Modeling Runners' Times in the Cherry Blossom Race

Gupta, Moro, Kannan

1. Business Understanding

The "Cherry Blossom Ten Mile Run & 5K Run-Walk" is an annual race conducted in Washington, D.C since 1973. It has grown in popularity in recent years with an increase in local and international participation. However, with growing participation comes the need for better planning, for example, traffic management, parking, public transportation to and from the race event, medical facilities, hospitality, security, and other amenities. It is important for the race planning committee to design the right plan of action for future races taking these factors into account.

The goal of this study is to provide a recommendation to the planning committee for potential improvements that can be made to future races based on historical trends. We focus on the data for the 10 mile women's division run from 1999 to 2012. These insights may help the committee to reorganize race divisions and race start times in order to conduct the race more effectively as well as provide better amenities.

2. Data Extraction Prep

Race records are accessible using the web page of the Cherry Blossom race [1]. The site allows two different methods to access historical race times. The first method allows us to use fixed weblinks to access the race data for each year (one link per year). The second method provides an interactive webpage which queries the online race database [2]. None of these methods have the option to download the records in a data friendly format like CSV or JSON, but it is possible to use a web crawler to download the records. The first method is a little more cumbersome since the data downloaded is in text format and it needs to be reformatted into a tabular format. Since the text format is not consistent from year to year, this would need custom code to be written based on the year. The second method provides a more consistent data format from year to year. Hence, we choose this method to download the data using code that leverages the interactive pages to directly query the race database. This methodology is akin to modern methodologies, such as API, that allow for transfer of data via web, as it utilizes a fixed query structure to obtain data consistently from the web.

While this approach is very structured and uses a short and clean coding structure, we need to be aware of some constrains and issues. The automated process is very slow as each web query returns a maximum of 20 records. Hence, more than 3,800 web queries are needed to download the entire data set of 75,866 records. The total run time takes more than 30 minutes using on a remote machine via internet. To mitigate this problem, we created parallel processes to query multiple years at the same time. This reduced the process time to 4 minutes using a medium-sized machine (8-cores). If we have access to a more powerful machine, then this processing time can be reduced further. Another method to reduce the dependency from the web queries, is to save the downloaded data in a local file that can be used for any future analysis. This option is viable after the initial download as the historical data is not expected to change in the future.

Even if the Cherry Blossom Race site allows users to query the race records, it does not mean users are authorized to download and use them. If this aspect is not considered, it may lead to legal issues in the future. Unfortunately, the Cherry Blossom race website doesn't provide any information on the legality or limitation on using the data. Another method to check if web crawling is allowed from the site is to search for the `robots.txt` file. This file is the standard method used by webpages to inform automated processes (crawlers) about limitations and directions on how to crawl the site [3]. From the `robots.txt` file from the Chery Blossom race website [4], we obtain the message depicted in Fig. 1. This file does not restrict web crawling on the website, as the tag `Disallow` is commented (`#`). This means we can download the data from the race website without any legal issues. Even if we did not find any explicit limitation, we suggest contacting the legal representatives of the race to validate that we are acting within their guidelines.

```
# See http://www.robotstxt.org/robotstxt.html for documentation on how to use the robots.txt file
#
# To ban all spiders from the entire site uncomment the next two lines:
# User-agent: *
# Disallow: /
```

Fig. 1 robots.txt file for Cherry Blossom Race website

3. Data Extraction (Execution)

3.1 Extract the raw results for all 14-years

The interactive page that queries the race database is accessible through the Cherry Blossom race webpage [2]. The page displays four fields the user can use to set the query parameters. The parameters are then used to create a web query that returns the records stored in the database. The database query is structured as follow:

```
http://www.cballtimeresults.org/performances?utf8=%E2%9C%93&section=[section_name]&page=[page_number]&
year=[year]&division=[division_name]
```

The parameters can be used as follow:

`[section_name]`	A string that filters data for the 5K or 10K events, for the objective of this case we are going to use the `10K` string
`[year]`	The year of the selected race, for this case we are going to use the range of values from `1999` to `2012`
`[division_name]`	Accepts the values: `Overall+Women` or `Overall+Man`. For this case we will use the women option
`[page_number]`	The number of the page that will be returned. Each page contains 20 records

The web address is created using the `getLink` function [7.1]. This returns a new web page containing a table with 20 records. The table can be identified in the page using the HTML tag **`<table>`**. This is obtained using the `getTable` function [7.2]. If there are no records in the selected page, an empty table is returned. As the objective of this data extraction is to parse all the records from year 1999 to 2012 for women in the 10K run, we need to create a set of nested loops to cycle through all the years and all the pages containing the records (Fig. 2).

