Project 2

Computer Intensive Statistical Methods

Erling Fause Steen og Christian Oppegård Moen

08 03 2022

Contents

Problem 2

In this problem we will use INLA to fit the same model as in the previous problem.

```
library("INLA")
library("ggplot2")
library("ggrepel")
```

```
# load('C:/Users/erlin/Documents/Beregningskrevende statistiske
# modeller/Øvinger 2022/rain.rda')
load("./rain.rda")
```

a) INLA model 1

To be able to compare the model with the previous Markov chain, we need to use the same priors as in Problem 1. In this model the intercept term is removed so we don't include a prior on this term. We do however need to include a prior for σ_u^2. We have

$$\tau_t \sim \tau_{t-1} + u_t$$

for $u_t \sim N(0, \sigma_u^2)$. In INLA priors are placed on the log precision rather than the variance. The precision is $1/\sigma_u^2$, and we place the prior on the hyperparameter

$$\theta = \log\left(\frac{1}{\sigma_u^2}\right).$$

We know from problem 1 that σ_u^2 has a distribution. This means that $\sigma_u^2 \sim \text{Gamma}(\alpha, \beta)$ and $\theta \sim \text{loggamma}(\alpha, \beta)$.

We therefore place this prior on θ and use $\alpha = 2$ and $\beta = 0.05$.

```r
# Prior placed on hyperparameter
alpha = 2
beta = 0.05
hyper = list(prec = list(prior = "loggamma", param = c(alpha, beta)))
```

In the chunck below a function that fits a INLA model with a given for given control.inla inputs is made. The function also returns the computation time using proc.time()[3].

```r
INLA_fit <- function(con.inla) {
    time_before = proc.time()[3]
    mod <- inla(n.rain ~ -1 + f(day, model = "rw1", constr = FALSE, hyper = hyper),
        data = rain, Ntrials = n.years, control.compute = list(config = TRUE), family = "binomial",
        verbose = TRUE, control.inla = con.inla)
    # computation time
    time = proc.time()[3] - time_before
    return(list(mod = mod, time = time))
}
```

We also make a function that plots the development of the means of the model with a 95 % credible interval.

```r
INLA_plot <- function(mod) {
    lower = mod$summary.fitted.values$`0.025quant`
    # upper bound
    upper = mod$summary.fitted.values$`0.975quant`
    # Plot means with 95 % credible interval
    ggplot(data = as.data.frame(mod$summary.fitted.values$mean), mapping = aes(x = 1:366,
        y = mod$summary.fitted.values$mean)) + geom_line(col = "red") + geom_ribbon(aes(ymin = lower,
        ymax = upper), alpha = 0.1) + xlab("Day") + ylab("predicted values")
}
```

We fit a model using simplified Laplace as the strategy for approximations and ccd as the strategy for integration.

```r
# control.inla input
control.inla = list(strategy = "simplified.laplace", int.strategy = "ccd")
fit1 <- INLA_fit(control.inla)
mod1 <- fit1$mod
time1 <- fit1$time
```

We look at predictions and uncertainties of INLA and plot the development of the means with a 95 % credible interval for the fitted values.

```r
INLA_plot(mod1)
```

We see in figure 1 that we get a similar graph as in problem 1. We look specifically at $\pi(\tau_1), \pi(\tau_{201})$ and $\pi(\tau_{366})$.

```r
# pi(tau_1)
mod1$summary.fitted.values[1, ]
```

```
##                           mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.001 0.1857516 0.02889716  0.1336351 0.1841907  0.2467199
##                           mode
## fitted.Predictor.001 0.1810586
```

