Predicting cherry blossom bloom dates with Bayesian normal regression

YC Lim-Hitchings

Introduction

In this analysis, a Bayesian linear regression approach is employed for the prediction of the cherry blossom bloom dates of 5 different locations. Based on a very cursory examination of the sources available online, it appears that temperature throughout the year is very much accepted as an important variable for the bloom dates. In addition to this, it was suspected that each species will also behave differently in terms of the bloom date.

Although, variables such as longitude, latitude, and altitude also seem attractive as predictors, these were not considered. This is because, it is postulated that the main effect of these variables is to alter the temperature of the location. However, this hypothesis is admittedly based on pure speculation from someone with little to no background knowledge of botany or meteorology.

As for the distribution of bloom dates given the temperature and species, a normal model was assumed. This assumption is based on the supposition that the bloom dates for a given year should be independent of previous years should be independent of bloom dates. As such a normal regression with species and temperature as predictors is specified. For simplicity, the temperature used will be limited to a running five day average for the year in question, up to day number 50 of each year.

We will begin by putting together the data provided with daily temperature data retrieved from the National Oceanic and Atmospheric Administration (NOAA) for the locations in North America and Kyoto. Due to some missing information in the temperatures provided, data from the website time and date was retrieved to supplement this gap. Concerning weather information for Liestal, this was retrieved from the open source API, open meteo.

This combined data set will then be explored to examine if the assumptions made have any validity. Finally, the model will be specified as describe using the `rstan` package and predictions made for the year of 2024.

Data tidying

First, we import the data for the bloom dates provided:

```
bloom_df <- read.csv("data/washingtondc.csv") %>%
  bind_rows(read.csv("data/liestal.csv")) %>%
  bind_rows(read.csv("data/kyoto.csv")) %>%
  bind_rows(read.csv("data/vancouver.csv")) %>%
  as.data.frame()
```

Due to reported issues in the **rnoaa** package, it was opted to retrieve the temperature data directly from the NOAA website. These data, along with the temperature data from other sources were combined into a single csv file named "weather.csv".

```
temp = read.csv("./data/weather.csv")
```

Once the data was imported, years and locations that corresponded to the available bloom date data were selected for. A selection of the first 50 days of each year were made. For locations where average daily temperature was not available, it was calculated from the mean of the maximum and minimum temperature. Any missing data was interpolated. These were then combined with the respective rows based on the year in the bloom date data. Additionally, a column specifying the species was added, which grouped all *prunus x yedoensis* species into the same group. Finally, a 5 day average for the temperature was calculated for each year at each location from day of year 1 to day of year 50.

```
# function to arrange temp data into dataframe by year from day 1:50
get_temp_day = function(x, temps){
  temp_year = filter(temps, year == x)
  temp_year_wide = pivot_wider(temp_year[1:50,-1],
                               names_from = date,
                               values_from = temp)
  temp_year_wide = as.data.frame(temp_year_wide)

  return((temp_year_wide))
}

# function to patch together temp data with bloom data
combine_temp_data = function(stationid, loc, tavg = FALSE){

  if(tavg == FALSE){
  temp_data = filter(temp, STATION == stationid)%>%
  mutate(DATE = as.Date(DATE, format = "%d/%m/%Y"),
```

```
          TMAX = na.approx(TMAX),
          TMIN = na.approx(TMIN))%>%
  transmute(year = parse_number(format(DATE, "%Y")),
            date = DATE,
            temp = (TMAX + TMIN) / 2)
  }else{temp_data = filter(temp, STATION == stationid)%>%
  mutate(DATE = as.Date(DATE, format = "%d/%m/%Y"),
         TAVG = na.approx(TAVG))%>%
  transmute(year = parse_number(format(DATE, "%Y")),
            date = DATE,
            temp = TAVG)}

station = filter(bloom_df, year >= temp_data$year[1], location == loc)

years = station$year

temps_list = lapply(years, get_temp_day, temps = temp_data)
temps_df = as.data.frame(matrix(unlist(temps_list),
                                nrow = length(temps_list),
                                ncol =50,
                                byrow= TRUE))
names(temps_df)= c(paste0("temp_doy_", 1:50))

bloom_temp = cbind(station, temps_df)

return(bloom_temp)
}


vancouver = combine_temp_data(stationid = "CA001108395",
                              loc = "vancouver",
                              tavg = TRUE)
dc = combine_temp_data(stationid = "USC00186350",
                       loc = "washingtondc")
kyoto = combine_temp_data(stationid = "JA000047759",
                          loc = "kyoto")
liestal = combine_temp_data(stationid = "Liestal",
                            loc = "liestal",
                            tavg = TRUE)

df_combined = na.omit(rbind (vancouver, dc, kyoto, liestal))
```

```r
df_combined = df_combined %>%
  mutate(species = ifelse(location == "kyoto",
                          "jamasakura",
                          ifelse(location == "liestal",
                                 "avium",
                                 "yedoensis")),
         temp_doy_1_5 = rowMeans(cbind(df_combined$temp_doy_1,
                                       df_combined$temp_doy_2,
                                       df_combined$temp_doy_3,
                                       df_combined$temp_doy_4,
                                       df_combined$temp_doy_5)),

         temp_doy_6_10 = rowMeans(cbind(df_combined$temp_doy_6,
                                        df_combined$temp_doy_7,
                                        df_combined$temp_doy_8,
                                        df_combined$temp_doy_9,
                                        df_combined$temp_doy_10)),

         temp_doy_11_15 = rowMeans(cbind(df_combined$temp_doy_11,
                                         df_combined$temp_doy_12,
                                         df_combined$temp_doy_13,
                                         df_combined$temp_doy_14,
                                         df_combined$temp_doy_15)),

         temp_doy_16_20 = rowMeans(cbind(df_combined$temp_doy_16,
                                         df_combined$temp_doy_17,
                                         df_combined$temp_doy_18,
                                         df_combined$temp_doy_19,
                                         df_combined$temp_doy_20)),

         temp_doy_21_25 = rowMeans(cbind(df_combined$temp_doy_21,
                                         df_combined$temp_doy_22,
                                         df_combined$temp_doy_23,
                                         df_combined$temp_doy_24,
                                         df_combined$temp_doy_25)),

         temp_doy_26_30 = rowMeans(cbind(df_combined$temp_doy_26,
                                         df_combined$temp_doy_27,
                                         df_combined$temp_doy_28,
                                         df_combined$temp_doy_29,
                                         df_combined$temp_doy_30)),
```

