsequently a higher Bayes factor. 3.) The addition of two kernels that are more capable of fitting a GP on the data direct the multi-kernel approach towards the discontinuous model (compared to the 2019 results).

Perhaps these results are more representative of the true effect of the smoking ban, since there exist such widely varying reported effects.

6. Discussion

Although the current implementation provides more functionality than the previous one, there remain a number of unimplemented features that could drastically improve the effectiveness of this approach, as well as some features that do not perform as well as desired:

1. BNQDflow does not offer effect size measures for different orders of discontinuities. Not does it offer effect size measures that quantify a difference latent variables as the effect size (e.g. frequency of periodic data).

2. The interface between the `EffectSizeMeasure` and `Analysis` objects is currently rather ambiguous. Instead of a dictionary, there should be a dedicated interface between the two classes so that their interaction is transparent.

3. There is no way to force both sub-regressions in $\mathcal{M}_1$ to have the same slope when using a linear kernel. This is something that is generally done in linear RDD to force any differences between the control and intervention group to be at the intervention point.

4. There also does not exist an option to force the 0th order discontinuity to be 0, and subsequently forcing the discontinuity into, for instance, the difference in slope.

5. A possible improvement that would be in line with the assumptions of RDD is to apply sparse GPs with a higher density of inducing variables around the intervention point, since we are particularly interested in the data close to the intervention point. Currently, it simply picks points at random. Additionally, the inducing variables are set to non-trainable, as per the relevant notebook by GPflow[5]. However, being able to train these variables such that they summarise the ”important” parts of the data may be desirable. Whether or not this is possible with the current training methods is unclear.

6. Currently, the ELBO score is calculated with the entire data set. And although it is faster than applying a regular GP, it can be made more computationally efficient by taking the mean ELBO score over multiple mini-batches.

7. It is common practice to standardise the data before fitting a GP to them. In future iterations, BNQDflow could standardise the data for the training procedure, but save the parameters used for the standardisation process, in order to reconstruct the original data if the user asks for it.

5. `https://gpflow.readthedocs.io/en/master/notebooks/advanced/gps_for_big_data.html#Likelihood-computation:-batch-vs.-minibatch`

8. Currently it is unclear why the sampling of the posterior distribution of the hyper-parameters does not work anymore. However, since this provides so much more information than a simple point estimate, this should really be fixed.

7. Conclusion

In this paper, a re-implementation of the theory of BNQD has been presented, and its performance investigated. Its performance on linear (4.1) and non-linear (4.2) data compared to the 2019 implementation is at worst similar, and at best more able to detect a discontinuity. The overall increase of the log Bayes factor, compared to the 2019 experiments, is assumed to be due to the different approach to optimising $\mathcal{M}_1$, resulting in a better fit on the data.

The posterior hyper-parameter sampling experiment (4.3) generally shows results that are congruent with the theory, although the flaws with its implementation might impugn the validity of the results.

The hyper-parameter prior experiment (4.4) shows that priors can indeed be used to constrain the possible regressions fit by BNQDflow. And if chosen properly, a prior can serve as an alternative to a larger data set.

The sparse GP model experiment (4.5) shows that the run time of the optimisation step likely increases cubically with the number of inducing points. Additionally, the means of the log Bayes factors remain consistent across all conditions. The estimated discontinuities also approach τ_{true} across all conditions, though the variance over the estimate only approaches that of a non-sparse GP starting from $n_{ind} = 316$. From this, we can conclude that we can comfortably remove some data, without negatively effecting our results.

The application of BNQDflow to real data shows that it can come to a conclusion that is congruent with previously found results.

In conclusion, this re-implementation shows an increase in performance in regards to the replicated experiments. Secondly, the new features that were previously unavailable build well on this Bayesian framework, though the HMC sampling functionality requires fine-tuning and more rigorous testing before it can be reliably applied. In its current state, BNQDflow is likely not a contender for contemporary QED libraries. However, in certain niche instances it may be used to give more detailed insight on the effects of certain interventions, and will hopefully be a factor in steering the scientific community towards a Bayesian framework for quasi-experimental designs.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Francesco Barone-Adesi, Antonio Gasparrini, Loredana Vizzini, Franco Merletti, and Lorenzo Richiardi. Effects of italian smoking regulation on rates of hospital admission for acute coronary events: a country-wide study. *PloS one*, 6(3):e17419, 2011.

Alessio Benavoli and Francesca Mangili. Gaussian processes for bayesian hypothesis tests on regression functions. In *Artificial intelligence and statistics*, pages 74–82, 2015.

Howard S Bloom. Modern regression discontinuity analysis. *Journal of Research on Educational Effectiveness*, 5(1):43–82, 2012.

Donald T Campbell and Julian C Stanley. *Experimental and quasi-experimental designs for research*. Ravenio Books, 2015.

Yanshuai Cao, Marcus A Brubaker, David J Fleet, and Aaron Hertzmann. Efficient optimization for sparse gaussian process regression. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2013.

Sai Hung Cheung and James L Beck. Calculation of posterior probabilities for bayesian model class assessment and averaging from posterior samples based on dynamic system data. *Computer-Aided Civil and Infrastructure Engineering*, 25(5):304–321, 2010.

David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

Nial Friel and Jason Wyse. Estimating the evidence – a review. *Statistica Neerlandica*, 66(3): 288–308, 2012. doi: 10.1111/j.1467-9574.2011.00515.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9574.2011.00515.x`.

Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

Sara Geneletti, Aidan G O'Keeffe, Linda D Sharples, Sylvia Richardson, and Gianluca Baio. Bayesian regression discontinuity designs: Incorporating clinical knowledge in the causal analysis of primary care data. *Statistics in medicine*, 34(15):2334–2352, 2015.

Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

GPy. Gpy: A gaussian process framework in python, since 2012. `https://sheffieldml.github.io/GPy/`.

Jinyong Hahn, Petra Todd, and Wilbert Van der Klaauw. Identification and estimation of treatment effects with a regression-discontinuity design. *Econometrica*, 69(1):201–209, 2001.

Markus Heinonen, Henrik Mannerström, Juho Rousu, Samuel Kaski, and Harri Lähdesmäki. Non-stationary gaussian process regression with hamiltonian monte carlo. In *Artificial Intelligence and Statistics*, pages 732–740, 2016.

Max Hinne, Marcel AJ van Gerven, and Luca Ambrogioni. Causal inference using bayesian non-parametric quasi-experimental design. *arXiv preprint arXiv:1911.06722*, 2019. `https://arxiv.org/abs/1911.06722`.

Jennifer A Hoeting, David Madigan, Adrian E Raftery, and Chris T Volinsky. Bayesian model averaging: a tutorial. *Statistical science*, pages 382–401, 1999.

Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, page 2, 2016.

Guido Imbens and Thomas Lemieux. The regression discontinuity design—theory and applications. *Journal of Econometrics*, 2008a.

Guido W Imbens and Thomas Lemieux. Regression discontinuity designs: A guide to practice. *Journal of econometrics*, 142(2):615–635, 2008b.

Evangelos Kontopantelis, Tim Doran, David A Springate, Iain Buchan, and David Reeves. Regression based quasi-experimental approach when randomisation is not an option: interrupted time series analysis. *BMJ*, 350, 2015. doi: 10.1136/bmj.h2750. URL `https://www.bmj.com/content/350/bmj.h2750`.

David S Lee and Thomas Lemieux. Regression discontinuity designs in economics. *Journal of economic literature*, 48(2):281–355, 2010.

David JC MacKay. Comparison of approximate methods for handling hyperparameters. *Neural computation*, 11(5):1035–1068, 1999.

Ioana E Marinescu, Patrick N Lawlor, and Konrad P Kording. Quasi-experimental causality in neuroscience and behavioural research. *Nature human behaviour*, 2(12):891–898, 2018.

Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18 (40):1–6, apr 2017. URL `http://jmlr.org/papers/v18/16-537.html`. `https://www.gpflow.org/`.

Ellen Moscoe, Jacob Bor, and Till Bärnighausen. Regression discontinuity designs are underutilized in medicine, epidemiology, and public health: a review of current and best practice. *Journal of clinical epidemiology*, 68(2):132–143, 2015.

Kun Il Park and Park. *Fundamentals of Probability and Stochastic Processes with Applications to Communications*. Springer, 2018.

Adrian E Raftery. Bayesian model selection in social research. *Sociological methodology*, pages 111–163, 1995.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

Lorenzo Richiardi, Loredana Vizzini, Franco Merletti, and Francesco Barone-Adesi. Cardiovascular benefits of smoking regulations: The effect of decreased exposure to passive smoking. *Preventive medicine*, 48(2):167–172, 2009.

Christian P Robert and Darren Wraith. Computational methods for bayesian model choice. In *Aip conference proceedings*, volume 1193, pages 251–262. American Institute of Physics, 2009.

Christopher Skovron and Rocıo Titiunik. A practical guide to regression discontinuity designs in political science. *American Journal of Political Science*, 2015:1–36, 2015.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.

Donald L Thistlethwaite and Donald T Campbell. Regression-discontinuity analysis: An alternative to the ex post facto experiment. *Journal of Educational psychology*, 51(6): 309, 1960.

Nikki van Leeuwen, Hester F Lingsma, Anton JM de Craen, Daan Nieboer, Simon P Mooijaart, Edo Richard, and Ewout W Steyerberg. Regression discontinuity design. *Epidemiology*, 27(4):503–511, 2016.

Eric-Jan Wagenmakers. A practical solution to the pervasive problems ofp values. *Psychonomic bulletin & review*, 14(5):779–804, 2007.

A. K. Wagner, S. B. Soumerai, F. Zhang, and D. Ross-Degnan. Segmented regression analysis of interrupted time series studies in medication use research. *Journal of Clinical Pharmacy and Therapeutics*, 27(4):299–309, 2002. doi: 10.1046/j.1365-2710.2002.00430.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1365-2710.2002.00430.x`.

Appendices

A. Calculation of the Posterior GP

$$
\begin{aligned}
f(x)|\mathcal{D} \sim \mathcal{GP}\left(\mu_{\mathcal{D}}(x), k_{\mathcal{D}}(x,x')\right), \\
m_{\mathcal{D}}(x) &= m(x) + \vec{k}(x,\vec{x}_{\mathcal{D}})\hat{K}_{\mathcal{D}}^{-1}(\vec{y}_{\mathcal{D}} - \vec{m}(\vec{x}_{\mathcal{D}})) \\
k_{\mathcal{D}}(x,x') &= k(x,x') - \vec{k}(x,\vec{x}_{\mathcal{D}})\hat{K}_{\mathcal{D}}^{-1}\vec{k}(\vec{x}_{\mathcal{D}},x') \\
\hat{K}_{\mathcal{D}} &= \hat{K}(\vec{x}_{\mathcal{D}},\vec{x}_{\mathcal{D}}) + \sigma^2\hat{I}
\end{aligned}
$$

Where $\vec{x}_{\mathcal{D}}$ and $\vec{y}_{\mathcal{D}}$ are the x and y values of the data respectively, σ^2 is the variance over the data points, $\hat{K}(\vec{x}_{\mathcal{D}}, \vec{x}_{\mathcal{D}})$ is the covariance matrix of the data points, and $\vec{k}(x, \vec{x}_{\mathcal{D}})$ is the vector with covariances between the data points and x.