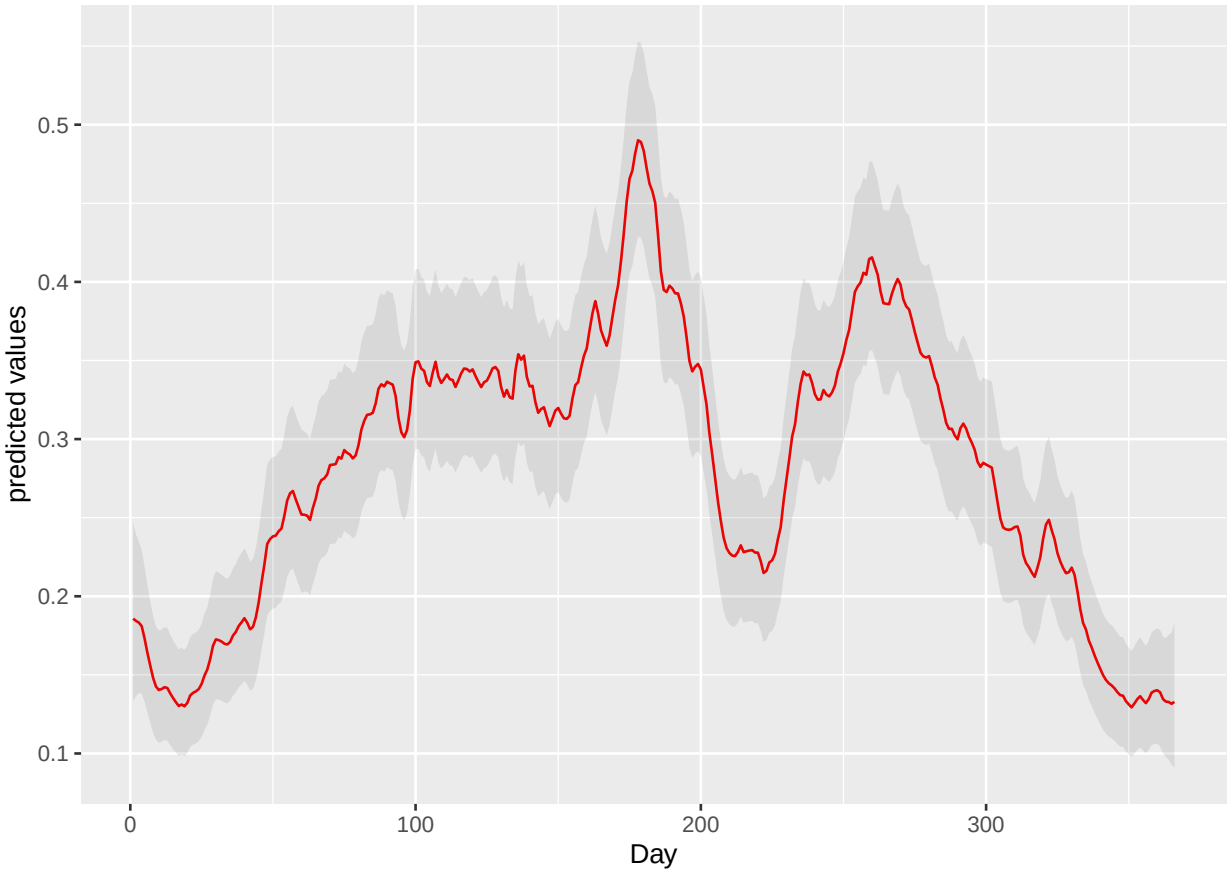

Figure 1: Plot of mean of fitted values of model 1

```
mod1$summary.fitted.values[201, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.201 0.3329557 0.02871629  0.2785433 0.3322929  0.3911323
##                            mode
## fitted.Predictor.201 0.3309675
```

```
mod1$summary.fitted.values[366, ]
```

```
##                            mean        sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.366 0.1328162 0.0234674 0.09103382 0.1313649  0.1828464
##                            mode
## fitted.Predictor.366 0.1284721
```

The computational time according to the proc.time()[3] is 1.22.

b) Robustness of the results to different control.inla inputs

We want to look at how robust the results of the two control.inputs are. The strategy we used for integration is ccd. The ccd integration is a less costly alternative compared to the grid strategy when the dimensions of the hyperparameter is large. The grid strategy gives the most accurate result, but the number of points grow exponentially with the dimension of θ. The ccd approach locates fewer points around the mode and is therefore less computationally expensive. Other options for integration strategies are 'eb', 'user' and 'user.std'. The 'auto' option which is the default integration strategy corresponds with the grid approach if the dimension of θ 2 or lower, and ccd if the dimension of θ is over 2. We have that the dimension of θ is 1, so the auto option will correspond with 'grid'.

The default option for strategy is the simplified Laplace option which is the option we have used here. Another option is the Gaussian approximation which is easy to apply and cheap to compute. However, there could be errors in location or due to skeweness. Simplified Laplace can correct these errors and is computationally faster than the Laplace approximation. This method is therefore regarded as a compromise between accuracy and computational cost. Other options are 'adaptive' and 'eb'.

We fit a couple of different models with different strategies and compare the results. We start by fitting some models with different integration strategies and the 'simplified.laplace' strategy for approximations. The first integration option we use is 'grid'.

```
# control.inla input
control.inla = list(strategy = "simplified.laplace", int.strategy = "grid")

fit2 <- INLA_fit(control.inla)
mod2 <- fit2$mod
time2 <- fit2$time
```

We plot the result with a 95 % credible interval.

```
INLA_plot(mod2)
```

As seen in figure 2, this looks very similar to the first model. We also look at $\pi(\tau_1)$, $\pi(\tau_{201})$ and $\pi(\tau_{366})$ for this model.