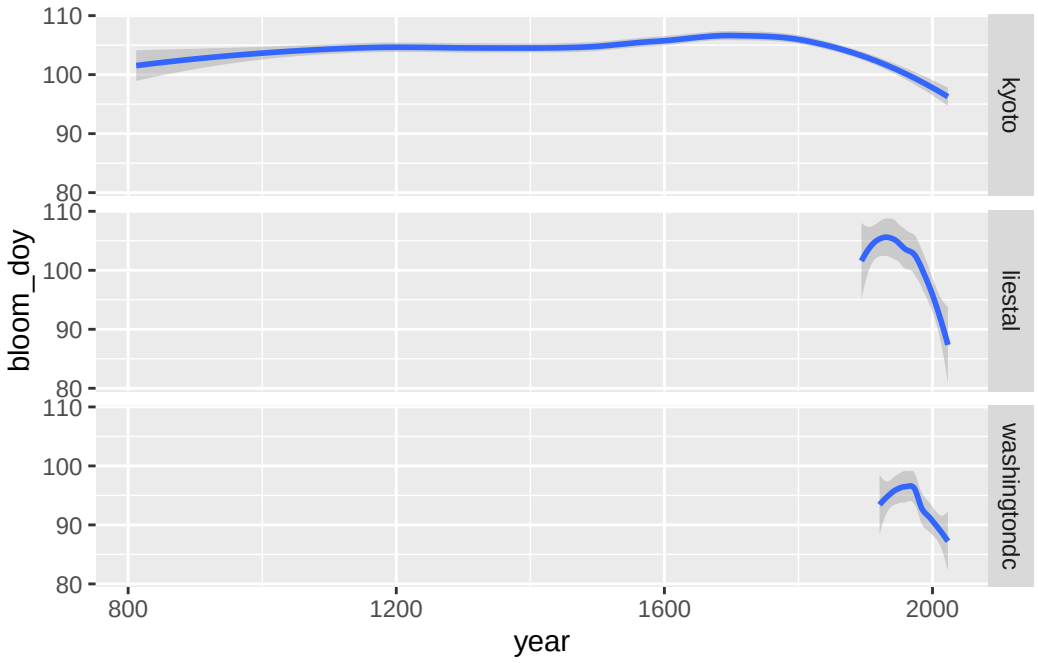
```
        temp_doy_31_35 = rowMeans(cbind(df_combined$temp_doy_31,
                                        df_combined$temp_doy_32,
                                        df_combined$temp_doy_33,
                                        df_combined$temp_doy_34,
                                        df_combined$temp_doy_35)),

        temp_doy_36_40 = rowMeans(cbind(df_combined$temp_doy_36,
                                        df_combined$temp_doy_37,
                                        df_combined$temp_doy_38,
                                        df_combined$temp_doy_39,
                                        df_combined$temp_doy_40)),

        temp_doy_41_45 = rowMeans(cbind(df_combined$temp_doy_41,
                                        df_combined$temp_doy_42,
                                        df_combined$temp_doy_43,
                                        df_combined$temp_doy_44,
                                        df_combined$temp_doy_45)),
        temp_doy_46_50 = rowMeans(cbind(df_combined$temp_doy_46,
                                        df_combined$temp_doy_47,
                                        df_combined$temp_doy_48,
                                        df_combined$temp_doy_49,
                                        df_combined$temp_doy_50)))
```

