Predicting Cherry Blossom Bloom Dates

Joshua Write and Taehoon Song

2/26/2022

Introduction

In this document, we demonstrate how we went about predicting the peak bloom dates in the coming decade for all four locations required by the competition. We use different modeling approaches due to different available data.

```
# Loading in packages and data
library(tidyverse)
library(forecast)
library(lubridate)
library(astsa)
library(here)
library(rnoaa)
library(gridExtra)

cherry <- read_csv(here('peak-bloom-prediction','data','washingtondc.csv')) %>%
  bind_rows(read_csv(here('peak-bloom-prediction','data','liestal.csv'))) %>%
  bind_rows(read_csv(here('peak-bloom-prediction','data','kyoto.csv'))) %>%
  select(location,year,bloom_date,bloom_doy)
```

Predicting peak bloom date for DC

In the field of phenology, plant growth is often modeled by using growing degree-day (GDD) as a predictor. Since the National Park Service(NPS) publishes dates of various growth stages for 2004 through 2021, we use this to build a growth model that helps us estimate the threshold GDD for which we can expect the plant to bloom. Details can be found in Phenological Models of Flower Bud Stages and Fruit Growth of 'Montmorency' Sour Cherry Based on Growing Degree-day Accumulation by C. Zavalloni, J. Adresen, and J. Flore. The authors use 4 degrees Celsius as the base for calculating GDD and fit a logistic function to the phenological stages from side green as a function of acculumated GDD.

$$y = \frac{k}{1 + \left[\frac{k-n_0}{n_0} \cdot \exp(-r \cdot \text{GDD})\right]}$$

where y is the phenological stage and k and r are empirical factors related to growth rate. Now, with just enough knowledge to be dangerous, we attempt to use this model to fit the growth stages published by the NPS as a function of cumulative GDD. Note that the authors use $n_0 = 2$ because they were interested in phenological stages after side green (Stage 2). To be consistent with the paper's authors, we label the phenological stages published by the NPS in a similar fashion.

Obtaining temperature and phenological data for DC

Since the species in the paper is different from what we're interested in, we must first identify the typical day of year when the cherry blossom trees in DC reach the green buds stage (Stage 3, as defined by the authors).

```
# Import historical weather data from DayMet
dc_tmin <- read_csv(here('peak-bloom-prediction','data','statistics_tmin.csv'))
dc_tmax <- read_csv(here('peak-bloom-prediction','data','statistics_tmax.csv'))

# DayMet data stops at 2020. Use rnoaa for 2021.
# Original code from demo analysis
# https://competition.statistics.gmu.edu/wp-content/uploads/2022/02/demo_analysis.html
getTemperature <- function (stationid,date_min,date_max) {
  tbls <- ghcnd_search(stationid = stationid, var = c("tmax","tmin"),
               date_min = date_min, date_max = date_max)
  tmax <- tbls[[1]] %>% select(id,tmax,date)
  tmin <- tbls[[2]] %>% select(tmin,date)
  temp <- inner_join(tmax,tmin,by="date") %>%
    group_by(date) %>%
    # Calculate average and convert unit from tenth degree Celsius to regular Celsius
    summarize(id,tavg=0.5*sum(tmax,tmin,na.rm=TRUE)/10) %>%
    ungroup() %>%
    mutate(year = year(date))
  return(temp)
}

dc_noaa <- getTemperature("USC00186350","2021-01-01","2022-01-31") %>%
  select(-id)

# Temperature data
dc_temp <- left_join(dc_tmin,dc_tmax, by = 'dt') %>%
  select(dt,value_mean.x,value_mean.y) %>%
  rename(tmin=value_mean.x,tmax=value_mean.y) %>%
  summarize(date=dt,tavg=0.5*(tmin+tmax), year=year(date)) %>%
  bind_rows(dc_noaa)

# Phenological stage data
dc_bloomdata <- cherry %>% filter(location=='washingtondc') %>%
               summarize(date=bloom_date,bloom_doy) %>%
               summarize(year=year(date),FullBloom=bloom_doy)

phenostage <- read_csv(here('peak-bloom-prediction','data',
                            'Phenological_Stage.csv')) %>%
  inner_join(dc_bloomdata,by='year')

summary(phenostage$GreenBuds)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   50.00   59.00   64.00   63.94   69.50   77.00
```

This result suggests that Stage 3 occurs as early as 50 days from the start of the year. Thus, we shift the dates by 49 days to start calculating GDD to ensure that we start calculating GDD as soon as Stage 3 occurs. We also set $n_0 = 3$ because we started from Stage 3 instead of 2.

