Kunnskap for en bedre verden

DEPARTMENT OF MATHEMATICAL SCIENCES

TMA4500 - INDUSTRIAL MATHEMATICS,
SPECIALIZATION PROJECT

Adaptive Gaussian Markov random fields for data with known shocks

Author:
Halvard Emil Sand-Larsen

31.01.2025

Table of Contents

Abstract

Gaussian Markov random fields are commonly used for spatial and temporal smoothing. They allow for lending strength from data points that are close either in time or space. When analysing data, they can be used in the latent layers of Bayesian hierarchical models. For data with shocks, a drawback of the GMRFs is that they tend to oversmooth the shocked data points. This can be remedied by introducing adaptive GMRFs. This article explores the theory behind adaptive GMRFs as well as their implementation with the INLA package in R. The simulation study shows that adaptive GMRFs perform better for some data with known shocks, and equally well for other data, compared to the standard GMRFs. This is reinforced by the analysis of Norwegian death rates, including the Spanish flu and World War 2 as shocked time points.

1 Introduction

Gaussian Markov random fields (GMRF) is a much used tool in data analysis. They facilitate smoothing, both in temporal and spatial contexts, and allows the models to lend strength from data points nearby, either in time or space. However, they have a tendency to oversmooth shocked time points, like large spikes in the data. A remedy for this is the adaptive GMRFs, which allow for extra hyperparameters that make the models more flexible. The idea is to have different hyperparameters for data with known shocks than for the rest of the data. This article only concerns data where the shock is known, and thus will not focus on methods which identifies shocked data points as a part of the model.

Section 2 gives a brief introduction to GMRFs while Section 3 shows the framework called Bayesian hierarchical models, which often use GMRFs. Adaptive GMRFs are introduced in Section 4 before being used and compared to standard GMRFs in a simulation study in Section 5. Section 6 uses both adaptive GMRFs and standard GMRFs to analyse Norwegian death rates for periods with known shocks, namely the Spanish flu and World War 2. All of the code referenced in the article, and the code used to generate all figures, can be found at https://github.com/Halvardgithub/Master-project.

2 Gaussian Markov random fields (GMRF)

As the goal of this paper is to compare models with adaptive GMRFs to models with standard GMRF, we need a fundamental understanding of GMRFs. This section aims to give a quick introduction to GMRFs and the theory in this section is based on chapter 2 and 3 from the book Gaussian Markov Random Fields: Theory and Applications (Rue and Held, 2005).

2.1 Multivariate normal distribution

A core concept of a GMRF, is the multivariate normal distribution. If the vector $\mathbf{x}$ (1 x N) follows a multivariate normal distribution, we can write that $\mathbf{x} \sim N(\boldsymbol{\mu}, \Sigma)$. Here, $\boldsymbol{\mu}$ is a (1 x N) mean-vector and Σ is a (N x N) symmetric positive definite covariance matrix. Some well known properties of $\mathbf{x}$ is that $E[\mathbf{x}] = \boldsymbol{\mu}$, $E[x_i] = \mu_i$ and that $\text{Cov}(x_i, x_j) = \Sigma_{ij}$. Furthermore, $\mathbf{x}$ has the following density

$$\pi(\mathbf{x}) = (2\pi)^{-N/2}|\Sigma|^{-1/2}\text{exp}\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

A useful property of a random vector following a multivariate normal distribution, is that we can split the random vector in two, for instance $\mathbf{x} = \binom{\mathbf{x}_A}{\mathbf{x}_B}$, where $\mathbf{x}_A$ and $\mathbf{x}_B$ are two disjoint subsets of $\mathbf{x}$ and $\mathbf{x}_A \cup \mathbf{x}_B = \mathbf{x}$. Similarly, $\boldsymbol{\mu} = \binom{\boldsymbol{\mu}_A}{\boldsymbol{\mu}_B}$ and $\Sigma = \begin{pmatrix}\Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB}\end{pmatrix}$. We note that $\Sigma_{AB} = 0$ iff. $\mathbf{x}_A$ and $\mathbf{x}_B$ are independent. This means that distributions with assumptions of independence have fewer non-zero values in Σ. If we are interested in $\mathbf{x}_A$ conditioned on $\mathbf{x}_B$ we get that $\mathbf{x}_A|\mathbf{x}_B \sim N(\boldsymbol{\mu}_{A|B}, \Sigma_{A|B})$ where $\boldsymbol{\mu}_{A|B} = \boldsymbol{\mu}_A + \Sigma_{AB}\Sigma_{BB}^{-1}(\mathbf{x}_B - \boldsymbol{\mu}_B)$ and $\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$. Lastly, we can also describe a multivariate normal by a precision matrix Q instead of the standard covariance matrix Σ. Q is defined as the inverse of Σ, so $Q = \Sigma^{-1}$. A key property of the precision matrix is that x_i is conditionally independent of x_j given the remaining $\mathbf{x}$-values iff $Q_{ij} = 0$. Note that the requirement of conditional independence is weaker than the independence requirement for Σ. Thus, the precision matrix Q always have fewer or equal non-zero elements compared to the covariance matrix Σ.

2.2 Definition of a GMRF

Consider the multivariate normal vector $\mathbf{x} \sim N(\boldsymbol{\mu}, \Sigma)$ of size N. If we define an associated undirected graph G with a set of nodes $\mathcal{V} = \{1, ..., N\}$ and a set of edges $\mathcal{E}$ where $\{i, j\} \in \mathcal{E}$ iff. $\Sigma_{ij} \neq 0$. Furthermore, we call G a labelled graph as the nodes are numbered from 1 and up to the number of nodes N. If the above are satisfied, $\mathbf{x}$ is a GMRF wrt. G. The following formal definition is taken from Rue and Held (2005).

Definition: A random vector $\mathbf{x} \in \mathbb{R}^N$ is called a GMRF with respect to a labelled graph $G = (\mathcal{V}, \mathcal{E})$ with mean $\boldsymbol{\mu}$ and symmetric positive definite precision matrix Q iff. its density can be written as

$$\pi(\mathbf{x}) = (2\pi)^{-N/2}|Q|^{1/2}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T Q(\mathbf{x}-\boldsymbol{\mu})\right)$$

and $Q_{ij} \neq 0 \iff \{i, j\} \in \mathcal{E} \quad \forall \quad i \neq j$.

So, as a GMRF is simply a multivariate normal with some added formal definitions and the results for a multivariate normal are clearly also applicable to a GMRF. We define $ne(i)$ as the set off all nodes directly connected to i in the labelled graph $\mathcal{G}$, and n_i as the size of $ne(i)$, the number of neighbouring nodes. The structure matrix R can be defined as follows.

$$R_{ij} = \begin{cases} n_i, & \text{if } i = j \\ -1, & \text{if } \{i, j\} \in \mathcal{E} \\ 0, & \text{else:} \end{cases}$$

We then define $Q = \tau R$. The precision matrix Q is often sparse, which will be beneficial for the computation time of matrix multiplications, inverses and so on. This will greatly reduce the time spent on simulating or fitting models with GMRFs, and Q will also need less storage space when stored as a sparse matrix.

A key part of a GMRF is the Markov property. This is closely related to the precision matrix, and its mentioned connection to conditional independence. Formally, the following statements from Rue and Held (2005) are equivalent for a GMRF $\mathbf{x}$:

- The pairwise Markov property: $x_i \perp x_j | \mathbf{x}_{-ij}$ if $\{i, j\} \notin \mathcal{E}$ and $i \neq j$

- The local Markov property: $x_i \perp \mathbf{x}_{-\{i,ne(i)\}} | \mathbf{x}_{ne(i)}$ for every $i \in \mathcal{V}$
- The global Markov property: $\mathbf{x}_A \perp \mathbf{x}_B | \mathbf{x}_C$ for all disjoint sets A, B and C where C separates A and B and they are all non-empty.

Above, a negative subscript indicates that the mentioned elements have been omitted from the vector. That C separates A and B means that any path from $a \in A$ to $b \in B$ will contain at least a point $c \in C$.

We can also specify the conditional distribution of a part of the GMRF given the rest of it. If we split the $\boldsymbol{\mu}$ and Q as we did for the multivariate normal earlier, we get that $\mathbf{x}_A | \mathbf{x}_B$ is a GMRF with mean $\boldsymbol{\mu}_{A|B} = \boldsymbol{\mu}_A + Q_{AA}^{-1} Q_{AB}(\mathbf{x}_B - \boldsymbol{\mu}_B)$ and precision matrix $Q_{A|B} = Q_{AA}$ with respect to the subgraph G^A, which is all the nodes in A and the edges between them from the graph for the first GMRF.

So far we have been operating with a parameterization of mean and precision matrix. However, this is not the canonical parameterization. This is instead defined by a vector $\mathbf{b}$ and a precision matrix Q and denoted as $\mathbf{x} \sim N_C(\mathbf{b}, Q)$. Note that the mean $\boldsymbol{\mu} = Q^{-1}\mathbf{b}$, so $N(\boldsymbol{\mu}, Q^{-1}) = N_C(\mathbf{b}, Q)$.

2.3 Simulation of a GMRF

The base case for simulating samples from a GMRF is to simulate a sample from a multivariate normal, $\mathbf{x} \sim N(\boldsymbol{\mu}, \Sigma)$. This is done by computing the Cholesky decomposition of Σ so that $\Sigma = LL^T$. Given a sample $\mathbf{z} \sim N(0, I)$, a sample from $\mathbf{x}$ is equal to $\boldsymbol{\mu} + L\mathbf{z}$. If we instead want to sample from $\mathbf{x} \sim N(\boldsymbol{\mu}, Q^{-1})$ we exchange $L\mathbf{z}$ with $\mathbf{v}$ where $\mathbf{v}$ is the solution to $L^T\mathbf{v} = \mathbf{z}$ which is generally solved by back substitution. These methods can be altered to fit a GMRF with a canonical parameterization as follows, from Rue and Held (2005).

1. Compute L, the Cholesky factorization of Q
2. Solve $L\mathbf{w} = \mathbf{b}$
3. Solve $L^T\boldsymbol{\mu} = \mathbf{w}$
4. Sample $\mathbf{z} \sim N(0, I)$
5. Solve $L^T\mathbf{v} = \mathbf{z}$
6. Compute $\mathbf{x} = \boldsymbol{\mu} + \mathbf{v}$

This framework is also applicable to conditional simulations for a GMRF. If $\mathbf{x} \sim N(\boldsymbol{\mu}, Q^{-1})$ and $\mathbf{x}_B = \mathbf{x}_{-A}$ we know that $\mathbf{x}_A | \mathbf{x}_B \sim N(\boldsymbol{\mu}_A - Q_{AA}^{-1} Q_{AB}(\mathbf{x}_B - \boldsymbol{\mu}_B), Q_{AA}^{-1})$. If we subtract $\boldsymbol{\mu}_A$ and change to the canonical parameterization we get that $\mathbf{x}_A - \boldsymbol{\mu}_A | \mathbf{x}_B \sim N_C(-Q_{AB}(\mathbf{x}_B - \boldsymbol{\mu}_B), Q_{AA})$. Now we can simulate from the algorithm above and simply add $\boldsymbol{\mu}_A$ afterwards.

Another useful case is when we add a linear constraint, so $\mathbf{x} \sim N(\boldsymbol{\mu}, Q^{-1})$ and we want to sample from $\pi(\mathbf{x}|A\mathbf{x} = \mathbf{e})$. One option here is to condition by Kriging, a procedure from geostatistics, which allows us to sample from $\mathbf{x} \sim N(\boldsymbol{\mu}, Q^{-1})$ before computing $\mathbf{x}^* = \mathbf{x} - Q^{-1}A^T(AQ^{-1}A^T)^{-1}(A\mathbf{x} - \mathbf{e})$, which has the desired distribution.

2.4 Intrinsic GMRF

Many smoothing models, for instance the ones introduced in Sections 3.1 and 3.2, use a precision matrix without full rank, and hence it will not be invertible. This complicates the previous algorithms for simulation as we do not have a Cholesky factorization. To handle this problem we introduce improper and intrinsic GMRF, which both allow for precision matrices without full rank.

Definition: (Improper GMRF) Q is an (N x N) SPSD matrix with rank $N - k > 0$ and $\mathbf{x}$ is an Improper GMRF wrt. the labelled graph $G = (\mathcal{V}, \mathcal{E})$ with rank $N - k$ if its density can be written as

$$\pi(\mathbf{x}) = (2\pi)^{-(N-k)/2}(|Q|^*)^{1/2}\text{exp}\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T Q(\mathbf{x} - \boldsymbol{\mu})\right)$$

$|\cdot|^*$ is the product of the non-zero eigenvalues of the matrix. We will still refer to $\boldsymbol{\mu}$ and Q as the mean and precision, even though neither of them exists in this case. An *Intrinsic GMRF* of the first order only adds one further constraint to a rank $N - 1$ improper GMRF, namely that $Q\mathbb{1} = 0$. This is both easy to construct and to verify.

3 Bayesian Hierarchical models

A Bayesian Hierarchical model commonly has three levels (Rue and Held, 2005, chapter 4). These models generally have a likelihood where some of the parameters again depend on other parameters. For instance, the likelihood for a data point i could be $Y_i|\lambda_i \sim Poisson(\lambda_i)$. The second layer defines the parameter from the likelihood through other parameters, often called a latent model, for instance $\text{log}\lambda_i = \mu + \alpha_i$ and assigns priors to the introduced parameters, namely μ and α_i. For example $\mu \sim N(10, 5)$ and $\alpha_i \overset{iid}{\sim} N(0, \sigma^2)$ respectively. The third layer assigns priors to the hyperparameters, in this case σ. An important sub-category is when the latent layer follows a Gaussian distribution, which we call a latent Gaussian model. To summarise, we can view the layers as

1. Define the likelihood $Y|x, \theta \sim \pi(Y|x, \theta)$
2. Define the latent model $x|\theta \sim \pi(x|\theta)$
3. Define the hyperpriors $\pi(\theta)$

An important type of Bayesian Hierarchical models are the general linear models (GLM). This setup expands on the standard linear model so it can handle more types of data, like binomial, Poisson and gamma (Fahrmeir *and others*, 2022, chapter 5). The key here is a transformation of the linear terms, $X\boldsymbol{\beta}$, through some link function $g(\mu)$ such that $\mu_i = g^{-1}(\eta_i) = g^{-1}(X_i\boldsymbol{\beta})$. There are many choices for g, but the logit is the most common. The GLM framework can further be expanded with random effects in the linear predictor term. These are generally some zero mean normally distributed variables which represent some difference from the overall mean for a given variable. An example could be a dataset containing data from different cities, where it is reasonable that these cities have slightly different properties, and to capture these differences we use random effects. This is very useful for grouped data, and we call this model a Generalised Linear Mixed Model (GLMM).

3.1 Temporal latent components

One basic model in time is the random walk of first order, commonly referred to as a RW1. This is a process where we assume that the next step only depends on the current value [chapter 3](Rue and Held, 2005). In addition there is some added noise in each step which generally follow some zero-centred Gaussian distribution. We also assume that these noise terms are independent and that we have equispaced time-points $\{1, ..., N\}$. For a fixed x_1 the RW1 is defined iteratively below:

$$x_{t+1} = x_t + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2) \quad t = 1, ..., N-1$$

This can also be described in terms of conditional distributions as follows.

$$x_{t+1}|x_t, \sigma \sim N(x_t, \sigma^2) \quad t = 1, ..., N-1$$

We can then define the joint density as

$$\begin{aligned}\pi(\mathbf{x}|\sigma) &= (2\pi\sigma^2)^{-(N-1)/2}(|R|^*)^{1/2}\text{exp}\left(-\frac{1}{2\sigma^2}\sum_{t=1}^{N-1}(x_{t+1}-x_t)^2\right) \\ &= (2\pi\sigma^2)^{-(N-1)/2}(|R|^*)^{1/2}\text{exp}\left(-\frac{1}{2}\mathbf{x}^T Q\mathbf{x}\right)\end{aligned}$$

where $Q = \frac{1}{\sigma^2}R = \tau R$ with precision parameter τ and structure matrix R defined below.

$$R_{ij} = \begin{cases} 1, & \text{if } i = j \in \{1, N\} \\ 2, & \text{if } i = j \notin \{1, N\} \\ -1, & \text{if } |i-j| = 1 \\ 0, & \text{else.} \end{cases}$$

The rows of Q all sum to 0 and the rank of Q is $N-1$, so this is an intrinsic GMRF. Therefore, the model is invariant to adding a constant, so when a random walk is included in a model with an intercept it is often required that $\sum_{t=1}^{n} x_t = 0$ to ensure the interpretability of the intercept.

For situations where we want a second order dependence in time, we can use the RW2. For known x_1 and x_2 it can be described by the conditional distributions below for $t = 2, ..., N-1$.

$$x_{t+1}|x_t, x_{t-1}, \sigma \sim N\left(2x_t - x_{t-1}, \sigma^2\right)$$

We can again find the joint density as for the RW1, and the only difference is the precision matrix $Q = \tau R$, with R defined as

$$R_{ij} = \begin{cases} 2, & \text{if } i = j \in \{1, N\} \\ 3, & \text{if } i = j \in \{2, N-1\} \\ 4, & \text{if } i = j \notin \{1, 2, N-1, N\} \\ -1, & \text{if } |i-j| \leq 2 \quad \text{and} \quad i \neq j \\ 0, & \text{else.} \end{cases}$$

Again $Q\mathbb{1} = \mathbf{0}$ and the rank of Q is $N-1$, thus the RW2 is an intrinsic GMRF. Simulations of RW1 and RW2 can be seen in Figure 1. Note that the RW2 is much smoother compared to the RW1, and that the RW2 reaches much larger values.

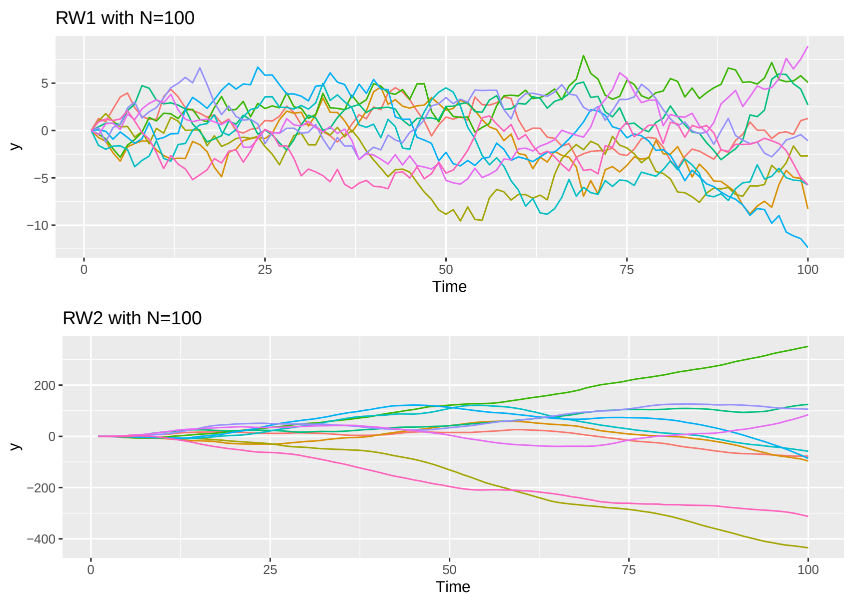

Figure 1: The top panel shows 10 simulations of a RW1 with $x_1 = 0$ and $\sigma = 1$. The bottom panel shows 10 simulations of a RW2 with $x_1 = x_2 = 0$ and $\sigma = 1$.

3.2 Spatial latent components

A common first order model in discrete space is the intrinsic conditional autoregressive model, or ICAR (Freni-Sterrantino *and others*, 2018). We typically have areal data, like the regions of a country, and we assume that neighbouring regions are spatially correlated. By neighbouring regions we mean regions that share a border with each other, and we write $i \sim j$ if region i and region j share a border. The model then assumes that each region is normal around the mean of its neighbours, and we are more confident the more neighbours a region has. We also assume that for all pairs of neighbours $\{i, j\}$ we can write $x_i - x_j \sim N(0, \tau^{-1})$ for a universal precision τ. In total, we get the following conditional distribution for an element of $\mathbf{x}$:

$$\pi(x_i|\mathbf{x}_{ne(i)}, \tau) = N\left(\frac{\sum_{j \in ne(i)} x_j}{n_i}, \frac{1}{n_i \tau}\right)$$

where $ne(i)$ is the set of the neighbours of region i, and we define n_i as the size of $ne(i)$. We can construct the associated precision matrix Q through the structure matrix R.

$$R_{ij} = \begin{cases} n_i, & \text{if } i = j \\ -1, & \text{if } i \sim j \\ 0, & \text{else.} \end{cases}$$

Where $i \sim j$ is true if they share a border and false otherwise.Then $Q = \tau R$. As $Q\mathbb{1} = 0$ and the rank of Q is $N - 1$, the ICAR is an intrinsic GMRF.

If we combine a structured random effect, like the ICAR above, with an unstructured random effect term we get a BYM model (Besag *and others*, 1991). In the expression below, u_i is from an ICAR model with a precision τ_u and v_i is the unstructured random effect which typically follows a $N(0, \tau_v^{-1})$ for some precision τ_v.

$$\eta_i = u_i + v_i$$

The BYM model allows us to increase or decrease estimates compared to the prediction from the neighbouring regions. This is important because it accounts for spatial heterogeneity. The BYM model can be used as a smoothing model on its own, or in the second layer of a Bayesian hierarchical model for disease mapping.

4 Adaptive GMRFs

Standard GMRFs work well for regular problems, but this is not always the case. In a temporal setting there could be spikes at certain time points, for instance due to war or financial crisis, depending on the outcome of interest. In a spatial setting these spikes can occur at country borders or some geographical divider, say a mountain range or a river. When this is the case, the assumption that all differences between neighbouring points follow the same normal distribution, which is assumed in both ICAR and RW1, no longer holds. In this chapter I will follow the ideas introduced in Aleshin-Guendel and Wakefield (2024). They propose to instead use adaptive GMRFs, which allows for multiple parameters to model the distributions between neighbours. The idea is to have different parameters for transitions which are considered shocked versus standard transitions. This will hopefully avoid over-smoothing of large transitions and also reduce the estimated variance for the standard transitions. Additionally, we assume that we know which data points are considered shocked.

4.1 General model to allow for different smoothing

We start with a general framework for an adaptive RW1, where we no longer assume that all transitions have the same precision. Instead, we assign each transition its own precision τ_t for $t = 1, ..., N-1$, for N time points. For some fixed x_1, the conditional distribution becomes

$$x_{t+1}|x_t, \tau_t \sim N(x_t, \tau_t^{-1}) \quad t = 1, ..., N-1.$$

And the joint density can be written as

$$\pi(\mathbf{x}|\tau_1, ..., \tau_{N-1}) \propto \left(\prod_{t=1}^{N-1} \tau_t^{1/2}\right) \exp\left(-\frac{1}{2}\mathbf{x}^T Q \mathbf{x}\right)$$

with the precision matrix Q defined as

$$Q_{ij} = \begin{cases} \tau_1, & \text{if } i = j = 1 \\ \tau_{N-1}, & \text{if } i = j = N \\ \tau_{i-1} + \tau_i, & \text{if } i = j \notin \{1, N\} \\ -\tau_i, & \text{if } i = j - 1 \\ -\tau_{i-1}, & \text{if } i = j + 1 \\ 0, & \text{else} \end{cases}$$

As for the standard RW1, $Q\mathbb{1} = \mathbf{0}$ and the rank of Q is $N-1$. Thus, the adaptive random walk is intrinsic, and we add the sum-to-zero constraint if it is used with an intercept.

We can also make the spatial case adaptive by modifying the ICAR, see Section 3.2. We again discard the notion that all transitions have the same precision and instead introduce a precision τ_{ij} for each pair of neighbours. We can define the distributions for the transitions as

$$x_i - x_j|\tau_{ij} \sim N(0, \tau_{ij}^{-1}).$$

The joint density of $\mathbf{x}$ is then

$$\pi(\mathbf{x}|\{\tau_{ij}\}_{i\sim j}) \propto (|Q|^*)^{1/2} \exp\left(-\frac{1}{2}\mathbf{x}^T Q \mathbf{x}\right).$$

$|Q|^*$ is the product of the non-zero eigenvalues of the matrix Q, which has the following elements.

$$Q_{ij} = \begin{cases} \sum_{k|i\sim k} \tau_{ik}, & \text{if } i = j \\ -\tau_{ij}, & \text{if } i \sim j \\ 0, & \text{else} \end{cases}$$

Again, $Q\mathbb{1} = \mathbf{0}$, the rank of Q is $N-1$ and we have an intrinsic GMRF which should have a sum-to-zero constraint if used with an intercept.