Exploring the data

We are now ready to explore the data. First looking at the overall trend:

```
bloom_df %>%
  filter(location != "vancouver") %>%
  ggplot(aes(y = bloom_doy, x = year))+
    geom_smooth()+
    facet_grid(rows = "location")
```

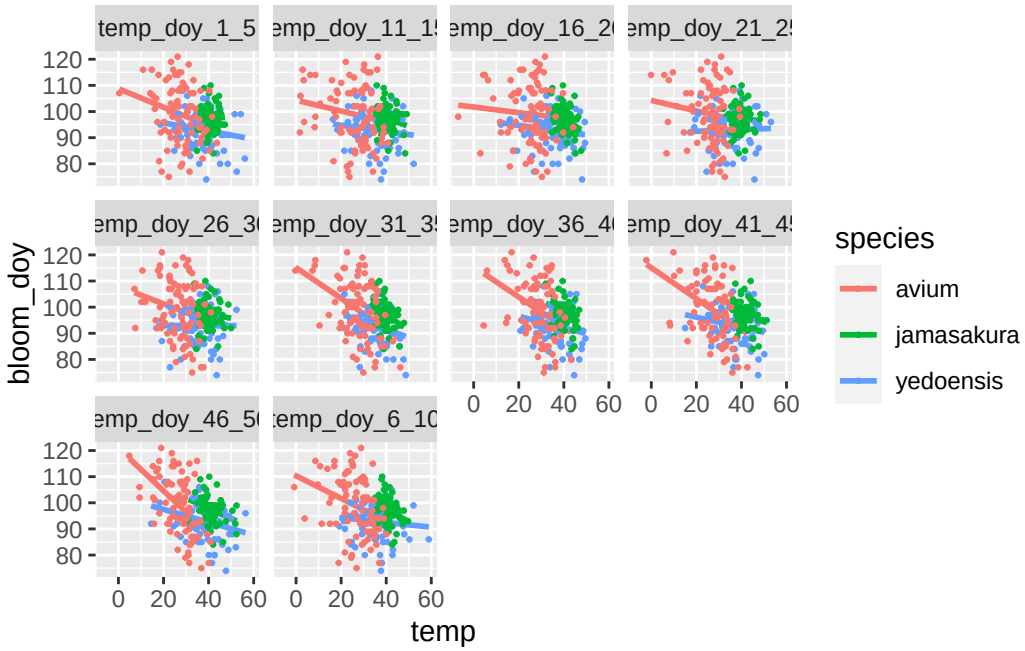
```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



We can see that over the years the bloom dates are happening earlier especially in recent times. This is consistent with the increase in temperature globally. Now we can look at the effect of the five day average temperature of the first 50 days on the bloom date.

```
df_combined_long = pivot_longer(df_combined,
                                cols = c(59:68),
                                names_to = "doys",
                                values_to = "temp")

