Modelling cyclist count with Gaussian Processes in Stan

Markus Paasiniemi

2017-08-23

Introduction

This report documents a Stan model that uses Gaussian processes to model cyclist count data in Helsinki between 2014 and 2016. The model is fitted via the RStan (Stan Development Team 2016) interface.

Data

The data includes daily average temperature, precipitation amount and number of cyclists passing the Baana bicycle corridor. The data can be obtained by running the `get_data.R`-script.

```
rm(list = ls())
library(tidyverse)
library(gridExtra)
library(feather)
library(readr)
library(lubridate)
library(ggjoy)
library(rstan)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

Prepare the data for the modelling by scaling the variables and take a subset of 200 observations for training the model.

```
# source(get_data.R)
(df_raw <- read_feather("data.feather"))
```

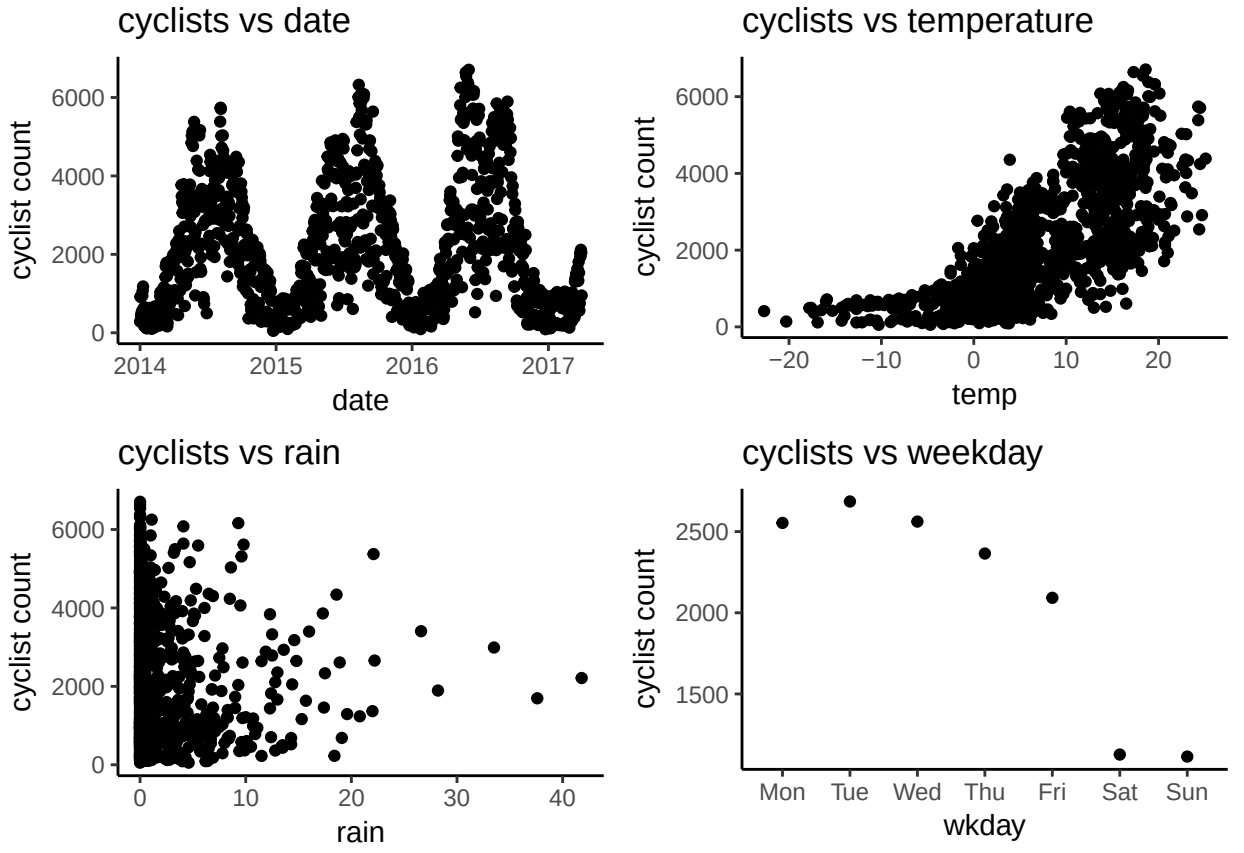
```
## # A tibble: 1,186 x 6
##          date  wkday day_ind  temp  rain count
##        <date> <fctr>   <dbl> <dbl> <dbl> <int>
##  1 2014-01-01    Wed    -592   3.6   0.0   289
##  2 2014-01-02    Thu    -591   1.1   0.0   921
##  3 2014-01-03    Fri    -590   1.7   0.0   912
##  4 2014-01-04    Sat    -589   2.6   2.7   422
##  5 2014-01-05    Sun    -588   4.5   0.0   497
##  6 2014-01-06    Mon    -587   2.4   0.8   545
##  7 2014-01-07    Tue    -586   3.4  11.1   944
##  8 2014-01-08    Wed    -585   5.1   1.4  1070
##  9 2014-01-09    Thu    -584   4.6  10.7  1181
## 10 2014-01-10    Fri    -583   1.8   3.0   618
## # ... with 1,176 more rows
```

```
df <- mutate_at(df_raw, c("temp", "rain", "count"), function(x) c(scale(x)))
# functions to revert the 0,1-scaling
stats <- select(df_raw, temp:count) %>% summarise_all(funs(m = mean, sd = sd))
inv_scale_count <- function(x) x*stats$count_sd + stats$count_m
```

```r
inv_scale_temp <- function(x) x*stats$temp_sd + stats$temp_m
inv_scale_rain <- function(x) x*stats$rain_sd + stats$rain_m

# take a subset for a training data
n_tr <- 200
set.seed(1)
inds_tr <- sample(nrow(df), n_tr)
df_tr <- slice(df, inds_tr)
```

Make a few plots to illustrate the possible relationships between the target and the explanatory variables.

