virus

May 18, 2020

1 Comparison of Infection Rates and Mortality Rates between Influenza and SARS-Cov-2 in the United States

1.1 Introduction

Sometime in November of 2019, Chinese hospitals in the province of Hubei began admitting patients who were displaying symptoms of Pneumonia. At first it is likely that doctors in these hospitals assumed that the uptick in Pneumonia cases were due to influenza infections; however, the patients began dying at a rate that exceeded the expected case fatality rates of most known strains of Pneumonia. Within a month it became clear that Chinese patients suffering from the effects of Pneumonia were either infected with a new strain of the flu, or a completely differnt pathogen altogether.

Unfortunately due to the communist nature of the Chinese government, this new outbreak was hidden from most of the world until early January, and even then information was slow to percolate two western media sources and epidemiologists. By the time the Chinese goverment made public the existance of this new pathogen, it had already spread to Europe, Asia and North America. European and Asian governments as well as governments in the America's were given little time to react to this new pathogen that appeared to have an infection rate and mortality rate that exceeded that of common influenza variants and possibly rivaled that of the Spanish Flu, which caused a global pandemic in 1918.

By the 7th of January, 2020, the Chinese goverment identified the pathogen responsible for the large increase in Pneumonia patients. The pathogen was identified to be a type of a Coronavirus, similar to the pathogens responsible for the 2003 SARS outbreak in China and the 2012 MERS outbreak in the middle east and Korea. The new Coronavirus has been commonly referred to as COVID 19, which stands for the Coronavirus Identification Year 2019; however, the formal idenfitication for this new virus is SARS-Cov-2.

As of the date of this report, epidemiologists across the globe are working at a fevered pace to understand the new Coronavirus. Doctors endeavor to understand the virus genome, its tramission methods, transmission rates, and how it interacts with infected hosts. Unfortunately the modern transporation infrastructure, combined with modern population densities, and the secrecy of the Chinese goverment has put the world at a disadvantage with respect to combatting this new illness. By the time the rest of the world was made aware of SAR-Cov-2, the virus had already spread to most places in the world where a rigorous transportation architecture was in place. The World Health Organization (WHO) declared the SARS-Cov-2 outbreak a pandemic on March 11th, 2020, which represents a global recognition of the severity of this disease. Currently the United States

Center for Disease Control (CDC) maintains a website where American citizens and citizens of most other countries can stay apprised of all relevant information regarding this new pathogen.

As of the date of this report, on March 22nd, 2020, over 169 countries and regions have reported SARS-Cov-2 related infections and deaths. However, the reported infection and mortality rates vary greatly between the various affected countries, with mortality rates that differ by more than an order of magnitude. As with any other pathogen, the societal response to the illness should be proportional to the risk of aquiring the illness and the consequences if infected. An under-reaction to a virus such as SARS-Cov-2 could cost tens of thousands, and possibly even millions of lives. However, an over-reaction to the illness can drive an economic recession, which could also cost tens of thousands to millions of lives through the detrimental effects of poverty.

1.2 Background

The United States Center for Disease Control (CDC) has been collecting data on influenza infections dating back to the 1997-1998 flu season. Since 1997, the United States has kept records of patients infected with the A (H1) and A (H3) influenza subtype's as well as the common B subtype and the Victoria as well as the Yamagoto sub types of the B strain. In additon, the United States has kept records of patients infected with the H3N2V influenza strain and in 2009, there was also an outbreak of the H1N1 flu type, which was similar to the same illness that caused the Influenza Pandemic of 1918.

The U.S. government, through the CDC, bases its annual flu response on two metrics, the probability that a member of society will become infected with the flu (P_i), and the probability that an infected person will perish due to the influenza infection (P_d), also known as the case fatality mortality rate. The product of P_i and P_d is known as the mortality rate (P_m). The relationship between P_m and P_i and P_d.

$$P_m = P_i \times P_d \tag{1}$$

The value of P_i is a product of the probability that an un-infected person will come into medically significant contact with an infected person (P_c) and the probability that the un-infected person will acquire the illness if they come into contact with an individual who is infected (P_{ii}). Medically significant contact is defined as an interaction between an infected and un-infected individual that poses a non-zero probability of disease transmission.

$$P_i = P_c \times P_{ii} \tag{2}$$

The term P_i is largely a function of population density and the transmission method for the illness. For instance, if an individual lives in an area containing just 0.1 to 1 person per square mile, the probability of an un-infected individual interacting with someone who is infected with an illness is very small compared to an un-infected individual living in an area containing 2-10 people per square mile. However, if the illness can be transmitted over the air and has an aerosol range of half a mile before the pathogen perishes, the person living in an area with a population density of 0.1 to 1 person per square mile may have a higher probability of acquiring an illness. The terms P_{ii}, and P_d are a function of an un-infected persons genome, age, and behavioral habits. The genetic composition of an individual can give that person an immunity to diseases that could otherwise be very infectuous or fatal to other members of the population. Age is also known to greatly affect the immune response of patients to illnesses that may otherwise not be of concern people of a different age. Typically the very young and the elederly population will have a lesser immune response to a pathogen than those of moderate ages. Finally a persons behavior can also greatly affect their ability to resist or survive an infection. For instance, people who smoke cigarettes greatly reduce the ability of their lungs to resist respitory illnesses.

The term P_i serves as an unmitigated risk factor for the flu, since it represents the fraction of a population that will be put at risk if nothing is done. The term P_d serves as the unmitigated consequence management term for government planners, since it describes how severe the outcome will be for the fraction of the population that is put at risk of infection. Once the risk and consequences are known, planners can weigh the merits of various responses to mitigate the risk or the consequences, where each response is likely evaluated based on its cost, effectiveness, and secondary consequences. **The secondary consequences of a response to a pathogen cannot be overlooked, as an over-reaction to a pathogen outbreak can be as harmful to human life as under-reacting.** It is a well established fact that an individuals general health and mortality is inversly proportional to their economic status. Most stringent measures to contain an outbreak will have a negative impact on the national and global economy. It is possible that an effort to contain an outbreak may save lives from a pathogen caused death; however, the long term economic effects could impoverish enough people that the detrimental effects of economic downturn results in more deaths than were saved by combatting the virus.

It is impossible to know how many people actually aquire the flu each year, since only those with severe infections feel the need to go to a hospital where they are officially diagnosed. However, the United States has a long history of sampling the population to estimate the number of un-diagnosed influenza patients for each diagnosed patient. Between the 2010-2011 flu season and the 2018-2019 season the CDC estimates that between 9.3 million Americans and 45 million Americans are infected with influenza each year, with approximately 12,000 to 61,000 flu related fatalities per year. These figures come from the CDC influenza burden website. This implies values of $P_i = 0.0890 \pm 0.0485$ and $P_d = 0.00132 \pm 0.00044$ where each number represents a percentage ranging from 0 to 1. It should be noted that the author of this report is not aware of how the CDC determines the true number of infections, which could have a very large effect on the values of P_i and P_d. However, as reported by the CDC, the mortality rate for influenza in the United States has a value of $Pm = 0.00011 \pm 0.000006$, which means that an average American citizen has a 1.11 chance in 10,000 of acquiring and dying of influenza, which is well within the risk factor of other natural causes of death such as automobile accidents, heart disease, etc...

Currently epidemiologists are working to better understand the SARS-Cov-2 virus; however, in the absense of data, policy makers struggle to determine the appropriate resoonse to the COVID-19 epidemic. The U.S. CDC as well as the United States National Institute for Allergyies and Infectious Disease (NIAD) is working to develop estimates for the values of P_c, P_{ii} and therefore P_i through the combination of modeling and experimental measurements on the SARS-Cov-2 virus itself; however, these will only be estimates for the infection rates. The true value of the infection rates will not be known for a very long time, and even once the outbreak is contained, it will take years to gain an estimate for the number of people who were infected but never diagnosed. However, we can gain a relatively quick estimate for the value of P_d by monitoring those who have been formally diagnosed and comparing the numbers of those diagnosed who perished, with the total number of patients diagnosed. The initial normalized mortality rates (P_d) can be used to guide a societal response to the virus and should be based on the effects in each nation or region. Each regional area has a different population density, genetic composition, age distribution, and average behavioral pattern. A financial and medical response that makes sense in an Asian country, may not make sense for a North American country, so the response in each country should be determined seperately.

1.3 Purpose of this Study

The SARS-Cov-2 outbreak may be the first data-driven pandemic in human history. With exception of the original Chinese response to the outbreak, infection and mortality data for countries across the globe is being shared publically, allowing anyone to analyze the data and draw conclusions for themselves. One of the best examples of data-sharing is the John's Hopkins Univerity SARS-Cov-2 dahsboard site, which gives users an easy to use interface to track the diseases spread and its effect on each separate country.

For right or wrong, many people are comparing the SARS-Cov-2 outbreak to a bad influenza season, and have in default, made the influenza risk and consequences the benchmark for the United States response to this virus. If the mortality and infection rates are worse than the flu, then the public largely favors drastic actions to combat the Coronavirus; however, if the risk and consequences are in the same general ball park, or only slightly worse than the flu, then the public is apprehensive about drastic actions, which will inevitebly damage the economy.

The purpose of this study is to mine the existing American databases for influenza as well as the current SARS-Cov-2 databases for the purpose of making relevant comparisons between the two pathogens. I should mention that the author of this study is not a doctor, virologist, or an epidemiologist, but is instead an engineer and data scientist. I will endeavor to only make conclusions that are empircal based on the data provided and not delve into a mechanistic study, which is beyond the intellectual depth of the author.

1.4 Package Imports and General Code

This document is produced in a Jupyter Lab notebook using the Python 3.6 programming language. This document is inherently meant to utylize a programming language within a word processer environemt to combine the analytical power of computer software with the graphical display capabilities of a word processor. This document can be read by technical experts as well as non-technical readers. If a non-technical reader is viewing this document, I recommend that you skip the code sections and proceed directly to the written sections as well as the plots and tables that highlight the points being made.

The section below contains code that is used to read in the various databases and that conducts basic formatting so the data can be used for analysis. In addition, all package imports are listed in the next section. The actualy analysis begins after the code section below.