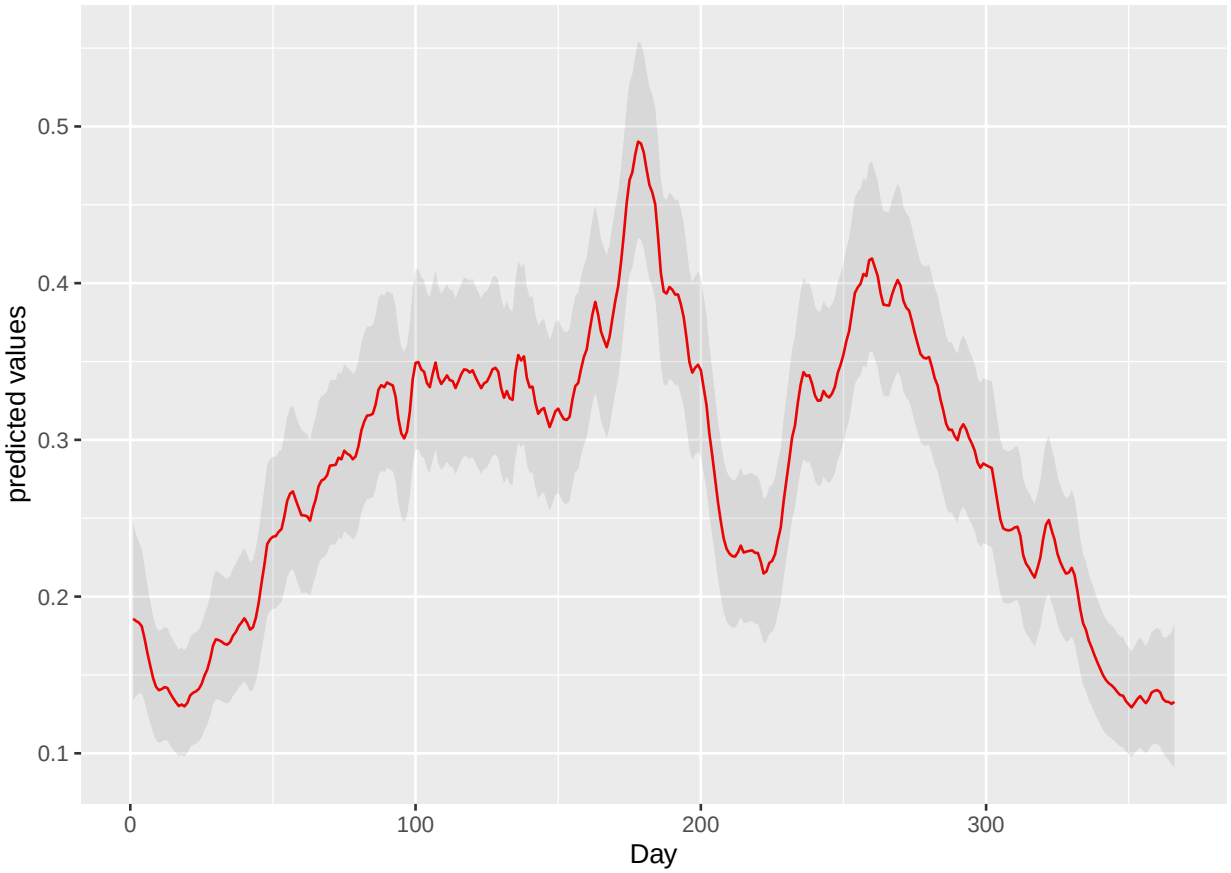

Figure 2: Plot of mean of fitted values of model with strategy=simplified and int.strategy=grid

```
mod2$summary.fitted.values[1, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.001 0.1858509 0.02906011  0.1336058 0.1842197  0.2473707
##                            mode
## fitted.Predictor.001 0.1809593
```

```
mod2$summary.fitted.values[201, ]
```

```
##                            mean        sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.201 0.3331174 0.0288756  0.2785767 0.3323867  0.3918271
##                            mode
## fitted.Predictor.201 0.3309331
```

```
mod2$summary.fitted.values[366, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.366 0.1328082 0.02353102 0.09088397 0.1313605  0.1829818
##                            mode
## fitted.Predictor.366 0.1284918
```

There are no significant differences to the first model. The computational time we get by using proc.time()[3] is 1.51, which is barely higher than the first model.

The next integration strategy considered is the 'eb' strategy.

```
control.inla = list(strategy = "simplified.laplace", int.strategy = "eb")
# Time before
fit3 <- INLA_fit(control.inla)
mod3 <- fit3$mod
time3 <- fit3$time
```

```
INLA_plot(mod3)
```

```
mod3$summary.fitted.values[1, ]
```

```
##                            mean         sd 0.025quant 0.5quant 0.975quant
## fitted.Predictor.001 0.1857206 0.02865648  0.1337232 0.184299  0.2457938
##                            mode
## fitted.Predictor.001 0.1814124
```

```
mod3$summary.fitted.values[201, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.201 0.332782 0.02848308  0.2785045 0.3322546  0.3900629
##                            mode
## fitted.Predictor.201 0.3311852
```

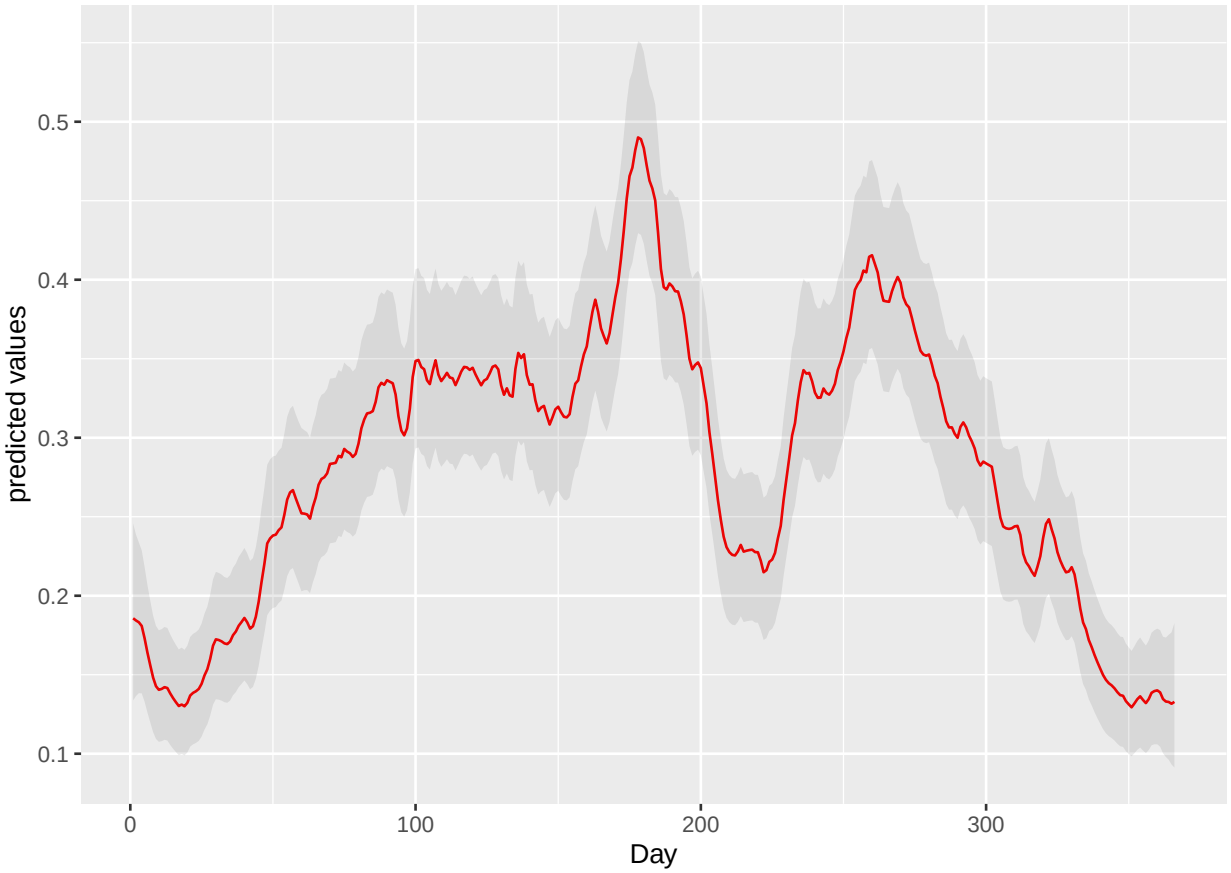

Figure 3: Plot of mean of fitted values with strategy=simplified laplace and int.strategy=eb

```
mod3$summary.fitted.values[366, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.366 0.1328144 0.02340002 0.09123294 0.1313497   0.182693
##                            mode
## fitted.Predictor.366 0.1283949
```

```
time3
```

```
## elapsed
##    1.18
```

The computation time is here 1.18, which is the same as the time when using 'grid'.

The results for the different integration strategies are very similar. There first difference can be seen in the third and fourth decimal for the different strategies.

We now look at different strategies for approximation and use the same method for integration strategy. We start by trying the gaussian approach

```
control.inla = list(strategy = "gaussian", int.strategy = "ccd")
fit4 <- INLA_fit(control.inla)
mod4 <- fit4$mod
time4 <- fit4$time
```

```
INLA_plot(mod4)
```

```
mod4$summary.fitted.values[1, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.001 0.1863651 0.02907272  0.1347588 0.1845145  0.2484831
##                            mode
## fitted.Predictor.001 0.1808184
```

```
mod4$summary.fitted.values[201, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.201 0.3333651 0.02873673  0.2791579 0.3326171  0.3918185
##                            mode
## fitted.Predictor.201 0.3311221
```

```
mod4$summary.fitted.values[366, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.366 0.1333908 0.02367606 0.09204402 0.1316544  0.1846206
##                            mode
## fitted.Predictor.366 0.1282184
```

The computation time is 1.17. We also try the Laplace approach.

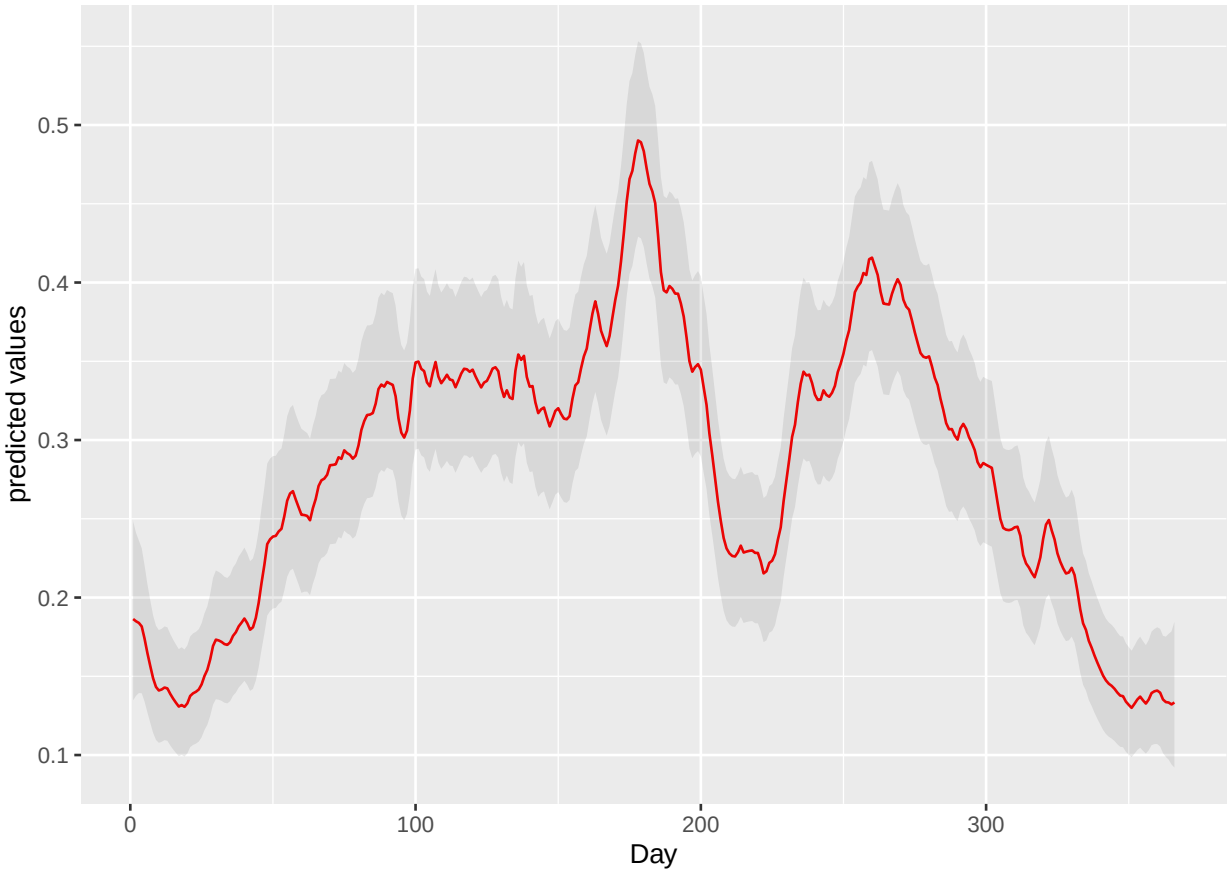

Figure 4: Plot of fitted values with strategy=gaussian and int.strategy=ccd

```
control.inla = list(strategy = "laplace", int.strategy = "ccd")
fit5 <- INLA_fit(control.inla)
mod5 <- fit5$mod
time5 <- fit5$time
```

```
INLA_plot(mod5)
```