```r
p_date <- ggplot(df_raw) + geom_point(aes(x = date, y = count)) +
  labs(y = "cyclist count", title = "cyclists vs date") + theme_classic()
p_temp <- ggplot(df_raw) + geom_point(aes(x = temp, y = count)) +
  labs(y = "cyclist count", title = "cyclists vs temperature") + theme_classic()
p_rain <- ggplot(df_raw) + geom_point(aes(x = rain, y = count)) +
  labs(y = "cyclist count", title = "cyclists vs rain") + theme_classic()
p_wkday <- group_by(df_raw, wkday) %>% summarise(count = mean(count)) %>%
  ggplot() + geom_point(aes(x = wkday, y = count)) +
  labs(y = "cyclist count", title = "cyclists vs weekday") + theme_classic()
grid.arrange(p_date, p_temp, p_rain, p_wkday)
```



There seems to be a relationship between daily temperature and cyclist count. From the figures it is not clear, if it is mainly a yearly pattern regardless of the temperature, or if it is the actual temperature that affects the cyclist count. Similarly, weekday seems to have a great influence on the cyclist count. For rain, however, there isn't a clear relationship, or at least any that would appear by simply plotting the data.

Model

Let's fit a GP model that tries to capture these effects. Based on the previous observations, the model will be as follows:

$$
\begin{aligned}
y &\sim \mathrm{N}(f, \sigma) \\
f &\sim \mathrm{GP}(0, K) \\
K &= \alpha_{pe} K_{pe}(x_d|\rho_{pe}, 7) + \alpha_{ard} K_{ard}(x_w|\rho_{ard}) \\
\sigma &\sim N(0, 1) \\
\alpha_{pe}, \alpha_{ard} &\sim N(0, 1) \\
\rho_{pe}, \rho_{ard}[1], \rho_{ard}[2] &\sim \mathrm{InvGamma}(5, 5),
\end{aligned}
$$

where x_d and x_w refer to the day-indices and weather-variables, K_{pe} and K_{ard} to a periodic and an exponentiated quadratic (with varying length-scales) covariance function, α's to the marginal standard deviations of the CF magnitudes and ρ's to the length-scales of the CFs.

To summarise, the latent mean function f is a GP with a covariance function K, consisting of a periodic component (with a period of 7, ie. a week) and a weather component (that consists of temperature and rain). The stan-code for the model is in the file `gp.stan`.

The data is scaled to have 0 mean, 1 standard deviation, so the priors for the noise σ and marginal standard deviation α are essentially weakly regularising on the components and roughly stating that both of the components explain some of the output, but there is likely also some unexplained variance.

For the length-scales, the prior distribution is an inverse gamma instead of a normal so that there is only a non-significant mass around zero. This is because when the length-scale is smaller than the smallest possible change in the variables, the likelihood plateaus and it causes numerical problems in fitting the model. On the other hand, in this particular case it is also not even desirable to detect functions that are more rapidly changing than the smallest differences in the data.

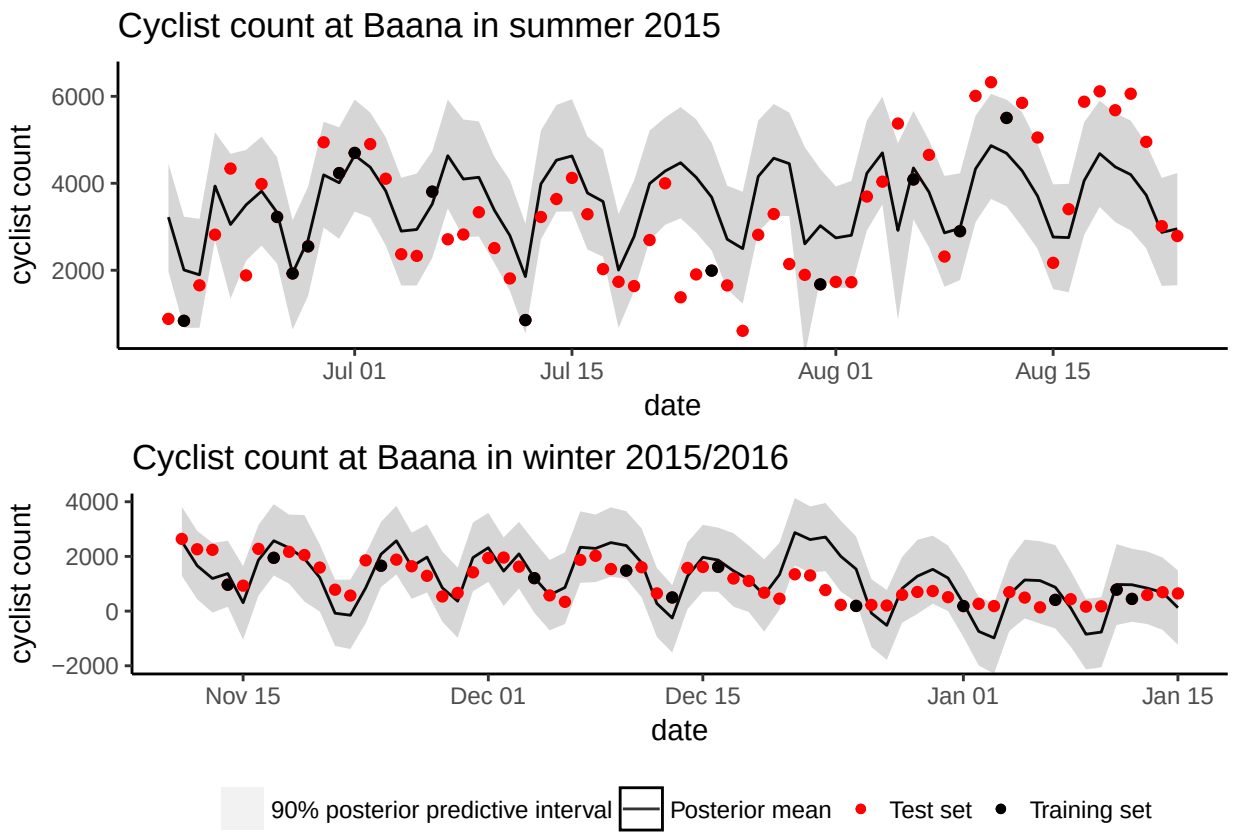
Fitting the model with MCMC takes around an hour, so load the results from the disk.