ggplot(df_combined_long, aes(x = temp, y=bloom_doy, color = species))+
  geom_smooth(method = "lm", se = FALSE)+
  geom_point(size = 0.5)+
  facet_wrap("doys")
```

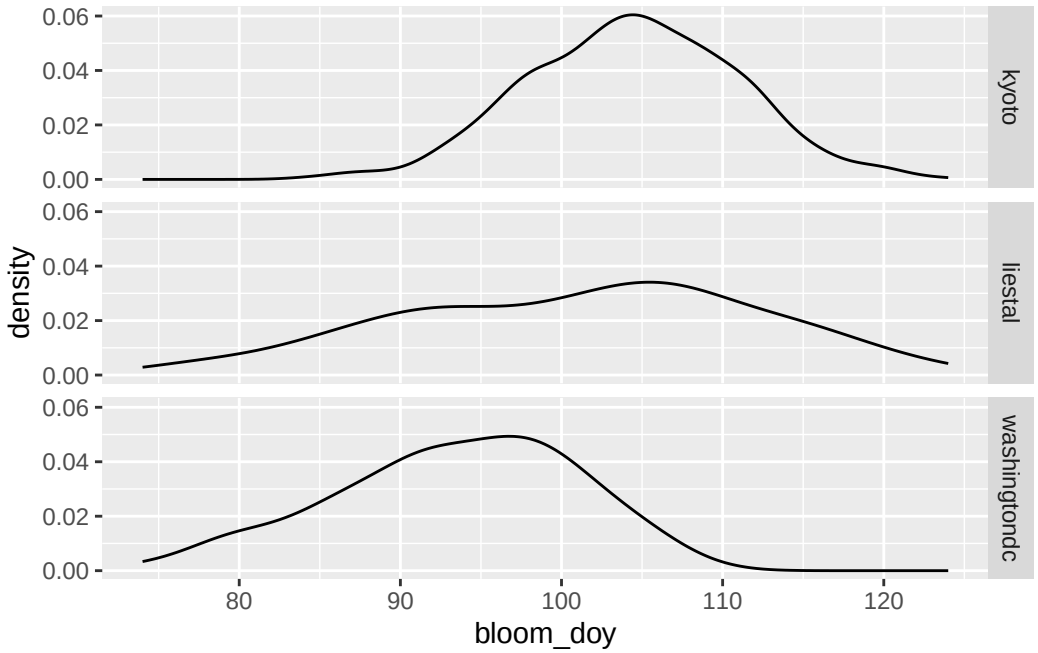
```
`geom_smooth()` using formula = 'y ~ x'
```



These plots seem to support the notion that there is a negative correlation between bloom day of year and that the different species behave differently.

Finally we evaluate if the normal assumption has any weight.

```
bloom_df %>%
  filter(location != "vancouver") %>%
  ggplot(aes(x = bloom_doy))+
    geom_density()+
    facet_grid(rows = "location")
```



We can see that based on the overall bloom data without taking into consideration the temperature, the bloom day of year is roughly normal. There appears to be the start of a bi-modal trend, which is likely due to the recent rise in global temperatures, a new mean is appearing from this new trend.

While not completely perfect, it appears that the model specified in the introduction could serve its purpose in predicting the bloom dates of the year 2024.

Specifiying the model

The model can thus be specified as such:

$Y = _0 + _1X_1 + 2 + … + {13} $

Where, β_0 represents the intersect, $\beta_1 - \beta_2$ represents the slope based on the species as dummy variables $X_1 - X_2$, and each following β and X pairs represent the slope based on each 5 day average temperature and the temperature itself respectively. Priors for each of these parameters have to be specified along with the parameter for the standard deviation σ.

Based on observations of the correlation plots, it appears reasonable to assume the following priors:

- $\beta_0 \sim N(105, 10)$

- $\beta_3 - \beta_1 2 \sim N(-0.2, 20)$
- $\sigma \sim \frac{1}{10}$

This model was specified using the **rstan** package.