[1]:
```python
# Package Imports
import pandas as pd
import numpy as np
from matplotlib import rcParams, pyplot as plt
import matplotlib
import matplotlib.dates as mdates
```

```
from datetime import datetime
from IPython.display import display
from scipy.signal import argrelextrema
import operator
%matplotlib inline

start_date = '1/22/20'
end_date = '5/17/20'

class ReadCovidCumData:
    def __init__(self, file: str):
        """

        :param file: The file location to include path-length
        """
        self.file = file
# ----------------------------------------------------------------------------

    def read_data(self, start_date: str, end_date: str) -> pd.DataFrame:
        """

        :param start_date: The start date in the format MM/DD/YY
        :param end_date:  The end date in the format MM/DD/YY
        :return df: A pandas data-frame containing all information between
                    the start and end date

        This function will read the csv files provided by JHU which contain
        the number of infections, deaths and recovered patients per day as
        cumulative values
        """
        head_dict, headers = self._header_dict(start_date, end_date)
        df = pd.read_csv(self.file, usecols=headers, dtype=head_dict)
        return df
# ----------------------------------------------------------------------------

    def filter_by_country(self, country: str, start_date: str,
                          end_date: str) -> pd.DataFrame:
        """

        :param country: The country for which data is required
        :param start_date: The start date in the format MM/DD/YY
        :param end_date: The end date in the format MM/DD/YY
        :return df: A pandas data-frame containing all information between
                    the start and end date

        This function will return a pandas data-frame that filters the
        JHU csv files by the country of interest.  The country must match
        the spelling in the JHU .csv files
        """
        df = self.read_data(start_date, end_date)
        mask = df['Country/Region'] == country
        return df[mask]
# ----------------------------------------------------------------------------

    def filter_by_region(self, country: str, region: str,
                         start_date: str, end_date: str) -> pd.DataFrame:
        """

        :param country: The country for which data is required
        :param region: The region within the country for which data is required
        :param start_date: The start date in the format MM/DD/YY
        :param end_date: The end date in the format MM/DD/YY
        :return df: A pandas data-frame containing all information between the
                    start and end date

        This function will return a pandas data-frame that filters the JHU csv
        files by the country and region of interest.  The country and region
        must match the spelling in the JHU .csv files
        """
        df = self.read_data(start_date, end_date)
        mask = (df['Country/Region'] == country) & (df['Province/State'] ==␣
↪region)
        return df[mask]
# ============================================================================

    def _header_dict(self, start_date, end_date):
        """

        :param start_date: The start date in the format MM/DD/YY
        :param end_date: The end date in the format MM/DD/YY
        :return headers: A dictionary containing the names of all headers
                         and the data types of each header
        """
        dates = self._date_range(start_date, end_date)
        headers = ['Province/State', 'Country/Region', 'Lat', 'Long']
        dattypes = [str, str, float, float]
        for date in dates:
            if date[0] == '0' and date[3] == '0':
                dat = date[1:3] + date[4:]
            elif date[3] == '0':
                dat = date[0:3] + date[4:]
            elif date[0] == '0':
                dat = date[1:]
            else:
                dat = date
            headers.append(dat)
            dattypes.append(int)
        return dict(zip(headers, dattypes)), headers
# ----------------------------------------------------------------------------

    def _date_range(self, start_date, end_date) -> pd.DataFrame:
        """

        :param start_date: The start date in the format MM/DD/YY
        :param end_date: The end date in ht eformat MM/DD/YY
        :return dt: A pandas date range covering the days between the
                    start and end dates
        """
        dt = pd.date_range(start=start_date, end=end_date).strftime('%m/%d/%y')
        return dt
# ============================================================================
# ============================================================================


class ProcessCovidData(ReadCovidCumData):
    def __init__(self, file: str):
        """

        :param file: The file location to include path-length
        """
        self.file = file
        ReadCovidCumData.__init__(self, file)
# ----------------------------------------------------------------------------

    def total_global_cases(self, start_date: str, end_date: str) -> pd.
 ↪DataFrame:
        """

        :param start_date: The start date in the format MM/DD/YY
        :param end_date:  The end date in the format MM/DD/YY
        :return df: A pandas data-frame containing all information between
                    the start and end date

        This function will read the csv files provided by JHU which contain
        the number of infections, deaths and recovered patients per day as
        cumulative values.  This function specifically produces a data frame
        that only contains the dates and the cumulative number of global␣
 ↪infections,
        deaths or recovered patients on that day
        """
        df = self.read_data(start_date, end_date)
        df = df.drop(['Province/State', 'Country/Region', 'Lat', 'Long'],␣
↪axis=1)
        df2 = df.sum(axis=0)
        df2.index = pd.to_datetime(df2.index, format="%m/%d/%y")
        df2.columns = ['index', 'Number']
        return df2
# ----------------------------------------------------------------------------

    def total_cases_by_country(self, country: str, start_date: str,
                               end_date: str) -> pd.DataFrame:
        """

        :param country: The country for which data is required
        :param start_date: The start date in the format MM/DD/YY
        :param end_date: The end date in the format MM/DD/YY
        :return df: A pandas data-frame containing all information between
                    the start and end date

        This function will return a pandas data-frame that filters the
        JHU csv files by the country of interest.  The country must match
        the spelling in the JHU .csv files.  The data-frame only contains the
        date and the cumulative number of infections, deaths, or recovered
        patients on that day in that state
        """
        df = self.filter_by_country(country, start_date, end_date)
        df = df.drop(['Province/State', 'Country/Region', 'Lat', 'Long'],␣
↪axis=1)
        df2 = df.sum(axis=0)
        df2.index = pd.to_datetime(df2.index, format="%m/%d/%y")
        df2.columns = ['Number']
        return df2
# ----------------------------------------------------------------------------

    def total_cases_by_region(self, country: str, region: str,
                              start_date: str, end_date: str) -> pd.DataFrame:
        """

        :param country: The country for which data is required
        :param region: The region within the country for which data is required
        :param start_date: The start date in the format MM/DD/YY
        :param end_date: The end date in the format MM/DD/YY
        :return df: A pandas data-frame containing all information between the
                    start and end date

        This function will return a pandas data-frame that filters the JHU csv
        files by the country and region of interest.  The country and region
        must match the spelling in the JHU .csv files.  The data-frame
        only contains the date and cumulative number of infections, deaths,
        or recovered patients on that day in that state and region
        """
        df = self.filter_by_region(country, region, start_date, end_date)
        df = df.drop(['Province/State', 'Country/Region', 'Lat', 'Long'],␣
 ↪axis=1)
        df2 = df.sum(axis=0)
        df2.index = pd.to_datetime(df2.index, format="%m/%d/%y")
        df2.columns = ['Number']
        return df2
# ----------------------------------------------------------------------------

    def global_cases_per_day(self, start_date: str, end_date: str) -> pd.
 ↪DataFrame:
        """

        :param start_date: The start date in the format MM/DD/YY
        :param end_date:  The end date in the format MM/DD/YY
        :return df: A pandas data-frame containing all information between
                    the start and end date

        This function will read the csv files provided by JHU which contain
        the number of infections, deaths and recovered patients per day as
        cumulative values.  This function specifically produces a data frame
        that only contains the dates and the number of global infections,
        deaths or recovered patients produced on that day
        """
        df = self.total_global_cases(start_date, end_date)
        df = df.diff()
        return df.drop(df.index[0])
# ----------------------------------------------------------------------------

    def country_cases_per_day(self, country: str, start_date: str,
                              end_date: str) -> pd.DataFrame:
        """

        :param country: The country for which data is required
        :param start_date: The start date in the format MM/DD/YY
        :param end_date: The end date in the format MM/DD/YY
        :return df: A pandas data-frame containing all information between
                    the start and end date

        This function will return a pandas data-frame that filters the
        JHU csv files by the country of interest.  The country must match
        the spelling in the JHU .csv files.  The data-frame only contains the
        date and the number of infections, deaths, or recovered
        patients produced on that day in that state
        """
        df = self.total_cases_by_country(country, start_date, end_date)
        df = df.diff()
        return df.drop([df.index[0]])
# ----------------------------------------------------------------------------

    def region_cases_per_day(self, country: str, region: str,
                             start_date: str, end_date: str) -> pd.DataFrame:
        """

        :param country: The country for which data is required
        :param region: The region within the country for which data is required
        :param start_date: The start date in the format MM/DD/YY
        :param end_date: The end date in the format MM/DD/YY
        :return df: A pandas data-frame containing all information between the
                    start and end date

        This function will return a pandas data-frame that filters the JHU csv
        files by the country and region of interest.  The country and region
        must match the spelling in the JHU .csv files.  The data-frame
        only contains the date and number of infections, deaths,
        or recovered patients produced on that day in that state and region
        """
        df = self.total_cases_by_region(country, region, start_date, end_date)
        df = df.diff()
        return df.drop([df.index[0]])
# ============================================================================
# ============================================================================


class ReadFluInfections:
    def __init__(self, file: str):
        """

        :param file: The file location to include path-length
        """
        self.file = file
# ----------------------------------------------------------------------------

    def read_data(self) -> pd.DataFrame:
        """

        :return df: A pandas data-frame containing all flu infection
                    information

        This function will read the csv files provided by the CDC which contain
        the number of infections per week for the entire range of data␣
↪collection
        """
        headers = ['YEAR', 'WEEK', 'A (2009 H1N1)', 'H3N2v',
                   'SEASON', 'A (H1+H3)', 'B (B+BVic+BYam)', 'DATE']
        data_type = [int, int, int, int, str, int, int, str]
        head_dict = dict(zip(headers, data_type))
        df = pd.read_csv(self.file, usecols=headers, dtype=head_dict)
        df['DATE'] = pd.to_datetime(df['DATE'])
        df = df.set_index('DATE')
        return df
# ==============================================================================
# ==============================================================================


class ProcessFluInfections(ReadFluInfections):
    def __init__(self, file: str):
        """

        :param file: The file location to include path-length
        """
        self.file = file
        ReadFluInfections.__init__(self, file)
# ------------------------------------------------------------------------------

    def filter_by_season(self, season: str) -> pd.DataFrame:
        """

        :param season: The flu season for which the user desires information.
                       The season should be in a YYYY-YYYY format
        :return df: A pandas data-frame containing the flu infections data for
                    the season entered by the user
        """
        df = self.read_data()
        mask = df['SEASON'] == season
        return df[mask]
# ==============================================================================
# ==============================================================================


class ReadFluMortality:
    def __init__(self, file: str):
        """

        :param file: The file location to include path-length
        """
        self.file = file
```

```python
# ----------------------------------------------------------------------------

    def read_data(self) -> pd.DataFrame:
        """

        :return df: A pandas data-frame containing all flu mortality
                    information

        This function will read the csv files provided by the CDC which contain
        the number of deaths per week for the entire range of data collection
        """
        headers = ['SEASON', 'WEEK', 'NUM INFLUENZA DEATHS',
                   'NUM PNEUMONIA DEATHS', 'DATE', 'TOTAL DEATHS',
                   'PERCENT P&I']
        data_type = [str, int, int, int, str, int, float]
        head_dict = dict(zip(headers, data_type))
        df = pd.read_csv(self.file, usecols=headers, dtype=head_dict)
        df['DATE'] = pd.to_datetime(df['DATE'])
        df = df.set_index('DATE')
        return df
# ============================================================================
# ============================================================================


class ProcessFluMortality(ReadFluMortality):
    def __init__(self, file):
        """

        :param file: The file location to include path-length
        """
        self.file = file
        ReadFluMortality.__init__(self, file)
# ----------------------------------------------------------------------------

    def filter_by_season(self, season: str) -> pd.DataFrame:
        """

        :param season: The flu season for which the user desires information.
                       The season should be in a YYYY-YYYY format
        :return df: A pandas data-frame containing the flu mortality data for
                    the season entered by the user
        """
        df = self.read_data()
        mask = df['SEASON'] == season
        return df[mask]
# ============================================================================
# ============================================================================
def convert_to_m_d(df: pd.DataFrame) -> pd.date_range:
    """

    :param df: A pandas data-frame containing pathogen information
    :return dates: A pandas date_range list ona. week by week basis

    This function is used to produce a date list for flu dataframes so
    they can be plotted on the same axis
    """
    dates = pd.date_range(start='2020-01-10', periods=len(df), freq='W')
    return dates
```