```
# data for the model
d <- list(N1 = n_tr, N2 = nrow(df), D_w = 2, y1 = df_tr$count,
          x1_d = df_tr$day_ind, x1_w = select(df_tr, temp, rain),
          x2_d = df$day_ind, x2_w = select(df, temp, rain),
          delta = 1e-9, sd_sig = 1, sd_a = 1, gamma = 7)
# m_gp <- stan_model(stanc_ret = stanc_builder("gp.stan"))
# fit <- sampling(m_gp, data = d, iter = 625, warmup = 500, seed = 1)
# e <- extract(fit)
# write_rds(fit, "fit.rds")
e <- read_rds("fit.rds") %>% extract()
```

Next the fit is examined visually. The code below produces a plot of the model fit for summer 2015 and winter 2015/2016 for demonstrating the model fit.

```
# a helper function to plot the posterior mean and credible intervals
get_stat <- function(var, qs) {
  apply(var, 2, function(x) c(mean(x), quantile(x, qs))) %>%
    t() %>% as_tibble() %>% setNames(c("mean", "lq", "uq"))
}
y_stats <- get_stat(e$y2, qs = c(0.05, 0.95)) %>%
  mutate_all(inv_scale_count) %>%
  bind_cols(select(df_raw, date, count))

plot_inds <- function(y_stats, inds, title, ylim, show_legend) {
  obj <- ggplot(slice(y_stats, inds), aes(x = date)) +
```

```
    geom_line(aes(y = mean, col = "Posterior mean")) +
    geom_ribbon(aes(ymin = lq, ymax = uq,
                    color = "90% posterior predictive interval"), alpha = 0.2) +
    geom_point(aes(y = count, col = "Test set")) +
    geom_point(aes(x = date, y = count, col = "Training set"),
               df_raw[intersect(inds, inds_tr), ]) +
    labs(title = title, x = "date", y = "cyclist count") +
    coord_cartesian(ylim = ylim) +
    scale_color_manual(name = "", values = c("Posterior mean" = "black",
                                             "Training set" = "black",
                                             "Test set" = "red",
                                             "90% predictive interval" = "gray")) +
    theme_classic() +
    theme(legend.position = "bottom")

  if (show_legend) {
    obj + guides(color = guide_legend(override.aes=list(
      shape=c(NA, NA, 16, 16), linetype = c(1, 1, 0, 0),
      fill = c("gray", rep("white", 3)))))
  } else {
    obj + guides(color = FALSE)
  }
}

inds_s15 <- 535:600 # summer15
title_s15 <- "Cyclist count at Baana in summer 2015"
ylim_s15 <- c(500, 6500)
inds_w16 <- 680:745 # winter15/16
title_w16 <- "Cyclist count at Baana in winter 2015/2016"
ylim_w16 <- c(-2000, 4000)