4.2 Model for data with known shocks

Both of the adaptive models above are very flexible. However, they introduce many parameters which need priors and increases the computational complexity when fitting the models. If we assume that our temporal data has known shocks, for instance war or drought during a subset of the time points, we can split the data into shocked and non-shocked time points. We then assign a precision τ_1 for transitions between non-shocked points and a precision τ_2 for transitions involving at least one shocked time point. We make a set containing the shocked time points and call it S. The transitions between neighbours then follow these distributions:

$$\begin{aligned} x_{t+1}|x_t, \tau_1, \tau_2 &\sim N(x_t, \tau_1^{-1}), \quad \text{if } \{t, t+1\} \notin S, \\ x_{t+1}|x_t, \tau_1, \tau_2 &\sim N(x_t, \tau_2^{-1}), \quad \text{if } t \in S \text{ or } t+1 \in S. \end{aligned}$$

We expect $\tau_1 > \tau_2 \rightarrow \sigma_1^2 < \sigma_2^2$, as there should be higher variance for transitions involving shocked data. Now we need the precision matrix Q. First, for $l \in \{1, 2\}$, we define $R_l = D_l - W_l$ with

$$W_{l,ij} = \begin{cases} I(|i-j| = 1 \text{ and } \{i, j\} \notin S), & \text{if } l = 1, \\ I(|i-j| = 1 \text{ and } i \in S \text{ or } j \in S), & \text{if } l = 2, \end{cases}$$

and $D_l = \text{diag}\left(\sum_{j=1}^{N} W_{l,1j}, ..., \sum_{j=1}^{N} W_{l,Nj}\right)$. Then, $Q = \tau_1 R_1 + \tau_2 R_2$ which is again intrinsic as $Q\mathbb{1} = \mathbf{0}$ and the rank of Q is $N - 1$.

The next step is to scale the model appropriately, and we will use the scaling method proposed in Sørbye and Rue (2014). The goal is to make interpretation and priors interchangeable between different applications and graph structures. For a structured random effect $\mathbf{x}$ with a structure matrix R we scale it by the geometric mean of the marginal variances computed by

$$\sigma_{GV}^2(\mathbf{x}) = \exp\left(\frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{1}{\tau_x}[R^-]_{ii}\right)\right)$$

where R^- is the generalised inverse of the structure matrix R. τ_x in the formula is the constant precision for all transitions. Ideally, $\sigma_{GV}^2(\mathbf{x}) = \frac{1}{\tau_x}$, and we scale R to achieve it. Let $\sigma^2(\mathbf{x})$ be the geometric mean when $\tau_x = 1$, then we denote the scaled structure matrix as $R_* = \sigma^2(\mathbf{x})R$. We multiply by the geometric mean as it increases the precision when $\sigma^2(\mathbf{x}) > 1$ and else lowers it so $\sigma^2(\mathbf{x}_*) = 1$ with $\mathbf{x}_* \sim N(0, R_*^-)$.

Since the adaptive model we want to scale has two parameters instead of one, we slightly alter the steps above. Now, let $\sigma^2(\mathbf{x})$ be the geometric mean variance when $\tau_1 = \tau_2 = 1$ for the adaptive GMRF $\mathbf{x}$. Then, the scaled precision matrix $Q^* = Q/\sigma^2(\mathbf{x}) = \tau_1 R_1^* + \tau_2 R_2^*$ where $R_l^* = R_l/\sigma^2(\mathbf{x})$.

4.3 Implementing the model with INLA

To implement the adaptive RW model we will use INLA in R. As the adaptive RW1 is not already defined in INLA it must be implemented from scratch. After defining the latent term with `inla.rgeneric.model()` we will use the function `inla()` to perform the model fitting. Sections 4.3.1 and 4.3.2 are based on chapter 11 in Gómez-Rubio (2020) and Rue (2021), as well as my own testing.

4.3.1 INLA's rgeneric

The function `inla.rgeneric.model()` is used to implement the adaptive model in INLA, which can be used as a latent model. The `model` in the function call is just the name of the model, say RW1 or AR1 for instance. This function needs to satisfy certain formalities to fit in with the INLA framework, and an overview of the function structure is shown in the code exempt below.

```r
inla.rgeneric.somemodel = function(
cmd = c("graph", "Q", "mu", "initial", "log.norm.const","log.prior",
"quit"), theta = NULL)
{
  interpret_theta <- function(){ <to be completed> }

  graph <- function(){ <to be completed> }
  Q <- function() { <to be completed> }
  mu <- function() { <to be completed> }
  log.norm.const <- function() { <to be completed> }
  log.prior <- function() { <to be completed> }
  initial <- function() { <to be completed> }
  quit <- function() { <to be completed> }

  if (!length(theta)) theta = initial()
  val = do.call(match.arg(cmd), args = list())
  return (val)
}
```

All the above functions need to be suitably defined, apart from `interpret_theta()`, although for some it is often enough to pass a default value. The check of the length of θ is not strictly necessary, but is a good practice to avoid runtime issues when theta is undefined. Now, let us take a look at each of the subfunctions.

`interpret_theta()` is a function that transforms the parameters of the model from INLA's internal scale to a desired scaling. This is optional, but often makes the code in the other functions more readable when used as a helper function. INLA's internal representation is a vector $\theta = [\theta_1, ..., \theta_p]$ for p parameters where $\theta_i \in \mathbb{R} \forall i$. The internal scaling is used since it allows for unconstrained optimization. The desired scaling will vary for different types of parameters, for instance a precision parameter $\tau = \exp(\theta) \in \mathbb{R}^+$. So, for a model with two precision parameters we could use the following function.

```r
interpret_theta <- function() {
    return(list(tau1 =exp(theta[1L]), tau2 = exp(theta[2L]))}
```

The function `graph()` should return the neighbourhood structure of the model. This is essentially just the structure matrix of the model where we have a connection if $Q_{ij} \neq 0$. Thus, we just return the precision matrix Q. For both `graph()` and `Q()` it is strictly speaking sufficient to only construct the upper triangular part, including the diagonal, as they are symmetric.

The function `Q()` needs to return the precision matrix of the model, preferably with a sparse representation. The most efficient way is to create the matrix as sparse, but it is often easier and more readable to define the full matrix and then convert it to a sparse representation with `inla.as.sparse()`. The downside is an increase in both the memory usage and the computation time.

`mu()` simply returns the mean of the model, and generally just returns `numeric(0)` if the mean is 0. Note that this is not a sum-to-zero constraint, which can be added later in the pipeline.

`log.norm.const()` needs to return the log of the normalizing constant for the prior distribution, not with the internal scale. The most straight forward approach is to return `numeric(0)` and let INLA compute it for you (Rue, 2021).

`log.prior()` defines the combined prior for all the parameters on the log-scale. Generally, we assign a prior on the desired scaling, so a prior for the precision τ, and then transform the prior to depend on the corresponding θ_i before taking the log of the prior. It then returns the log prior evaluated at the values stored in θ.

`initial()` returns a vector of the initial values for the parameters used in the model. These are the same parameters as in `interpret_theta()`, and they must be returned on the internal scale.

`quit()` is called when everything else is finished and can be used for clean up if necessary. For the uses in this article it is sufficient to use

```
    quit <- function() {return(invisible())}
```

The last three lines of the framework simply makes sure that θ is defined and that all the parts of the model is returned in the correct format.

As implementing this function for the adaptive RW1 is somewhat complex, I started with the simpler standard RW1 and the specific framework is shown below.

```
inla.rgeneric.RW1.model = function(
  cmd = c("graph", "Q", "mu", "initial",
   ↪  "log.norm.const","log.prior", "quit"),
  theta = NULL)
{
  #Input:
  #N is the number of timepoints

  interpret_theta <- function() {return(list(tau = exp(theta[1L])))}

```

```r
  graph <- function() {return(Q())}

  Q <- function() {
    R <- toeplitz(c(2, -1, rep(0, N - 2)))# 2 on diag and -1 on
    ↪  firstdiags
    R[1, 1] <- R[N, N] <- 1 # 1 for first and last diag element
    gv <- exp(1 / N * sum(log(diag(INLA:::inla.ginv(R))))) #the
    ↪  geometric variance
    R_star <- gv * R

    p <- interpret_theta()
    Q <- p$tau * R_star
    return(inla.as.sparse(Q))
  }

  mu <- function() {return(numeric(0))}

  initial <- function() {return(4)}#default for precisions: initial =
  ↪  4

  log.norm.const <- function() {return(numeric(0))} #Inla computes it

  log.prior <- function() {#default: shape = 1, rate = 0.00005 for
  ↪  tau
    p <- interpret_theta()
    prior <- dgamma(p$tau, shape = 1, rate = 0.00005, log = TRUE) +
    ↪  log(p$tau)
    return(prior)
  }

  quit <- function() {return(invisible())}

  #to ensure theta is defined
  if (!length(theta)) theta = initial()

  vals <- do.call(match.arg(cmd), args = list())
  return(vals)
}
```

This is a bare bones implementation of a latent RW1 model. `Q()` and `log.prior()` is the meat of the necessary code, while the other functions are rather straight forward. Note that we assume the input N is known inside the function. `Q()` first defines the structure matrix as defined in Section 3.1 before scaling it by the geometric variance and the precision parameter τ. For `log.prior()` we simply choose a gamma prior for τ and we use INLA's default values for shape and rate for a precision parameter. The initial value of 4 is also to match INLA's default choice. To ensure the model works as intended I compared it with INLA's own RW1 model. Figure 2 shows that the mean and standard deviation of the fitted values coincide. The estimated hyperparameters are also practically equal, although there are small differences. Worth noting that the user defined function has a much greater runtime with 7.15s versus 1.34s for the test example. For further details see Section A in the Appendix.

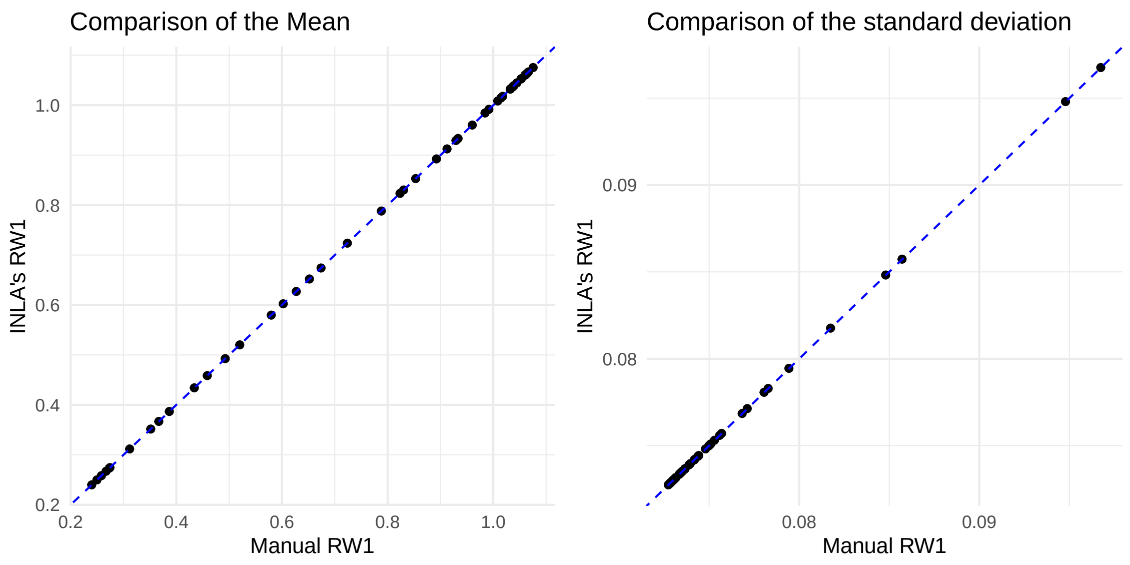

Figure 2: The mean and standard deviation for the fitted values from two similar INLA models. One uses INLA's RW1 while the other uses a `rgeneric` implementation of a RW1.

To expand this to the adaptive RW1 with 2 precision parameters, should be straight forward. For instance `interpret_theta()` and `initial()` simply returns two elements compared to the RW1. For `log.prior()` we simply add another prior of the same form for the added parameter as we assume that the two precision parameters are independent. In `Q()` we now require an additional input, namely the conflict_years, to construct the two matrices $R1$ and $R2$. Definition and scaling of the structure matrices are in Section 4.2. In total we get the following model.

```r
inla.rgeneric.AdaptiveRW1.model = function(
  cmd = c("graph", "Q", "mu", "initial",
  ↪  "log.norm.const","log.prior", "quit"),
  theta = NULL)
{
  #Input:
  #N is the number of timepoints
  #conflict_years

  interpret_theta <- function() { return(list(tau1 = exp(theta[1L]),
                                              tau2 = exp(theta[2L])))}

  graph <- function() {return(Q())}

  Q <- function() {
    R1 <- matrix(0, nrow = N, ncol = N) #non-conflict
    R2 <- matrix(0, nrow = N, ncol = N) #conflict
    for( i in 1:(N - 1)){
      if(i %in% conflict_years | (i + 1) %in% conflict_years) {
        R2[c(i, i+1), c(i, i+1)] <- R2[c(i, i+1), c(i, i+1)] + c(1, -1, -1, 1)
      }
      else {
        R1[c(i, i+1), c(i, i+1)] <- R1[c(i, i+1), c(i, i+1)] + c(1, -1, -1, 1)
      }
    }
    gv <- exp(1 / N * sum(log(diag(INLA:::inla.ginv(R1 + R2))))) #scaling constant
    R_star_list <- list(R1 = R1*gv, R2 = R2*gv)

    p <- interpret_theta()
    Q <- R_star_list$R1 * p$tau1 + R_star_list$R2 * p$tau2
    return(inla.as.sparse(Q)) #sparse representation
  }

  mu <- function() {return(numeric(0))}

  initial <- function() {return(c(4, 4))}#Default initial for precisions is 4

  log.norm.const <- function() {return(numeric(0))}

  log.prior <- function() {#default: shape = 1, rate = 0.00005
    p <- interpret_theta()
    prior <- dgamma(p$tau1, shape = 1, rate = 0.00005, log = TRUE) + log(p$tau1) +
    dgamma(p$tau2, shape = 1, rate = 0.00005, log = TRUE) + log(p$tau2)
    return(prior)
  }

  quit <- function() {return(invisible())}

  #to ensure theta is defined
  if (!length(theta)) theta = initial()

  vals <- do.call(match.arg(cmd), args = list())
  return(vals)
}
```

In the code on github the `rgeneric` definition uses an additional input `prior_str` to choose between the three different priors used, see Section 5.1.2.

When `inla.rgeneric.model()` is defined, the next step is to make the model with `inla.rgeneric.define()`. The necessary inputs are a rgeneric function followed by the model specific arguments, N for the RW1 and N and `conflict_years` for the adaptive RW1. The variable we assign the output from the define function to becomes the name of the model which can be used in the INLA formula like a standard latent model. For the adaptive RW1 we get:

```r
    ARW1_model <-
     ↪  inla.rgeneric.define(inla.rgeneric.AdaptiveRW1.model,
                          N = N, R_star_list = R_star_list_W)
    formula <- y ~ f(time, model = ARW1_model)
```

Both `y` and `time` corresponds to a column-name in the dataframe for training, more on this later.

4.3.2 The INLA formula and the INLA function

For all latent models, predefined or from rgeneric, we can add restrictions in the INLA formula. These are added inside the `f()` as the argument "extraconstr". For N data points the form of the restriction is $A\mathbf{x} = e$, where A is a (i X N) matrix and e a (i X 1) vector, where i is the number of restrictions you want to enforce. A and e are defined by the user and $\mathbf{x}$ is the latent effects. A sum-to-zero constraint for the ARW1_model above can be enforced as follows, where N is the number of data points.

```r
formula_ARW1 <- y ~ f(time, model = ARW1_model,
        extraconstr = list(A = matrix(1, nrow = 1, ncol = N), e = 0))
```

When using `f()` with a predefined INLA model it is possible to define the prior and initial values for the parameters of the model. The example below uses INLA's RW1 with a loggamma prior for θ, `param` assigns the parameters of the distribution and `initial` sets the initial value for θ, so on the internal scale.

```r
f(time, model = "rw1", hyper = list(prec = list(prior = "loggamma",
        param = c(1, 0.00005), initial = 4)))
```

The expression above uses the default values for the RW1 from the INLA documentation and is equal to using `f(time, model = "rw1")`. To combine multiple latent models, we add functions `f()` on the right side of `~`.

The formula is then passed to the function `inla(formula, family, data, ...)`. The variable-names used in the formula must match the columns in the data. The family specifies the observational layer in the Bayesian hierarchical model, see Section 3. The parameters in the family, for instance the precision in a Gaussian family, can also be further defined here by `control.family`. If the precision is known, it can be fixed for each data point. An example call when the precision is known to be 100 for all data points:

```r
res <- inla(formula_ARW1, family = "gaussian", data = test_data,
        control.family = list(hyper = list(prec =
                    list(initial = log(100), fixed = TRUE))))
```

A useful optional argument is `control.compute` for computing a multitude of model statistics.

4.3.3 Some mistakes to avoid

As INLA runs in C, with an interface in R, there are some issues caused by different environments in R and C. For instance, defining functions in R and using them inside the `rgeneric` definition will not work. This is also the case for functions imported from libraries. So, using a self-defined function `gen_inv` or `ginv` from "MASS" to compute the generalised inverse of a matrix, will both fail. However, INLA has made some useful functions as a part of their package, so using `INLA:::inla.ginv` successfully computes the generalised inverse.

Another issue I had was in regards to scopes in R. It turned out that the formula object in INLA used the scope it was defined in, regardless of which scope in R the INLA call was called from. This is illustrated in the code below for some suitable `test_data`.

```r
P <- 75
formula_RW1 <- y ~ f(time, model = "rw1")  + f(us, model = "iid")
res_RW1 <- inla(formula_RW1, family = "gaussian", data = test_data,
                control.compute = list(cpo = TRUE),
                control.family = list(hyper = list(prec =
                list(initial = log(1), fixed = TRUE))),
                scale = P)

test_function <- function(formula, df, Prec) {
res_RW1 <- inla(formula, family = "gaussian", data = df,
                control.family = list(hyper = list(prec =
                list(initial = log(1), fixed = TRUE))),
                scale = Prec)
  return(res_RW1)
}
res1 <- test_function(formula_RW1, test_data, 75)
```

Outside of the function scope, the INLA call runs smoothly. However, when running the same call, with the parameters passed through a function, suddenly INLA does not know what `Prec` is supposed to be. It gives the following error:

```
Warning in set.warn("scale", nm) :
  Argument 'scale=Prec' expanded to NULL or gave an error.
```

I believe this is because the INLA call uses the scope associated with the formula, which is assigned when the formula is defined. So, if we instead define the formula inside the function, it runs smoothly. This is demonstrated in the code block below, which gives no errors.

```r
test_function2 <- function(df, Prec) {
  formula_RW1_2 <- y ~ f(time, model = "rw1")  + f(us, model = "iid")
  res_RW1 <- inla(formula_RW1_2, family = "gaussian", data = df,
                  control.family = list(hyper = list(prec =
                                  list(initial = log(1), fixed =
                                   ↪  TRUE))),
                scale = Prec)
  return(res_RW1)
}
res2 <- test_function2(test_data, 75)
```

Additionally, another way of scaling seems to avoid the problem altogether. The code below does not fix the precision, and then scale it, it simply fixes it at the desired value immediately. The code below runs without errors, also when substituting `log(Prec)` with just `Prec`.

```r
test_function3 <- function(formula, df, Prec) {
  res_RW1 <- inla(formula, family = "gaussian", data = df,
                  control.family = list(hyper = list(prec =
                                  list(initial = log(Prec), fixed =
                                   ↪  TRUE))))
  return(res_RW1)
}
res3 <- test_function3(formula_RW1, test_data, 3)
summary(res3)
```

I am not sure why the second method of scaling works when the first method fails, but the lesson is to be on guard when using INLA inside function scopes in R when the formula is defined elsewhere.

5 Simulation study

The goal is to recreate parts of the simulation study in Aleshin-Guendel and Wakefield (2024), where they compare using a standard RW1 versus and adaptive RW1 in the latent layer of Bayesian hierarchical models for simulated data with different properties. I will also implement the models myself and use a different model parametrization and prior for the adaptive RW1. Furthermore, I will also expand some of their simulation study by introducing new mean trends in the simulated data, more on this in Section 5.1.1. In total, this includes simulating the data and implementing the models with INLA and then evaluating the model performance.

We expect the adaptive model to outperform the standard RW1 model for shocked data.

5.1 Design of the simulation study

The design of the simulation study is important as choices here will influence the results and conclusion. A key part is the data simulation where we need to simulate both shocked and non-shocked data, and ideally multiple types of shock as well. To include multiple types of shock, and also combining them in different ways, makes the simulation study more robust and we will try a few different mean trends for the data. Precise definitions of the models used are also important. For this we will use quite similar models where the only difference will be if we use a RW1 or an adaptive RW1 in the latent layer. The last part is the model evaluation criteria used to evaluate and compare the models. These must be chosen with care, as different criteria prioritize different aspects of the model.

5.1.1 Data generation

The data $\mathbf{y}$ are simulated from a normal distribution around η and we are using $N = 30$ time points. We denote the variance for the normal distribution as V and use $V \in \{\frac{1}{75}, \frac{1}{150}, \frac{1}{300}\}$. We choose N and V to align with the values used in Aleshin-Guendel and Wakefield (2024). Thus,

$$y_t|\eta_t \sim N(\eta_t, V) \quad \forall t \in \{1, ..., 30\}.$$

$\eta_t = \mu_t + b_t$ where μ is one of five mean trends and $b_t \overset{\text{iid}}{\sim} N(0, \tau^{-1})$ for some precision τ. The mean trends μ are called flat, delta, triangle, sine and offset sine, where the first three align with the simulated data from Aleshin-Guendel and Wakefield (2024). The means and a realization is shown in Figure 3. The flat mean is not shocked, while delta and triangle have a shock at $t \in S = \{9, ..., 15\}$. The sine mean follows a harmonic trend, additionally the offset sine has a delta shock for $t \in S$. Another element of shock is the precision τ in the white noise term $\mathbf{b}$. We use a precision $\tau_1 = 20$ for all $t \notin S$ and either $\tau_2 = 20$ or $\tau_2 = 10$ for $t \in S$. This means that either have the same noise for all time points, or we have more variance, lower precision, for $t \in S$. When $\tau_1 = \tau_2$ I will reference it has having constant precision, while non-constant precision references the case where $\tau_1 \neq \tau_2$. So,

$$\eta_t \sim N(\mu_t, \frac{1}{\tau_1}), \quad t \notin S$$
$$\eta_t \sim N(\mu_t, \frac{1}{\tau_2}), \quad t \in S.$$

In total this gives 30 different data configurations, and for each of them we make $n = 100$ simulations.

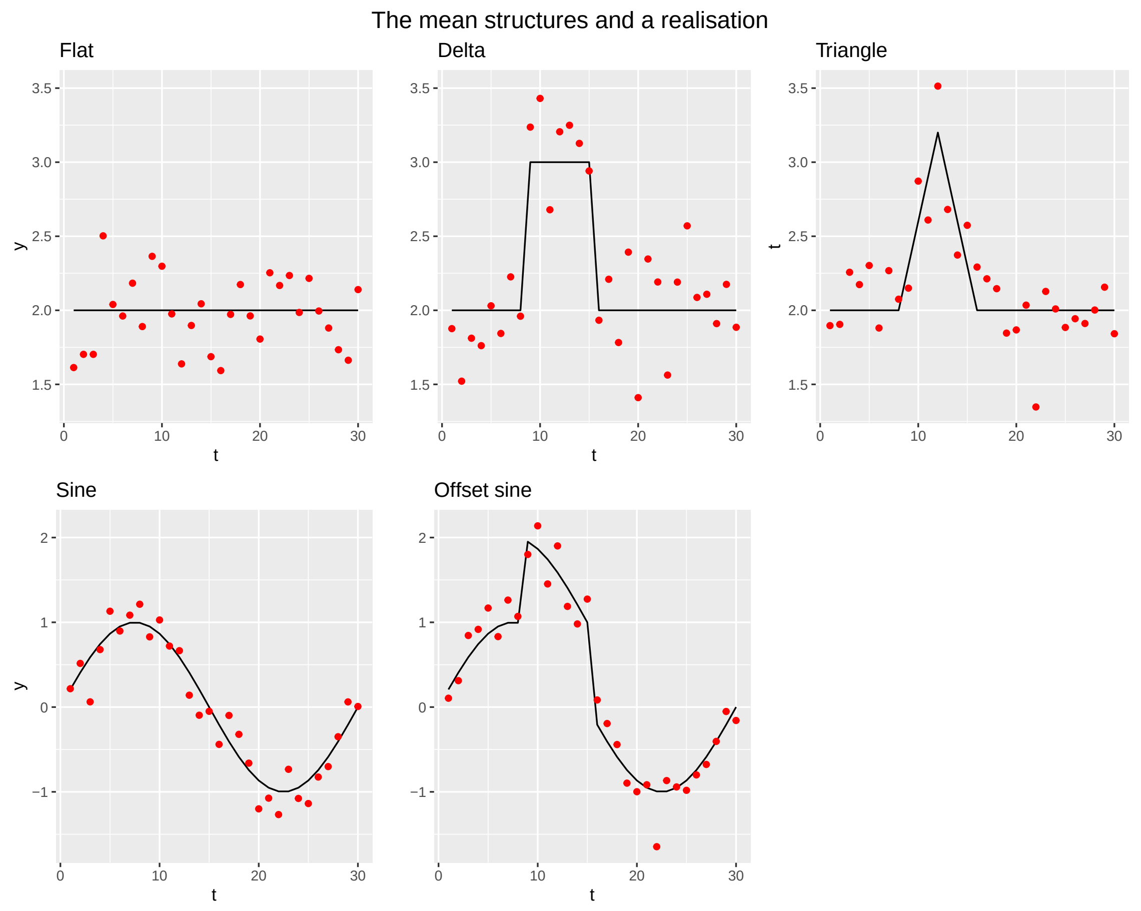

Figure 3: The mean μ in black with a simulated realization as red points. The data simulation uses constant precision $\tau = 20$ and observational variance $V = 1/300$.