1.	**for each** year from 1999 to 2012
2.	**for each** page number starting from 1
3.	**compute and send** the web query
4.	**identify** the data table using the tag **`<table>`** from the returned page
5.	**parse** the records from table
6.	**exit** this loop if the table has no records

Fig. 2 Steps to download the data from the Cherry Blossom Website

This process is performed using the **`getAllTables`** function [7.3]. The output of this steps is a data frame containing 75,866 records and 22 columns. We also add metadata to this table such as link to page and query parameters which allows us to debug any potential issues later.

3.2 Clean-up the raw results

Once the raw table has been obtained, we proceed with cleaning the data and obtaining the columns needed for the analysis, in the right format. This process is performed using the **`transformAllTables`** function [7.4]. The data returned is very clean and needed only minor transformations as described below.

The **`Name`** field from the raw data contains the name of the athlete and the gender in parenthesis. We converted this field in two separate fields – "name" and "gender". The **`PiS/TiS`** and **`PiD/TiD`** fields from the raw data contain both position of the athlete and total athletes, for their gender and division groups. These fields have been split in new numeric columns containing the position of the athlete and total participants. The **`Hometown`** field containing the town and the state has been split in "Hometown" and "HomeState" fields. When the athlete is from a non-USA state the country is indicated in the `HomeState` field. To correct this behavior, a new field **`HomeCountry`** is populated with text 'USA' when the `HomeState` field contains an USA state, otherwise it is populated with the non-USA country indicated in the `HomeState` field. The **`Time`** and **`Pace`** fields have been parsed as time format, and then converted to number of minutes. The **`Age`** field has been converted to numeric format (originally parsed as string). The resulting data frame contains the following columns:

Column	Format	Description
Race	Text	The race code, `10M` for this case, obtained from the query parameter
Name	Text	Name of the athlete
Gender	Text	Gender of the athlete, `W` for this case
Age	Numeric, integer	Age of the athlete
Time	Time	Time of athlete run
Pace	Time	One mile time
PiS	Numeric, integer	Position of the athlete within its section (Women)
TiS	Numeric, integer	Total athletes in the section (Women)
Division	Text	Division Code
PiD	Numeric, integer	Position of the athlete within its division
TiD	Numeric, integer	Total athletes in the division
Hometown	Text	Athlete hometown
HomeState	Text	Athlete home State
HomeCountry	Text	Athlete home Country
Year	Numeric, integer	Year of the race (query parameter)
divisionTitle	Text	Title of the division (query parameter)
Page	Numeric, integer	Database page where the record originated
Link	Text	Link to the database page where the record originated
TimeMins	Numeric, decimal	Time of the athlete race in minutes
PaceMins	Numeric, decimal	One mile time of the athlete, in minutes.

3.3 Missing Data Analysis

Fig. 3 shows the cross map of missing values in the data. These values, that show up as "NR" (Not Reported) in the dataset, are cast to "NA" values. The total number of 'NR' data for each column in the dataset is shown in the chart along with the correlation in their missing pattern. For instance, the fifth bar from left shows the number of rows in the dataset with 'NR' for **`Division`**. We could also see a pattern in which, whenever **`Division`** is 'NR', the associated values of **`Age, Pid,`** and **`Tid`** were also 'NR'.

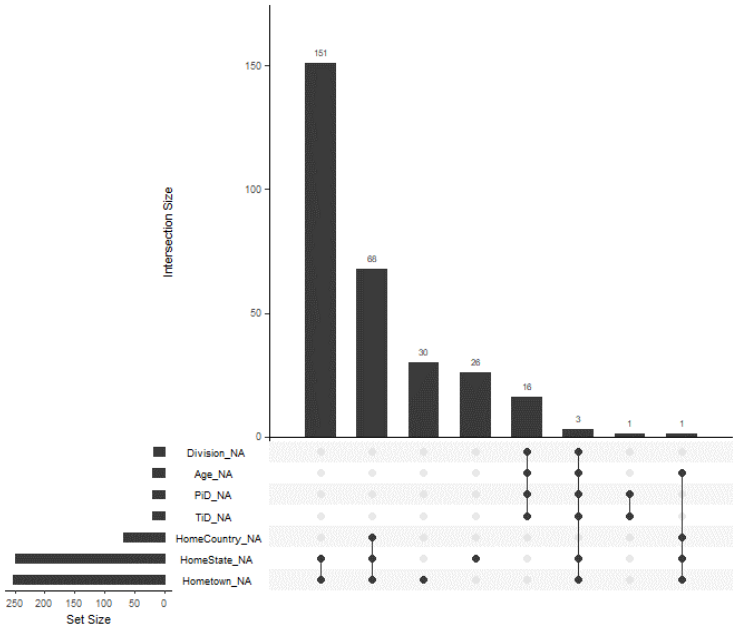

Fig. 3 Cross map of missing values in the data

For our exploratory data analysis, we focused on four fields that had 'NR' as an entry: **`Age, PiD, TiD,`** and **`Division`**. There are 21 records with 'NR' values across these four fields. This is consistent with the data in the online database and is probably related to missing entries. Without these 21 records, the dataset will have a total of 75,845 records which is only minute reduction from the original dataset. Hence, we are proceeding with the removal of these records. In addition, we decided to retain the observations with the missing **`HomeState`** information as we did not have a good way to impute these and we did not want to discard these race times.