p_s15 <- plot_inds(y_stats, inds_s15, title_s15, ylim_s15, FALSE)
p_w16 <- plot_inds(y_stats, inds_w16, title_w16, ylim_w16, TRUE)
grid.arrange(p_s15, p_w16)
```



Looks like the model fits the data relatively well and is able to learn at least a lot of the patterns in the cycling data. On the other hand it seems that the weekly pattern is stronger in the summer than in the winter, causing the model to overestimate the daily changes during the winter, and underestimate them during the summer.

In addition, even the posterior mean for the cyclist count for some of the days is way below zero, which is of course non-sensical. Of course this might happen as the normal model is not really the appropriate one for count data and it is likely that lognormal, poisson or negative binomial model might be more appropriate. For the purpose of this report, however, I want to model the components (covariance functions) as additive for simplicity.

While the latter cannot be fixed, the former can. A solution is to multiply the periodic CF with a squared exponential CF (Rasmussen and Williams 2006). For the exponentiated quadratic the interpretation is that for a lenght-scale ρ, the functions on an interval $[0, 365]$ pass zero $365/(\rho\pi)$ times on average (Stan Development Team 2017).

To have the periodic effect be different for winter and summer, it would make sense to have the long-term component cross zero between once and twice a year. With this information it is actually natural to specify the prior on this scale, ie. to have a parameter θ for the number of times the long-term component passes zero (on average) and then have the length-scale be $\rho_{se} = 365/(\theta\pi)$.

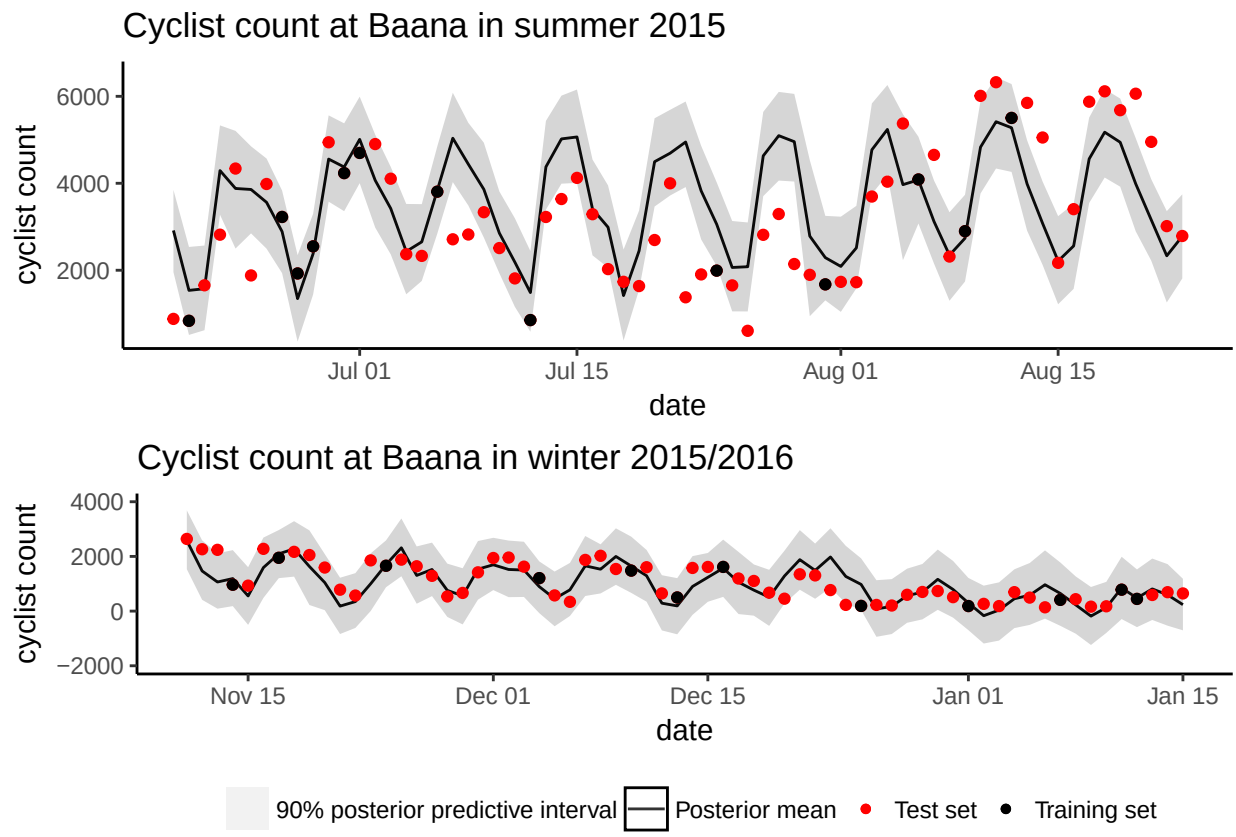
In addition to the scale for the prior being more intuitive, it also makes sense that θ has symmetric prior rather than ρ_{se}, which means that deviations to either side of the mean of θ are equally likely. It also turns out that at least for this data set this formulation also makes the sampling more efficient.

To summarise, the updated model is now:

$$
\begin{aligned}
y &\sim \mathrm{N}(f, \sigma) \\
f &\sim \mathrm{GP}(0, K) \\
K &= \alpha_{pe} K_{pe}(x_d|\rho_{pe}, 7) K_{se}(x_d|\rho_{se}, 7) + \alpha_{ard} K_{ard}(x_w|\rho_{ard}) \\
\sigma &\sim N(0, 1) \\
\alpha_{pe}, \alpha_{ard} &\sim N(0, 1) \\
\rho_{pe}, \rho_{ard}[1], \rho_{ard}[2] &\sim \mathrm{InvGamma}(5, 5) \\
\theta &\sim N(1.5, 0.25) \\
\rho_{se} &= 365/(\theta\pi).
\end{aligned}
$$

Let's fit the updated model (or once again load the fitted model from the disk):