5.1.2 The full adaptive model

We will use a Bayesian hierarchical model for both the adaptive model and the standard model. For the adaptive model we use a latent layer with an intercept β_0, the adaptive RW1 $\mathbf{x} = \{x_1, ..., x_N\}$ and an iid normal unstructured effect $\mathbf{u}$. Note that both At the observational layer we use a Gaussian model and fix the precision τ so $\tau = 1/V$ for the V used to simulate the data in question. This makes the linear predictor $\boldsymbol{\eta}$ approximate the hidden mean underneath the observed data, which is the sum of the mean trend and the white noise $\mathbf{b}$ from the previous section. This is an important reason for using the RMSE that we introduce later in Section 5.1.3. Thus, the model becomes

$$\begin{aligned} y_t | \eta_t, V &\sim N(\eta_t, V) \\ \eta_t &= \beta_0 + x_t + u_t \end{aligned}$$

The last layer assigns priors to the hyperparameters. Both the adaptive RW1 $\mathbf{x}$ and the iid normal $\mathbf{u}$ depend on their own hyperparameter for precision, say τ_x and τ_u. For the iid normal we use the model implemented by INLA and keep the default prior for the precision τ_u, namely $Gamma(1, 0.00005)$, with $\theta_0 = 4$. θ_0 is the initial value on the internal scale in INLA and equates to choosing a precision $\tau_u^0 = \exp(4)$ as there is a log-link between the internal and external scaling, see Section 4.3. For the adaptive RW1 we define the prior inside the `rgeneric` function and use one of the following priors.

- $Gamma(1, 0.00005)$
- $Gamma(1, 0.005)$
- $PC(1, 0.01)$

The PC, or penalized complexity, prior for a precision τ corresponds to using an exponential prior on σ where the rate λ is chosen so $P(\sigma > u) = \alpha$ for a $PC(u, \alpha)$ prior. This favours smaller sigma's, and thus larger precisions. We use $\theta_0 = 4$ for all the priors above. The adaptive model is implemented in R as

```
1 formula_ARW1 <- y ~ f(x, model = ARW1_model,
2     extraconstr = list(A = matrix(1, nrow = 1, ncol = N), e = 0)) +
3                 f(u, model = "iid")
4 res_ARW1 <- inla(formula_ARW1, family = "gaussian", data = test_data,
5         control.family = list(hyper = list(prec =
6             list(initial = log(P), fixed = TRUE))))
```

where $P = 1/V$ for the V from the data simulation and `ARW1_model` is defined in Section 4.3.2. We also include a sum-to-zero constraint for the adaptive RW1 on line 2 because it is an intrinsic GMRF, see Section 2.4.

The standard model uses a RW1 instead of the adaptive RW1 in line 1 and removes the sum-to-zero constraint as this is included automatically. Furthermore, we scale the model and with `scale.model = T` and also assign the prior for the precision τ. The rest is the same. INLA's RW1 will be used instead of the user defined because of runtime considerations, see Section A.

5.1.3 Model evaluation

We will use two model criteria to evaluate and compare the models. The first is the root mean square error (RMSE) between the simulated η for the data and the total latent effect fit by the given model in INLA, which for an INLA model `res` is retrieved as `res$summary.fitted.values$mean`.

$$RMSE(\boldsymbol{\eta}, \hat{\boldsymbol{\eta}}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\eta_i - \hat{\eta}_i)^2}.$$

Models with lower RMSE are preferred.

The second criteria is average proper logarithmic scoring(LS) from Gneiting and Raftery (2007). The minus in the definition below makes a lower LS indicate a better model fit.

$$\mathrm{LS}(\mathbf{p}, \boldsymbol{y}) = -\frac{1}{N}\sum_{i=1}^{N} \log p_i(y_i).$$

The main part is the scoring $p_i(y_i)$ which computes the likelihood of observing the observed value y_i. An approximation of this likelihood that can be predicted by INLA is the conditional predictive ordinate (CPO). The CPO uses the posterior distribution for a point y_i when training a model on the remaining data (Gómez-Rubio, 2020, Chapter 2.4). INLA predicts an approximation of the CPO, as it is time consuming to fit N models. INLA also gives a list of the error in each estimated CPO which is stored in `res$cpo$failure` where 0 indicates a good estimation and deviation from 0 indicates a problematic point where it is advised to make INLA compute the CPO explicitly.

5.2 Results

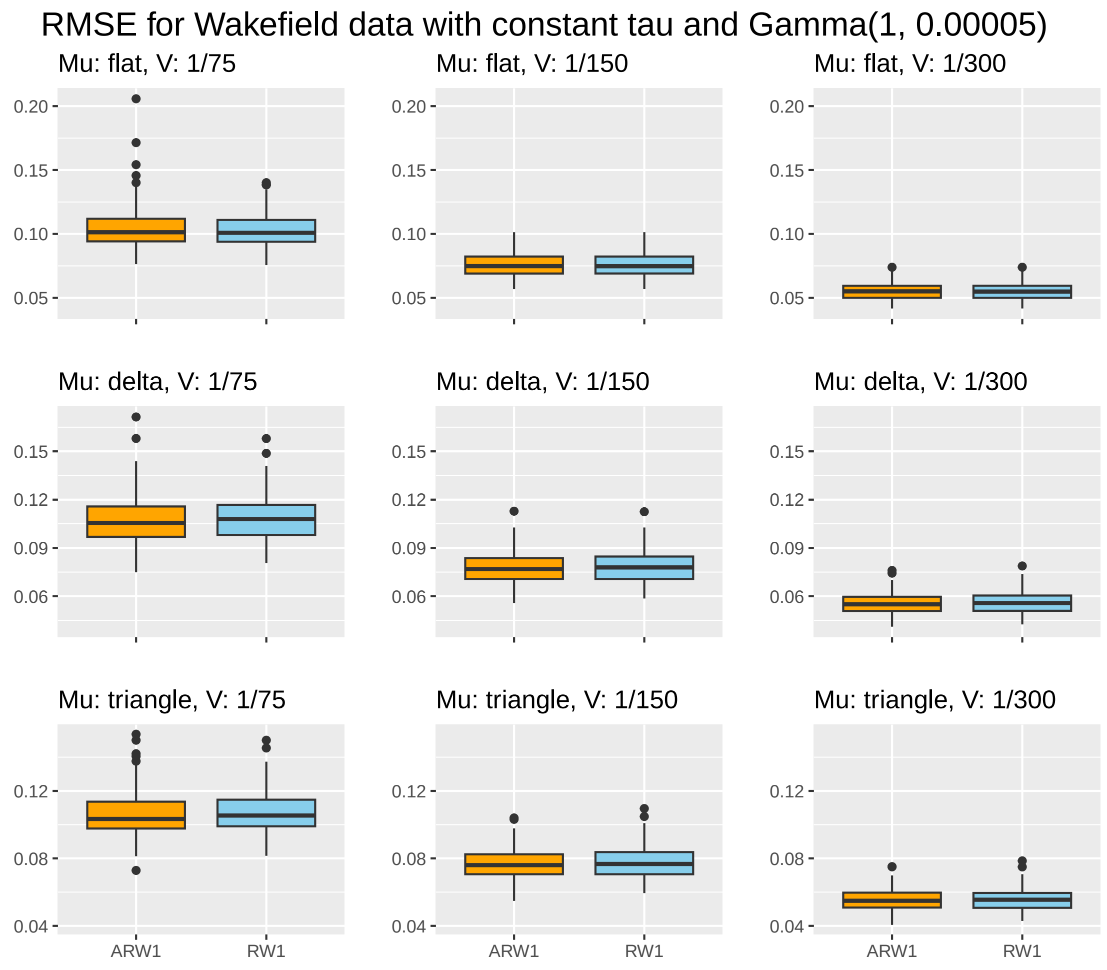

Figure 4: RMSE for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

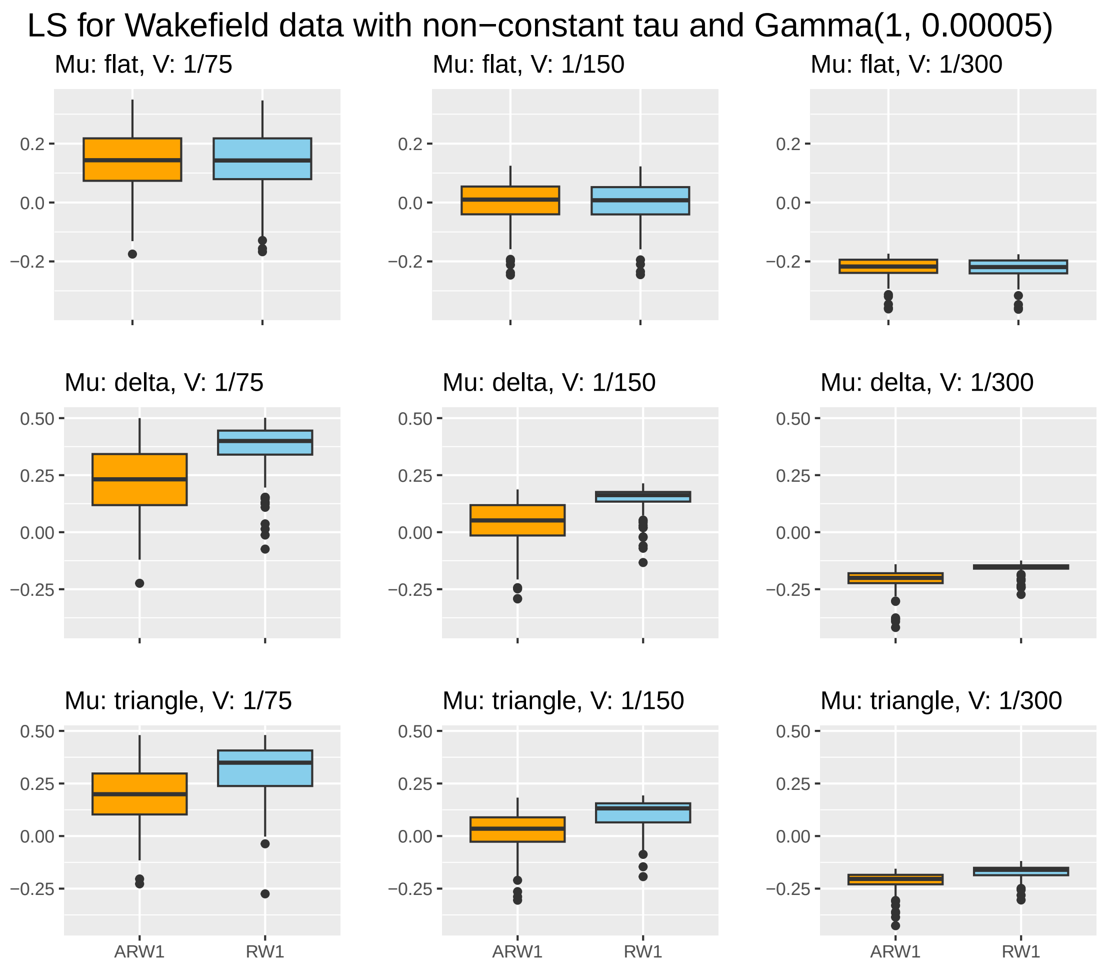

Figure 5: LS for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

The results for the mean trends flat, delta and triangle in Figure 4 and Figure 5 are noticeably different from the results presented in the Appendix of Aleshin-Guendel and Wakefield (2024). In particular, the figures for RMSE are similar, but the figures for LS have a significantly different scale. This could be because they use a different parametrization called BYM2 (Riebler *and others*, 2016), and thus different priors. Of note, I was unable to reproduce the figures in their article even with the code they supplied at their GitHub repository. In their code I changed `num_reps` from 200 to 100, as that was used in their article. Also changed `reps` to `num_reps` 4 places to make the code run, if not it gave an error when running process.results.R. I am unsure why their code does not produce the figures in their article. Even though I am unable to fully reproduce their simulation study, my results still show similar patterns and agree with my expectations.

The two models performed equally well in terms of RMSE. This holds for the mean trends flat, delta and triangle regardless of which prior was used. Note that the RMSE decreased when the observational variance V decreased for the three mean trends. This is as expected, as the models should fit the data better when there is less variance. Of note, there are some outliers for individual time series where the adaptive model is worse. For instance in Figure 4 in the top left plot, this was unexpected.

In terms of LS, there was a clear difference in model performance. Now, the adaptive RW1 is better for the mean trends delta and triangle. However, for a flat mean trend, their is no difference between the models. As for RMSE, we again observe an expected decreasing pattern for all the mean trends when the observational variance V decreases. Note that both delta and triangle mean trends have a larger LS than the flat mean for the same variance V, which is also as expected.

The two different latent terms result in models that perform very similarly for a flat mean trend, even with non-constant precision τ. For the shocked mean trends delta and triangle the adaptive model performs better in terms of LS, and equally well in terms of RMSE. However, their are some outliers among individual data points where the standard model performs better. These results are not sensitive to the choice of prior applied to the precision parameters in the random walks, namely $Gamma(1, 0.00005), Gamma(1, 0.005)$ and $PC(1, 0.01)$, and the other figures can be found in Appendix Section B. So, for the simulated data with known shocks the adaptive RW1 on average fits the trends in the data better and with less uncertainty.

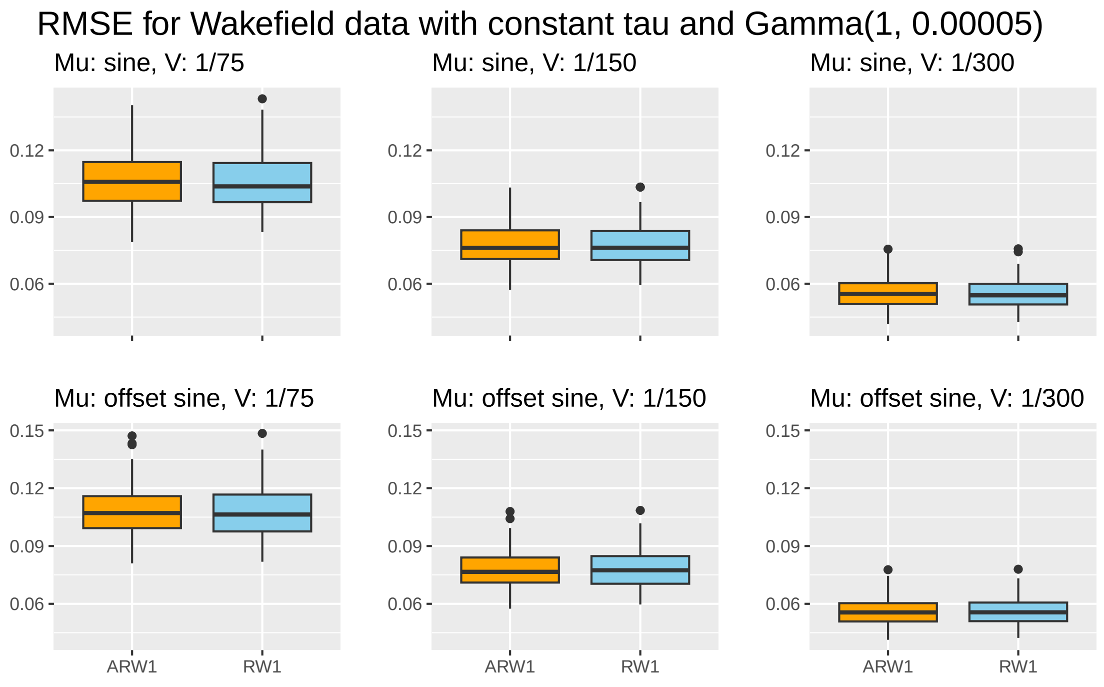

Figure 6: RMSE for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

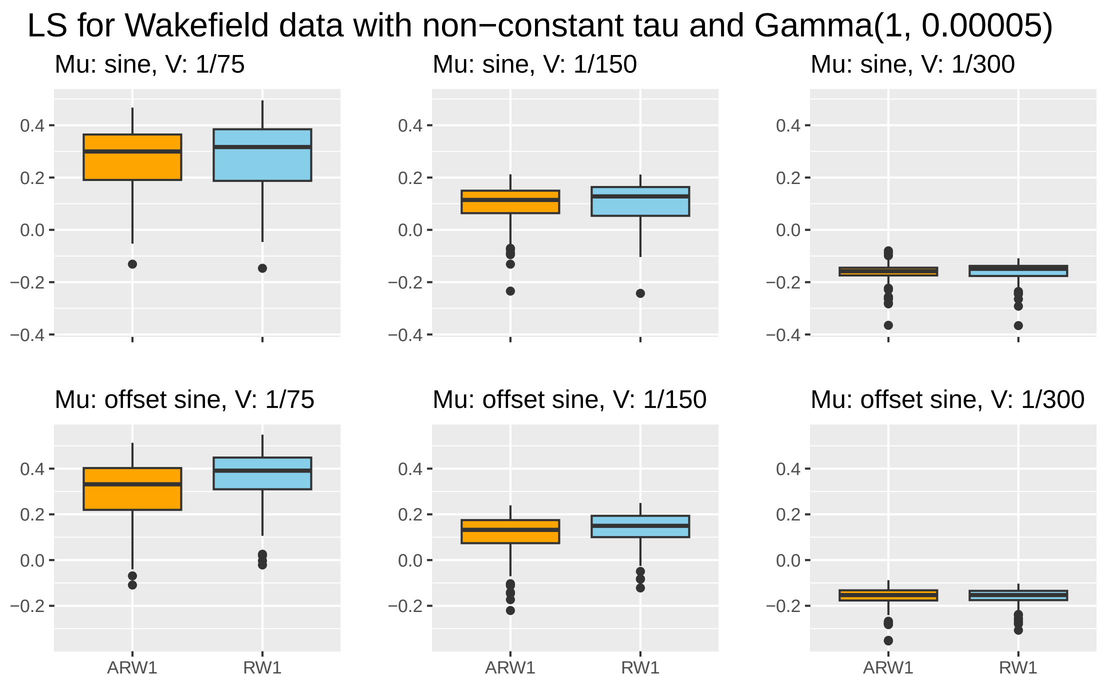

Figure 7: LS for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

For the novelty means introduced in this article, sine and offset sine, the results for constant precision τ can be seen in Figure 6. The figure shows that the RW1 and the adaptive RW1 perform practically the same in this setting. Furthermore, the models also perform similarly for the other priors and in terms of LS, which can be seen in Appendix Section C.1. The only setting where there is an observable difference is for LS with the largest variance $V = 1/75$ and the mean trend offset sine, where the adaptive model is better.

When we simulate the data with non-constant precision τ there are some discrepancies between the two models. For the LS this is shown in Figure 7 the adaptive RW1 outperforms the RW1, especially for the mean offset sine with larger observational variance V. For larger variance it also appears that the adaptive RW1 is slightly better for the mean sine as well. For the smallest observational variance the models are inseparable. These trends also appear for the other priors. However, they are not present for the RMSE, where the models still perform equally well. See Section C.2 for the other figures with non-constant precision τ.

6 Data analysis of death rates from Norway

For the applied study we want to analyse deaths in the Norwegian population in a temporal setting. We are specifically interested in periods where we expect shocks in the data. For instance the Spanish flu in 1918-1919 (Borza, 2001) and WW2 from 1940-1945. The data comes from the Human mortality database (HMD, 2024), which is publicly accessible, but requires creating an account. Note that the data in the Human mortality database is updated and revised periodically. All the data can be found in the table "Complete Data Series" and I used the dataset for deaths with a 5-year age interval and data for each calender year, which is located in the column 5x1 and the row named Deaths. As we need data for the Norwegian population for scaling the deaths I likewise fetched population data with the same granulation for age and calender year from the row named Population size. I copied the data from the supplied txt files and removed the first line with information about the datasets.

The Spanish flu mainly killed young people and the same could be said for WW2, as mainly young men went to war. This is also supported by the data, see Figure 8, where we clearly observe the effect of the Spanish flu and WW2 for the age group 20-24 while these shocks are less clear for older age groups, say 60-64. Therefore, I will investigate the death rates for the ages 20-24 from 1900 to 1970 with the shocked years being determined as $S = \{1918, 1919, 1940, 1941, 1942, 1943, 1944, 1945\}$.

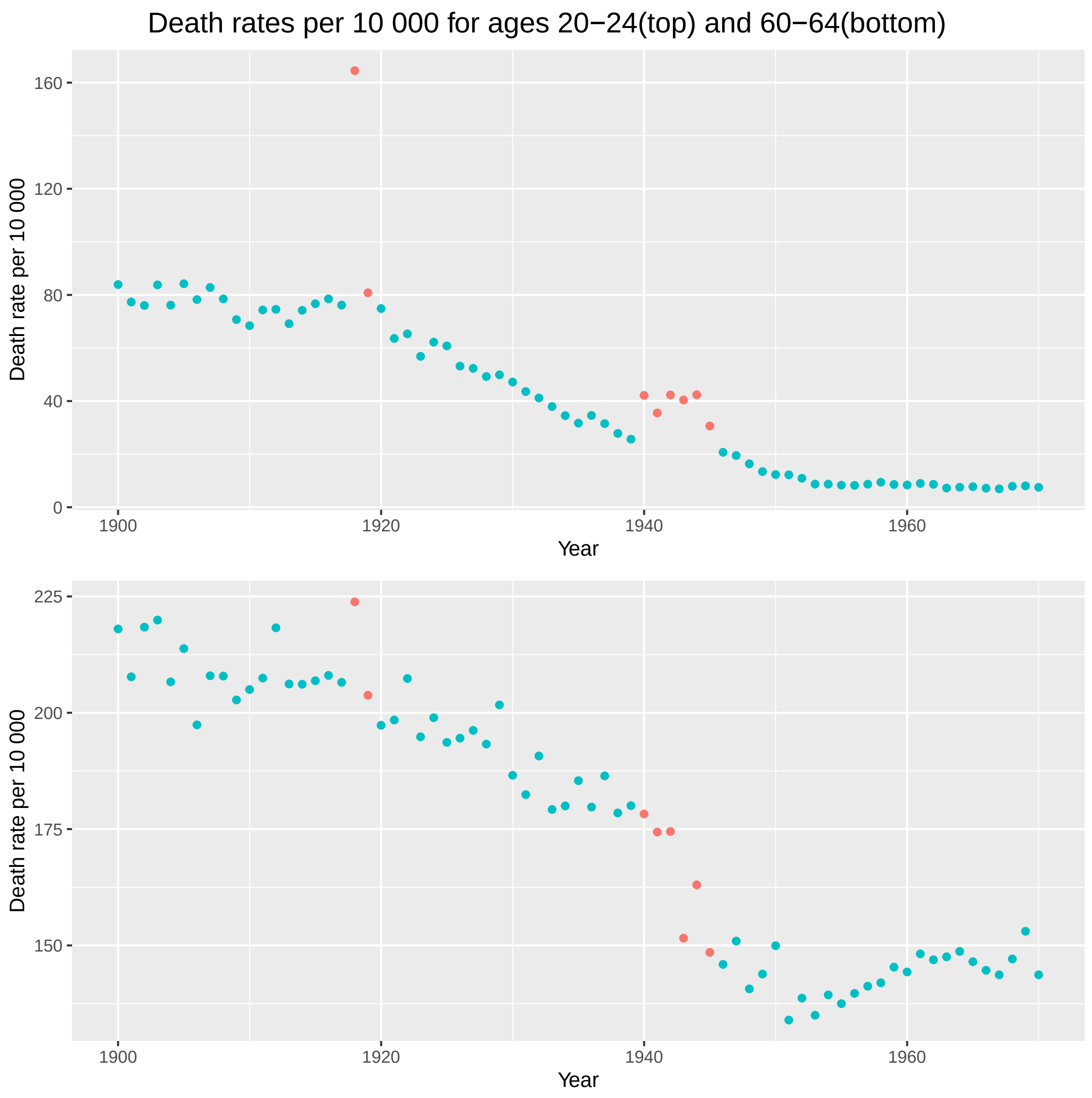

Figure 8: The death rate per 10 000 for two age groups from 1900 to 1970. For $t \in S$ the points are red. On the top we see ages 20-24 and on the bottom we see 60-64. Note that the y-axis are not on the same scale.