Building the growth model

As mentioned above, we calculate the GDD using a base of 4 degrees Celsius and starting from 49 days from the beginning of the year. Then, we fit the known phenological stages for the cherry blossoms in DC to the calculated GDDs.

```r
thresh <- 4
GDD_final <- dc_temp %>%
  mutate(NewYear=year(date %m-% period("49 day"))) %>%
  mutate(DD = ifelse(tavg - thresh > 0, tavg - thresh, 0)) %>%
  group_by(NewYear) %>%
  mutate(GDD = cumsum(DD),doy = yday(date)) %>%
  ungroup()

# obtain GDD for each phenological stage for DC
stages <- tibble(stage=names(phenostage)[2:7],stagenum=seq(3,8))
pheno_GDD <- phenostage %>%
  select(-GreenBuds) %>% #This stage is the starting point for our model.
  pivot_longer(cols=FloretsVisible:FullBloom,names_to = "stage",values_to="doy") %>%
  left_join(GDD_final,by=c("year","doy")) %>%
  summarize(year,stage,doy,GDD) %>%
  left_join(stages,by='stage')

# Fit growth model
n0 <- 3 #Starting phenological stage for our model
growth_model <- nls(stagenum ~ k/(1+(k-n0)/n0*exp(-r*GDD)),
                    data=pheno_GDD,start=list(k=1,r=0.01))
summary(growth_model)
```

```
##
## Formula: stagenum ~ k/(1 + (k - n0)/n0 * exp(-r * GDD))
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## k 10.223870   1.362473   7.504 4.72e-11 ***
## r  0.011973   0.001598   7.495 4.93e-11 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.9374 on 88 degrees of freedom
##
## Number of iterations to convergence: 8
## Achieved convergence tolerance: 7.797e-06
```

The model results show that both k and r are significant. However, inference is not of interest to us. We only care about prediction performance.

Checking performance on historical data

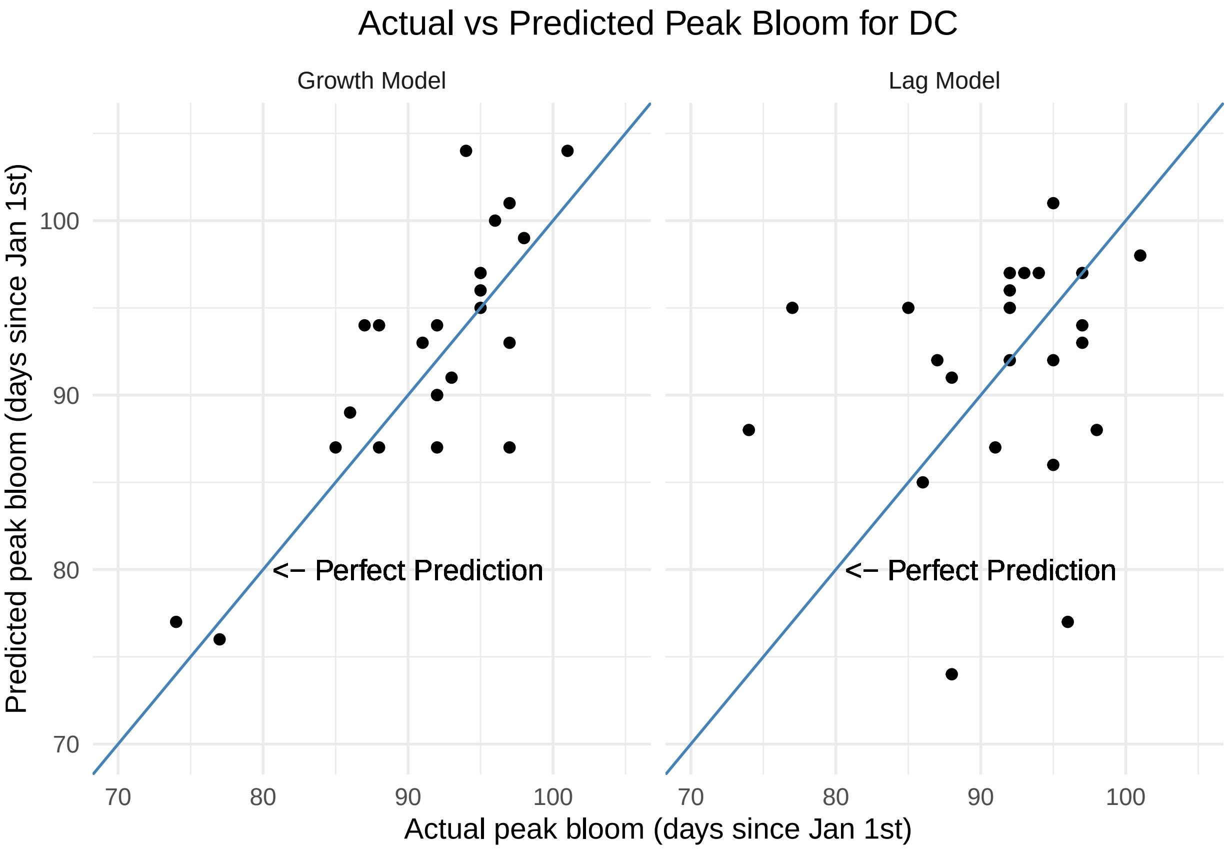
Now that we have the growth model, we can test its prediction performance out on historical data that we did not use.

```
testdata <- GDD_final %>% filter(year<=2003) %>% #Used data from 2004 onward
  group_by(year) %>%
  filter(doy>=50 & doy <= 200) %>%
  ungroup() %>%
  select(date,GDD)

preds <- tibble(date=testdata$date, GDD=testdata$GDD,
                pred=predict(growth_model,newdata=testdata)) %>%
  mutate(year=year(date))

# Predicted date is the earliest date when predicted stage rounds to Stage 8
# Compare predicted date to actual historical dates
comp <- preds %>% group_by(year) %>%
  filter(round(pred) >= 8) %>%
  filter(row_number()==1) %>%
  ungroup() %>%
  mutate(pred.doy=yday(date)) %>%
  inner_join(dc_bloomdata,by='year') %>%
  summarize(year,GDD,pred.doy,act.doy=FullBloom,err_GDD=abs(pred.doy-act.doy)) %>%
  # compare to using simply using a lag predictor
  mutate(lag_bloom = lag(act.doy),err_lag=abs(act.doy - lag_bloom))

data.frame(MAE_gdd=mean(comp$err_GDD[which(!is.na(comp$err_lag))]),
       MAE_lag=mean(comp$err_lag,na.rm=TRUE))
```

```
##    MAE_gdd  MAE_lag
## 1 3.347826 6.304348
```