```
y_stats_lp <- get_stat(e_lp$y2, qs = c(0.05, 0.95)) %>%
  mutate_all(inv_scale_count) %>%
  bind_cols(select(df_raw, date, count))

p_s15_lp <- plot_inds(y_stats_lp, inds_s15, title_s15, ylim_s15, FALSE)
p_w16_lp <- plot_inds(y_stats_lp, inds_w16, title_w16, ylim_w16, TRUE)
grid.arrange(p_s15_lp, p_w16_lp)
```



The fit to data seems significantly better, and the confidence intervals are also a lot narrower which means that the updated model indeed seems to be an improvement.

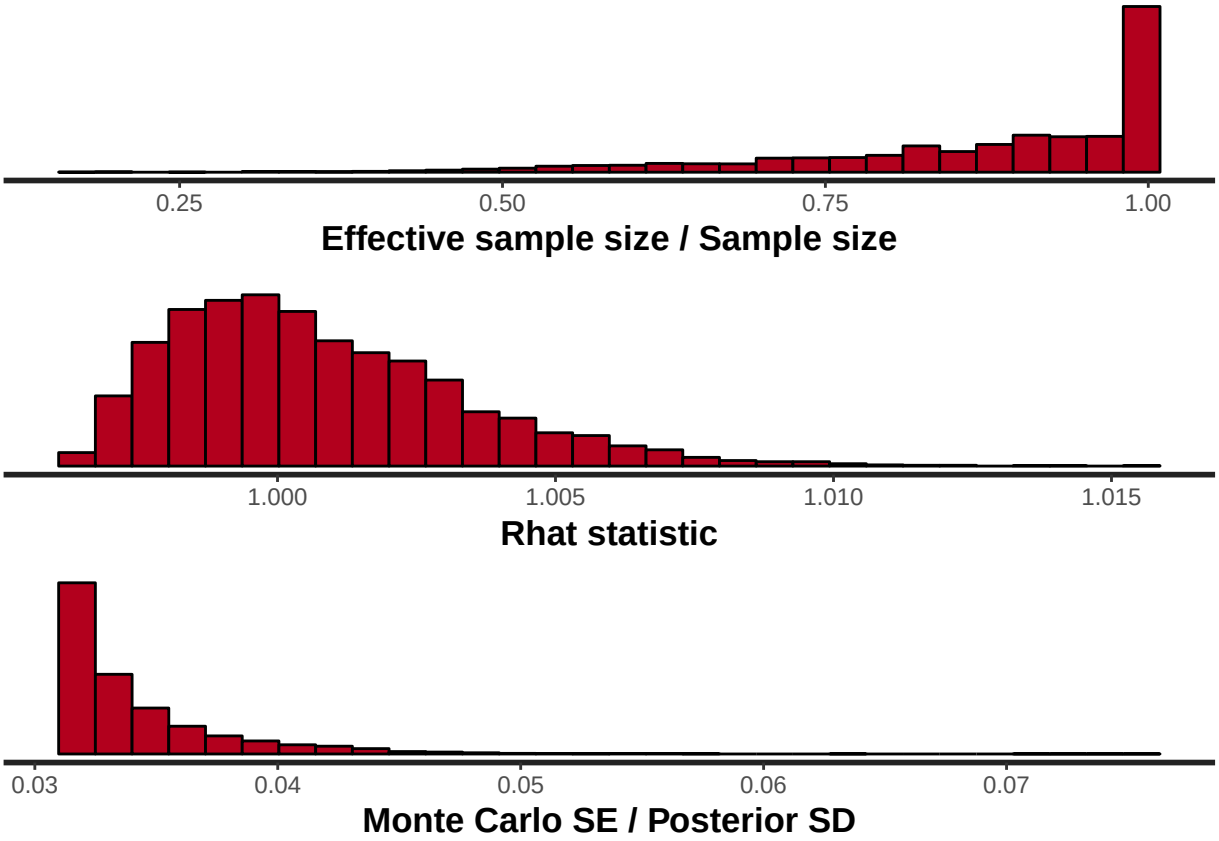
Next, let's fit the model with slightly more samples to ensure there are no convergence problems and add a few more data sets on which to evaluate the GP to visualise the results. To visualise the effects of weather, build the following data frames:

```
n_w <- 20
# effects of temperature, when its not raining
x_te <- tibble(temp = seq(min(df$temp), max(df$temp), length.out = n_w),
               rain = min(df$rain))
# effects of rain when it's 'hot;
x_ra_h <- mutate(x_te, temp = (18-stats$temp_m)/stats$temp_sd,
                 rain = seq(min(df$rain), max(df$rain), length.out = n_w))
# effects of rain when it's 'cold'
x_ra_c <- mutate(x_te, temp = (5-stats$temp_m)/stats$temp_sd,
                 rain = seq(min(df$rain), max(df$rain), length.out = n_w))
x_w <- tibble(temp = x_te$temp, rain = x_ra_h$rain)
```

Fit the full model with more samples and the additional data sets.

Let's do a few checks to ensure that there are no apparent problems related to the fit.

```
grid.arrange(stan_ess(fit_all, bins = 30),
             stan_rhat(fit_all, bins = 30),
             stan_mcse(fit_all, bins = 30), ncol = 1)
```



Look's like there's nothing particularly alarming about the markov chains, the Rhats are below 1.02 for all the parameters, Monte Carlo SE is small relative to the SD of the estimates and futhermore, the relative effective sample size does not seem too low.

```
test_inds <- setdiff(1:nrow(df), inds_tr)
tibble(lq = c(0.25, 0.125, 0.05), uq = c(0.75, 0.875, 0.95)) %>% rowwise() %>%
  do(stats = get_stat(e_all$y2, qs = c(.$lq, .$uq))) %>%
  do(calib = mean((df$count >= .$stats$lq & df$count <= .$stats$uq)[test_inds])) %>%
```

```
  unlist() %>% setNames(c("50%", "75%", "90%"))
```

```
##       50%       75%       90%
## 0.5547667 0.7657201 0.8904665
```

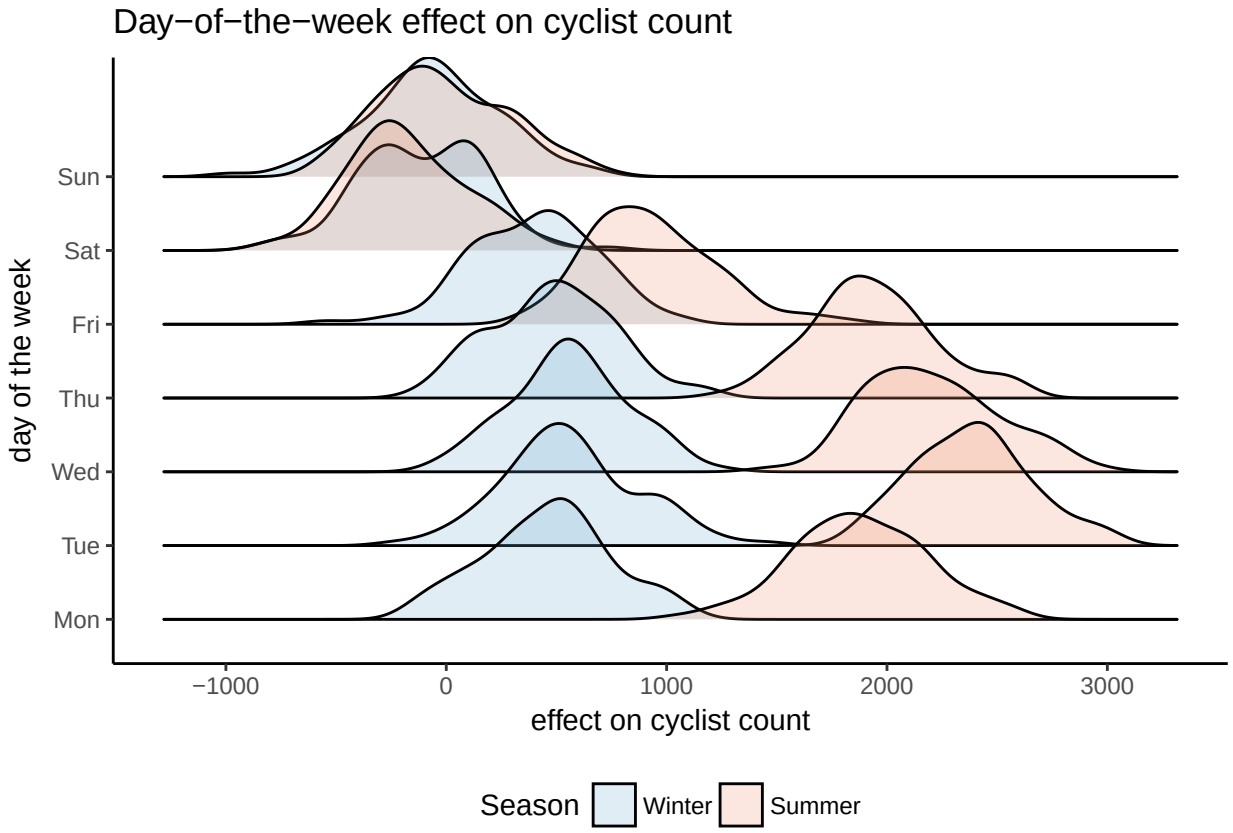
In addition, the predictions seem well calibrated, ie. 50% of the observations are contained in approximately 50% of the predictive intervals etc.

Results

As shown in the previous section, the model fits to the observed data pretty nicely. What about the the effects of weather (and week of the day) on the data?

Let's start by visualising the component with the cyclical day of the week effect (along with a long yearly trend). The count is transformed so that the mean (over all samples and days) of sunday-effect is be zero. This is done because the actual values are not interesting, but rather the differences between days. Furthermore, the results are somewhat experimentally visualised using a joyplot to test a 'new shiny method'.

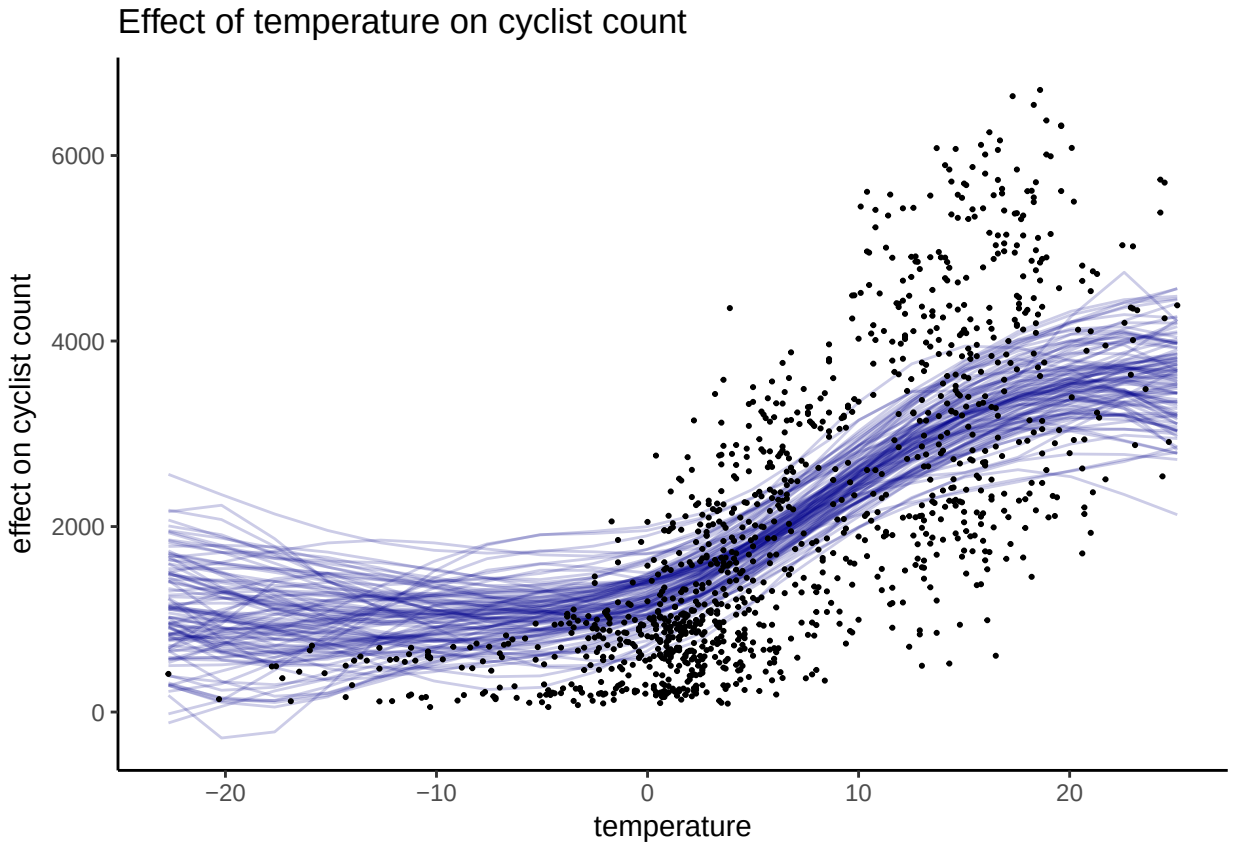
```
# get samples from the given variable
get_samp <- function(var, x, n_samp = 100) {
  as_tibble(t(var[1:n_samp, ])) %>% bind_cols(x = x) %>%
    gather(sample, val, -x)
}

dow_samp <- get_samp(e_all$f_pe, df$date) %>%
  mutate(val = val*stats$count_sd) %>%
  rename(date = x) %>%
  left_join(select(df_raw, date, wkday), "date") %>% # add wkday
  group_by(sample, wkday)
 # transform the component so that mean of cyclists on sunday is 0.
dow_samp$val <- dow_samp$val - mean(filter(dow_samp, wkday == "Sun")$val)

dow_samp_winter <- filter(dow_samp, month(date) %in% c(12,1,2)) %>%
  group_by(sample, wkday) %>%
  summarise(count = mean(val))

dow_samp_summer <- filter(dow_samp, month(date) %in% 6:9) %>%
  group_by(sample, wkday) %>%
  summarise(count = mean(val))

ggplot(bind_rows(dow_samp_winter, dow_samp_summer, .id = "id")) +
  geom_joy(aes(x = count, y = wkday, fill = id), alpha = 0.2, bandwidth = 100) +
  labs(x = "effect on cyclist count", y = "day of the week",
       title = "Day-of-the-week effect on cyclist count") +
  scale_fill_manual(values = c("#67a9cf", "#ef8a62"),
                    labels = c("Winter", "Summer"), name = "Season") +
  theme_classic() + theme(legend.position = "bottom")
```



The figure corresponds to our intuition pretty nicely. The day-of-the-week effect is a lot higher during Summer, which could be because of e.g. work commute. In addition, the effect is slightly lower on fridays, which also makes sense.

What about the weather effect?

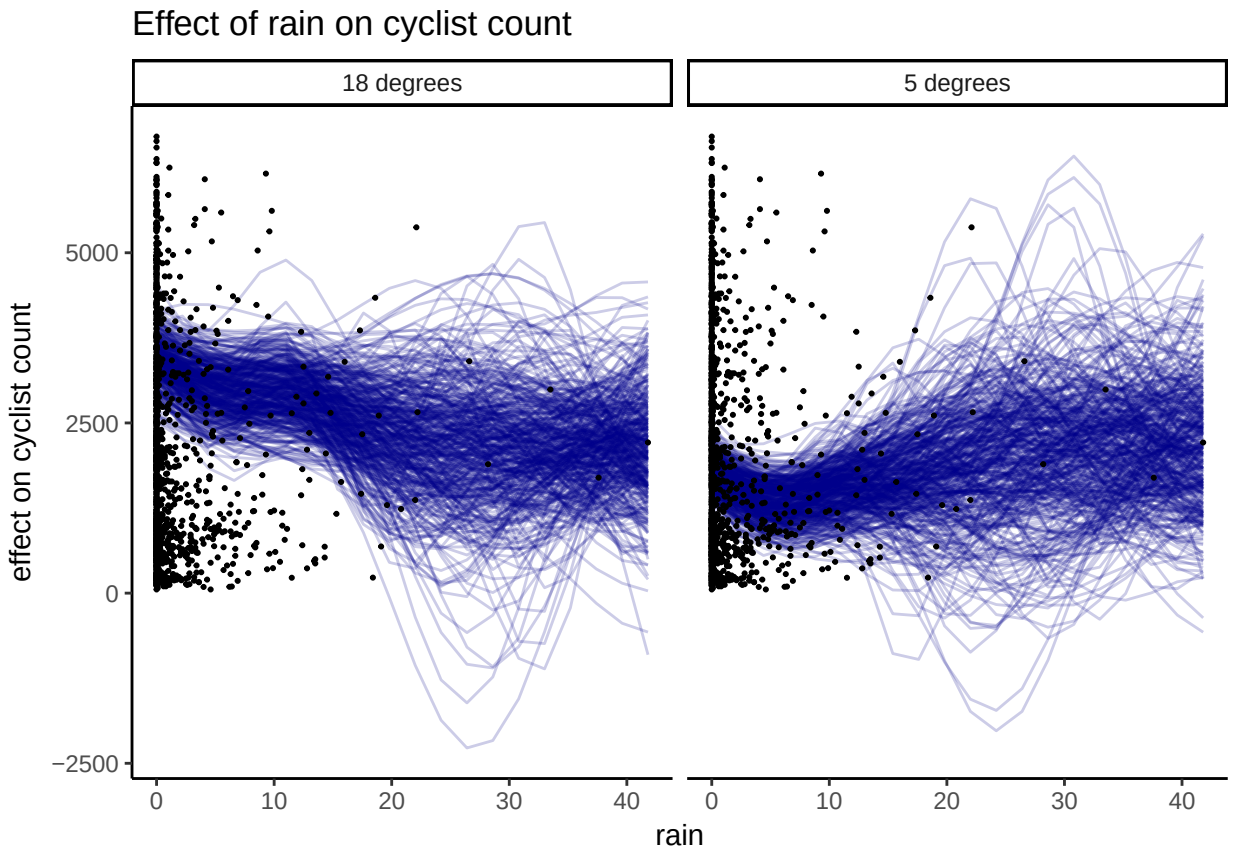
```
get_samp(e_all$f_te, inv_scale_temp(x_w$temp)) %>%
  mutate(val = inv_scale_count(val)) %>%
  rename(temp = x) %>%
  ggplot() +
  geom_line(aes(x = temp, y = val, group = sample), alpha = 0.2, col = "darkblue") +
  geom_point(aes(x = temp, y = count), df_raw, size = 0.4) +
  labs(y = "effect on cyclist count", x = "temperature",
       title = "Effect of temperature on cyclist count") +
  theme_classic()
```



The blue lines are draws from the posterior of the weather-effect-component (holding the amount of rain at zero), and the black dots are the actual data. Again the results are intuitive and relatively clean: the temperature is positively associated with the number of cyclists, as expected. On the other hand, there seems to be some non-linearity in the effect, and especially after the temperature drops under zero, the effect vanishes, which again is in line with intuition.

What about the effects of rain on cyclist count? Due to possible non-linearity, the effect of rain is evaluated at two temperature levels, at 18 and 5 degrees.

```
rain_samp <- get_samp(e_all$f_ra_c, inv_scale_rain(x_w$rain), 400) %>%
  bind_rows(get_samp(e_all$f_ra_h, inv_scale_rain(x_w$rain), 400), .id = "temp") %>%
  mutate(val = inv_scale_count(val)) %>%
  rename(rain = x) %>%
  mutate(temp = (function(x) ifelse(x == "1", "5 degrees", "18 degrees"))(temp))

ggplot(rain_samp) +
  geom_line(aes(x = rain, y = val, group = sample), alpha = 0.2, col = "darkblue") +
  facet_grid(~temp) +
  geom_point(aes(x = rain, y = count), select(df_raw, -temp), size = 0.4) +
  labs(y = "effect on cyclist count", x = "rain",
       title = "Effect of rain on cyclist count") +
  theme_classic()
```



For rain, there does not really seem to be anything clear going on. It might be that when the temperature is higher, there is a negative association with rain and cycling, where as with lower temperature does not really affect cycling.

Conclusion

Relatively unsurprisingly, weather conditions affect cycling. In addition, there are clear weekly patterns in cycling. As a byproduct this report has also demostrated Stan's feasibility with Gaussian process regression. On one hand, fitting the models with MCMC is still relatively slow compared to the other GP toolboxes available, but the flexibility and ease of using the same workflow as with all other Bayesian models make it a great tool fitting GPs.

References

Rasmussen, Carl Edward, and Christopher KI Williams. 2006. *Gaussian Processes for Machine Learning*. Vol. 1. MIT press Cambridge.

Stan Development Team. 2016. "RStan: The R Interface to Stan." http://mc-stan.org/.

———. 2017. "Stan Modeling Language: User's Guide and Reference Manual." http://mc-stan.org/.