6.1 Models

As in the simulations study we will again use a few different models to explore the adaptive GMRFs in different settings. All the models will include a temporal random walk $\mathbf{x}$ in the latent layer, either a RW1 or an adaptive RW1. For the observational layer we will either use a Poisson or a negative binomial likelihood. While the Poisson likelihood keeps the mean and variance equal, the negative binomial likelihood allow the variance to exceed the mean. These are both used with count data, which deaths are, and we will scale the deaths by the population so the latent layer models the death rate on the log-scale, because we use a log-link with both likelihoods.

For $t = 1, ..., 71$ and a Poisson likelihood we have

$$
\begin{aligned}
&y_t \sim Poisson(E_t * \exp(\eta_t)) \\
&\eta_t = \beta_0 + x_t.
\end{aligned}
$$

The rate in the Poisson is a product of the population E_t and the exponential of the linear predictor η_t, which approximates the death rate on the log-scale. For the negative binomial likelihood we get

$$
\begin{aligned}
&y_t \sim NB(n, p_t) \text{ with mean } \mu_t = n\frac{1-p_t}{p_t} \\
&\mu_t = E_t \exp(\eta_t) \\
&\eta_t = \beta_0 + x_t.
\end{aligned}
$$

Here we use a log-link between the mean of the negative binomial and the linear predictor η_t. We again also scale by the population E_t so the linear predictor approximates the log of the death rate. In both cases the intercept β_0 will use the default prior in INLA. For $\mathbf{x}$ we will either use a RW1, an adaptive RW1 with two total precision parameters or an adaptive RW1 with an independent precision for each shock period, leading to three hyperparameters. We will evaluate the models with RMSE and LS, as well as a plot with credibility intervals to see how well they fit the data. The RMSE is calculated between the mean of the fitted values and the scaled death rate, so

$$
RMSE(\mathbf{y}, \mathbf{E}, \boldsymbol{\eta}) = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(\eta_t - \frac{y_t}{E_t})^2}
$$

where $\mathbf{y}$ is the death counts, $\mathbf{E}$ is the population and $\boldsymbol{\eta}$ is the fitted values. For the model with separate precisions for the different shocks we need another `rgeneric` definition, which is included in the Appendix Section D.

6.2 Results

With the Poisson likelihood the models perform similarly in terms of interpolating the data. In Figure 11 all the models visually fit the death rate per 10 000 well and have a similar uncertainty in the latent layer. The observational variance is also the same as we are using the Poisson likelihood. Figure 9 shows that the RW1 has the best RMSE, while the two adaptive models have a better LS. Note that the two adaptive models are essentially equal in terms of RMSE and LS. In Figure 13 we observe that the random walks in the latent layers for each model follows the same trends and have a similar uncertainty. Both the model fits and the evaluation criteria were indifferent to the choice of prior and the figures for the $Gamma(1, 0.00005)$ and $Gamma(1, 0.005)$ priors can be seen in Section E.1.

	Method	RMSE	LS
1	RW1	3.631839e−05	5.727307
2	ARW1	4.855061e−05	5.323251
3	ARW1_3tau	4.896343e−05	5.328210

Figure 9: RMSE and LS for the three models with a Poisson likelihood and a $PC(1, 0.01)$ prior.

	Method	RMSE	LS
1	RW1	3.765023e−04	5.995746
2	ARW1	6.664163e−05	5.366324
3	ARW1_3tau	6.504811e−05	5.368691

Figure 10: RMSE and LS for the three models with a negative binomial likelihood and a $PC(1, 0.01)$ prior.

If we instead assume that the model follows a negative binomial distribution we get quite different results. From Figure 12 it is clear that the model with the RW1 struggles with the shocked time points. The model greatly underestimates the spike at 1918, and is also visibly off for the $t \in \{1940, ..., 1945\}$. Both the adaptive models have no issues with fitting the data. The 95% credibility interval is also much wider for the RW1 compared to the adaptive models. This is further supported by Figure 14, where the RW1 has a much wider credibility interval and also greatly underestimates the spike at 1918, compared to the two adaptive models. From the model criteria in Figure 10 the RW1 is worse for both RMSE and LS, while the two adaptive models are essentially interchangeable in terms of RMSE and LS. These results are similar regardless of the prior used for the precision in the random walks, and the figures for the $Gamma(1, 0.00005)$ and $Gamma(1, 0.005)$ priors can be seen in Section E.2.

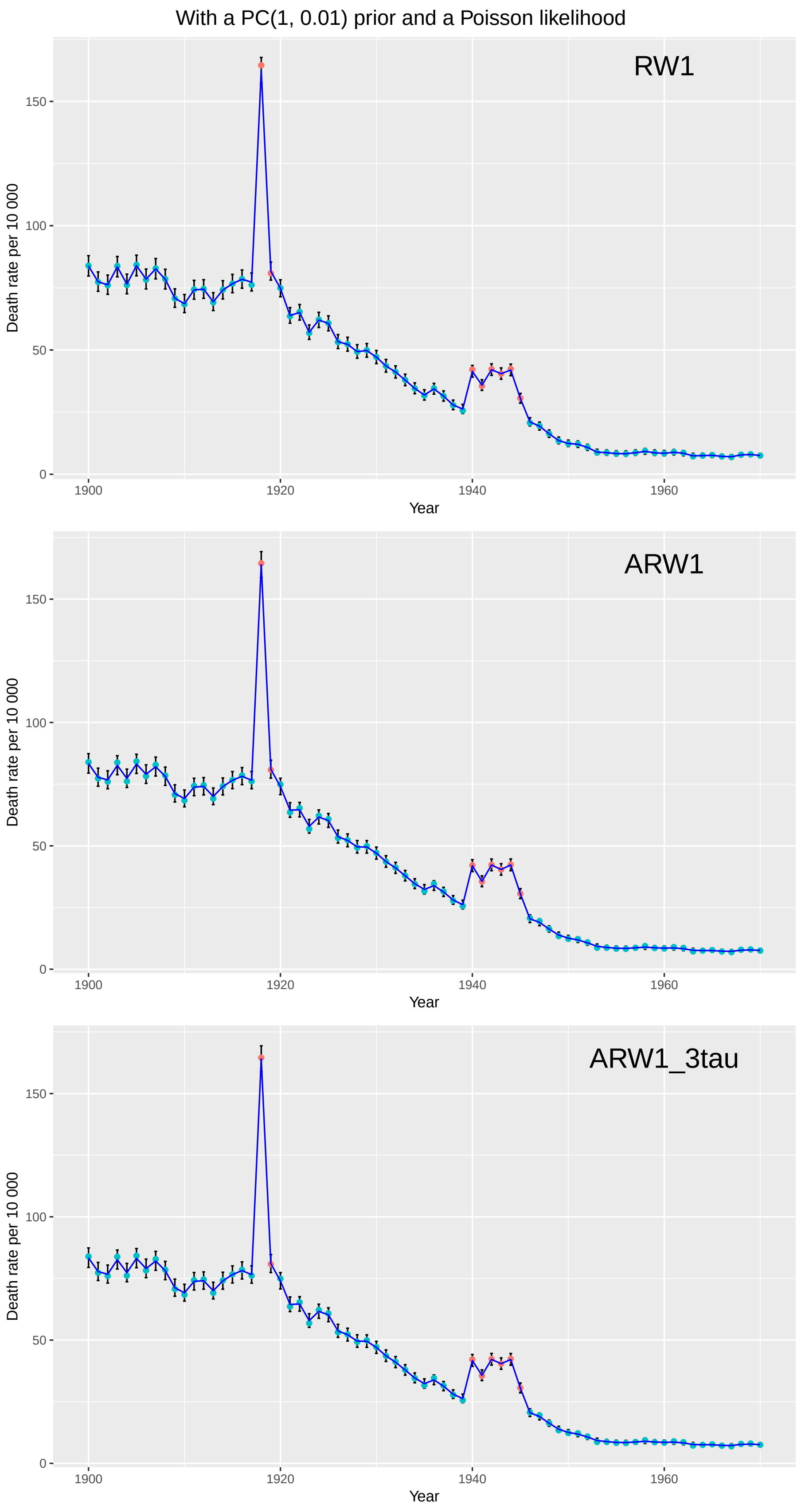

Figure 11: The model fit and the actual data for the death rate per 10 000 per year. The shocked points are in red. The models were fit using a Poisson likelihood and a $PC(1, 0.01)$ prior. The 95% credibility interval is shown for each point.

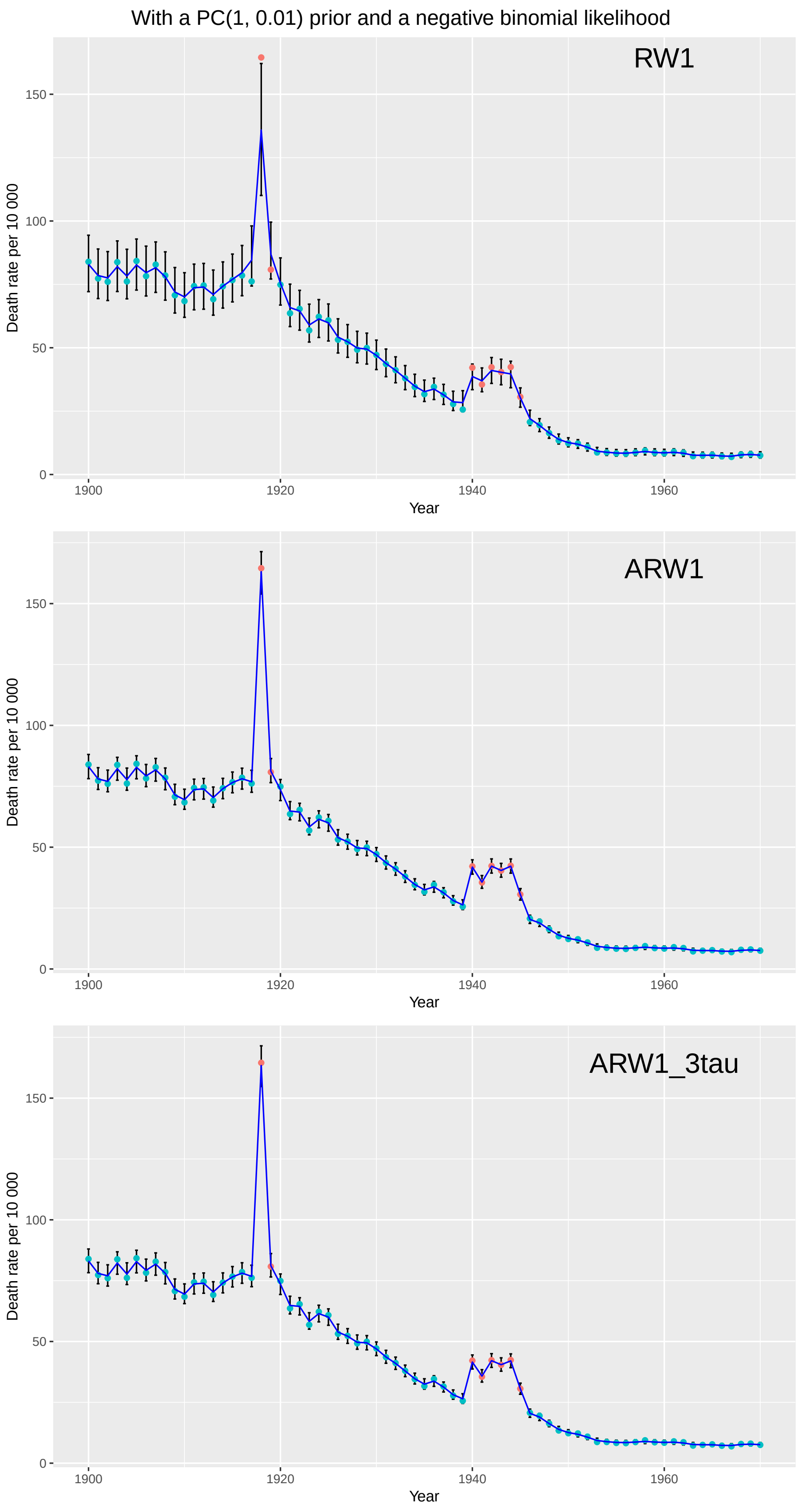

Figure 12: The model fit and the actual data for the death rate per 10 000 per year. The shocked points are in red. The models were fit using a negative binomial likelihood and a $PC(1, 0.01)$ prior. The 95% credibility interval is shown for each point.

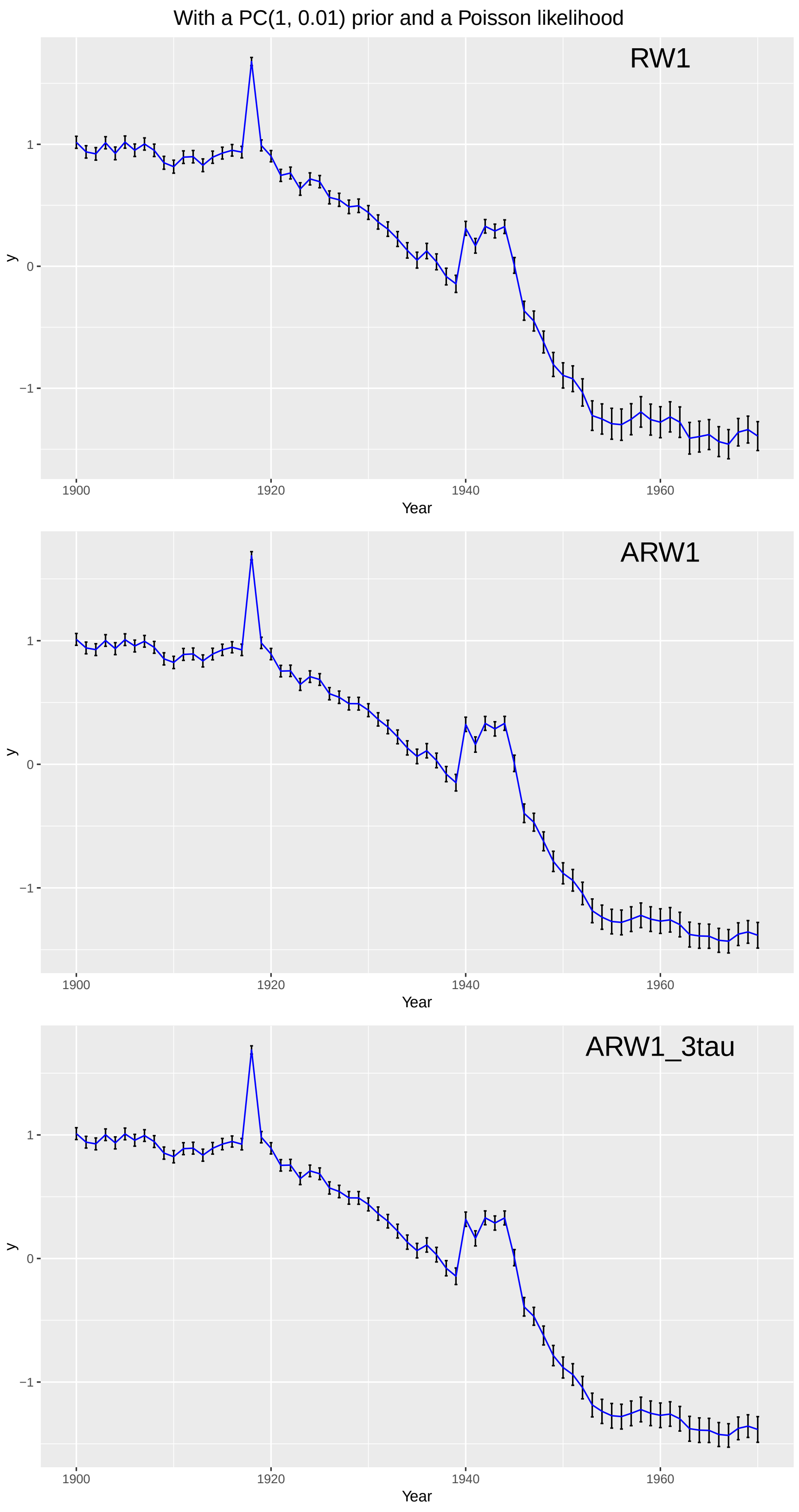

Figure 13: The random walk components in the latent layer for each model. The models were fit using a Poisson likelihood and a $PC(1, 0.01)$ prior. The 95% credibility interval is shown for each point.

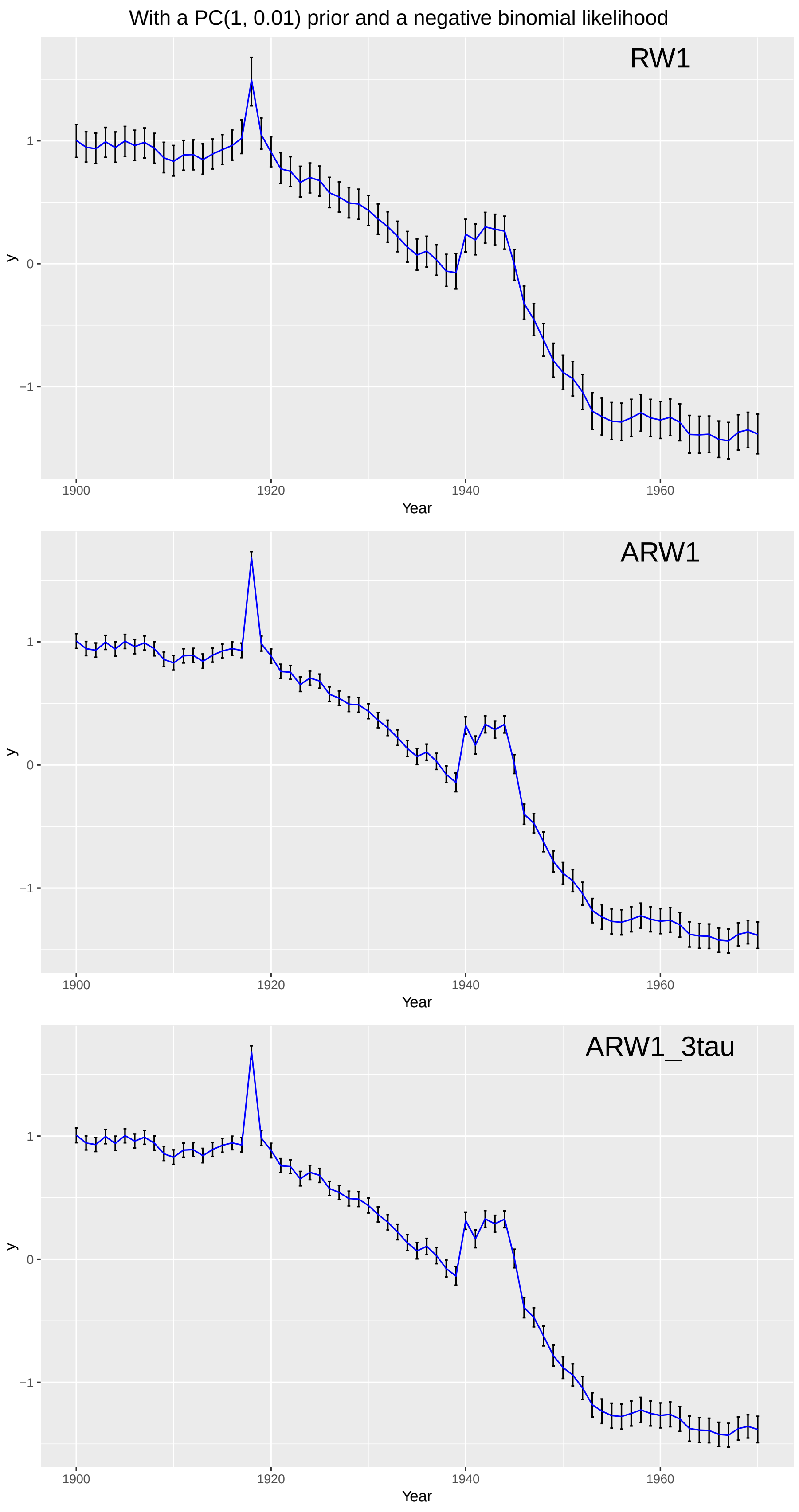

Figure 14: The random walk components in the latent layer for each model. The models were fit using a negative binomial likelihood and a $PC(1, 0.01)$ prior. The 95% credibility interval is shown for each point.

Of note, all the models performer better in terms of RMSE and LS when using a Poisson likelihood compared to a negative binomial likelihood. Also worth noting that the model with a RW1 becomes a lot worse, both in terms of LS and RMSE, as well as the visible model fit. The adaptive models have a small increase in RMSE and LS. This holds for all the three priors. Furthermore, the two adaptive models performed equally regardless of likelihood and priors, and it seems like the inclusion of a third hyperparameter was unnecessary with the data used.

7 Conclusion

For data with known shocks, we have seen that adaptive Gaussian Markov random fields are better at fitting the data than the standard GMRFs. However, not all types of shock benefit equally from using adaptive GMRFs. For instance, the change between constant and non-constant precision τ in the simulation study had no real effect. The shocks to the mean, for instance delta and triangle, showed large improvements when using adaptive models with regards to LS, but it had no effect on the RMSE. For the mean trends sine and offset sine the RMSE was again the same for both models in the different settings. However, the adaptive RW1 had a better LS for the trend offset sine. For data with no shocks the models with adaptive GMRFs performed on par with models with standard GMRFs, which means that standard GMRFs would be preferred because it has fewer model parameters. It is worth noting that for individual time series, the adaptive models can perform worse than the standard models, at least in terms of RMSE and LS. However, on average the adaptive models are on par, and often better than the standard models.

For the Norwegian death data we saw that the models have different strengths. When using a negative binomial likelihood, the adaptive models were better. However, when using a Poisson likelihood it was less clear. The RW1 had a better RMSE and a worse LS, but was at least on par with the adaptive models with one or two additional hyperparameters. Overall, I believe it was unnecessary to use adaptive random walks for this data analysis as the Poisson likelihood gave the best results for all models, and the RW1 was on par with the adaptive models in this case.

To conclude, the simulation study showed that there are situations where adaptive models are preferred, for instance data with known shocks. At least when the shock is sufficiently impactful. Additionally, the application for Norwegian death rates illustrated that the standard models can compete with the adaptive models, even for data that has clear shocks in terms of spikes. Overall, adaptive GMRFs can be a powerful analytical tool, but it sometimes results in more complex models than necessary.

References

Aleshin-Guendel, S. and Wakefield, J. (2024). Adaptive gaussian markov random fields for child mortality estimation, *Biostatistics*.

Besag, J., York, J. and Mollié, A. (1991). Bayesian image restoration, with two applications in spatial statistics, *Annals of the Institute of Statistical Mathematics*.

Borza, T. (2001). Spanskesyken i norge 1918-19 [spanish flu in norway 1918-19], *Tidsskrift for den Norske laegeforening* **121(30)**: 3551–3554.

Fahrmeir, L., Kneib, T., Lang, S. and Marx, B. D. (2022). *Regression*, Springer Berlin. Heidelberger Platz 3, 14197 Berlin, Germany.

Freni-Sterrantino, A., Ventrucci, M. and Rue, H. (2018). A note on intrinsic conditional autoregressive models for disconnected graphs, *Spatial and Spatio-temporal Epidemiology* **26**: 25–34. Available from: https://www.sciencedirect.com/science/article/pii/S1877584517301600.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation, *Journal of the American Statistical Association* **102**(477): 359–378.

Gómez-Rubio, V. (2020). *Bayesian Inference with INLA*, Chapman & Hall/CRC Press. Boca Raton, FL.

HMD (2024). *Norway*, https://www.mortality.org/Country/Country?cntr=NOR[Accessed 18.12-2024].

Riebler, A., Sørbye, S. H., Simpson, D. and Rue, H. (2016). An intuitive bayesian spatial model for disease mapping that accounts for scaling, *Statistical Methods in Medical Research* **25**: 1145–1165.

Rue, H. (2021). *Defining a latent model in R or C.*

Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: Theory and Applications*, Chapman & Hall/CRC. 6000 Broken sound parkway NW.

Sørbye, S. H. and Rue, H. (2014). Scaling intrinsic gaussian markov random field priors in spatial modelling, *Spatial Statistics* **8**: 39–51. Spatial Statistics Miami. Available from: https://www.sciencedirect.com/science/article/pii/S2211675313000407.