3.4 Exploratory Analysis

Number of Participants

Fig. 4 shows the number of women participants over the years (Appendix 7.5). There has been a steady increase in the participation over the years (participation has increased more than 4x from 1999 to 2012) and this is something that the organizers should be aware of as it can lead to race logistics and management issues.

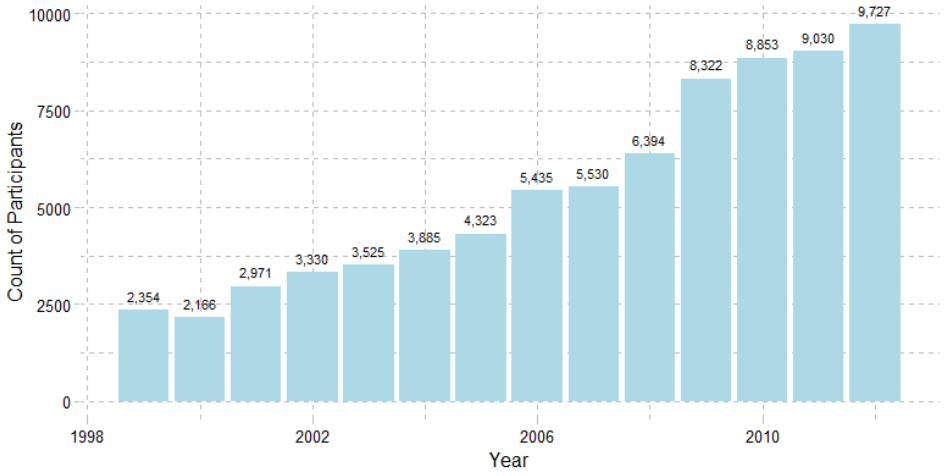

Fig. 4 Number of Participants by Year

Out of State Participants

Fig. 5 shows the "Out of State" participants over the years (Appendix 7.6). "Out of State" in this context is defined as participants whose **`HomeState`** is other than Washington DC, and its two neighboring states Maryland and Virginia. We felt that it was appropriate to include the two neighboring states since Washington DC is very small in size and people often commute from these two neighboring states to Washington DC daily. We can observe that the number of "Out of State" participants is increasing over time (increased more than 6x from 1999 to 2012) and it seems that the upward trend is accelerating recently. Again, this is noteworthy since the organizers may need to take this into account since the race location needs to have the appropriate hotel vacancies to accommodate these "Out of State" participants.