1.5 Databases

1.5.1 Influenza Infection Rates

The estimated number of influenza infections on a per week basis is extracted from the USCDC Flu View website. The CDC influenza infection database only accounts for patients that were formally diagnosed at a hospitals or primary care providers. For every diagnosed patient, their can be a substaintially large number of un-diagnosed infected people. In addition, those displaying the symptoms of influenza caused pnemonia may be classified as having pneumonia and not influenza. The CDC influenza database does not describe the total number of influenza infected patients in the United States, but can still give valuable insights into how the flu seasons and flu outbreaks progress as a function of time. The time progression information for influenza may also be used as an analog fo other pathogens while researchers collect data specific to the new pathogen.

The CDC Flu View website will provide a user with a zip file containing four `.csv` files. The files relevant to influenza infection rates are titled `WHO_NREVSS_Combined_prior_to_2015_16.csv` and `WHO_NREVSS_Public_Health_Labs.csv`. Stating in the 2007-2008 influenza season the CDC collected data on the number of patients infected on a week by week basis, where the number of infections were classified by those infected with H1N1 strain, those infected with the A (H1) strain-subtype, the A (H3) strain-subtype, the H3N2v strain, the B strain, as well as those who were infected with the A strain, but subtyping was not performed, and those infected with the A strain; however, the subtype could not be determined. The CDC continued to collect influenza data in this format until the end of the 2014-2015 season and have placed the information in the `WHO_NREVSS_Combined_prior_to_2015_16.csv` file. Starting in the 2015-2016 flu season, the CDC changed their data collection format to classify infections by those infected with the H1N1 strain, the A (H3) strain-subtype, those who were infected with the A strain but subtyping was not performed, those with the B strain, Victoria subtype of the B strain, those with the Yamagoto subtype of the B strain, and those with H3N2v. The results of data collection between the 2015-2016 flu season to present day are stored in the file titled `WHO_NREVSS_Public_Health_Labs.csv`. In both data sets the flu season is assumed to begin on the 40th week of each year and end on the 39th week of the following year.

In order to simplify the access of data for all influenza seasons starting in 1997 to present, the files are combined into a single single `.csv` file titled `US_Flu_Infections.csv`. In order to combine the different collection formats into one intelligible database, we will combine all A type infections, including those which could not be subtypes or were not subtyped into one classification. In addition, all B strain infections are combined into one classification, and the H3N2v and H1N1 strains are maintained as their own seperate classification. The code shown below reads in the raw data from the CDC and transforms it into a single `.csv` file titled `US_Flu_Infections.csv`. In addition to the data-wrangling mentioned in the paragraphs above, a date in the format mm/dd/yy is given to each week.