Appendix

A Comparison of the standard RW1 and the user defined RW1

Code for comparing the two RW1 models. Note that we need to scale the RW1 defined in INLA and we apply a sum-to-zero constraint for the user defined RW1. The created figure, in the main text, shows that both the mean and the standard deviation for the fitted values overlap. Furthermore, the estimated hyperparameters also have roughly the same quantiles, note that we need to convert the internal theta parameter for the user defined RW1 to the precision scale with an exponential link function.

```r
exp_func <- function(x){
  return(exp(x))
}
set.seed(15)
t <- 1:N
y <- sin(t/N*2) + rnorm(N, sd = 0.2)
test_data <- as.data.frame(list(y = y, time = t, us = 1:N))

N <- 50 #is defined further up as well
RW1_model <- inla.rgeneric.define(inla.rgeneric.RW1.model1, N = N)

#The INLA formula for a latent model with intercept and user defined
 ↪  RW1
formula_M <- y ~ f(time, model = RW1_model,
                extraconstr = list(A = matrix(1, nrow = 1, ncol = N),
                ↪  e = 0)) +
  f(us, model = "iid")
res_M <- inla(formula_M, family = "gaussian", data = test_data)

#The standard RW1 model from INLA
formula_I <- y ~ f(time, model = "rw1", scale.model = T) + f(us,
 ↪  model = "iid")
res_I <- inla(formula_I, family = "gaussian", data = test_data)

#comparing the fitted hyperparameters
summary(res_M)
summary(res_I)

inla.qmarginal(c(0.025, 0.5, 0.975), inla.tmarginal(exp_func,
 ↪  res_M$marginals.hyperpar$`Theta1 for time`))

mean_plot <- ggplot() +
  geom_point(aes(x = res_M$summary.fitted.values$mean,
                 y = res_I$summary.fitted.values$mean)) +
  geom_abline(intercept=0,slope=1, linetype="dashed", color="blue") +
  labs(title = "Comparison of the Mean", x = "Manual RW1", y =
   ↪  "INLA's RW1") +
  theme_minimal()
sd_plot <- ggplot() +
  geom_point(aes(x = res_M$summary.fitted.values$sd,
                 y = res_I$summary.fitted.values$sd)) +
  geom_abline(intercept=0,slope=1, linetype="dashed", color="blue") +
  labs(title = "Comparison of the standard deviation", x = "Manual
  ↪  RW1", y = "INLA's RW1") +
  theme_minimal()

comp_plot <- ggarrange(mean_plot, sd_plot, ncol = 2)
```

The generated output is shown below. The hyperparameters align well, but I am not sure what causes the difference in `Marginal log-Likelihood`. Also worth noting that the user defined function has a much greater runtime with 7.15*s* versus 1.34*s*.

```
> summary(res_M)
Time used:
    Pre = 0.708, Running = 6.32, Post = 0.118, Total = 7.15
Fixed effects:
            mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept) 0.77 0.025       0.72     0.77       0.82 0.77   0

Random effects:
  Name          Model
    time RGeneric2
   us IID model

Model hyperparameters:
Precision for the Gaussian observations
mean       sd 0.025quant 0.5quant 0.975quant    mode
32.30 7.14e+00      20.36    31.59      48.32   30.29
Theta1 for time
3.65 5.06e-01       2.62     3.67       4.61    3.73
Precision for us
22006.56 2.43e+04    1445.27 14377.72   86503.34 3928.46

Marginal log-Likelihood:  -7.97

> summary(res_I)
Time used:
    Pre = 0.484, Running = 0.571, Post = 0.281, Total = 1.34
Fixed effects:
            mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept) 0.77 0.025       0.72     0.77       0.82 0.77   0

Random effects:
  Name          Model
    time RW1 model
   us IID model
```

```

Model hyperparameters:
Precision for the Gaussian observations
mean       sd 0.025quant 0.5quant 0.975quant    mode
32.30     7.13      20.34    31.60      48.25   30.34
Precision for time
43.80    22.67      13.69    39.20     100.39   30.80
Precision for us
22003.76 24286.63    1445.09 14377.77   86476.24 3928.07

Marginal log-Likelihood:  -59.47

> inla.qmarginal(c(0.025, 0.5, 0.975), inla.tmarginal(exp_func,
↪  res_M$marginals.hyperpar$`Theta1 for time`))
[1] 13.78134 39.31982 99.48916
```

B Plots for the trends flat, delta and triangle

The result plots for the simulated data with trends flat, delta and triangle. The precision τ is constant or non-constant. Finally, we apply three different priors to the precision parameter in the random walk in each case.

B.1 Plots with constant precision

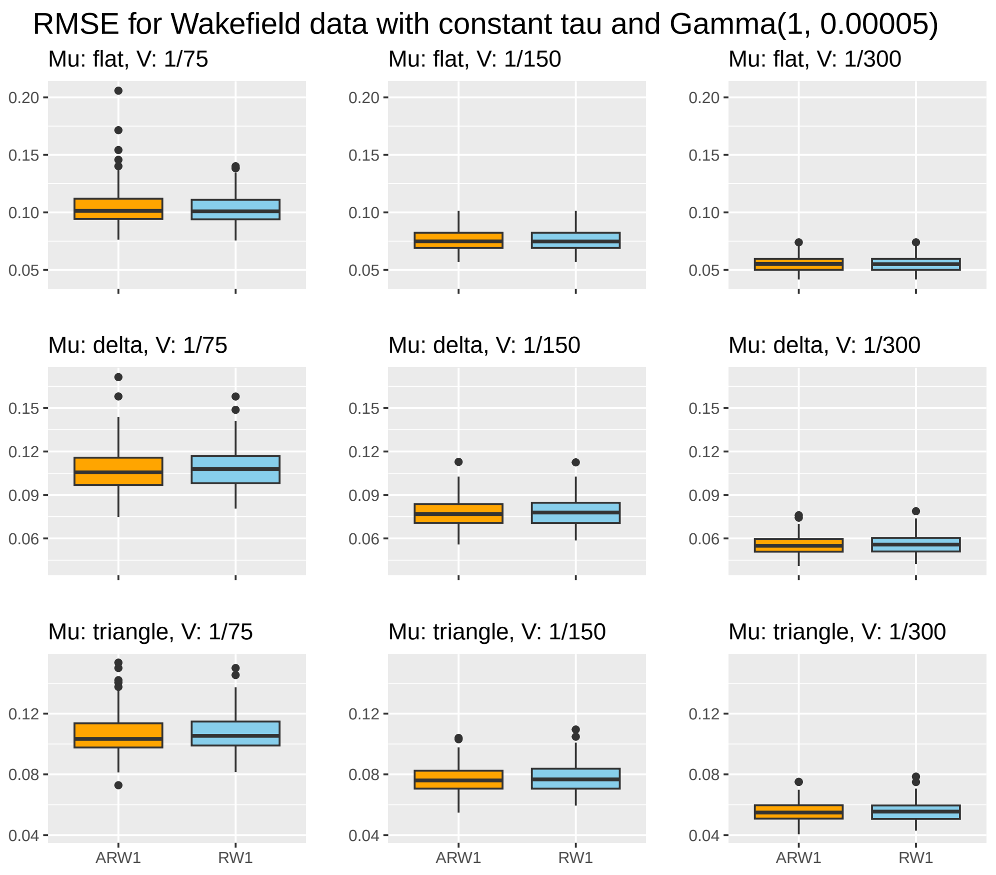

Figure 15: RMSE for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

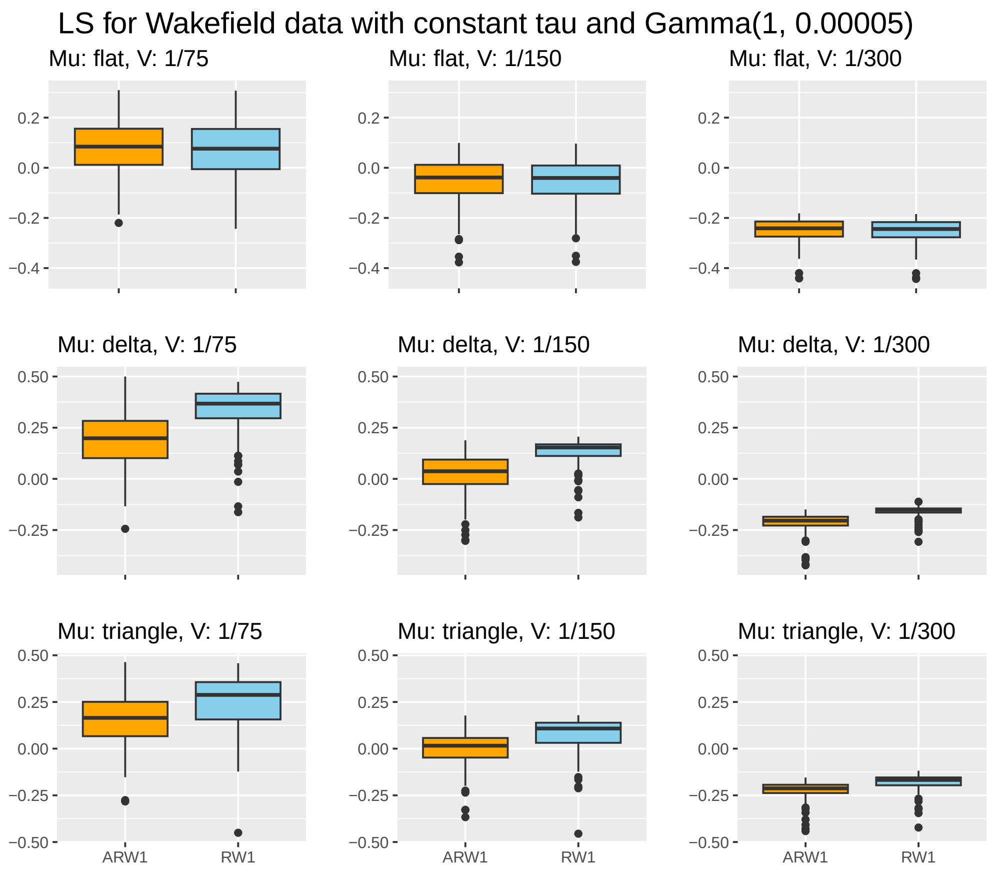

Figure 16: LS for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

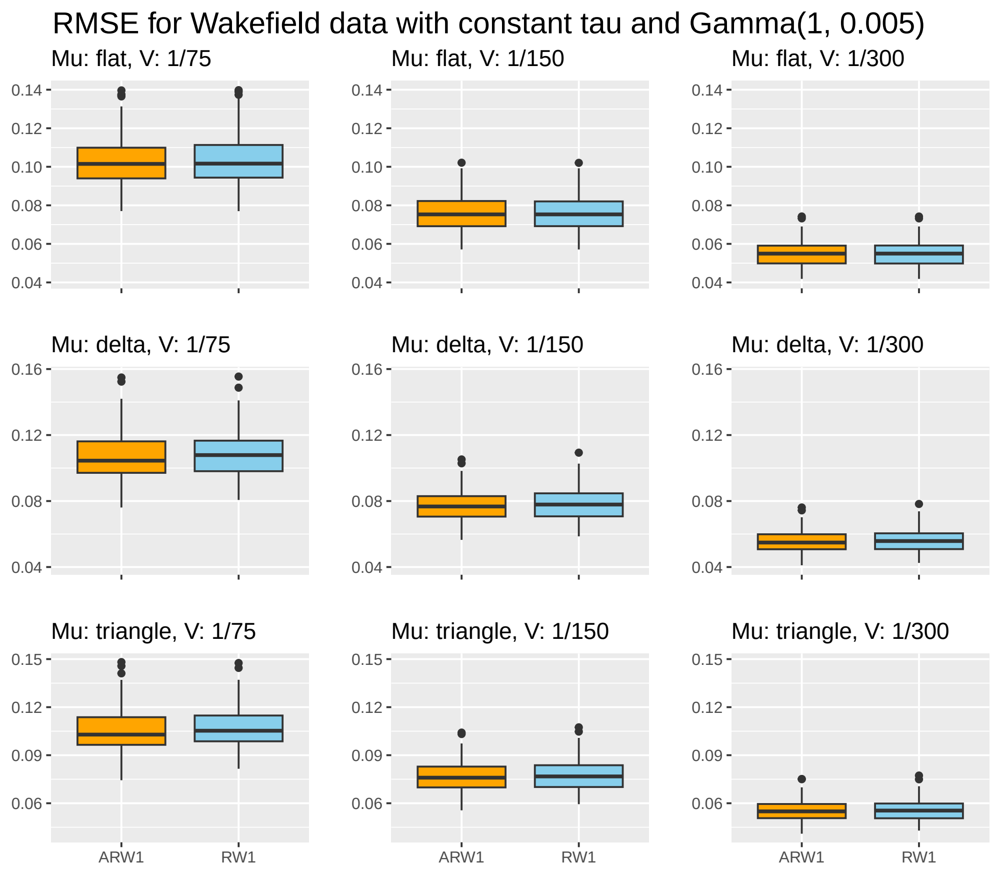

Figure 17: RMSE for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.005)$.

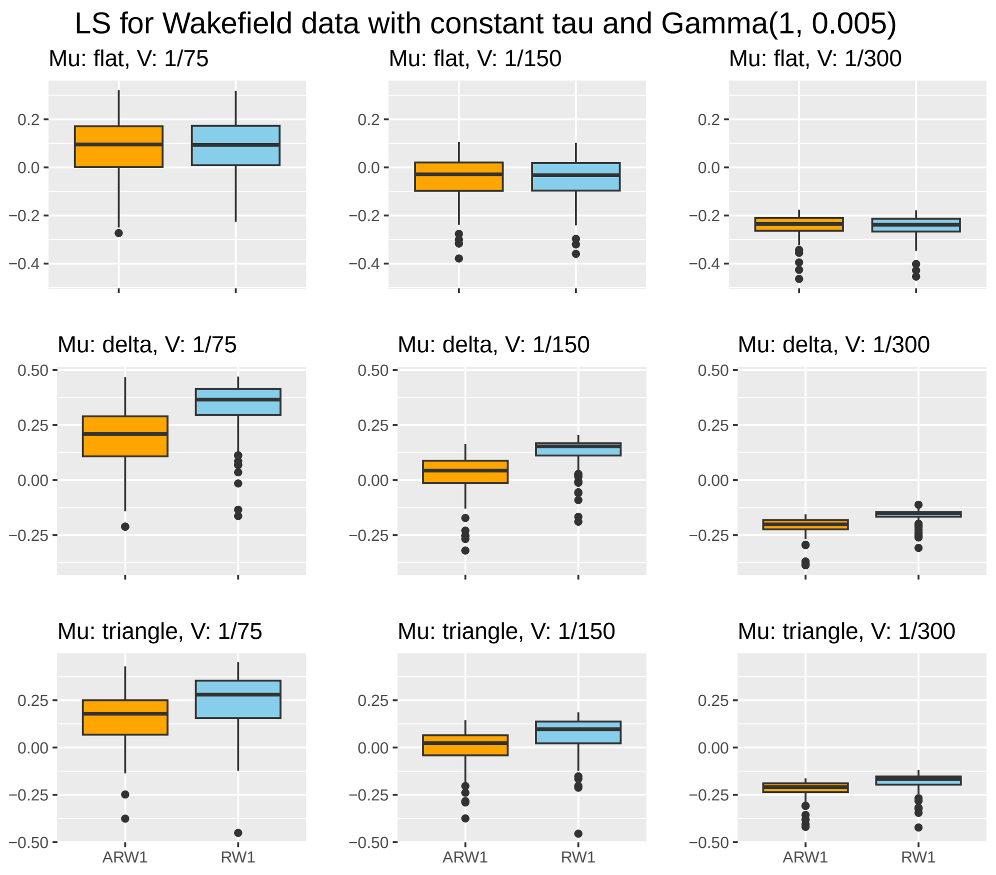

Figure 18: LS for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.005)$.

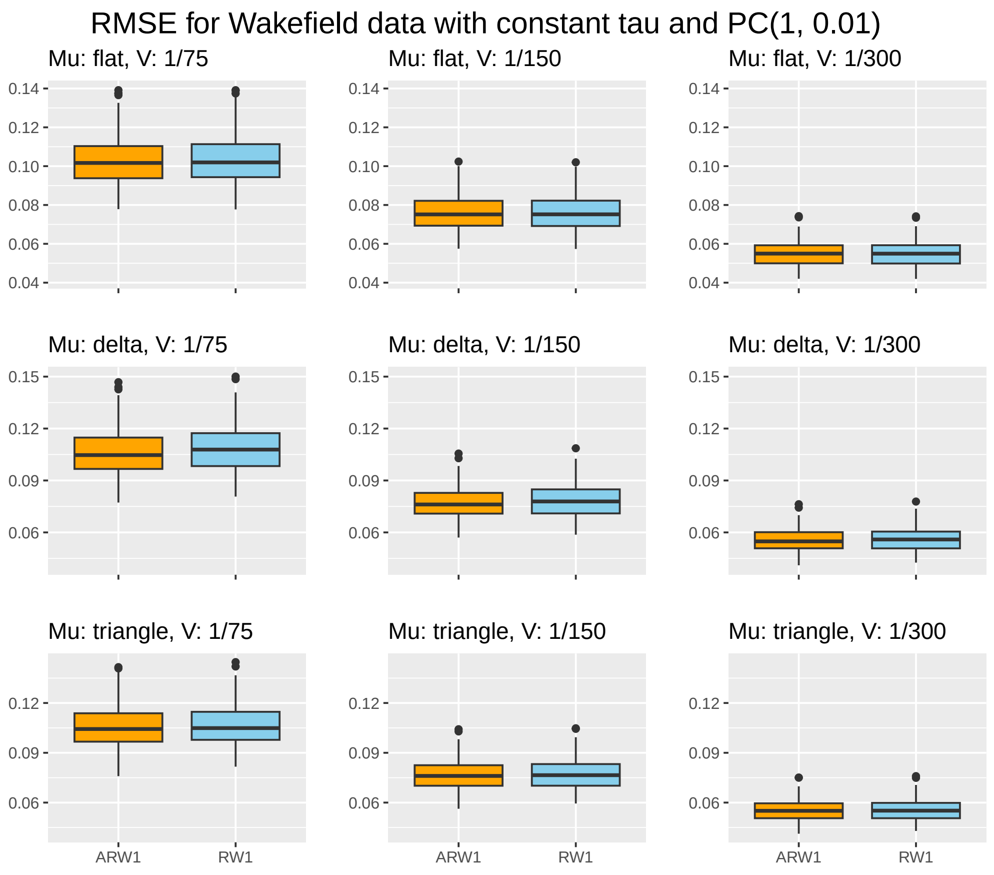

Figure 19: RMSE for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $PC(1, 0.01)$.

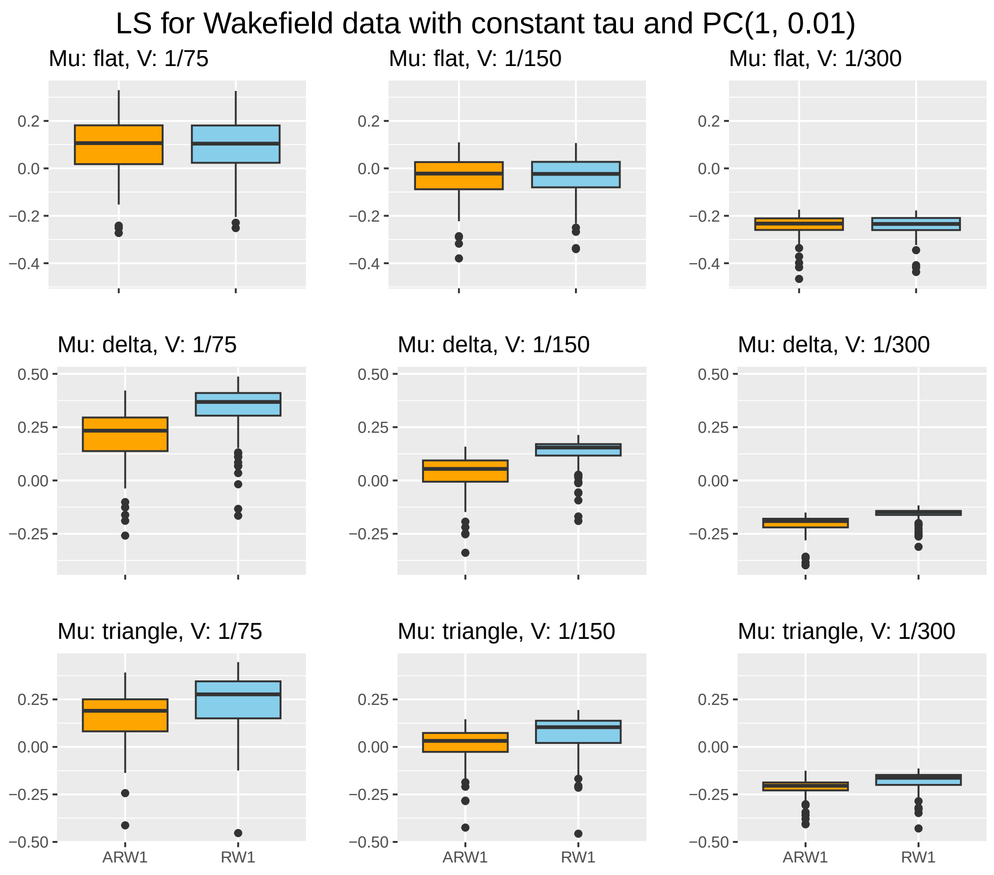

Figure 20: LS for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $PC(1, 0.01)$.