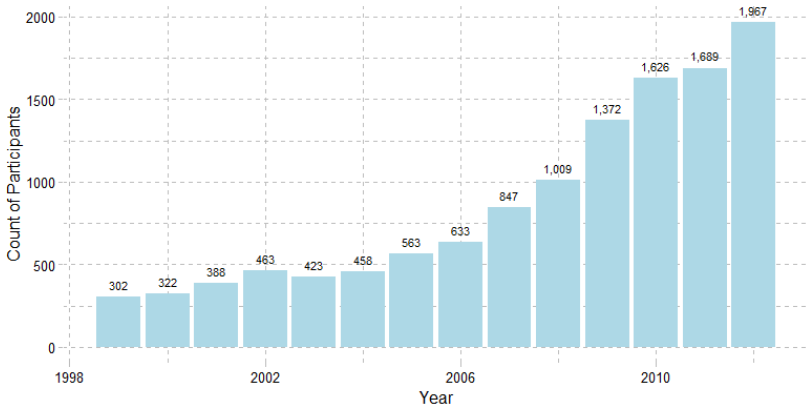

Fig. 5 Number of Out of State Participants over the years

4. Business Analysis

4.1 Age Profile of Runners

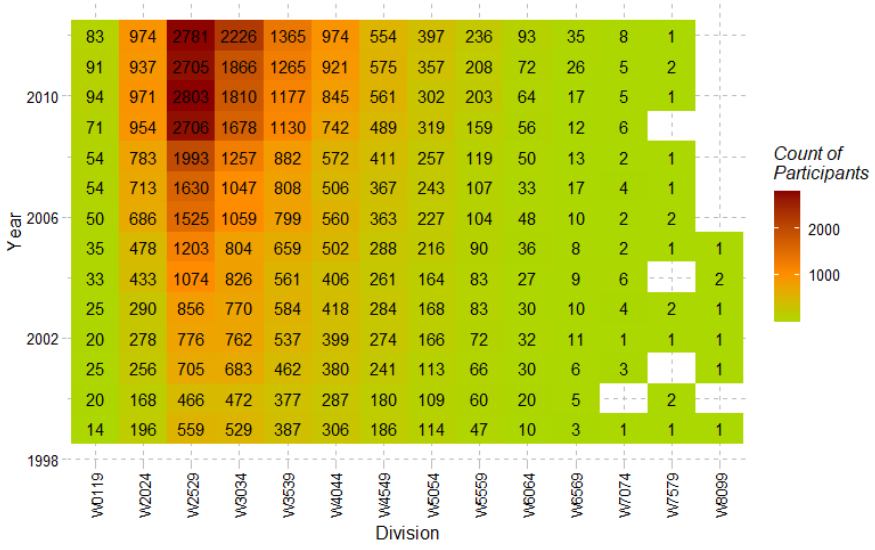

Fig. 6 Number of Participants by Age Bracket and Year

Fig. 6 shows the number of participants in each age bracket by year (Appendix 7.7). From this view, it seems that while the number of participants has increased across the board in almost all age brackets, the largest increase is in the W2024 – W4044 brackets. This can be an issue from a race management point of view, and we will evaluate this further in this report later.

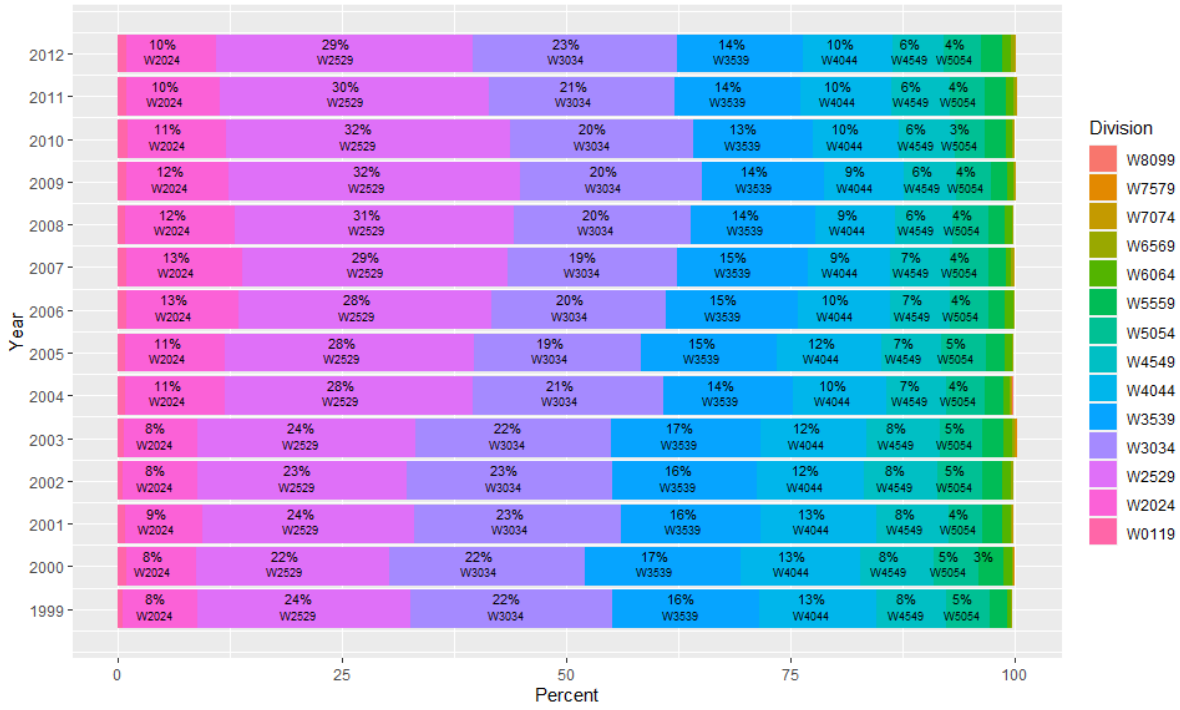

Fig. 7 Percentage of Participants by Age Bracket and Year

A complementary view to Fig. 6 is shown in Fig. 7 (Appendix 7.8). This figure shows the percentage of participants in each age bracket from 1999 to 2012. It can be seen that as the years progress, the percentage of participants from younger age brackets has increased (The cumulative percentage of runners from age bracket W0119 – W3034 has increased as per Fig. 7). The biggest percentage increase is seen in the W2529 bracket. This is however beginning to level off in the most recent years.