[2]:
```
# Read WHO_NREVSS_Combined_prior_to_2015_16.csv file
file = 'Data/Raw_Data/WHO_NREVSS_Combined_prior_to_2015_16.csv'
columns = ['YEAR', 'WEEK', 'A (2009 H1N1)', 'A (H1)', 'A (H3)',
           'A (Subtyping not Performed)', 'A (Unable to Subtype)',
           'B', 'H3N2v']
dattype = [int, int, int, int, int, int, int, int, int]
headers = dict(zip(columns, dattype))
df = pd.read_csv(file, skiprows=[0], usecols=columns, dtype=headers)

# Add SEASON column
season = []
for i in range(len(df)):
    if df['WEEK'][i] >= 40 and df['WEEK'][i] <= 53:
        start = df['YEAR'][i]
        end = df['YEAR'][i] + 1
        val = str(start) + '-' + str(end)
        season.append(val)
    else:
        start = df['YEAR'][i] - 1
        end = df['YEAR'][i]
        val = str(start) + '-' + str(end)
        season.append(val)
df['SEASON'] = season

# Add columns for combined A types and drop all irrelevant columns
df['A (H1+H3)'] = df['A (H1)'] + df['A (H3)'] + df['A (Subtyping not␣
 ↪Performed)'] + \
                  df['A (Unable to Subtype)']
df['B (B+BVic+BYam)'] = df['B']

df = df.drop(['A (H1)', 'A (H3)', 'A (Subtyping not Performed)',
              'A (Unable to Subtype)', 'B'], axis=1)

# Create a date series for this data set
dates = pd.date_range(start='28/9/97', end='28/9/15', freq='W').strftime('%m/%d/
 ↪%y')
df['DATE'] = dates
# ==============================================================================
# ==============================================================================
# Read WHO_NREVSS_Public_Health_Labs.csv file
file = 'Data/Raw_Data/WHO_NREVSS_Public_Health_Labs.csv'
columns = ['YEAR', 'WEEK', 'A (2009 H1N1)', 'A (H3)',
           'A (Subtyping not Performed)', 'B', 'BVic',
           'BYam', 'H3N2v']
dattype = [int, int, int, int, int, int, int, int, int]
headers = dict(zip(columns, dattype))
df2 = pd.read_csv(file, skiprows=[0], usecols=columns, dtype=headers)

# Add SEASON column
season = []
for i in range(len(df2)):
    if df2['WEEK'][i] >= 40 and df2['WEEK'][i] <= 53:
        start = df2['YEAR'][i]
        end = df2['YEAR'][i] + 1
        val = str(start) + '-' + str(end)
        season.append(val)
    else:
        start = df2['YEAR'][i] - 1
        end = df2['YEAR'][i]
        val = str(start) + '-' + str(end)
        season.append(val)
df2['SEASON'] = season

# Add columns for combined A types and drop all irrelevant columns
df2['A (H1+H3)'] = df2['A (H3)'] + df2['A (Subtyping not Performed)']
df2['B (B+BVic+BYam)'] = df2['B'] + df2['BVic'] + df2['BYam']

df2 = df2.drop(['A (H3)', 'A (Subtyping not Performed)',
                'B', 'BVic', 'BYam'], axis=1)
# ==============================================================================
# ==============================================================================

## Create a date series for this data set
dates = pd.date_range(start='10/04/15', end='03/13/20', freq='W').strftime('%m/
 ↪%d/%y')
df2['DATE'] = dates

# Join the dataframes into one
df3 = pd.concat([df, df2])
df3.to_csv('Data/Processed_Data/US_Flu_Infections.csv', index=False)
```

1.5.2 Influenza Mortality Rates

The mortality rate assumed to have been caused by influenza infections in the United States can be found on the CDC Flu View website in a file titled `National_Custon_Data.csv`. The `.csv` file contains the total number of people who die in the United States each week as well as the fraction that expire due to influenza as well as the fraction of those who expire due to pneumonia and influenza, the assumption being that the pneumonia may have been a complication due to an influenza infection. However, whether or not the pneumonia related deaths in the `.csv` file was caused by an influenza infection is not known as an absolute.

The format of the `National_Custon_Data.csv` file is similar but not identical to the infection rate files. The code below reads in the semi-raw data and corrects the season data from a YYYY-YY format to a YYYY-YYYY format. In addition the original file displayed numbers with commas demarking the thousands place within a number (i.e. 1,347), which causes problems when the computer reads the value. The commas are removed in order to simplify the process of reading the data. The file is described as a semi-raw format since it was manually manipulated before incorporating into this code repository in order to more easliy fix some chronological issues with the raw format. The output file is titled `US_Flu_Mortality.csv`. In addition to the data-wrangling mentioned in the paragraphs above, a date in the format mm/dd/yy is given to each week.

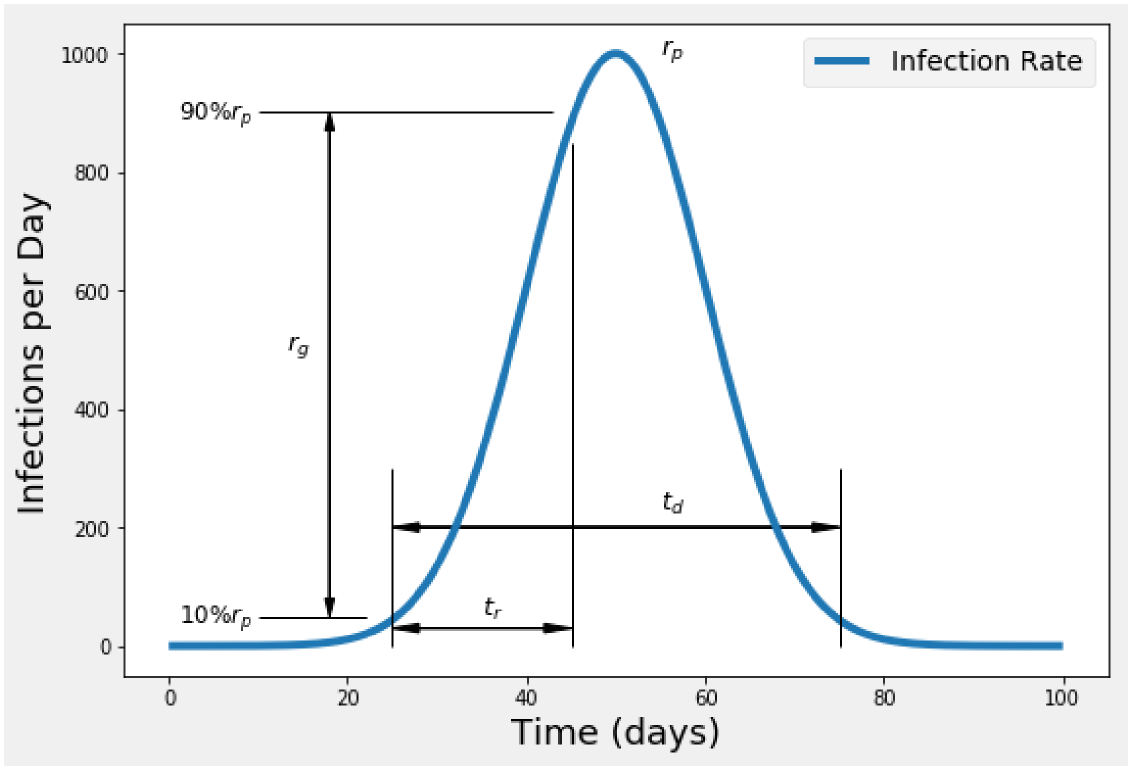
[3]:
```python
# Read National_Custom_Data.csv file
file = 'Data/Raw_Data/National_Custom_Data.csv'
columns = ['SEASON', 'WEEK', 'NUM INFLUENZA DEATHS',
           'NUM PNEUMONIA DEATHS', 'TOTAL DEATHS',
           'PERCENT P&I']
df = pd.read_csv(file, usecols=columns)

# Replace SEASON Dates
season = []
for i in range(len(df)):
    one = df['SEASON'][i][0:5]
    two = df['SEASON'][i][5:7]
    val = str(one) + '20' + two
    season.append(val)
df['SEASON'] = season
df['NUM PNEUMONIA DEATHS'] = df['NUM PNEUMONIA DEATHS'].str.replace(',', '')
df['NUM INFLUENZA DEATHS'] = df['NUM INFLUENZA DEATHS'].str.replace(',', '')
df['TOTAL DEATHS'] = df['TOTAL DEATHS'].str.replace(',', '')

# Add date range
dates = pd.date_range(start='09/23/13', end='03/01/20', freq='W').strftime('%m/
 ↪%d/%y')
df['DATE'] = dates
df.to_csv('Data/Processed_Data/US_Flu_Mortality.csv', index=False)
```