Our growth model has a much lower mean absolute error (MAE) compared to simply using the previous year's bloom date as a predictor. This is good because this is the bare minimum a useful model should be able to do. This is more apparent when we plot the two models against the actual peak bloom days since January 1st.

```
plt_data <- comp %>% rename(Actual=act.doy,GrowthModel=pred.doy,LagModel=lag_bloom) %>%
  pivot_longer(cols=c("GrowthModel","LagModel"),names_to="type",values_to="doy") %>%
  filter(!is.na(err_lag))
plt_data$type <- factor(plt_data$type,levels=c("GrowthModel","LagModel"),
                        labels=c("Growth Model","Lag Model"))

# Plot Actual vs Predicted for Growth Model
ggplot(plt_data,aes(x=Actual)) + geom_point(aes(y=doy)) +
  facet_grid(~type) +
  geom_abline(slope=1,col="steelblue") +
  geom_text(aes(x=90,y=80,label="<- Perfect Prediction")) +
  theme_minimal() +
  ggtitle("Actual vs Predicted Peak Bloom for DC") +
  xlab("Actual peak bloom (days since Jan 1st)") +
  ylab("Predicted peak bloom (days since Jan 1st)") +
  xlim(c(70,105)) + ylim(c(70,105)) +
  theme(plot.title=element_text(hjust=0.5))
```



Obtaining historical weather data

Using the function shown in the Appendix of the Demo Analysis for the competition, we obtain historical weather data for the four locations.

```
# library(rnoaa)
#
# locs <- c("USC00186350","GME00127786","JA000047759","CA001108395")
# city <- c("DC","Liestal","Kyoto","Vancouver")
#
# # Original code from demo analysis
# # https://competition.statistics.gmu.edu/wp-content/uploads/2022/02/demo_analysis.html
# get_temperature <- function (stationid) {
#   tbls <- ghcnd_search(stationid = stationid, var = c("tmax","tmin"),
#                date_min = "2017-01-01", date_max = "2022-01-31")
#   tmax <- tbls[[1]] %>% select(id,tmax,date)
#   tmin <- tbls[[2]] %>% select(tmin,date)
#   temp <- inner_join(tmax,tmin,by="date") %>%
#     group_by(date) %>%
#     # Calculate average and convert unit from 0.1 Celsius to 1 Celsius
#     summarize(id,tavg=sum(tmax,tmin,na.rm=TRUE)/20) %>%
#     ungroup() %>%
#     mutate(year = year(date))
#   return(temp)
# }
```

```
#
# # Obtain historical data
# temp_list <- lapply(locs,get_temperature)
#
# # ghcnd_search returns a list. Extract and append each element to create one dataset.
# temps <- tibble()
# for(i in 1:length(temp_list)){
#   temps <- bind_rows(temps,temp_list[[i]])
# }
#
# # Add labels to station IDs
# loclabels <- as_tibble(cbind(locs,city))
# temps <- temps %>% left_join(loclabels,by=c("id"="locs")) %>% select(-id)
```

The resulting temperature data looks like the following: