Business Case:

The Credit Union Cherry Blossom Ten Mile Race is an annual road race that takes place in Washington D.C. . The race has been held since 1973 and during that time valuable race result data has been collected and is available on the race’s official website. The objective of this analysis is to extract race result data for female runners between the years of 1999 to 2012, a total of 14 years. Using this data, we hope to be able to help race planners gain new insight into patterns and trends as they relate to female runners of the race.

A few questions explored in this analysis include: Have age distributions changed over the years? Have race times increased or decreased both in total and by age groups for female runners. We will also look for trends and other insight within the data that may be valuable to bring to the attention of race planners. With a better understand of female participants race planners can make adjustments in routes, sponsorship outreach, and marketing that could help increase female participation and improve the overall race experience.

Data Extraction(Prep)

To collect the necessary race data our team will be using software to extract the race results data published directly from the Cherry Blossom Ten Mile Race website. This technique is known as web scrapping and is widely used to collect data from the internet. The race results data published on the Cherry Blossom website is freely available to the public and there are no known restriction to either collecting (scrapping) this data or analyzing it.

The web scrapping process is a fairly straightforward one as it relates to this project. The website itself is constructed using Hypertext Markup Language or HTML. HTML is what is known as a markup language but put simply there are tags within HTML that give the webpage it’s structure. In viewing the underlying HTML code, we can find the tags that encapsulate the data we are interested in. Once we know the relevant tags related to the content, we wish to scrape in this case race results we instruct our software to search through the websites HTML code find the tags we are interested in and scrape the data contained in the tags. This process is repeated for each webpage of race results from 1999 to 2012 until all the raw race results data has been extracted. The next section will review how the raw data is then transformed to a usable state for analysis.

Data Extraction (Execution):

Raw data extraction: After separating the data into individual rows, we had to extract the variables from the raw data. We use the headers at the top and the row with the equal signs to guide our data extraction. For most years, the breaks in the row of equal signs correspond to a new variable. We use the extract variables function below to create matrices from the raw data. The function takes the raw data that is separated by row, finds the locations of the breaks in the row with the equal signs, and uses that location as a guide where to split every line in the raw data into a new variable. These variables are then named using the header row above the equal sign line. We found that year 2001 did not have a header row but did have the equal sign separator. 2002 and 2001 had the same header row structure, so we copy the header row from 2002 and use it for 2001. 2011 had a problem initially with parsing using UTF-8. Changing the htmlParse function in the extractResTable3 function to have encoding = 'latin1' solved the problem. It was also found that 2006 did not have a separation in the equal sign row between location and time, so we manually changed an equal sign to a space in the row so our function would automatically separate these variables. Our extractVariables function was successfully run on all raw data objects and outputted a list of individual matrices for each year.

Clean up: After extracting raw results, the data needed significant amounts of cleaning. There were NA's and outliers all over the place. We had to assign header names and change data types. Header names were addressed in our extractVariables function, and the remaining variables are ("name", "home", "ag", "gun", "net", "time"). Name is the name of the runner, home is their home location, ag is their age, gun is their gun time (if applicable), net is their net time (if applicable) and time is their race time (if applicable). If we have a net time variable, we use that to calculate time in the final dataframe. If we have a gun time variable but no net time variable, we use that to calculate time in the final dataframe. If we don't have a gun or net time but we have a time variable, we use that to calculate time in the final dataframe. After addressing header names, we changed the age variable to numeric. We had to do some data parsing to change the time variable to numeric as well: the data came in as hours:minutes:seconds. We were interested in analyzing the time variable in terms of minutes, so we multiplied hours by 60, divided seconds by 60 and kept the minutes, and added those 3 variables together to get total minutes. This result was then turned into a numeric variable.

Missing values and outliers: There were outliers and NA values in both of our numeric columns: age and time. These needed to be dealt with. Looking at box plots, it was clear that 2003 was definitely an issue, and 2009, 2001 and 2011 could potentially be issues with young runners. To deal with 2003, we found that the age column fluctuated by a column in either direction as we went down the raw data. Increasing the search by 1 column for the age variable solved the issue with 2003 and 2011. In 2001, there was a racer with an age of 0. When we went back and looked at the raw data, it was found that this racer was listed as 0, so we just removed this racer from the data. In 2011, there was a racer that was 7 years old. This was confirmed in the raw data. While the racer was young, it was determined that it was possible for a 7-year-old to run a 10-mile race, so the data point was kept. There were still a few outliers in the data frame, but those were kept in there for analysis in our EDA. Now that we dealt with NA's and outliers in the age variable, we decided to look at the time variable. Stars and hashtags in the time variable were messing up our multiplication initially. Removing those in the initial time calculation function removed a significant amount of NA's in many years. There were still NA's in 2002 and 2006. 2006 had times that were in the location variable. It was determined that the data wasn't being parsed correctly due to a missing break in the equal sign row. The break was manually inputted, which fixed the NA's in 2006. The one remaining NA in 2002 was a footer line, which was removed. Looking at box plots of the run times by year, nothing stood out as problematic in terms of outliers, so we proceed with our analysis.