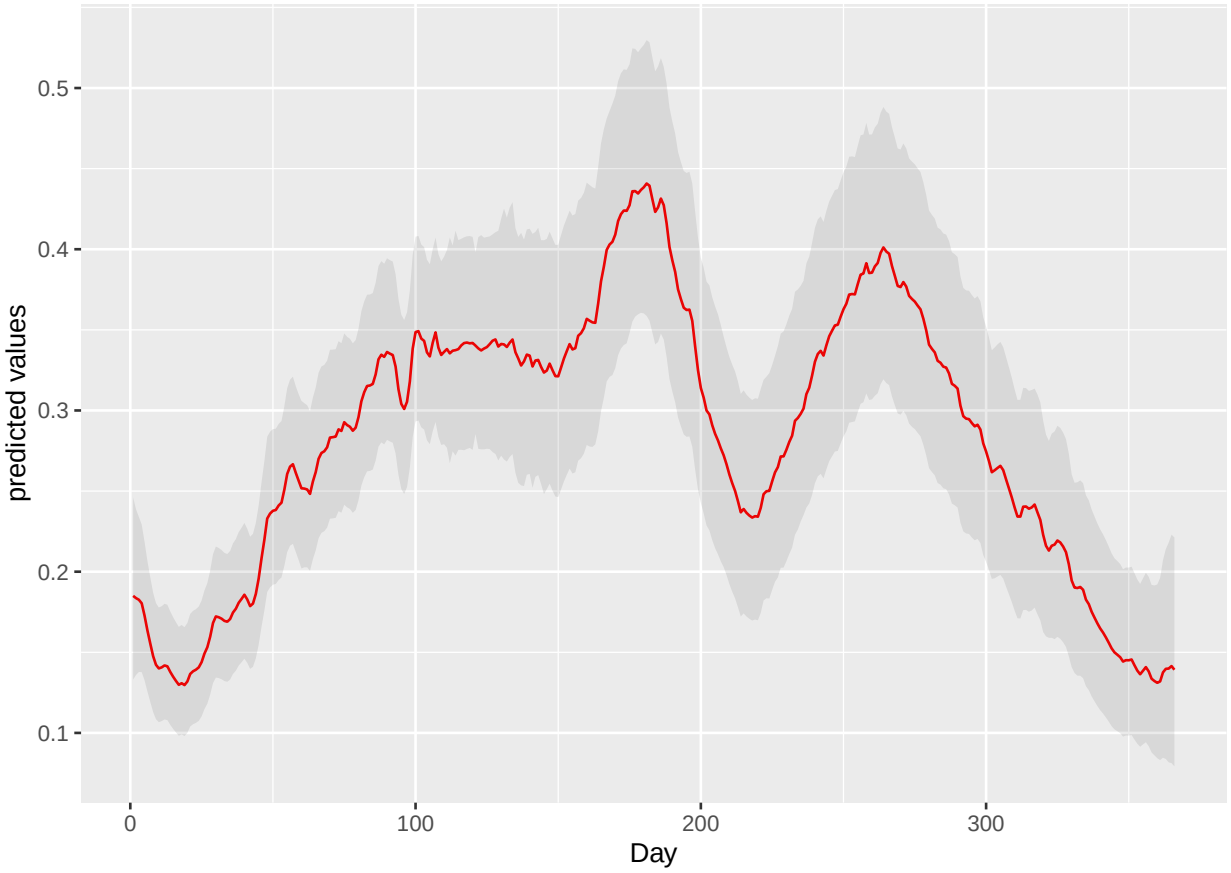

Figure 5: Plot of fitted values with strategy=laplace and int.strategy=ccd

```
mod5$summary.fitted.values[1, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.001 0.1850093 0.02885914  0.1331305 0.1834054  0.2460466
##                            mode
## fitted.Predictor.001 0.1802024
```

```
mod5$summary.fitted.values[201, ]
```

```
##                            mean         sd 0.025quant 0.5quant 0.975quant
## fitted.Predictor.201 0.3079166 0.03889765  0.2361191 0.306333  0.3885468
##                            mode
## fitted.Predictor.201 0.3031761
```

```
mod5$summary.fitted.values[366, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.366 0.1391876 0.03626712 0.07927257 0.1353535  0.2212466
##                           mode
## fitted.Predictor.366 0.12768
```

```
time5
```

```
## elapsed
##    1.64
```

The running time is 1.64. The last possible strategy is 'adaptive'. We fit a model with this strategy.

```
control.inla = list(strategy = "adaptive", int.strategy = "ccd")
fit6 <- INLA_fit(control.inla)
mod6 <- fit6$mod
time6 <- fit6$time
```

```
INLA_plot(mod6)
```

```
mod6$summary.fitted.values[1, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.001 0.1863651 0.02907272  0.1347588 0.1845145  0.2484831
##                            mode
## fitted.Predictor.001 0.1808184
```

```
mod6$summary.fitted.values[201, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.201 0.3333651 0.02873673  0.2791579 0.3326171  0.3918185
##                            mode
## fitted.Predictor.201 0.3311221
```

```
mod6$summary.fitted.values[366, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.366 0.1333908 0.02367606 0.09204403 0.1316544  0.1846206
##                            mode
## fitted.Predictor.366 0.1282184
```

The running time for fitting this model is 1.18. The last combination in control.inla we try is using Laplace as strategy and grid as strategy for the integration. From prior knowledge of the strategies, we expect this method to have the longest running time.

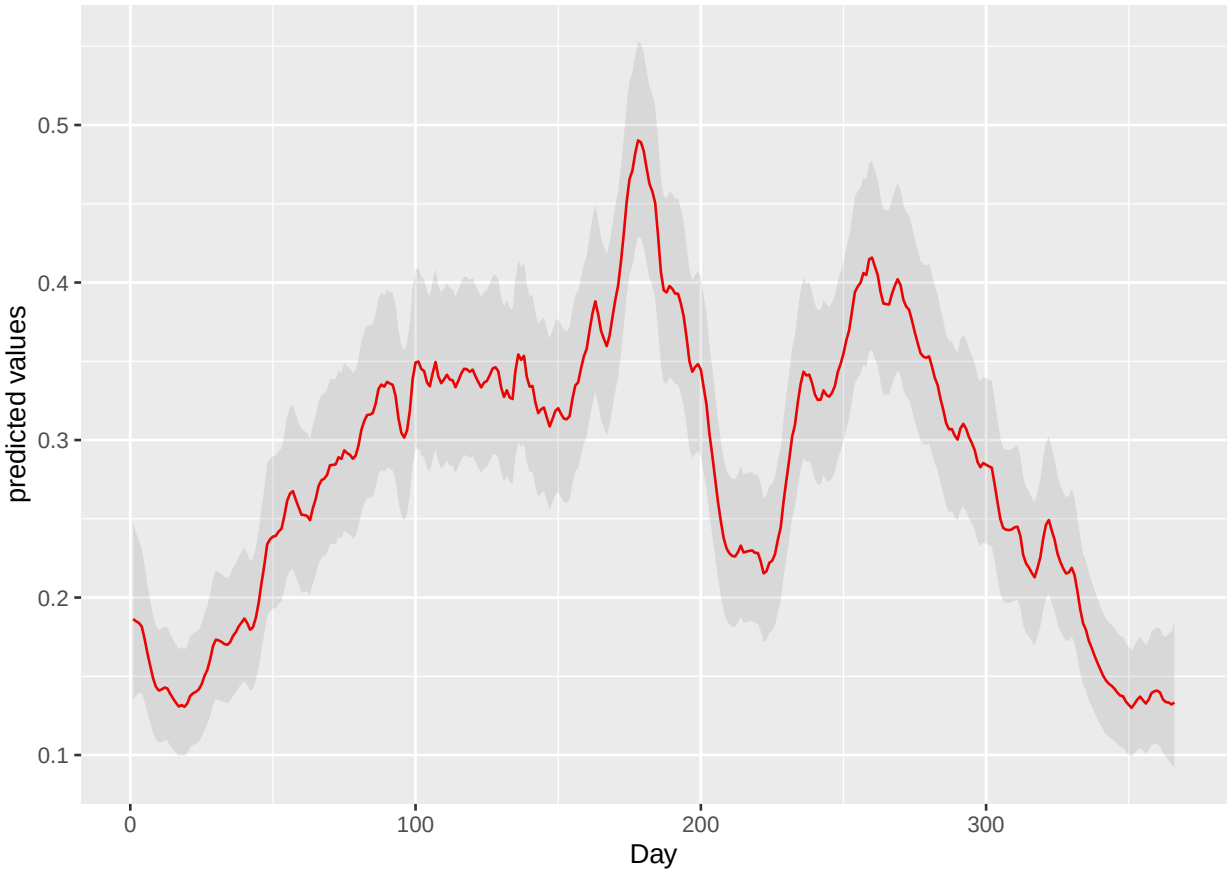

Figure 6: Plot of fitted values with strategy=laplace and int.strategy=ccd

```
control.inla = list(strategy = "laplace", int.strategy = "grid")
fit7 <- INLA_fit(control.inla)
mod7 <- fit7$mod
time7 <- fit7$time
```

```
INLA_plot(mod7)
```

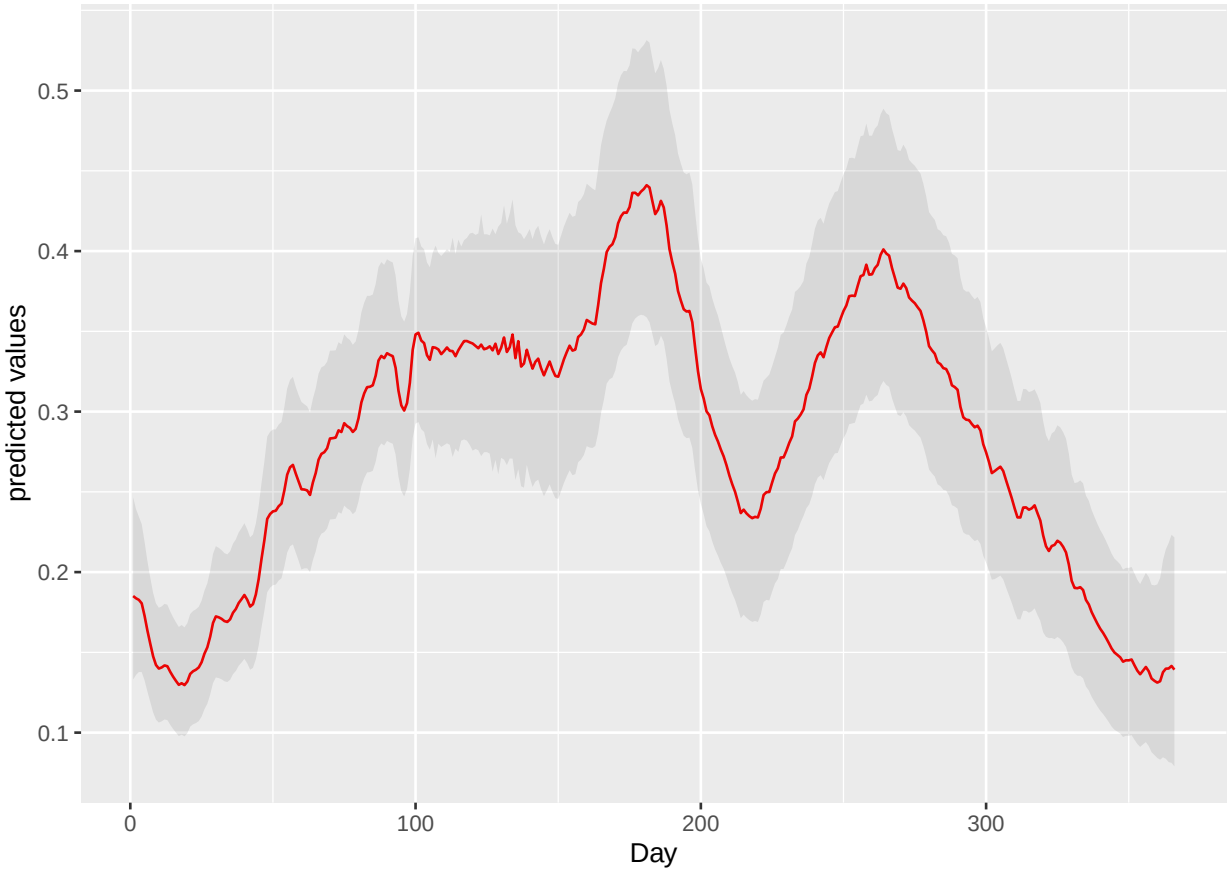

Figure 7: Plot of fitted values with strategy=laplace and int.strategy=ccd

```
mod7$summary.fitted.values[1, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.001 0.1851027 0.02901654  0.1331077 0.1834301   0.246658
##                            mode
## fitted.Predictor.001 0.1801034
```

```
mod7$summary.fitted.values[201, ]
```

```
##                            mean         sd 0.025quant 0.5quant 0.975quant
## fitted.Predictor.201 0.3081046 0.03903862  0.2360806 0.306502  0.3891139
##                            mode
## fitted.Predictor.201 0.3033422
```

```
mod7$summary.fitted.values[366, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.366 0.1392134 0.03640165 0.07899797 0.1353778  0.2216105
##                            mode
## fitted.Predictor.366 0.1277549
```

As seen in figure 7, the results are very similar to the results of the other models. The computation time is 2.23, which is as expected the highest running time so far.