4.2 Race Times

Fig. 8 shows a boxplot of race times over the years (Appendix 7.9). There is a slight upward trend in the race times since 2003, but this is quite minimal and may not be statistically different between the years. However, this chart can be misleading since the race times can be very different between the race brackets (Fig. 9 generated using Appendix 7.10). Hence, we also plot the race times by age bracket over the years (Fig. 10) which is a more representative comparison (Appendix 7.11). Note that this figure excludes the four oldest age brackets since there were very few participants in these brackets. From Fig. 10, we see that while there is an increase in the median race time across brackets since 2003, some brackets (such as W3034, W3539, W4044) seem to have a more substantial rise in the race time compared to other brackets (such as W4549, W5054). It is also observed that the race times are levelling off in recent years.

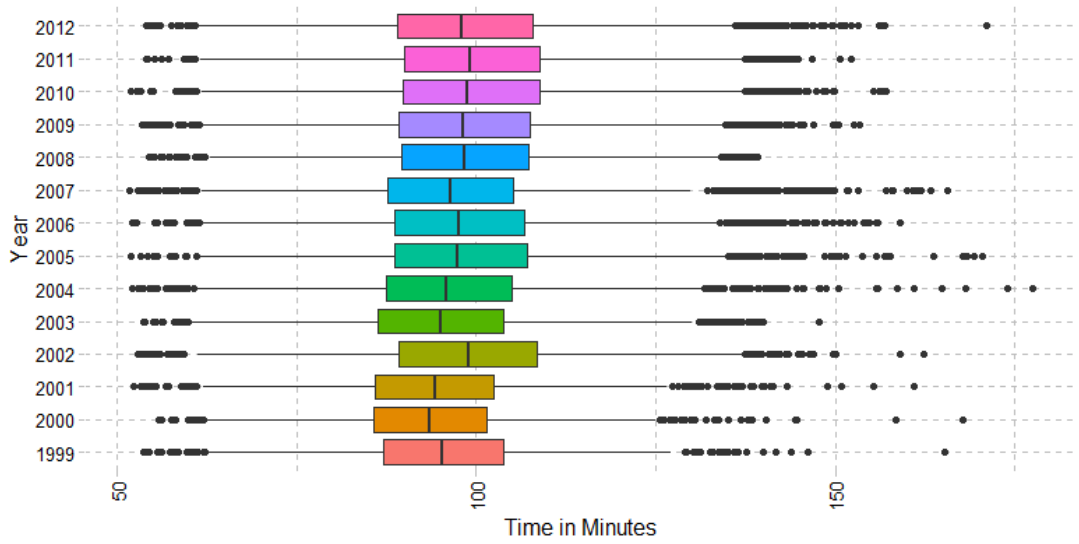

Fig. 8 Boxplot of Race Times per Year

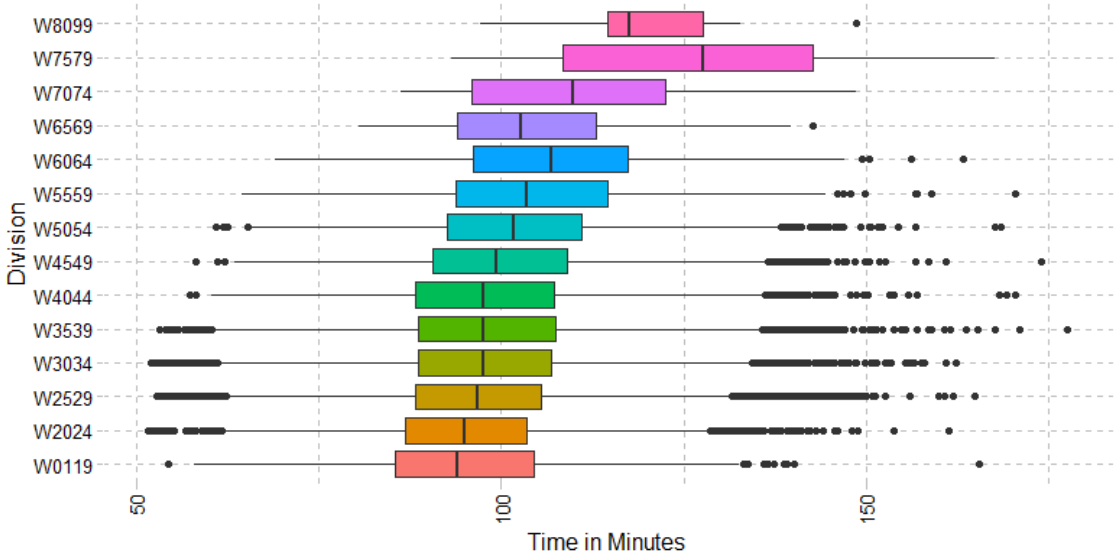

Fig. 9 Race Times by Age Bracket

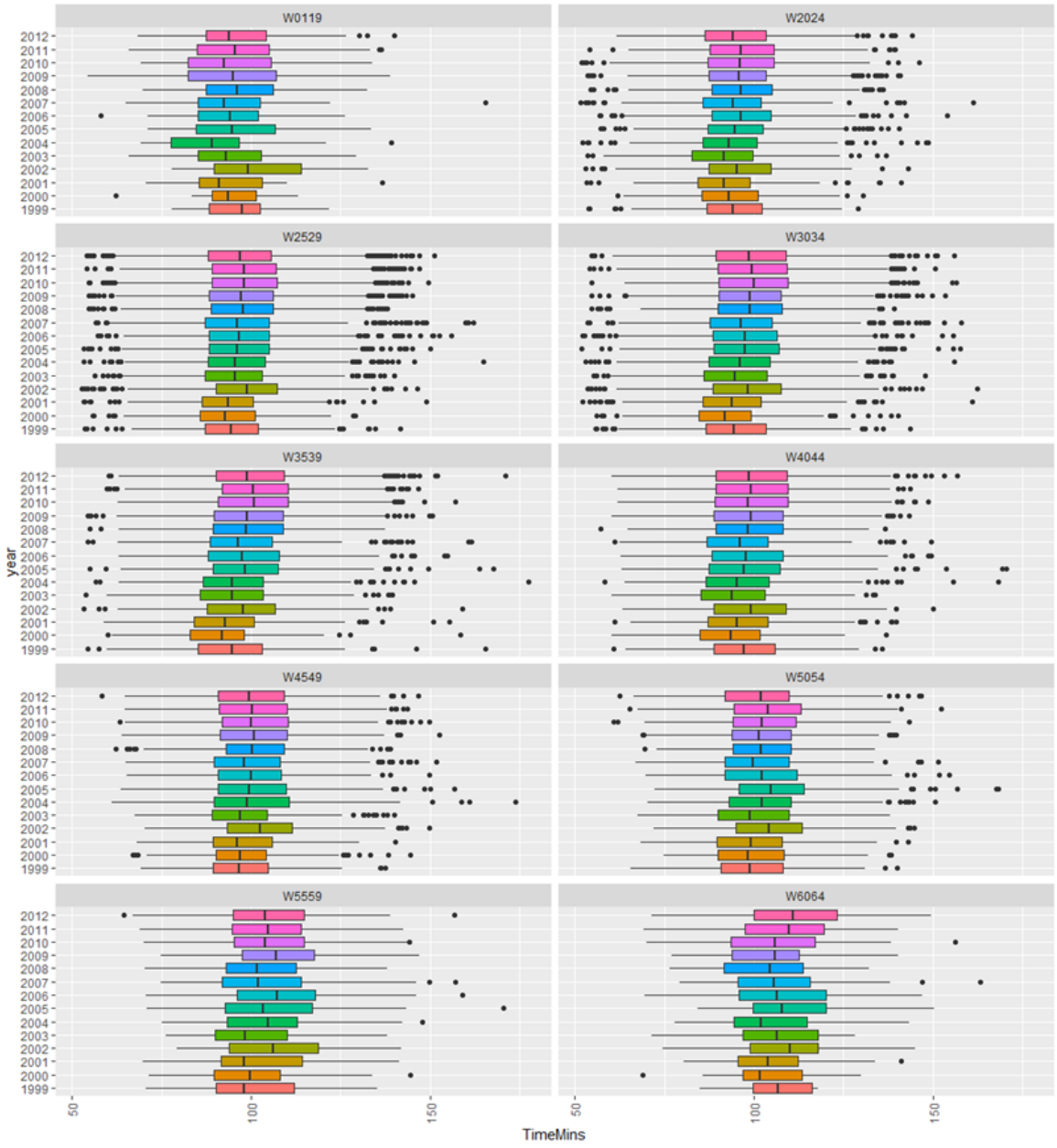

Fig. 10 Boxplot of Race Times per Year and Age Bracket

5. Conclusion

Based on our analysis, we recommend the following for the future races:

Splitting up Age Brackets

From our analysis, we noted that Age brackets W2529 and W3034 showed a significant increase in participation. Our recommendation would be to split up these brackets into 2 sub brackets each. This will ensure a more manageable number of participants in each bracket and less crowding on the racetrack.

Staggering Race Times

There has been an increase in the race times for almost all brackets but most notably the W3034, W3539, W4044 brackets. Hence, we recommend that the start time of especially these three brackets be staggered a little more than the other brackets. Again, this will lead to less crowding on the racetrack.

Facilities

There has been an overall participation in the race over the years and the trend shows no sign of levelling off. Increased participation also means that the organizers need to have more facilities available at the event. This includes more medical staff for emergencies, and police presence to manage the flow of traffic, crowd control, etc. This also means that more parking facilities need to be arranged with the city ahead of time and easy public transportation may need to be provided from the designated parking lots to and from the event. This public transport assistance can help with traffic and crowd management in and around the racetrack.

Accommodation

We also recognized increased "Out of State" participation over the years. The organizers should also consider partnering with local hotels and possibly provide block accommodation to these participants at a discounted price. This is likely to attract even more participation from this base in the future.