```
set.seed(1)
model = stan_glm(
  bloom_doy ~ species + temp_doy_1_5 + temp_doy_6_10 + temp_doy_11_15 + temp_doy_16_20 + t
  data = df_combined, family = "gaussian",
  prior_intercept = normal(102, 10),
  prior = normal(-0.2, 20, autoscale = TRUE),
  prior_aux = exponential(1/10, autoscale = TRUE),
  chains = 4, iter = 5000*2)
```

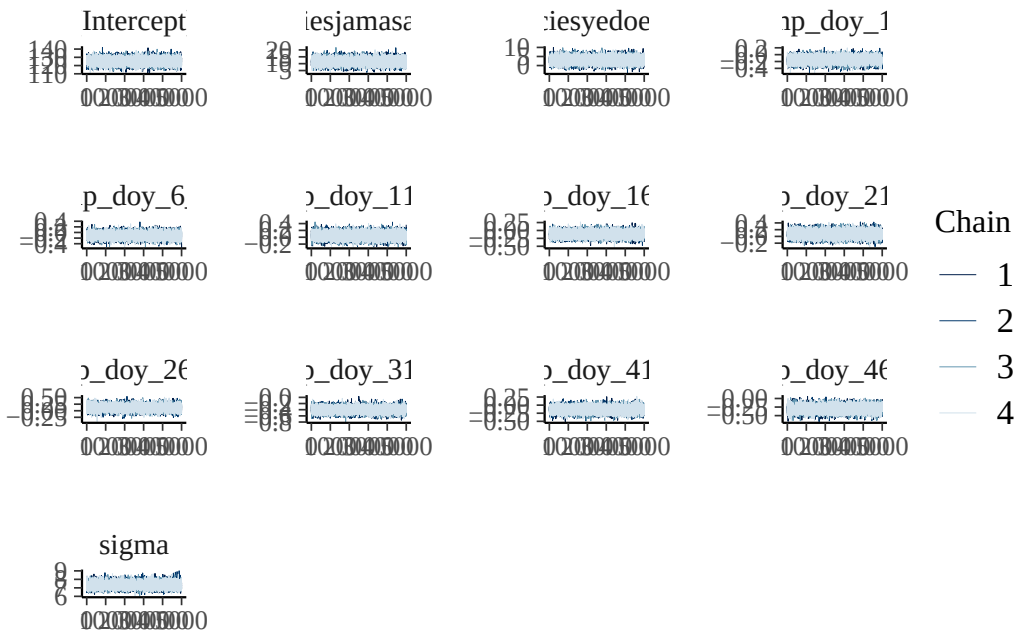
```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.576 seconds (Warm-up)
Chain 1:                0.594 seconds (Sampling)
Chain 1:                1.17 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
```

```
Chain 2: Gradient evaluation took 0 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 2: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 2: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 2: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 2: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 2: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 2: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 2: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 2: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 2: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 2: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 2: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.561 seconds (Warm-up)
Chain 2:                0.612 seconds (Sampling)
Chain 2:                1.173 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 3: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 3: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 3: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 3: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 3: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 3: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 3: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 3: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 3: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 3: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 3: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 3:
```

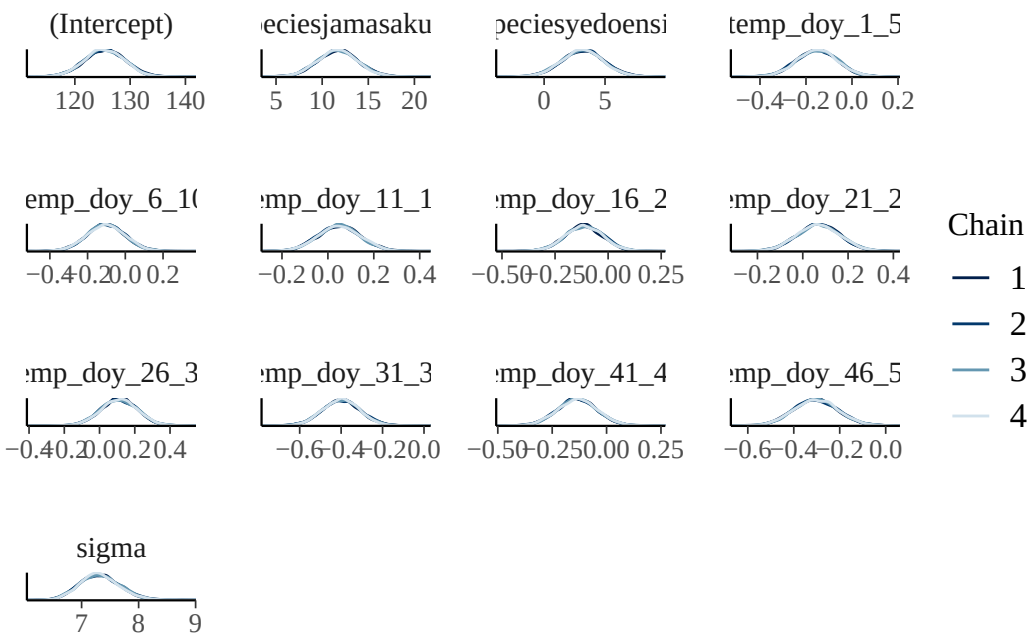
```
Chain 3:  Elapsed Time: 0.559 seconds (Warm-up)
Chain 3:                0.54 seconds (Sampling)
Chain 3:                1.099 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 4: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 4: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 4: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 4: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 4: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 4: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 4: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 4: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 4: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 4: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.54 seconds (Warm-up)
Chain 4:                0.605 seconds (Sampling)
Chain 4:                1.145 seconds (Total)
Chain 4:
```

To check if the simulations stabilized we run the respective trace plots, density plots, and autocorrelation plots.