B.2 Plots with non-constant precision

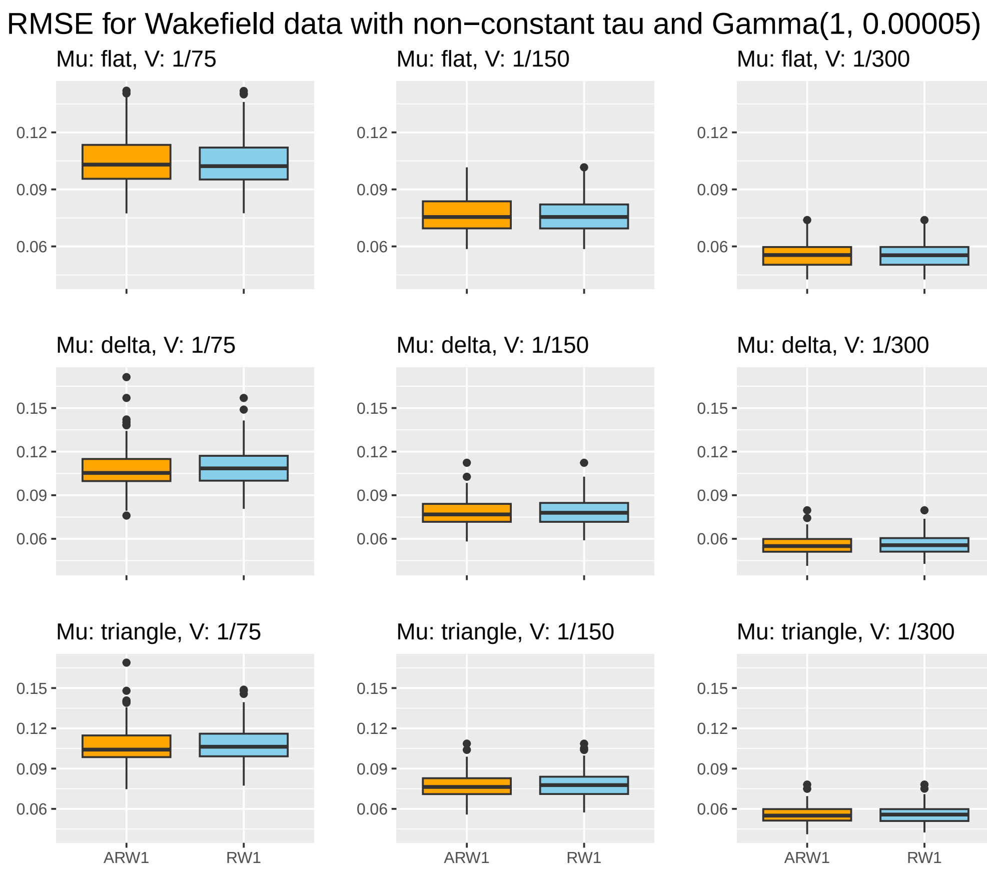

Figure 21: RMSE for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

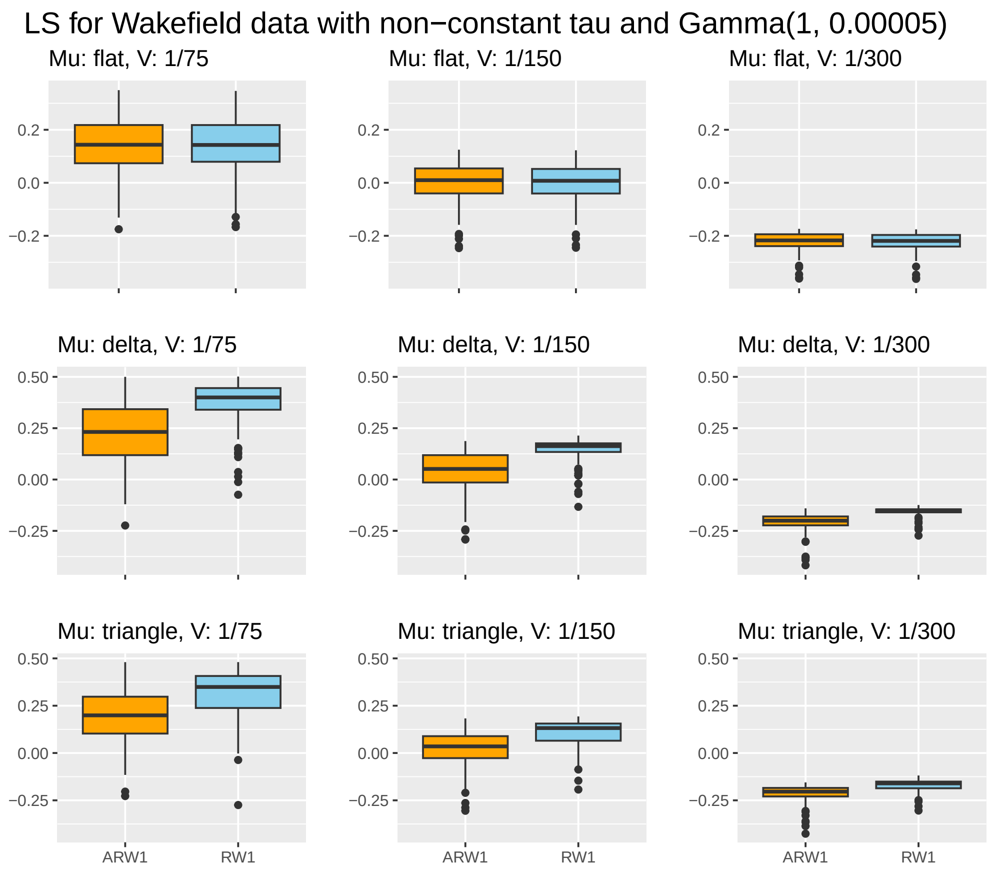

Figure 22: LS for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

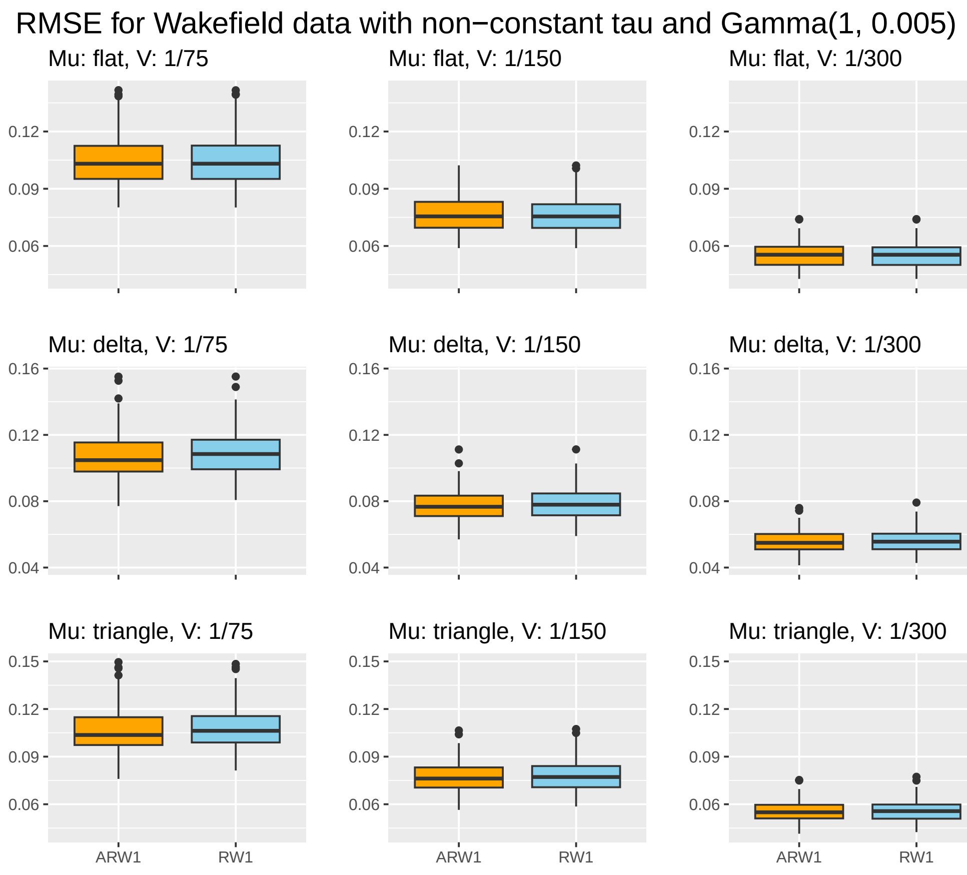

Figure 23: RMSE for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.005)$.

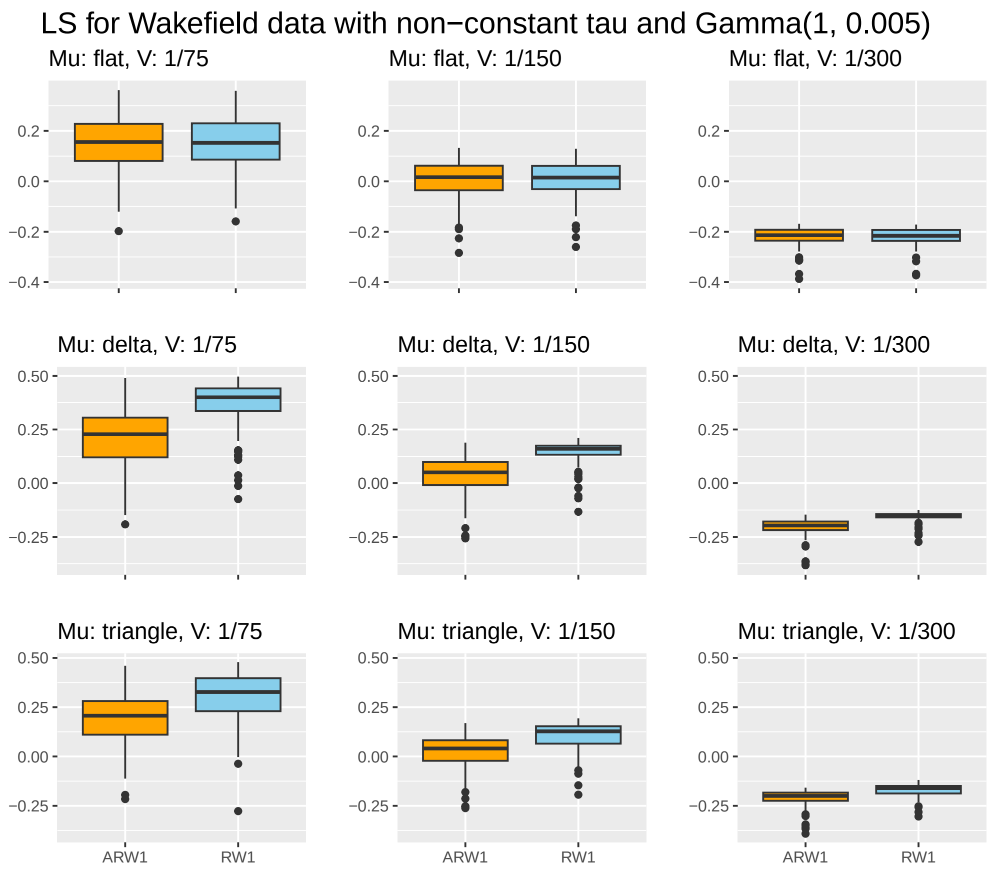

Figure 24: LS for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.005)$.

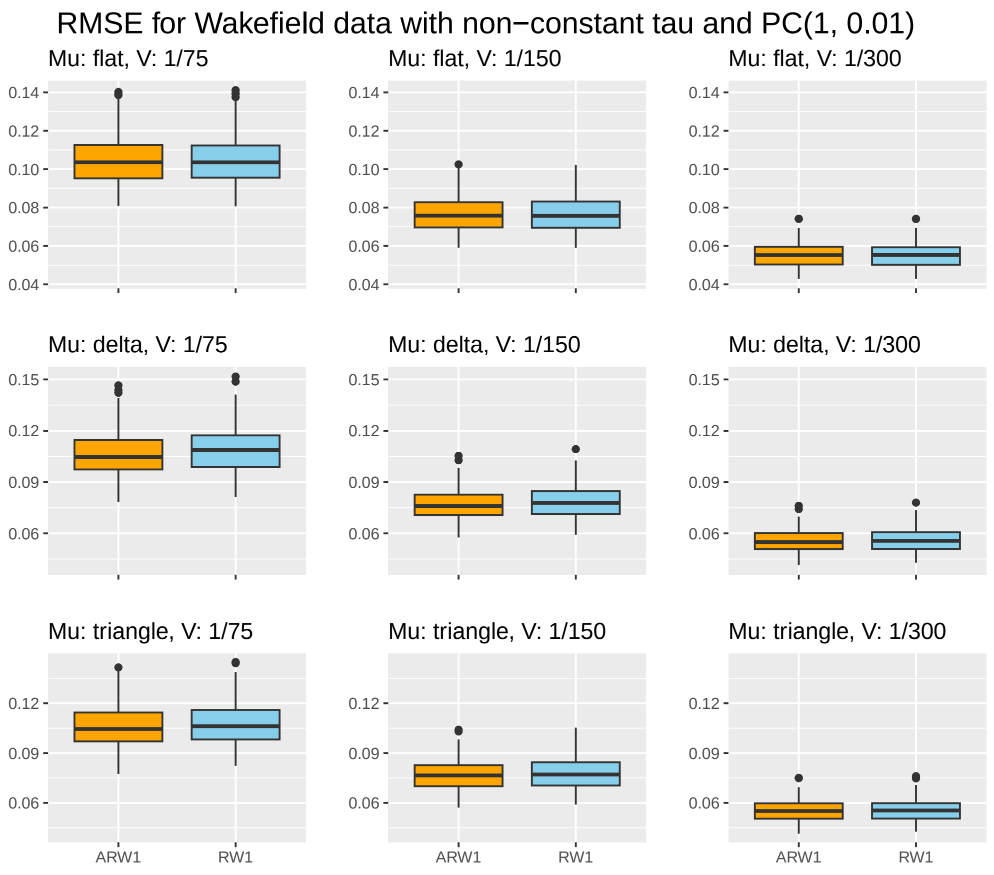

Figure 25: RMSE for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $PC(1, 0.01)$.

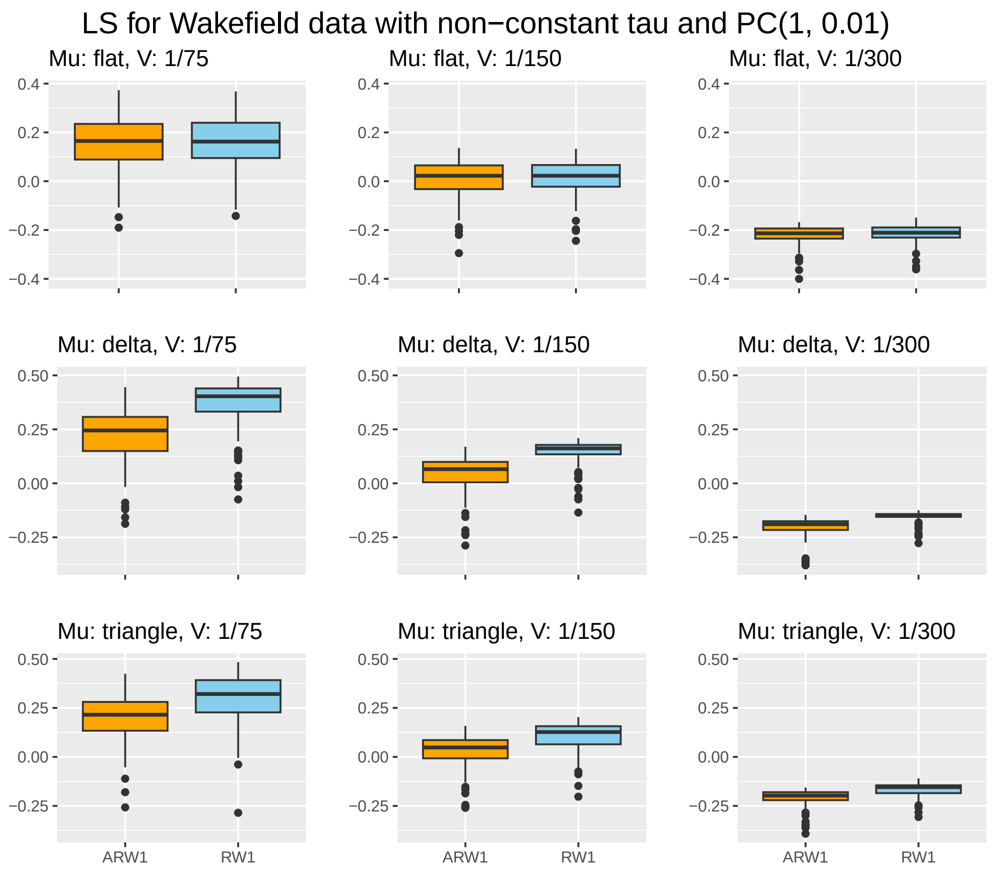

Figure 26: LS for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $PC(1, 0.01)$.

C Plots for the sine data

The result plots for the sine data with and without an offset. The precision τ is constant or not constant. Finally, we apply three different priors to the precision parameter in the random walk in each case.

C.1 Plots with constant precision

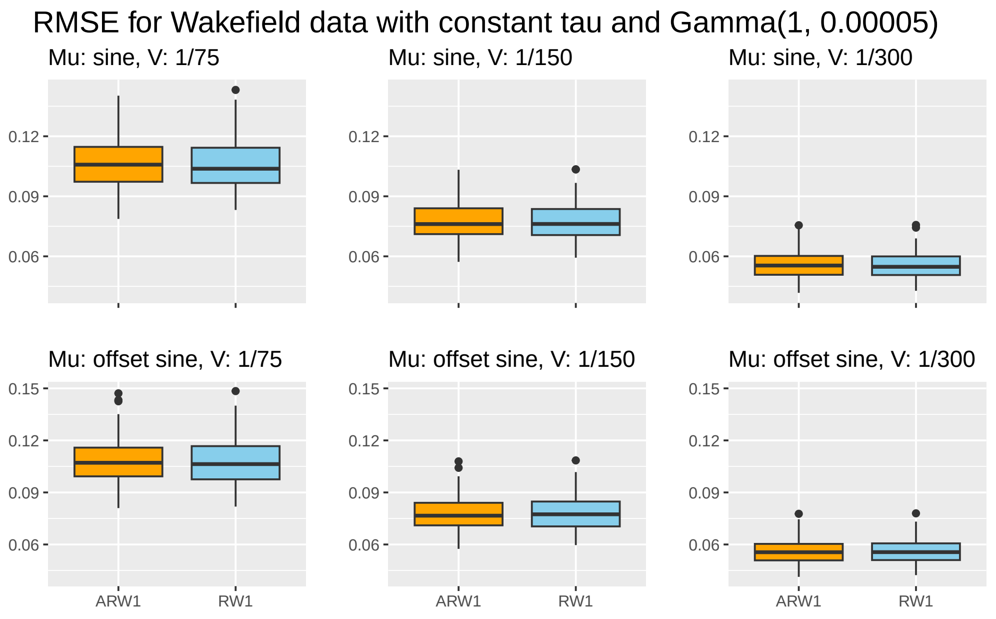

Figure 27: RMSE for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

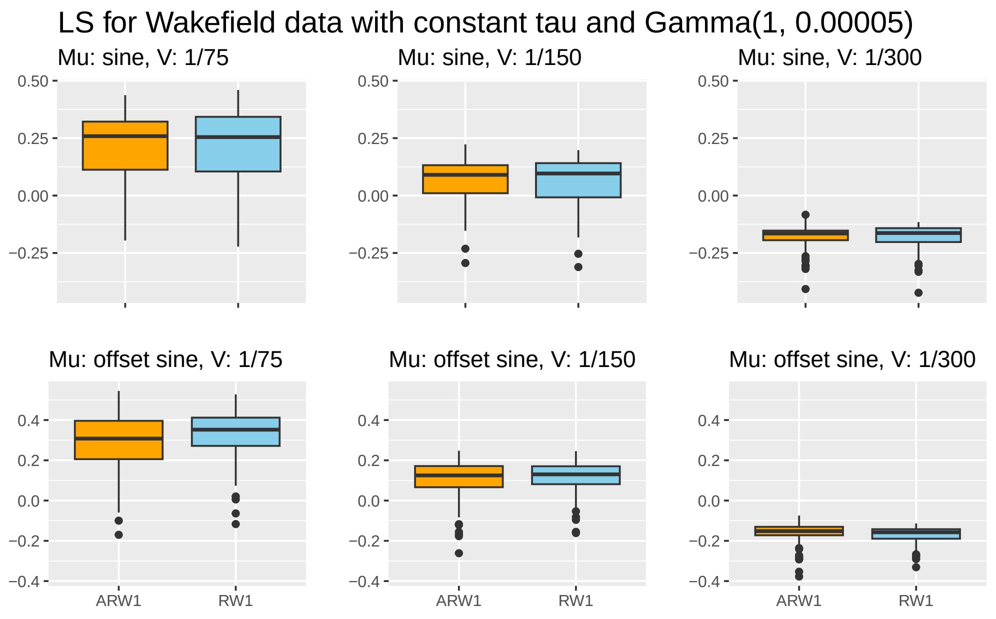

Figure 28: LS for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

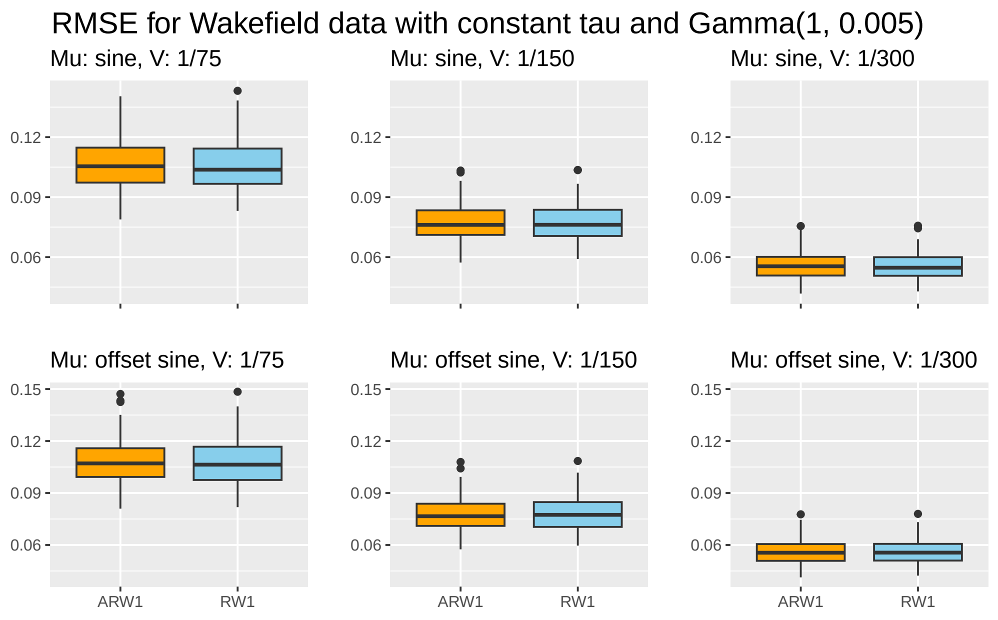

Figure 29: RMSE for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.005)$.

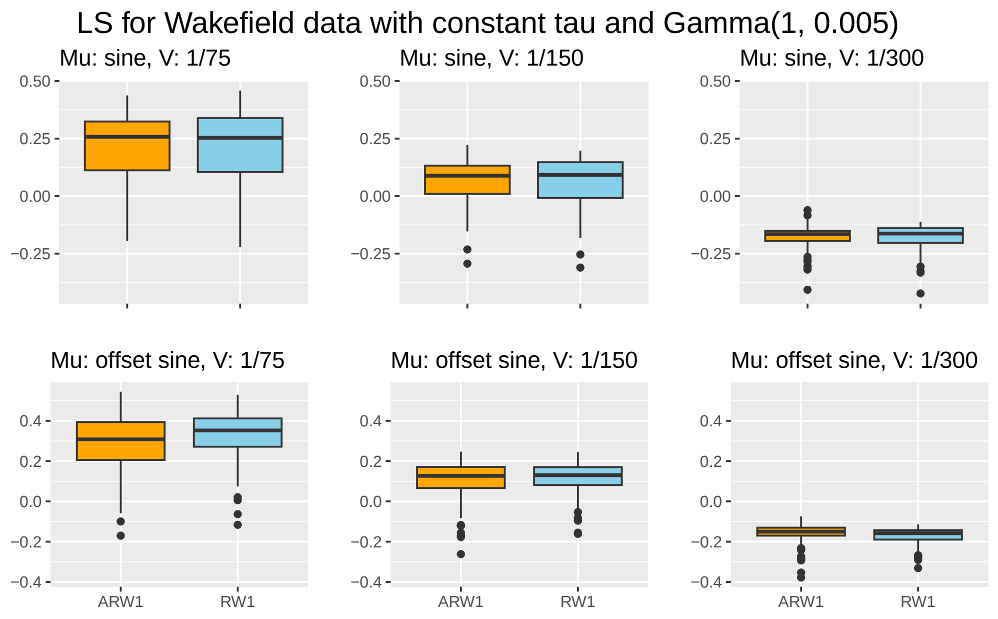

Figure 30: LS for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.005)$.

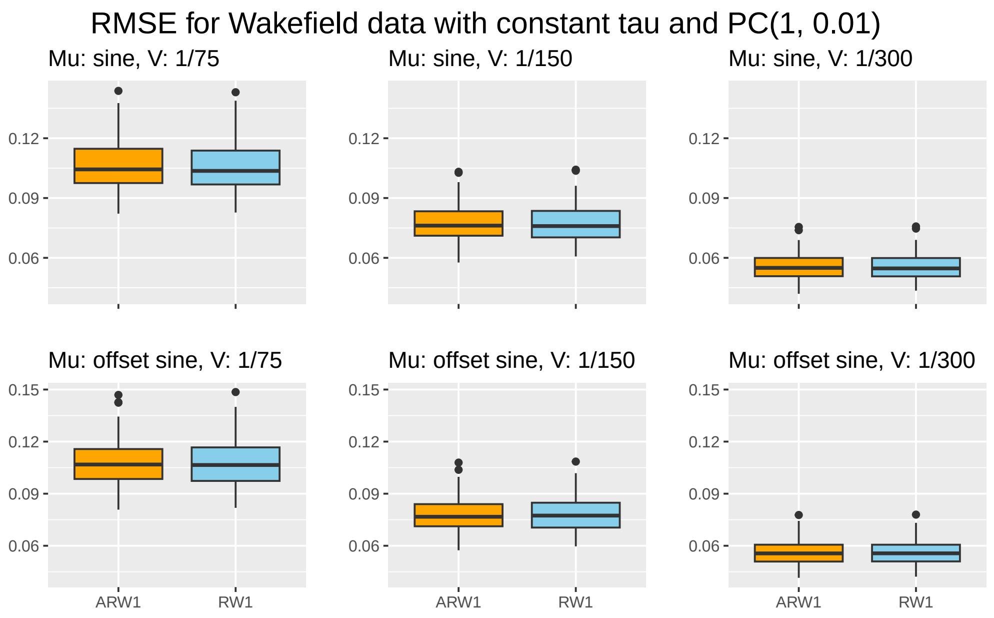

Figure 31: RMSE for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $PC(1, 0.01)$.

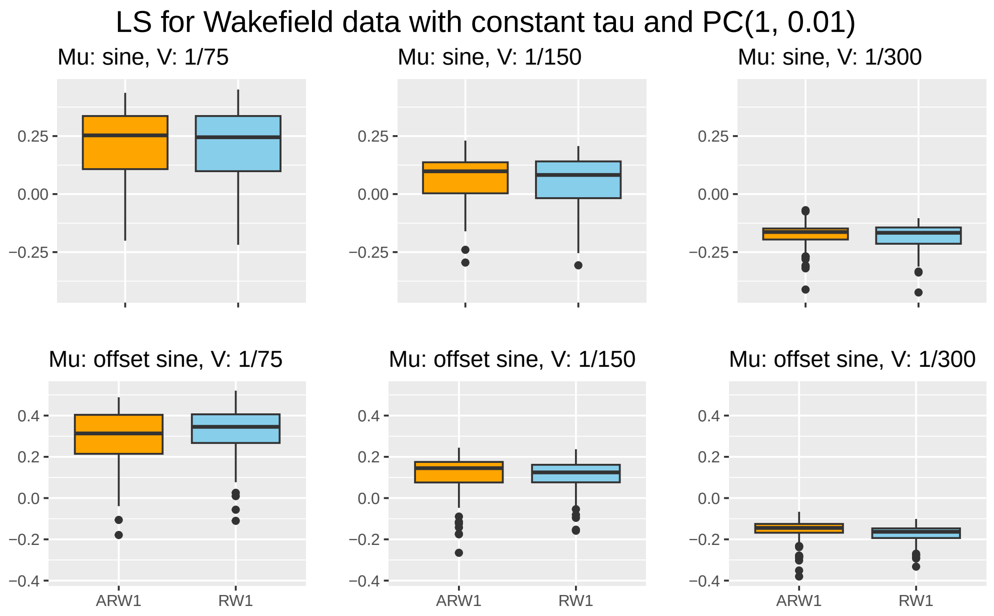

Figure 32: LS for simulated data with constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $PC(1, 0.01)$.