1.5.3 COVID-19 Infection and Mortality Database

The existance of the SARS-Cov-2 virus has only recently become public knowledge and the number of people who are known to be infected is changing daily. The Johns Hopkins University has compiled a database of all known or suspected infections, deaths, and recovered personal due to the COVID-19 virus. The database is a compilation of information collected from the World Health Organization (WHO) and the U.S. Center fro Disease Control (USCDC) and is published on GitHub. The GitHub repository consists of summary reports and a time series database, which is used for this analysis. The time series database consists of three `.csv` files of the exact same format. One file titled `time_series_19-covid-Confirmed.csv` contains the list of known infections for every country and region by the date they were discovered, the second file titled `time_series_19-covid_Deaths.csv` contains a list of all known deaths due to the COVID-19 virus for every country and region by date. Finally, the third file titled `time_series_19-covid-Recovered.csv` contains a list of all known persons to have recovered from the COVID-19 infection by country and region as a function of date; however, this file will not be used in this report. The format of the three COVID-19 files is very different than the Influenza files; however, no changes are required to begin analysis.

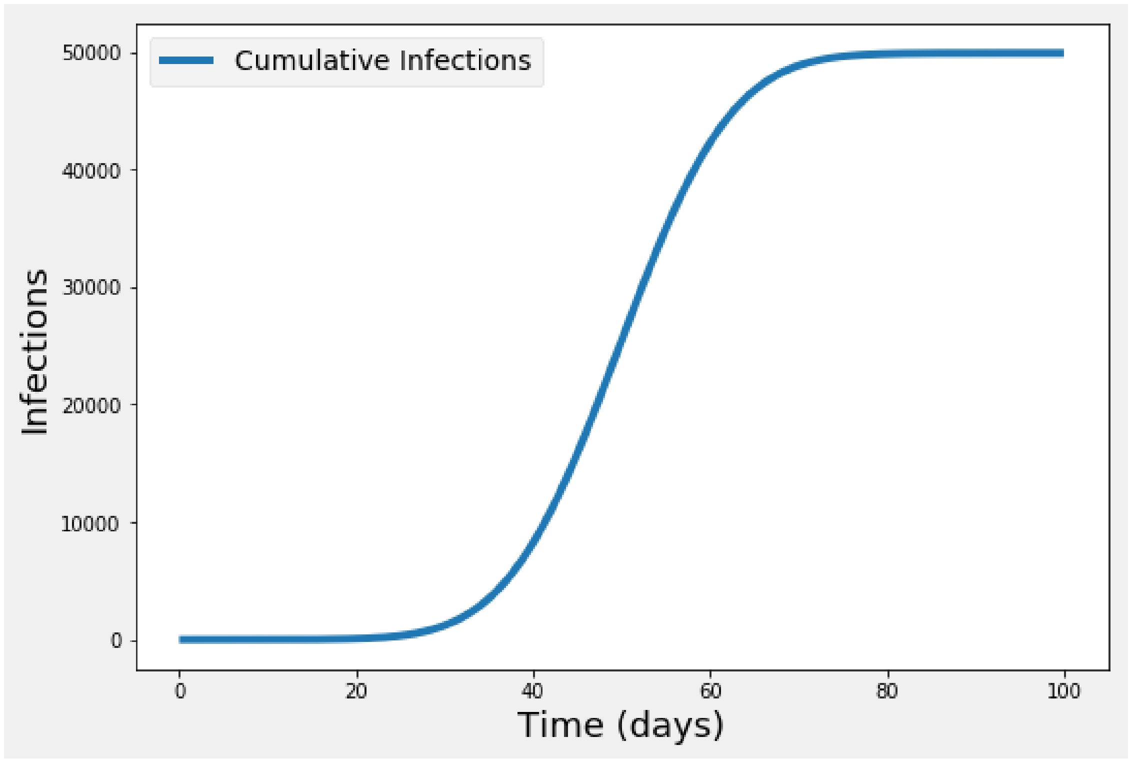
1.6 Basics of Pathogen Analysis

In this analysis we will focus our attention on the number of infections as a function of time and the number of fatalities as a function of time. In both cases we will evaluate the number of new infections or deaths per unit time and develop some basic metrics that help us conduct a simplistic analysis on the progression of a pathogen. The plot of infections per unit time as a function of time, most often assumes a Gaussian or Maxwellian distribution; however, in this case we will assume that the data conforms to a Gaussian profile. A guassian infection rate pulse is evaluated on a number of metrics, the first of which is r_p, which represents the peak infection rate, which is displayed as the maximum point on the Gaussian curve in the figure below. The figure shows a blue curve which represents the number of new infections diognosed per week as a function of time. This plot is commonly referred to as the Probability Distribution Function (PDF).



People are infected with the influenza virus throughout the year; however, for most of the time, the number of cases reported per week is within a statistical steady state value. However, during the time frame of early spring, when the temperatures are cooler, the virus is able to flourish and the number of cases increases above a statistical threshold to a peak value (r_p). In order to ensure that the flu season is not assumed to have begun prematurely, a threshold value must be choosen that will represent a barrier that the number of new cases must exceed before a new flu season or outbreak is assumed. This study will assume an outbreak or flu-season threshold of $10\%r_p$, which conforms to terminology in other statistical fields that must consider Gaussian trends. The geometric growth period r_g is the growth that happens between the $10\%r_p$ and $90\%r_p$ values. The growth period, sometimes referred to as the rise time (r_t) is measured as the time that elapses between the $10\%r_p$ and $90\%r_p$ values. Following the end of the groth period, the trend begins to inflect and eventually the number of new cases recorded per week begins to decrease. Finally, the outbreak duration, sometimes referred to as the pulse width is the time that evolves between the $10\%r_p$ value during the rise time and the same value on the decay side of the pulse.