Younger Age Brackets

We also saw a shift in the demographics toward the younger age group. It may also benefit the race organizers to add post-race events such as concerts or party to cater to this base.

In conclusion, it is also important to know that this analysis is solely based on the 10 mile women's division run. In order to get a more complete picture, we would recommend repeating this exercise with the men's division as well along with the 5K Run/Walk.

6. References

[1] "Cherry Blossom Webpage," [Online]. Available: http://www.cherryblossom.org/.

[2] "Cherry Blossom Race Results Database," [Online]. Available: http://www.cballtimeresults.org/ performances.

[3] "Robots.txt Overview," [Online]. Available: https://moz.com/learn/seo/robotstxt.

[4] "Cherry Blossom robots.txt," [Online]. Available: http://www.cballtimeresults.org/robots.txt.

7. Appendix

7.1 getLink Function

This function creates the web address to query the race database

```
getLink = function(year, division, section, page=1){
  #this function just create the link with the query parameters
  paste0( 'http://www.cballtimeresults.org/performances'
          ,'?utf8=%E2%9C%93&section=',section
```

```
          ,'&year=',year, '&division=',division, '&page=', page)
}
```

7.2 getTable Function

This function reads a page, identifies the table, and returns a data frame with the records

```
getTable = function(year, division, section, page){
  #this function reads the page and convert to a dataframe

  #get the link to the apge
  link=getLink(year, division, section, page=page)

  #read the page, and grab to 'table' tag
  t=read_html(link)  %>%
    html_nodes("table")  %>%
    html_table(fill=TRUE)

  #get the table and add metadata for the query parameters
  out = t[[1]] %>%
    mutate(year=year, divisionTitle=division, section=section, page=page, link=link)
}
```

7.3 getAllTables Function

This function returns all the raw data from the race database, by looping trough the requested years and the pages. The function leverages parallel processes for faster download time.

```
getAllTables  = function(years, division, section, useCachedWebData = FALSE){
  library(progress)
  library(doParallel)
  library(doSNOW)

  if(!useCachedWebData) {
    #init parallel process with max number of cores
    cl <- makeCluster(detectCores())
    doSNOW::registerDoSNOW(cl)

    #progress bar for the parallel loop
    pb <- progress::progress_bar$new(total = length(years),format='[:bar] :percent :eta')
    progress <- function(n) pb$tick()

    #start a parellel loop per each year
    dataRaw=NULL
    dataRaw = foreach(y=years
                      ,.combine=rbind,.export=c('getTable','getLink')
                      ,.options.snow = list(progress=progress)) %dopar%
      {
        library(foreach)
        library(rvest)
        library(dplyr)
        isCompleted=FALSE

        #loop to parse all pages
        dataRaw=foreach(p=c(1:1000),.combine=rbind) %do%
          if(!isCompleted) {
            message('getting year:',y, ' page:',p,appendLF = F)
            #get the table of the current page
            table = getTable(year=y
                             ,division=division
                             ,section=section
                             ,page=p)
```

```
            message(' rows:',nrow(table))
            isCompleted = nrow(table)==0 #if there is record, we are at the last page
            return(table)
          }
        return(dataRaw)
      }
    stopCluster(cl)
    #save the raw data
    saveRDS(dataRaw,file='../../data/dataRaw.rds')
  } else {
    #read the raw data
    dataRaw= readRDS(file='../../data/dataRaw.rds')
  }
  return(dataRaw)
}
```