C.2 Plots with non-constant precision

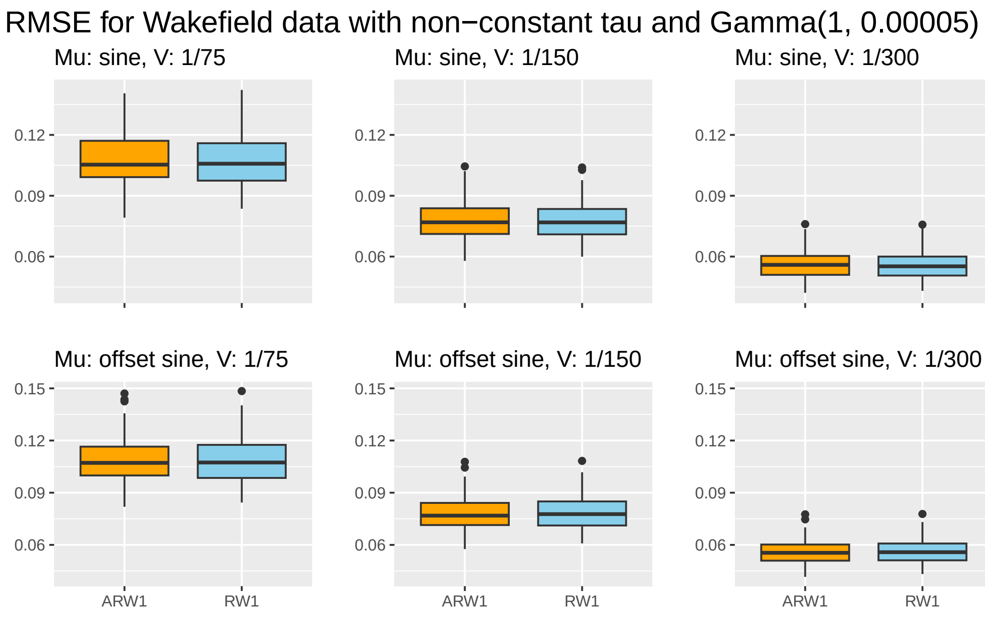

Figure 33: RMSE for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

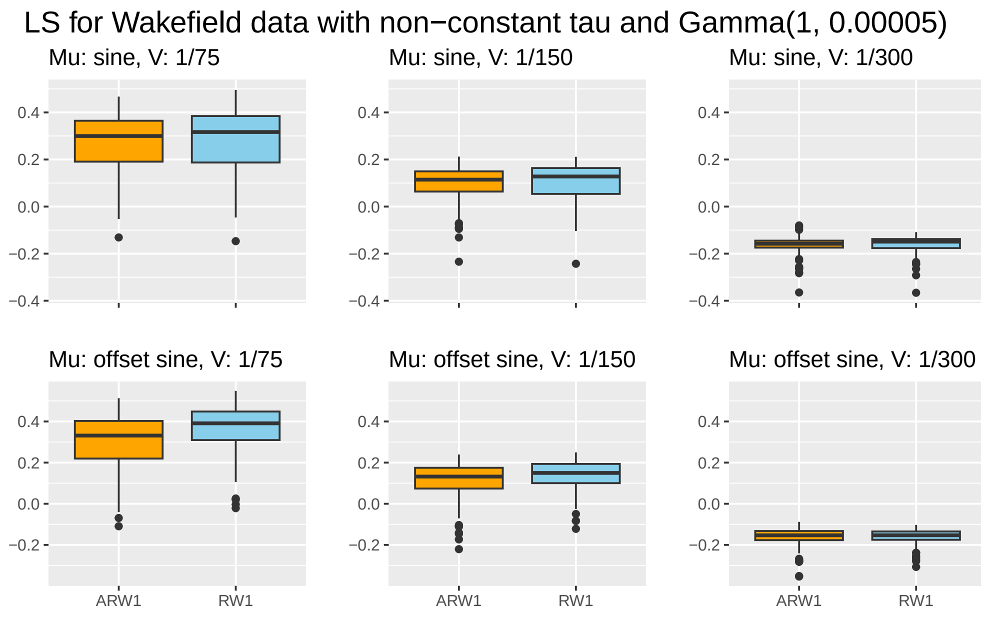

Figure 34: LS for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.00005)$.

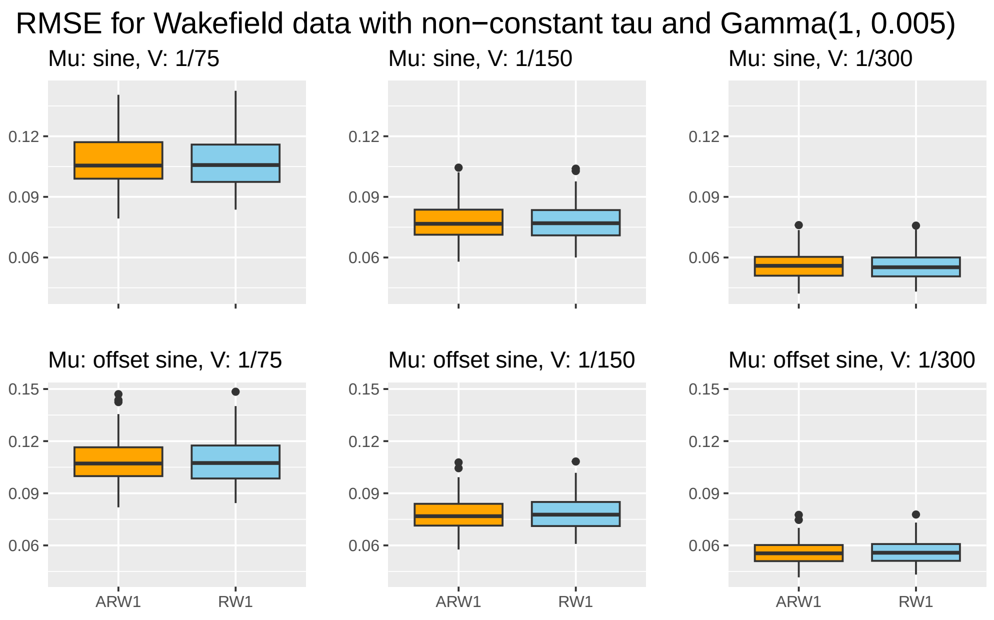

Figure 35: RMSE for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.005)$.

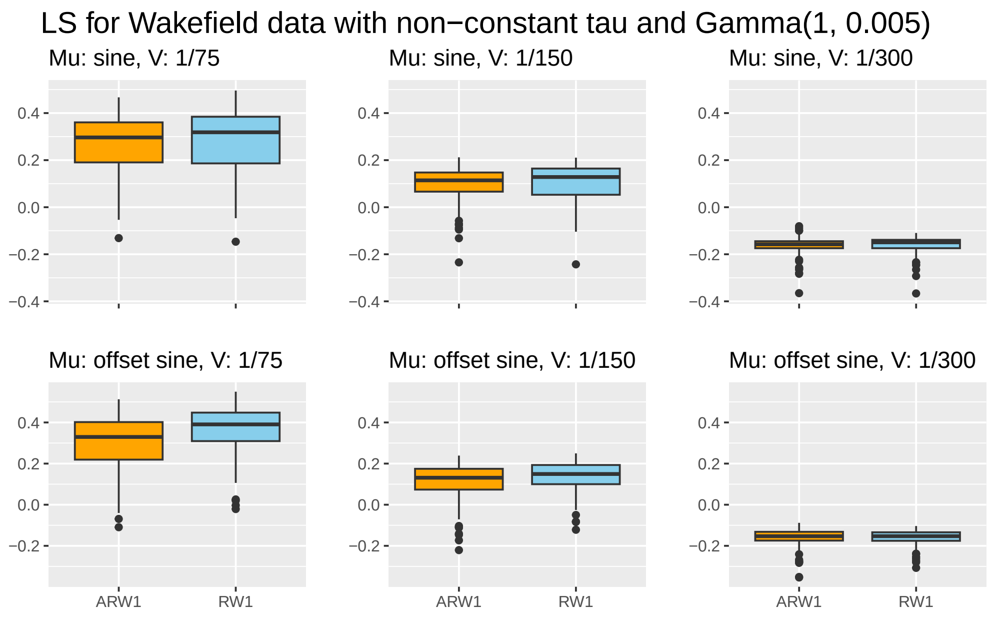

Figure 36: LS for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $Gamma(1, 0.005)$.

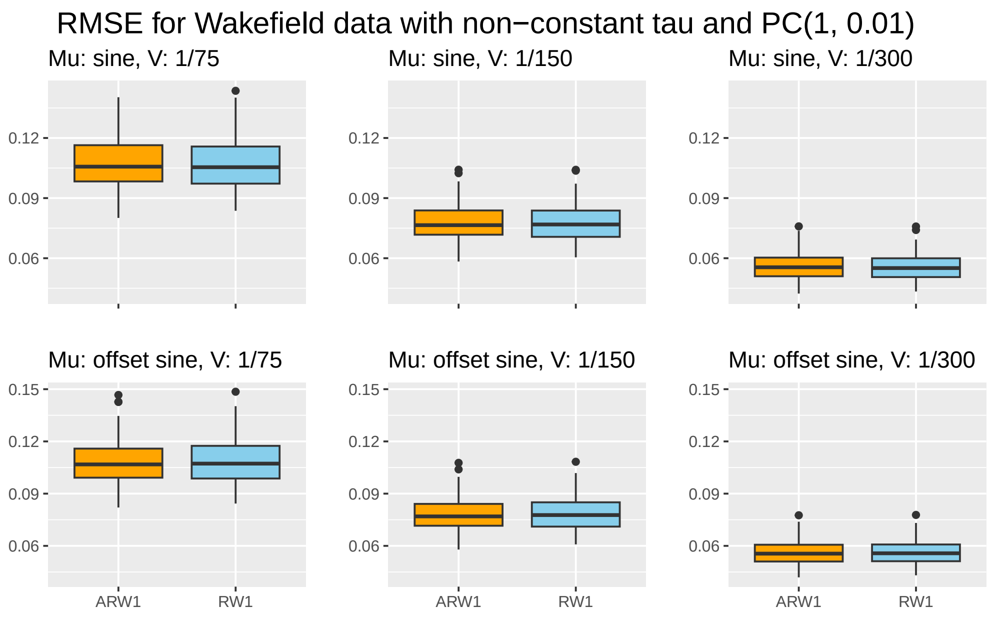

Figure 37: RMSE for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $PC(1, 0.01)$.

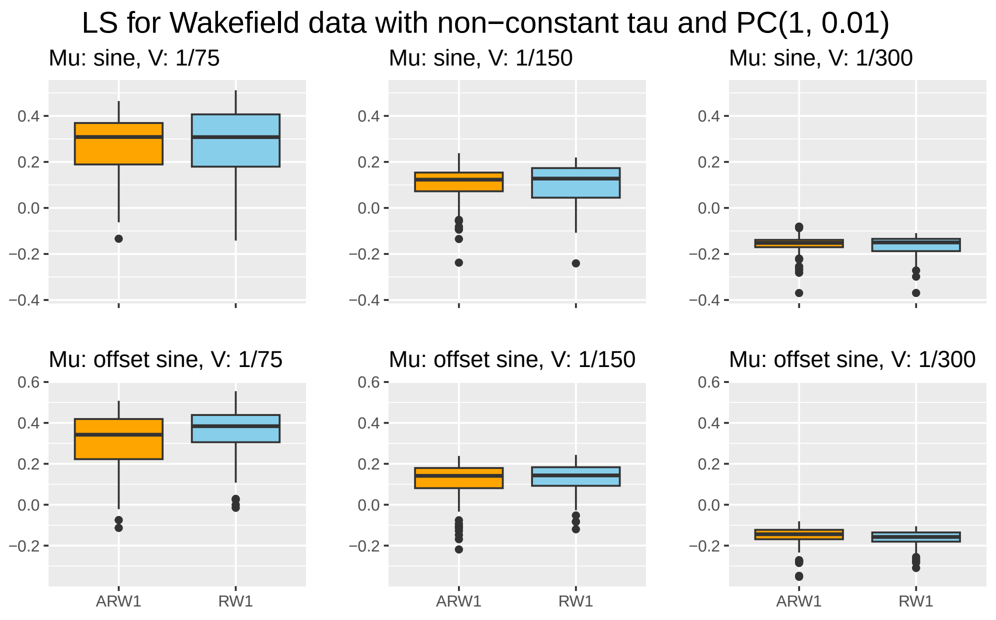

Figure 38: LS for simulated data with non-constant precision τ and mean trend and observational noise specified above each plot. The prior used for the precision in the random walks were $PC(1, 0.01)$.

D The ARW1 with three precisions

To implement an adaptive RW1 with a precision for each shocked event is a simple expansion of the basic adaptive RW1 model with two total precisions. Now, we need to pass two lists of shocked points, and I will assume there are no transitions involving both the shock events. If this is not the case, it becomes more complex. We need to make three structure matrices, one for each precision, and also have three parameters in interpret theta and for the log.prior. In total we get the `rgenric` function below.

```
inla.rgeneric.AdaptiveRW1.model_3tau = function(
  cmd = c("graph", "Q", "mu", "initial",
  ↪  "log.norm.const","log.prior", "quit"),
  theta = NULL)
{
  #Input:
  #N is the number of timepoints
  #conflict_years1 is the first event
  #conflict_years2 is the second event
  #prior_str is either Gamma0.005, Gamma0.00005 or PC

  envir = parent.env(environment())

  interpret_theta <- function() { return(list(tau1 = exp(theta[1L]),
            tau2 = exp(theta[2L]), tau3 = exp(theta[3L])))}

  graph <- function() {return(Q())}

  Q <- function() {
    R1 <- matrix(0, nrow = N, ncol = N) #non-shocked
    R2 <- matrix(0, nrow = N, ncol = N) #shocked 1
    R3 <- matrix(0, nrow = N, ncol = N) #shocked 2
    for( i in 1:(N - 1)){
      if(i %in% conflict_years1 | (i + 1) %in% conflict_years1) {
        R2[c(i, i+1), c(i, i+1)] <- R2[c(i, i+1), c(i, i+1)] + c(1,
        ↪  -1, -1, 1)
      }
      else if(i %in% conflict_years2 | (i + 1) %in% conflict_years2)
      ↪  {
        R3[c(i, i+1), c(i, i+1)] <- R3[c(i, i+1), c(i, i+1)] + c(1,
        ↪  -1, -1, 1)
      } else{
        R1[c(i, i+1), c(i, i+1)] <- R1[c(i, i+1), c(i, i+1)] + c(1,
        ↪  -1, -1, 1)
      }
    }
    gv <- exp(1 / N * sum(log(diag(INLA:::inla.ginv(R1 + R2 + R3)))))
    ↪  #scaling constant
    R_star_list <- list(R1 = R1*gv, R2 = R2*gv, R3 = R3*gv)

    p <- interpret_theta()
    Q <- R_star_list$R1 * p$tau1 + R_star_list$R2 * p$tau2 +
    ↪  R_star_list$R3 * p$tau3
    return(inla.as.sparse(Q)) #sparse representation
  }
```

```r

  mu <- function() {return(numeric(0))}

  initial <- function() {return(c(4, 4, 4))}#Default initial for
   ↪  precisions is 4

  log.norm.const <- function() {return(numeric(0))}

  log.prior <- function() {#default: shape = 1, rate = 0.00005
    p <- interpret_theta()
    if(prior_str == "PC"){
      prior <- inla.pc.dprec(p$tau1, u = 1, alpha = 0.01, log=TRUE) +
       ↪  log(p$tau1) +
          inla.pc.dprec(p$tau2, u = 1, alpha = 0.01, log = TRUE) +
           ↪  log(p$tau2) +
          inla.pc.dprec(p$tau3, u = 1, alpha = 0.01, log = TRUE) +
           ↪  log(p$tau3)
      return(prior)
    } else if(prior_str == "Gamma0,005"){
      prior <- dgamma(p$tau1, shape = 1, rate = 0.005, log = TRUE) +
       ↪  log(p$tau1) +
               dgamma(p$tau2, shape = 1, rate = 0.005, log = TRUE) +
                ↪  log(p$tau2) +
               dgamma(p$tau3, shape = 1, rate = 0.005, log = TRUE) +
                ↪  log(p$tau3)
      return(prior)
    }
    prior <- dgamma(p$tau1, shape = 1, rate = 0.00005, log = TRUE) +
     ↪  log(p$tau1) +
             dgamma(p$tau2, shape = 1, rate = 0.00005, log = TRUE) +
              ↪  log(p$tau2) +
             dgamma(p$tau3, shape = 1, rate = 0.00005, log = TRUE) +
              ↪  log(p$tau3)
    return(prior)
  }

  quit <- function() {return(invisible())}

  #to ensure theta is defined
  if (!length(theta)) theta = initial()

  vals <- do.call(match.arg(cmd), args = list())
  return(vals)
}
```

E Plots for the Norway death data

E.1 Plots with the Poisson likelihood

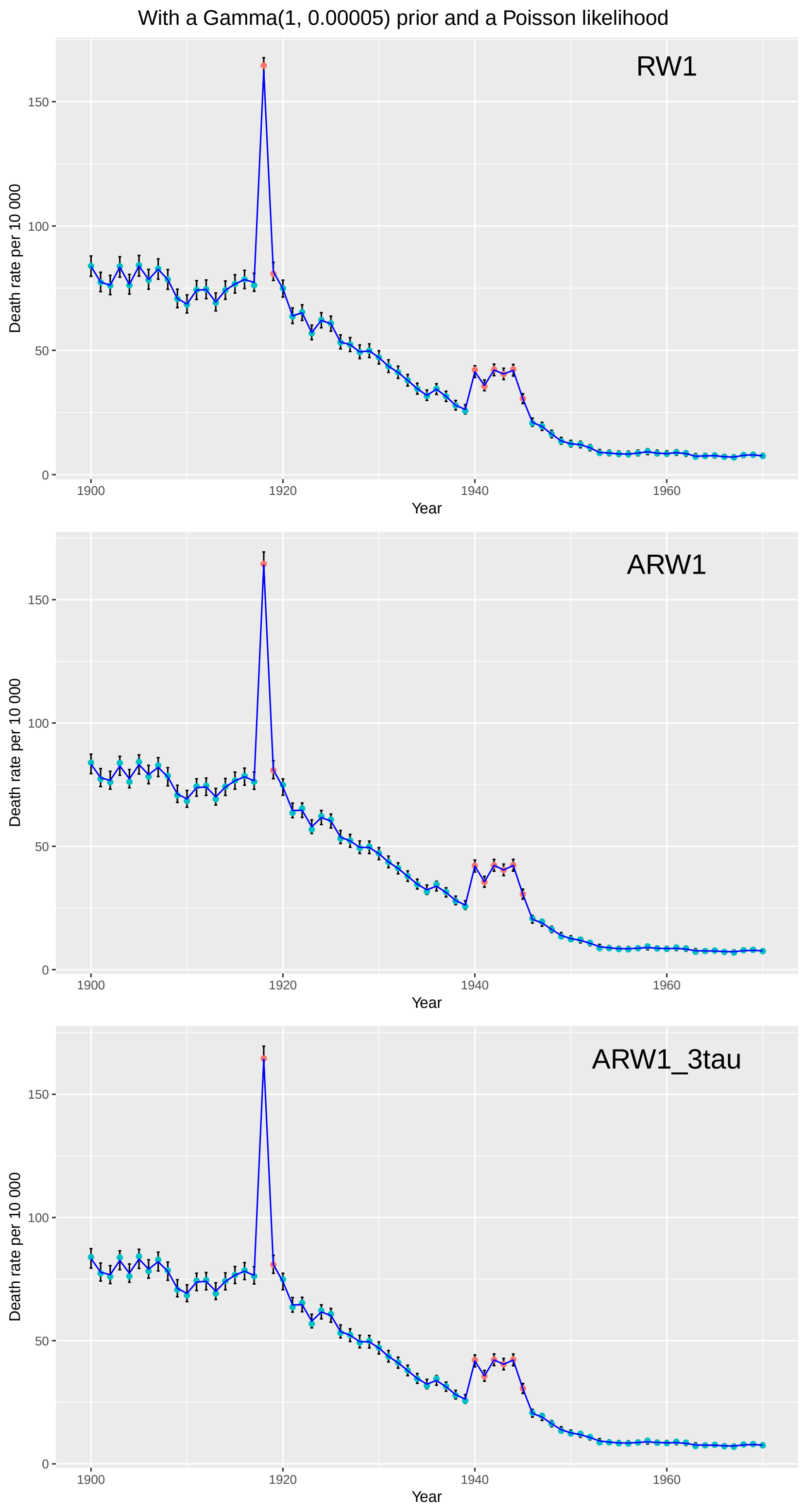

Figure 39: The model fit and the actual data for the death rate per 10 000 per year. The shocked points are in red. The models were fit using a Poisson likelihood and a *Gamma*(1, 0.00005) prior. The 95% credibility interval is shown for each point.

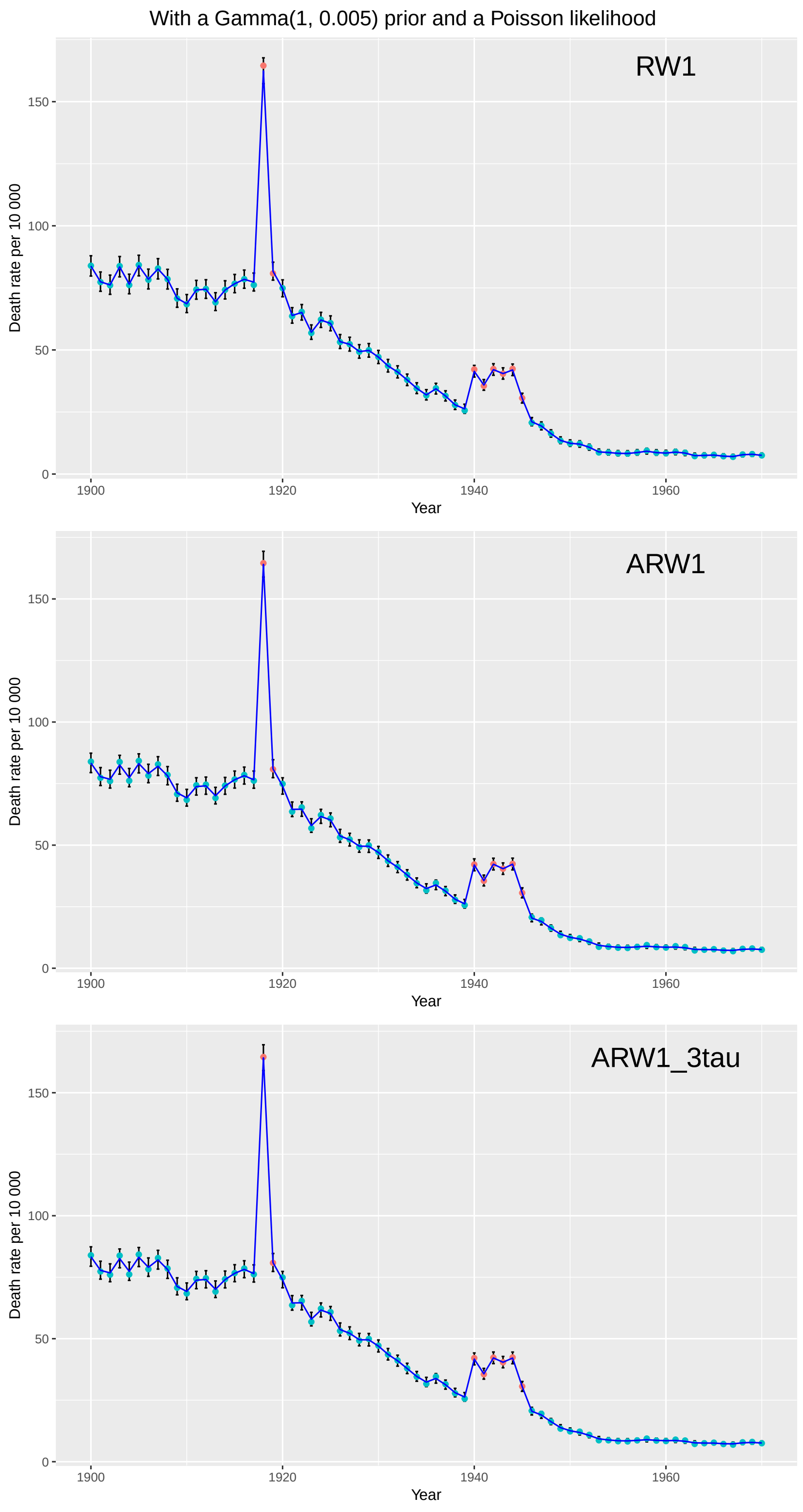

Figure 40: The model fit and the actual data for the death rate per 10 000 per year. The shocked points are in red. The models were fit using a Poisson likelihood and a $Gamma(1, 0.005)$ prior. The 95% credibility interval is shown for each point.

	Method	RMSE	LS
1	RW1	3.649927e−05	5.727412
2	ARW1	4.984560e−05	5.319114
3	ARW1_3tau	5.011074e−05	5.321661

Figure 41: RMSE and LS for the three models with a Poisson likelihood and a $Gamma(1, 0.00005)$ prior.