The different inputs for control.inla gives very similar results. This imply that the results are very robust for the two control.inputs we have used. We do however notice a significant longer running time when using Laplace for approximations. The running time is also larger when using `grid`compared to using `ccd`.

c) INLA model 2

The first difference of this model compared to the first model is that the intercept is not removed. this means that we have

$$y_t|\eta_t \sim \text{Bin}(n_t, \pi(\eta_t)), \quad \pi(\eta_t) = \frac{\exp(\eta_t)}{1+\exp(\eta_t)} = \frac{1}{1+\exp(-\eta_t)}.$$

where $\eta_t = \alpha + \tau_t$, and α is the intercept.
In addition, unlike the previous model, this model uses the argument constr=TRUE. This means that there is a sum-to-zero constraint, and the sum of $(\tau_1, .., \tau_n)$ is

$$\sum_{i=1}^{n} \tau_i = 0.$$

We fit the model and compare it to the model from 2a.

```
alpha = 2
beta = 0.05
hyper = list(theta = list(prior = "loggamma", param = c(alpha, beta)))
control.inla = list(strategy = "simplified.laplace", int.strategy = "ccd")

mod8 <- inla(n.rain ~ f(day, model = "rw1", hyper = hyper, constr = TRUE), data = rain,
    Ntrials = n.years, control.compute = list(config = TRUE), family = "binomial",
    verbose = TRUE, control.inla = control.inla)
```

The fitted values of the models in 2a and 2c are plotted together.

```
ggplot() + geom_line(data = as.data.frame(mod1$summary.fitted.values$mean), mapping = aes(x = 1:366,
    y = mod1$summary.fitted.values$mean), color = "black") + geom_line(data = as.data.frame(mod8$summary
    mapping = aes(x = 1:366, y = mod8$summary.fitted.values$mean), color = "red") +
    xlab("Day") + ylab("Fitted values") + geom_label_repel()
```

From figure 8 we can't see any significant differences. The estimated values of the mean of $\pi(\tau_1)$, $\pi(\tau_{201})$ and $\pi(\tau_{366})$ is printed below.