```r
mcmc_trace(model, size = 0.1)
```

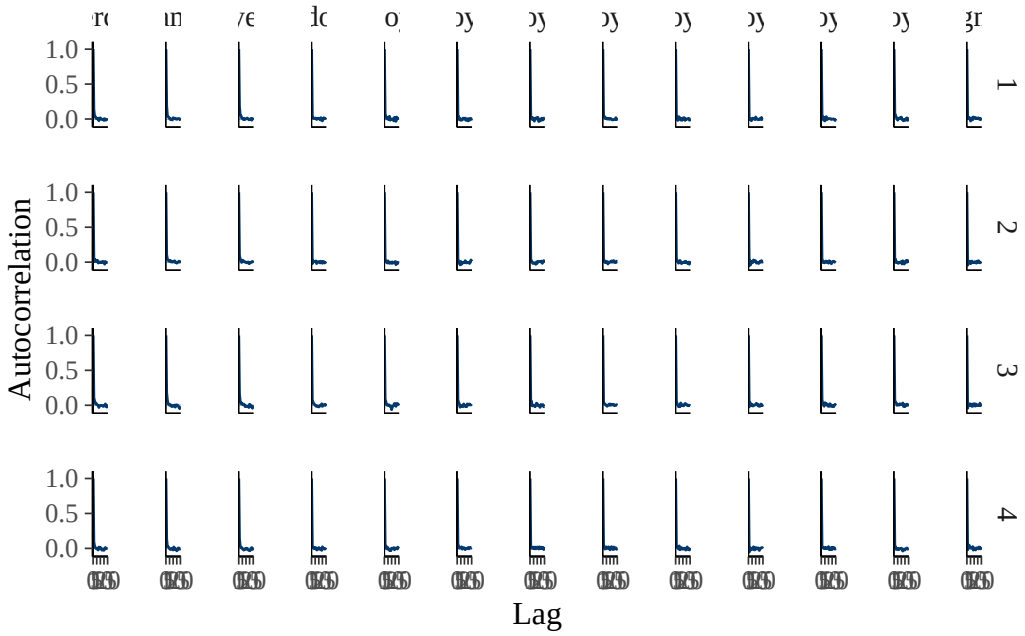



```
mcmc_dens_overlay(model)
```



```
mcmc_acf(model)
```

```
Warning: The `facets` argument of `facet_grid()` is deprecated as of ggplot2 2.2.0.
i Please use the `rows` argument instead.
i The deprecated feature was likely used in the bayesplot package.
  Please report the issue at <https://github.com/stan-dev/bayesplot/issues/>.
```



All three results are promising in terms of the stability of the MCMC chain.

Predictions

We are now ready to make our predictions. Based on the posterior distribution of our model, we can simulate new data based on the predictors, that is the five day average temperature of the first 50 days of this year, along with the species for the five given locations.

```
get2024_temp = function(stationid, loc, tavg = FALSE){

  if(tavg == FALSE){
  temp_data = filter(temp, STATION == stationid)%>%
```

```
  mutate(DATE = as.Date(DATE, format = "%d/%m/%Y"),
         TMAX = na.approx(TMAX),
         TMIN = na.approx(TMIN))%>%
  transmute(year = parse_number(format(DATE, "%Y")),
            date = DATE,
            temp = (TMAX + TMIN) / 2)
  }else{temp_data = filter(temp, STATION == stationid)%>%
  mutate(DATE = as.Date(DATE, format = "%d/%m/%Y"),
         TAVG = na.approx(TAVG))%>%
  transmute(year = parse_number(format(DATE, "%Y")),
            date = DATE,
            temp = TAVG)}

temps_df = get_temp_day(2024, temp_data)
names(temps_df)= c(paste0("temp_doy_", 1:50))

return(temps_df)
}


vancouver = get2024_temp(stationid = "CA001108395", tavg = TRUE)
dc = get2024_temp(stationid = "USC00186350")
#extrapolate missing data for dc
dc[50] = dc[49]
kyoto = get2024_temp(stationid = "JA000047759")
liestal = get2024_temp(stationid = "Liestal", tavg = TRUE)
nyc = get2024_temp(stationid ="USW00094728" )

predict_set = rbind(dc, liestal, kyoto, vancouver, nyc)

predict_set = predict_set %>%
  mutate(temp_doy_1_5 = rowMeans(cbind(predict_set$temp_doy_1,
                                       predict_set$temp_doy_2,
                                       predict_set$temp_doy_3,
                                       predict_set$temp_doy_4,
                                       predict_set$temp_doy_5)),

         temp_doy_6_10 = rowMeans(cbind(predict_set$temp_doy_6,
                                        predict_set$temp_doy_7,
                                        predict_set$temp_doy_8,
                                        predict_set$temp_doy_9,
```