The integral of the infection rate as a function of time will produce the plot of cumulative infections as a function of time. This trend, typically follows an S-curve geometry and is shown in the figure below. THis plot is commonly referred to as the Cumulative Distribution Function (CDF).



Another metric that is very useful for comparison between pathogens is the exponential factor (λ) and the doubling time (t_d). A typical flu season or pathogen outbreak increases in magnitude at an exponential rate. Within a period of days the number of infected persons can double, triple or even quadruple, before the outbreak is contained. We can model this exponenital increase with the equation below where A is the population of infected persons at the beginning of an outbreak and B represents the number of infected persons after time t, and exp represents the exponential term e.

$$B = Aexp(\lambda t) \tag{3}$$

The term λ helps mathematical determine the rate at which a pathogen outbreak progresses. However, we can re-write the equation to describe the amount of time it takes for the population to double.

$$t_2 = \ln(2)/\lambda \tag{4}$$

The value of λ and t_2 are typically determined on the CDF from the 10% to 90% rise time locations.

1.7 Influenza Analysis

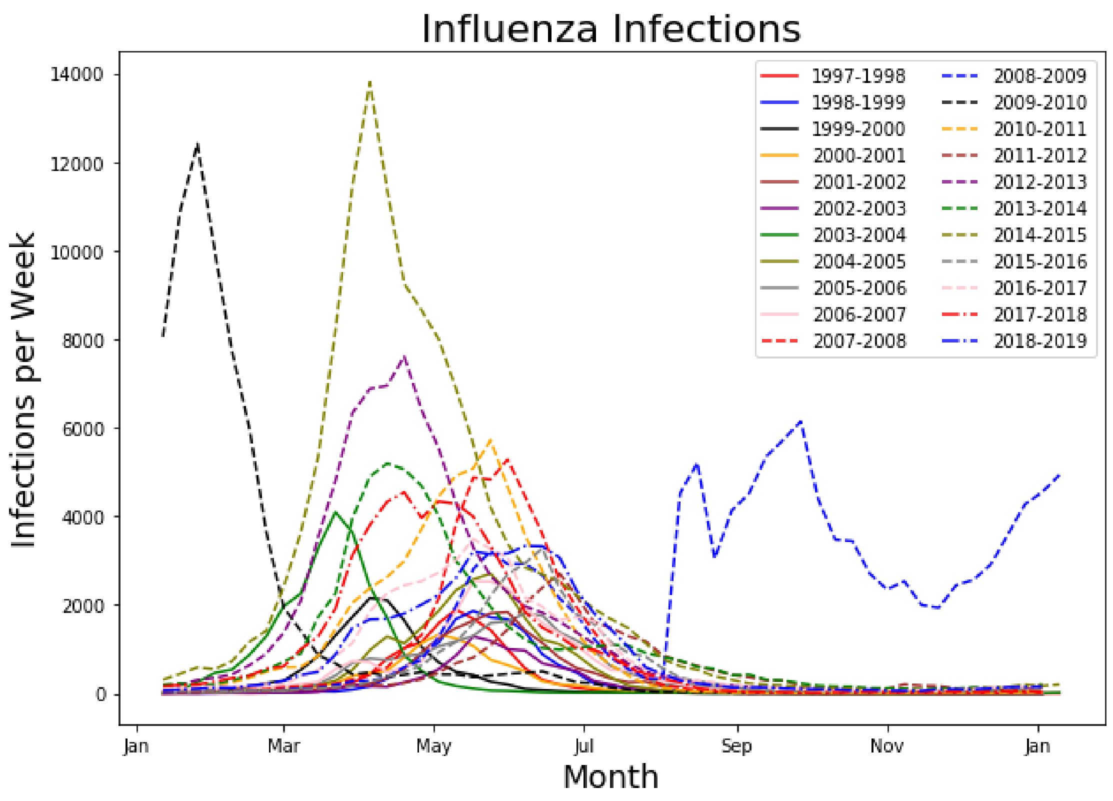
1.7.1 Influenza Probability Distribution Functions

The U.S. Center for Disease Control estimates that every year between 39 and 200 million American are infected with the flu, which normally consists of the H1 and H3 variants of the A virus, the

B-virus, to include its Yamagoto and Victoria strains, as well as the H1N1 and H3V2 variants of the pathogen. While millions of American's are infected, the symptoms felt by most are so minor that most do not seek medical attention and are therefore not officially diagnosed. Regardless, we can still learn allot by evaluating the database for those who were officially diagnosed. The following code snippet mines the `US_Flu_Infections.csv` database for the number of weekly diagnosed infections ranging from the 1997-1998 flu season untl the current 2019-2020 flu season. When plotted, the diognoses rate constitutes a probability distribution function which is plotted in a figure below. While there are some variations on the infectious period, it is clear that most flu seasons begin around late February, peak in early June and end by September, and for the most part follows a Gaussian distribution. The 2008-2009 and 2009-2010 seasons represent a rare outbreak of the H1N1 strain of the virus, and the 2014-15 season also represents an outbreak above and beyond that of an average flu season.

[4]:
```
# Instantiate class to read database
flu = ProcessFluInfections('Data/Processed_Data/US_Flu_Infections.csv')

# Determine how many unique seaons exist in the database and their names
total_df = flu.read_data()
seasons = np.array(list(total_df['SEASON']), str)
seasons = np.unique(seasons)

# Create a list of data-frames, one for each season
df = [flu.filter_by_season(seas) for seas in seasons]

# - The influenza dataframes contain a column for each type of flu,
#   we will now create a colum that accounts for all influenza
#   infections
for i in range(len(df)):
    df[i]['Total Infections'] = df[i]['A (H1+H3)'] + df[i]['H3N2v'] + df[i]['A␣
↪(2009 H1N1)'] + \
                                df[i]['B (B+BVic+BYam)']

# Create parameters required for plotting
dates = [convert_to_m_d(data) for data in df]
colors = ['red', 'blue', 'black', 'orange', 'brown',
          'purple', 'green', 'olive', 'gray', 'pink',
          'red', 'blue', 'black', 'orange', 'brown',
          'purple', 'green', 'olive', 'gray', 'pink',
          'red', 'blue', 'black']
linestyles = np.repeat('-', 10)
lines = np.array(['-.', '-.', '-.'])
linestyles = np.concatenate((linestyles, np.repeat('--', 10)))
linestyles = np.concatenate((linestyles, lines))

# Plot data
plt.rcParams["figure.figsize"] = (10, 7)
fig, td_plot = plt.subplots()
```

```
td_plot.set_xlabel('Month', fontsize=18)
td_plot.set_ylabel('Infections per Week', fontsize=18)
td_plot.set_title('Influenza Infections', fontsize=22)
td_plot.xaxis.set_major_formatter(mdates.DateFormatter('%b'))
for i in range(len(df) - 1):
    td_plot.plot(dates[i], df[i]['Total Infections'], color=colors[i],
                 linestyle=linestyles[i], label=seasons[i])
plt.legend(ncol=2)
plt.show()
plt.close()
```