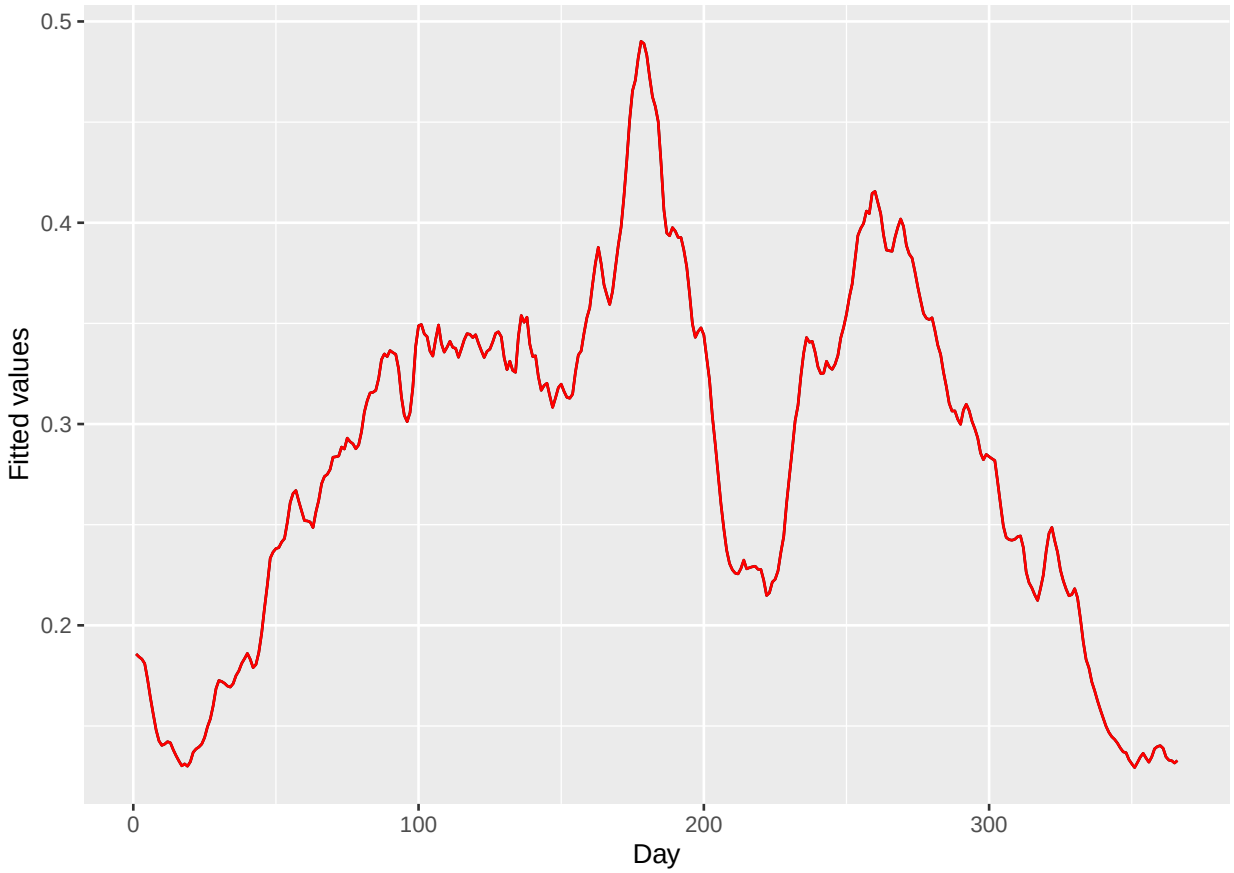

Figure 8: Fitted values of model 1 and 8

```
mod8$summary.fitted.values[1, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.001 0.1857527 0.02889691  0.1336364 0.1841919  0.2467203
##                          mode
## fitted.Predictor.001 0.18106
```

```
mod8$summary.fitted.values[201, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.201 0.3329546 0.02871598  0.2785428 0.3322919  0.3911305
##                            mode
## fitted.Predictor.201 0.3309665
```

```
mod8$summary.fitted.values[366, ]
```

```
##                            mean         sd 0.025quant  0.5quant 0.975quant
## fitted.Predictor.366 0.1328184 0.02346735 0.09103601 0.1313671  0.1828485
##                            mode
## fitted.Predictor.366 0.1284745
```

```
intercept <- mod8$summary.fixed$mean
```

We can't see any significant differences between the predictions of the two models. The estimated intercept term is -0.9876664. In the model η_t is given by

$$\eta_t = \log\left(\frac{\pi(\eta_t)}{1-\pi(\eta_t)}\right) \implies \tau_t = \log\left(\frac{\pi(\eta_t)}{1-\pi(\eta_t)}\right) - \alpha$$

In the model in 2a, we have

$$\tau_t = \log\left(\frac{\pi(\tau_t)}{1-\pi(\tau_t)}\right)$$

Below, we plot the estimated mean values of τ_t for $t = 1, .., T$ for both models.

```
tau_1 = mod1$summary.random$day$mean
tau_8 = mod8$summary.random$day$mean
label1 = c("Model 1", "Model 8")
ggplot() + geom_line(data = as.data.frame(tau_1), mapping = aes(x = 1:366, y = tau_1,
    color = "Model 1")) + geom_line(data = as.data.frame(tau_8), mapping = aes(x = 1:366,
    y = tau_8, color = "model 8")) + xlab("Day") + ylab("estimated values of tau") +
    labs(color = "Models")
```

We see that the estimated τ values are different between the two models. The reason for this is as stated that there is a sum-to-zero constraint on the second model. However, the shape of the graphs are similar. The intercept term makes up for the fact that there are different constraints and we get very similar results. This can also be seen when calculating the posterior distribution.

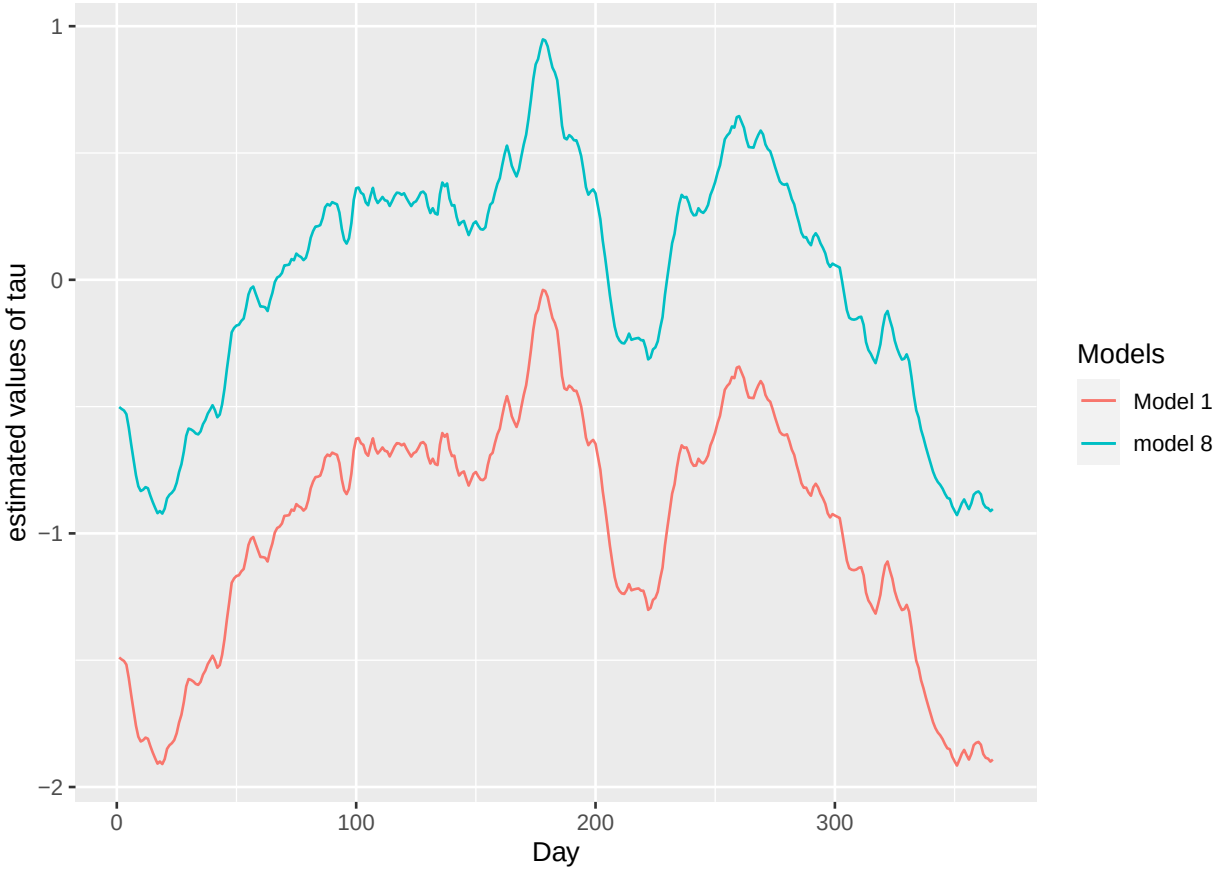

Figure 9: τ-values of model 1 and 8

```
inla.doc("loggamma")
```