```
                                predict_set$temp_doy_10)),

    temp_doy_11_15 = rowMeans(cbind(predict_set$temp_doy_11,
                                    predict_set$temp_doy_12,
                                    predict_set$temp_doy_13,
                                    predict_set$temp_doy_14,
                                    predict_set$temp_doy_15)),

    temp_doy_16_20 = rowMeans(cbind(predict_set$temp_doy_16,
                                    predict_set$temp_doy_17,
                                    predict_set$temp_doy_18,
                                    predict_set$temp_doy_19,
                                    predict_set$temp_doy_20)),

    temp_doy_21_25 = rowMeans(cbind(predict_set$temp_doy_21,
                                    predict_set$temp_doy_22,
                                    predict_set$temp_doy_23,
                                    predict_set$temp_doy_24,
                                    predict_set$temp_doy_25)),

    temp_doy_26_30 = rowMeans(cbind(predict_set$temp_doy_26,
                                    predict_set$temp_doy_27,
                                    predict_set$temp_doy_28,
                                    predict_set$temp_doy_29,
                                    predict_set$temp_doy_30)),

    temp_doy_31_35 = rowMeans(cbind(predict_set$temp_doy_31,
                                    predict_set$temp_doy_32,
                                    predict_set$temp_doy_33,
                                    predict_set$temp_doy_34,
                                    predict_set$temp_doy_35)),

    temp_doy_36_40 = rowMeans(cbind(predict_set$temp_doy_36,
                                    predict_set$temp_doy_37,
                                    predict_set$temp_doy_38,
                                    predict_set$temp_doy_39,
                                    predict_set$temp_doy_40)),

    temp_doy_41_45 = rowMeans(cbind(predict_set$temp_doy_41,
                                    predict_set$temp_doy_42,
                                    predict_set$temp_doy_43,
```

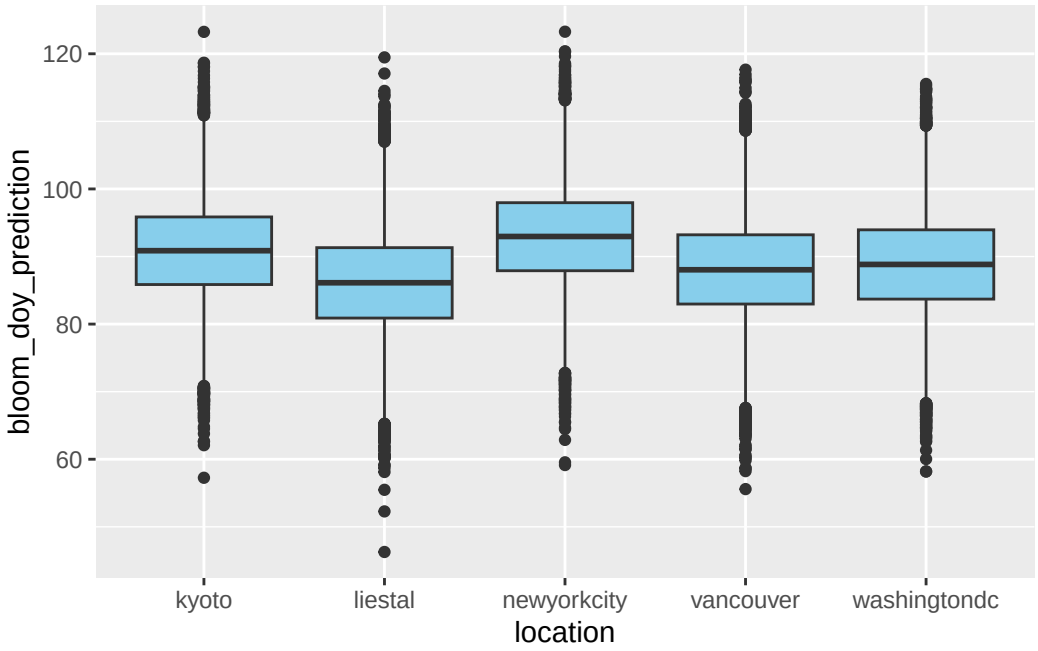
```
                                        predict_set$temp_doy_44,
                                        predict_set$temp_doy_45)),