	Method	RMSE	LS
1	RW1	3.648250e−05	5.727432
2	ARW1	4.976891e−05	5.319228
3	ARW1_3tau	5.002654e−05	5.321779

Figure 42: RMSE and LS for the three models with a Poisson likelihood and a $Gamma(1, 0.005)$ prior.

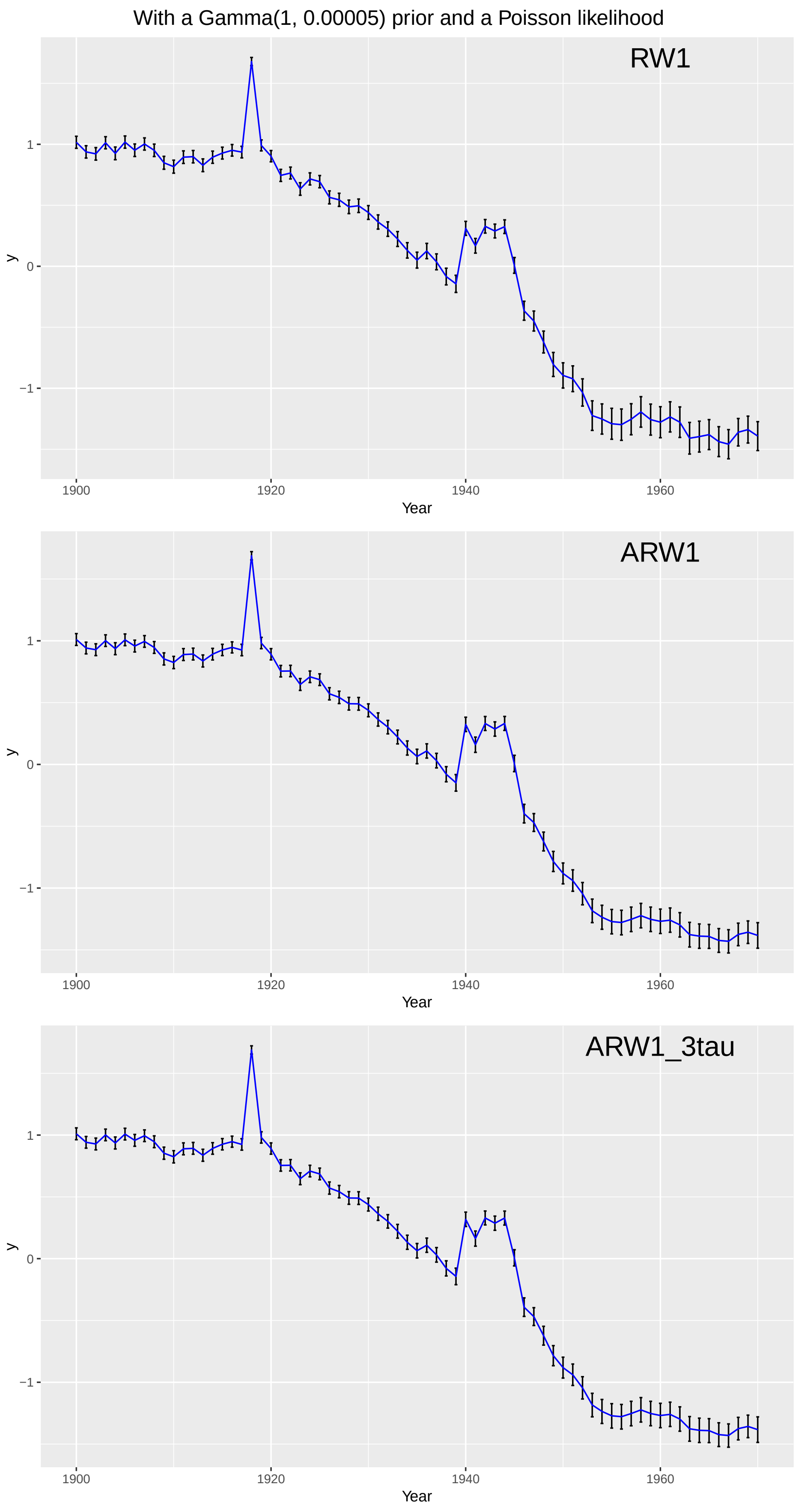

Figure 43: The fitted values of the random walk components in the latent layer for each model. The models were fit using a Poisson likelihood and a $Gamma(1, 0.00005)$ prior. The 95% credibility interval is shown for each point.

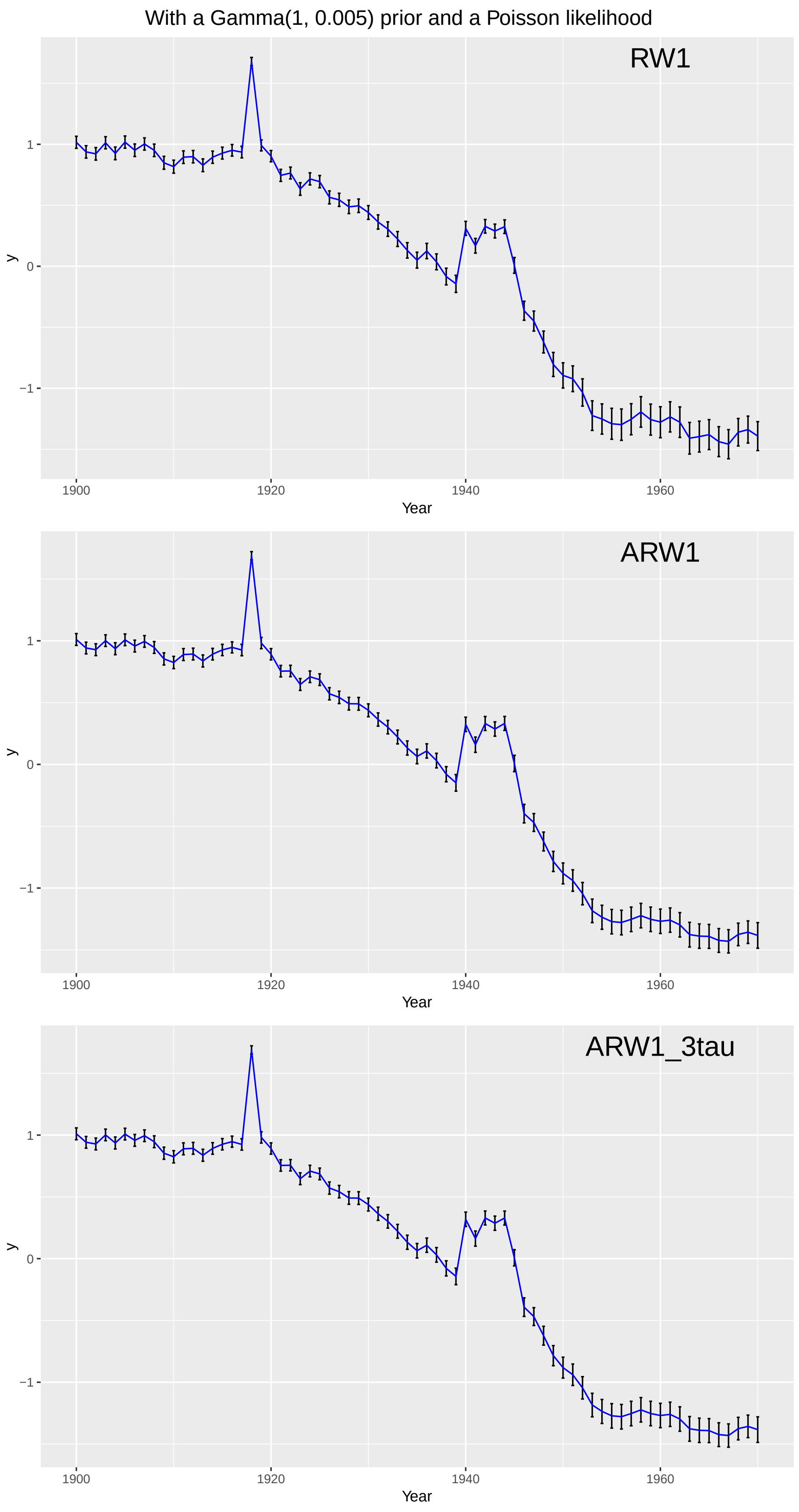

Figure 44: The fitted values of the random walk components in the latent layer for each model. The models were fit using a Poisson likelihood and a $Gamma(1, 0.005)$ prior. The 95% credibility interval is shown for each point.

E.2 Plots with the negative binomial likelihood

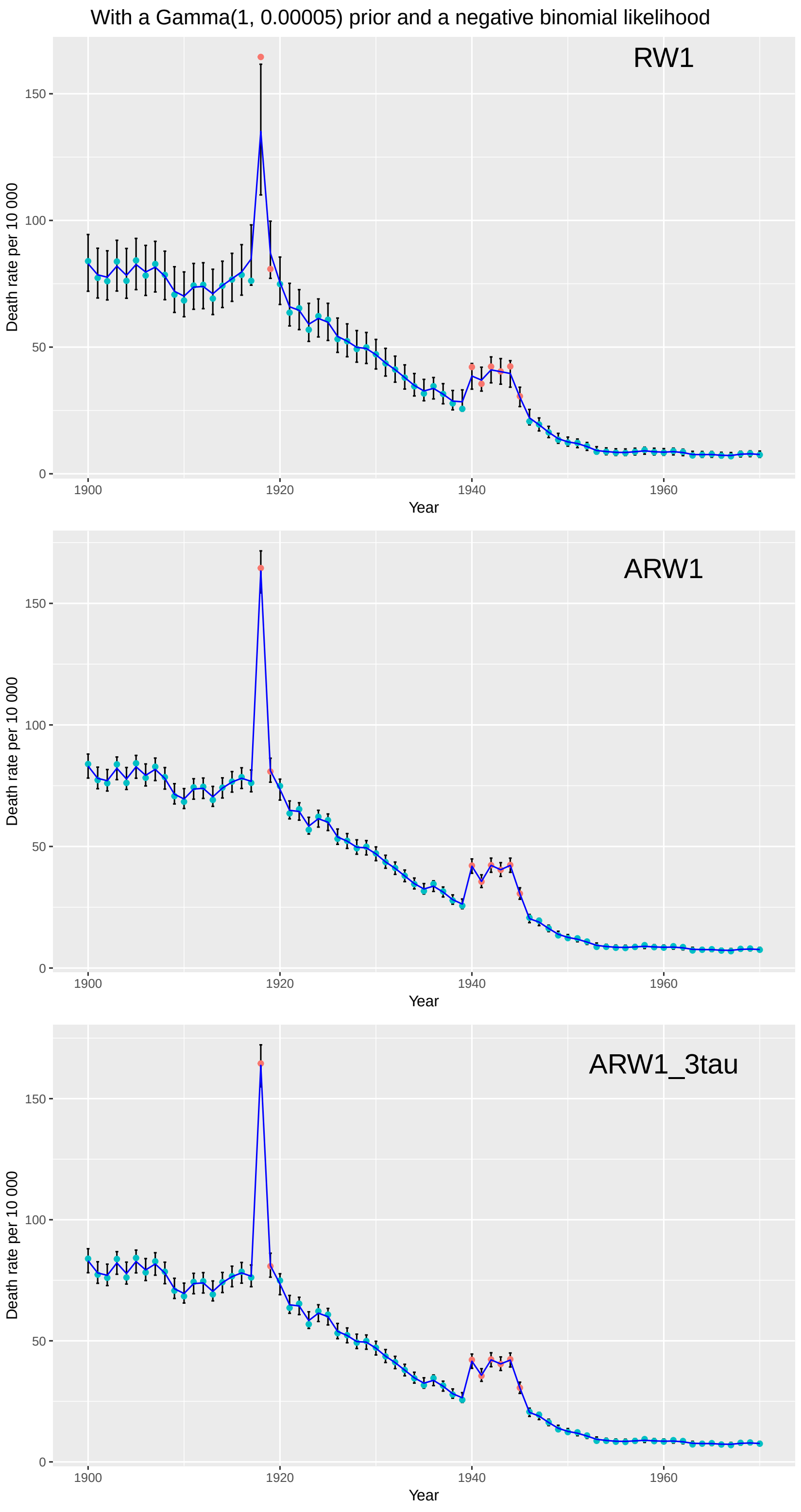

Figure 45: The model fit and the actual data for the death rate per 10 000 per year. The shocked points are in red. The models were fit using a negative binomial likelihood and a $Gamma(1, 0.00005)$ prior. The 95% credibility interval is shown for each point.

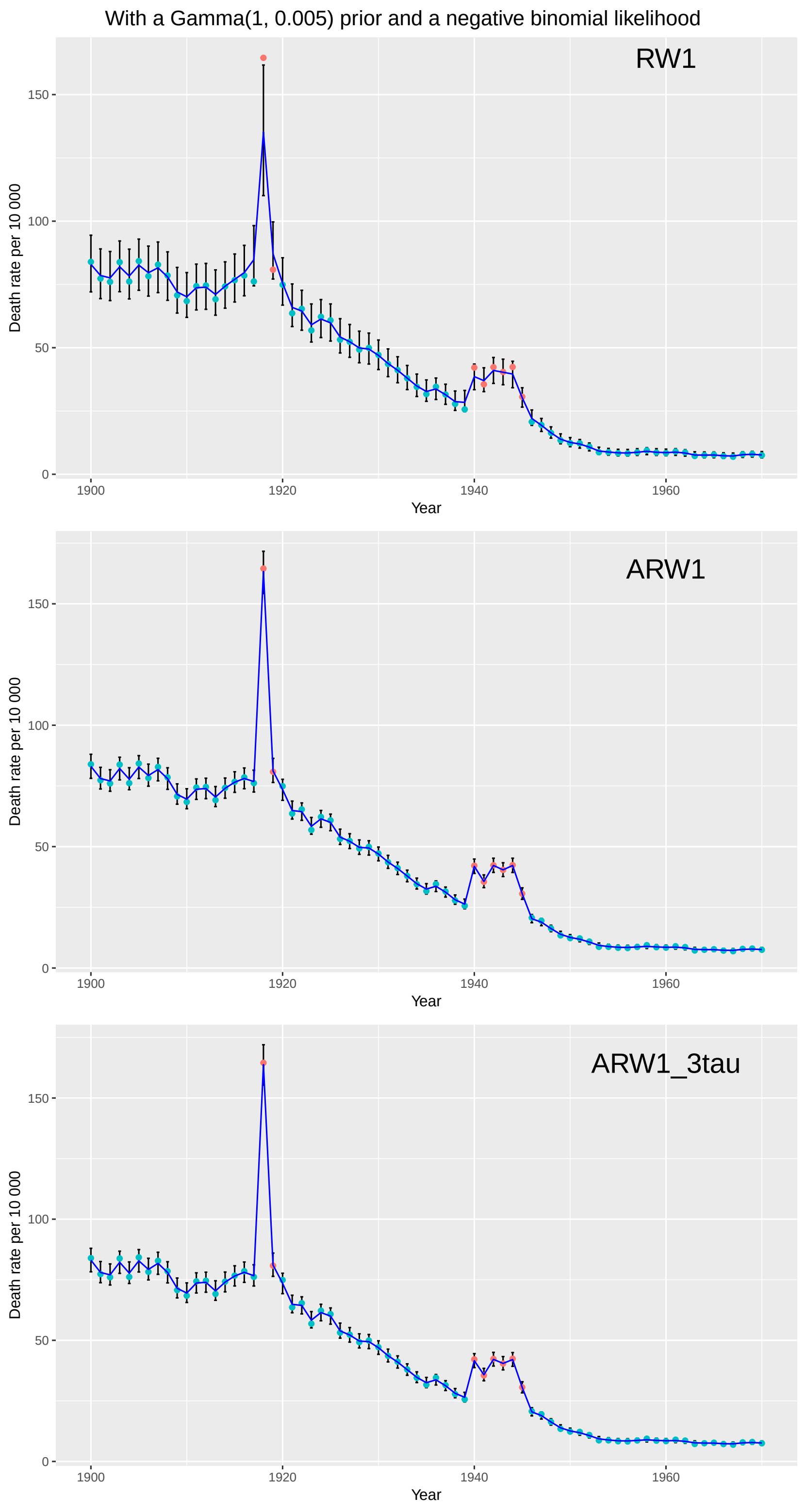

Figure 46: The model fit and the actual data for the death rate per 10 000 per year. The shocked points are in red. The models were fit using a negative binomial likelihood and a $Gamma(1, 0.005)$ prior. The 95% credibility interval is shown for each point.

	Method	RMSE	LS
1	RW1	3.860039e−04	5.997008
2	ARW1	6.762554e−05	5.359335
3	ARW1_3tau	6.738881e−05	5.358977

Figure 47: RMSE and LS for the three models with a negative binomial likelihood and a $Gamma(1, 0.00005)$ prior.

	Method	RMSE	LS
1	RW1	3.855639e−04	5.996992
2	ARW1	6.782872e−05	5.360094
3	ARW1_3tau	6.568605e−05	5.358168

Figure 48: RMSE and LS for the three models with a negative binomial likelihood and a $Gamma(1, 0.005)$ prior.

Now the plots for the fitted values of the random walks used.

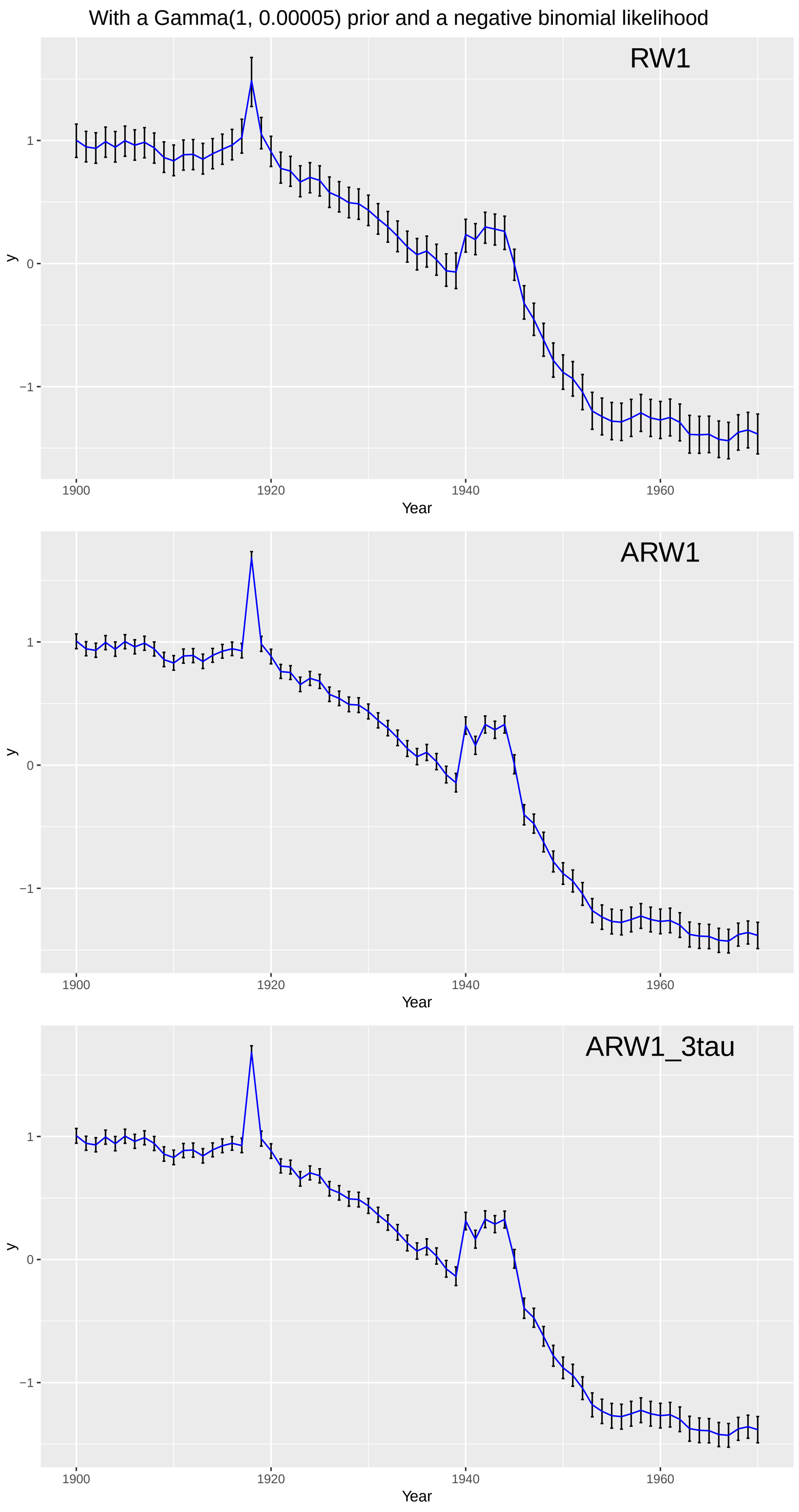

Figure 49: The fitted values of the random walk components in the latent layer for each model. The models were fit using a negative binomial likelihood and a $Gamma(1, 0.00005)$ prior. The 95% credibility interval is shown for each point.

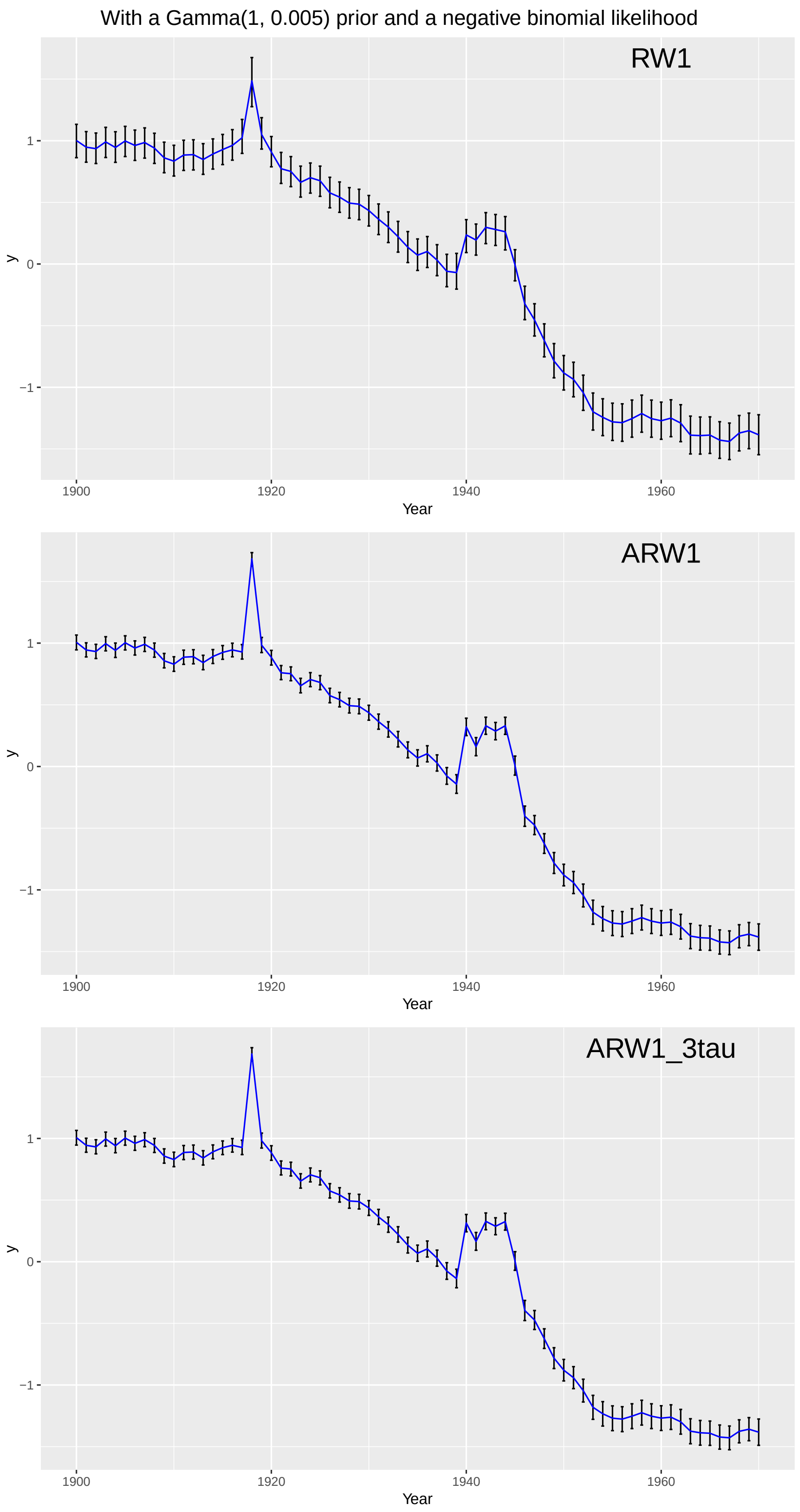

Figure 50: The fitted values of the random walk components in the latent layer for each model. The models were fit using a negative binomial likelihood and a $Gamma(1, 0.005)$ prior. The 95% credibility interval is shown for each point.