7.4 transformAllTables function

This function cleans and converts the raw data to the correct format. It also splits the columns that contains multiple information (like town ad state)

```
transformAllTables =function(dataRaw){
  #fix columns and content
  data = dataRaw %>%
    separate(col='Name',c('Name','Gender'),sep='[\\(\\)]'
             ,extra='drop',remove=TRUE) %>% #split name cols to name and gender
    separate(col='PiS/TiS',c('PiS','TiS'),sep='\\/'
             ,extra='drop',remove=TRUE) %>% #split PiS/TiS cols
    separate(col='PiD/TiD',c('PiD','TiD'),sep='\\/'
             ,extra='drop',remove=TRUE) %>% #split PiD/TiD cols
    separate(col='Hometown',c('Hometown','HomeState')
             ,sep=',',extra='merge',remove=TRUE,fill='right') %>% #split Hometown and state
    mutate(Hometown=trimws(Hometown) #remove trim space
           ,Hometown =  ifelse(Hometown %in% c('NR',''),NA, Hometown) #convert NR and '' to NA
           ,HomeState = trimws(HomeState) #remove trim space
           ,HomeState = ifelse(HomeState %in% c('NR',''),NA, HomeState) #convert NR and '' to NA
           ,HomeState = trimws(toupper(ifelse(is.na(HomeState),Hometown,HomeState))) #if state is null,
                                                                                     it is saved in the hometown
           ,HomeCountry = ifelse(HomeState %in% toupper(c('DC',state.abb)), 'USA',HomeState) #check if
                                                                                              is a USA state
           ,Hometown = ifelse(toupper(Hometown) == HomeState,NA,Hometown) #set town to NA if is the same
                                                                          of state
           ,HomeState = ifelse(HomeState == HomeCountry,NA,HomeState) #set State  to NA if is = to
                                                                      country
           ,Race = trimws(substr(Race,5,100)) #gram race type race cols
           ,Time = parse_date_time(gsub('\\*\\*','',Time)
                                   , orders = c("HMS", "MS")) #convert time col to time format
           ,TimeMins = second(Time)/60 + minute(Time)  + hour(Time)*60  #convert time to minus
           ,Pace = parse_date_time(gsub('\\*\\*','',Pace)
                                   , orders = c("HMS", "MS"))  #convert pace col to time format
           ,PaceMins = second(Pace)/60 + minute(Pace)  + hour(Pace)*60  #convert pace time to mins
           ,Age = as.numeric(Age) #convert age to numeric
           ,PiS = as.numeric(PiS) #convert PiS to numeric
           ,TiS = as.numeric(TiS) #convert TiS to numeric
           ,PiD = as.numeric(PiD) #convert PiD to numeric
           ,TiD = as.numeric(TiD) #convert TiD to numeric
    )

  #fixing missing home state (the state is save as town)
  return(data )
}
```

7.5 Participation by Year

```
p = plotdata %>%
  ggplot(aes(x = year, y = count, fill=as.factor(year))) +
  geom_bar(stat = "identity") +
  labs(fill='Year')
print(p)
```

7.6 "Out of Town" Participation

```
closeStates = c("DC", "VA", "MD")

data %>%
  dplyr::filter(!(HomeState %in% closeStates)) %>%
  group_by(year) %>%
  summarise(Count = n()) %>%
  ggplot(aes(x=year, y=Count, fill=as.factor(year))) +
    labs(fill='Year') +
    geom_bar(stat = "identity")
```

7.7 Participation by Year and Age Bracket (Count)

```
participants_year_div = data %>%
  group_by(year, Division) %>%
  summarise(num_participants = n())

participants_year_div %>%
  ggplot(aes(x=Division, y=year, fill=num_participants)) +
  geom_tile() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  scale_fill_gradient2(low = "green", mid='darkorange', high = "darkred", na.value = NA
                       ,midpoint=quantile(participants_year_div$num_participants, 0.9)[[1]]) +
  geom_text(data = participants_year_div, label = participants_year_div$num_participants)
```

7.8 Participation by Year and Age Bracket (Percentage)

```
plotdata_by_year = data %>%
  group_by(year) %>%
  summarise(count_year = n())

plotdata_by_year_div = data %>%
  group_by(year, Division) %>%
  summarise(count_year_div = n())

plotdata = plotdata_by_year %>%
  plyr::join(plotdata_by_year_div, by = "year", type = "full") %>%
  mutate(percent = round(count_year_div/count_year*100,1))

plotdata %>%
  ggplot(aes(x = year, y = percent, fill =forcats::fct_rev( Division))) +
  geom_bar(stat = "identity", position = "stack") +
  coord_flip() +
  labs(fill='Division') +
  geom_text(data=filter(plotdata,percent>2.5),aes(label = paste0(scales::percent(percent/100,accuracy=1)))
            , position = ggplot2::position_stack(vjust=0.5)
            , check_overlap = T, size=3,vjust=-0.5) +
  geom_text(data=filter(plotdata,percent>2.5),aes(label = Division)
            , position = ggplot2::position_stack(vjust=0.5)
            , check_overlap = T, size=2.5,vjust=1)  +
  scale_x_continuous(breaks = unique(plotdata$year))
```

7.9 Race Time by Year

```
p = data %>%
  mutate_at("year", as.factor) %>%
  ggplot(aes(x=year, y=TimeMins, fill=year)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  coord_flip()
print(p)
```

7.10 Race Time by Age Bracket

```
p = data %>%
  # dplyr::filter(!(Division %in% c("W8099", "W7579"))) %>%
  mutate_at("year", as.factor) %>%
  ggplot(aes(x=Division, y=TimeMins, fill=Division)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  coord_flip()

print(p)
```

7.11 Race Time by Year and Age Bracket

```
p = data %>%
  dplyr::filter(!(Division %in% c("W8099", "W7579", "W7074", "W6569"))) %>%
  mutate_at("year", as.factor) %>%
  ggplot(aes(x=year, y=TimeMins, fill=year)) +
  geom_boxplot() +
  facet_wrap(. ~ Division, ncol=2) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  coord_flip()

print(p)
```