          temp_doy_46_50 = rowMeans(cbind(predict_set$temp_doy_46,
                                        predict_set$temp_doy_47,
                                        predict_set$temp_doy_48,
                                        predict_set$temp_doy_49,
                                        predict_set$temp_doy_50)))

species = c("yedoensis", "avium", "jamasakura", "yedoensis", "yedoensis")
predict_set = cbind(species, predict_set)

set.seed(1)
prediction = posterior_predict(
  model,
  newdata = predict_set
)

location = c("washingtondc", "liestal", "kyoto", "vancouver", "newyorkcity")

cherry_predictions = as.data.frame(cbind(location, round(colMeans(prediction)), round(colM

names(cherry_predictions) = c("location", "prediction", "lower", "upper")

write.csv(cherry_predictions, "cherry-predictions.csv", row.names = FALSE)

print(cherry_predictions)
```

```
      location prediction lower upper
1 washingtondc         89    73   104
2      liestal         86    71   102
3        kyoto         91    76   106
4    vancouver         88    73   103
5  newyorkcity         93    78   108
```

We can also visualize this prediction:

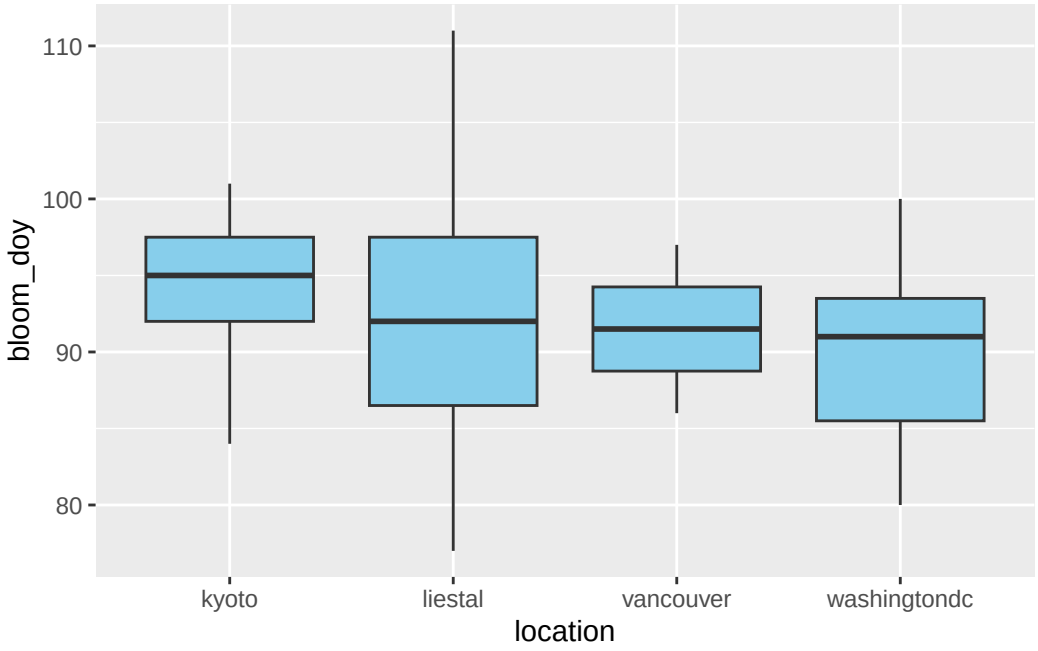
```
prediction_df = prediction %>%
  as.data.frame()

names(prediction_df) = location
```

```r
prediction_df_long = pivot_longer(prediction_df, cols = c(1:5), names_to = "location", val

ggplot(prediction_df_long, aes(y = bloom_doy_prediction, x = location))+
  geom_boxplot(fill = "skyblue")
```



Comparing this result to the previous years data, the model predicts a much earlier bloom day than usual. However, it is reflective of the bloom dates of more recent years as shown below:

```r
bloom_df %>%
  filter(year>2000)%>%
  ggplot(aes(y = bloom_doy, x= location))+
  geom_boxplot(fill = "skyblue")
```



Conclusions

There are of course many improvements that can be made to this model such as the use of hierarchical regression modelling, and taking into account other predictors such as humidity, rainfall, etc. However, the use of a regression model with only a few predictors permits a compromise between simplcity and complexity.