1.7.2 Peak Infection Rates

We can now process the database to determine the peak number of weekly infections that occured in each flu season (r_p). It appears that the 2014-2015 season produced the highest number, with a peak value of 13,798 infections in one week. On the other hand, the 2002-2003 year produced the lowest peak weekly infection value of 1,285 infections in one week. From this data we will use the average peak value for r_p to use in the determination of the $0.1r_p$ and $0.9r_p$ values. The data shows an average peak infection rate of 4315 infections per week, which results in a $0.1r_p$ value of 431 infections per week. The $0.9r_p$ value will be 431 infections less than the peak value in that flu season. It should be noted that the 2019-2020 flu season is still in progress, and will likely not peak until early June.

[5]:
```
# Drop the 2019-2020 flu season since it is strill in progress
total = [max(df[i]['Total Infections']) for i in range(len(df) - 1)]
new_seasons = [seasons[i] for i in range(len(df) - 1)]

data = {'Season': new_seasons, 'rp (infec/week)': total}
display(pd.DataFrame(data))
print('')
print('{}{:7.2f}'.format('Average Peak Weekly Flu Infections: ', int(np.
 ↪average(total))))
print('{}{:7.2f}'.format('Maximum Peak Weekly Flu Infections: ', np.max(total)))
print('{}{:7.2f}'.format('Minimum Peak Weekly Flu Infections: ', np.min(total)))
```

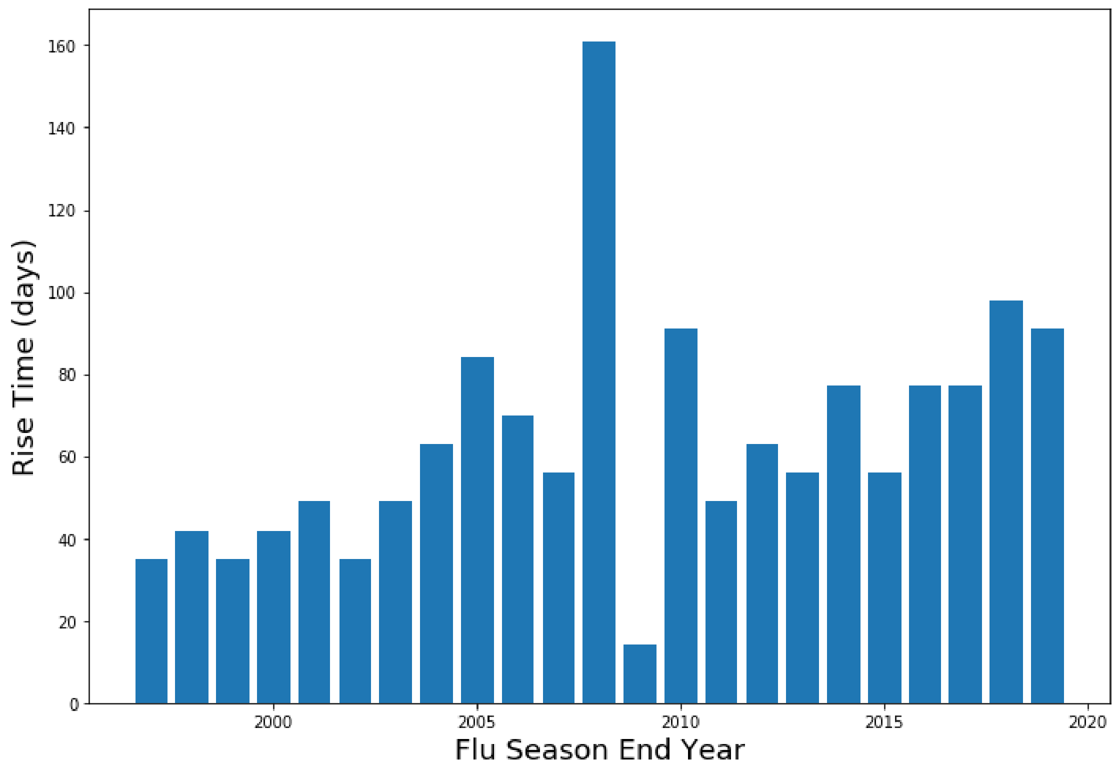
```
       Season  rp (infec/week)
0   1997-1998             1889
1   1998-1999             1860
2   1999-2000             2151
3   2000-2001             1330
4   2001-2002             1833
5   2002-2003             1285
6   2003-2004             4090
7   2004-2005             2705
8   2005-2006             1802
9   2006-2007             2532
10  2007-2008             5277
11  2008-2009             6140
12  2009-2010            12409
13  2010-2011             5716
14  2011-2012             2710
15  2012-2013             7602
16  2013-2014             5187
17  2014-2015            13798
18  2015-2016             3274
19  2016-2017             3482
20  2017-2018             4538
21  2018-2019             3326
```

```
Average Peak Weekly Flu Infections: 4315.00
Maximum Peak Weekly Flu Infections: 13798.00
Minimum Peak Weekly Flu Infections: 1285.00
```

1.7.3 Influenza Rise Time

The following code will determine the rise time for each season, which is defined as the time that elapses between the 10% pulse height and the 90% pulse height. This term defines hte length of the unmitigated outbreak phase. Following the 90% pulse height location, the rate at which the number of infections is increasing on a per week basis begins to decrease, until the trend goes negative at the peak heigh time. The pulse rise times vary from 14 days to 161 days, with 63 days being the average. The plot below is one where a small value indicates a very infectous disease.

```
[6]: def rise_time(threshold: int, df: pd.DataFrame) -> int:
         """

         :param threshold: The threshold number of infections in a one
                           week timeframe that can be used as the estimate
                           for the outbreak beginning date and end date
         :param df: A pandas dataframe containing a date index and
                    a 'Total Infections' column
         :return days: The pulse rise time in days
         """
         peak_value = max(df['Total Infections'])
         for i in range(len(df)):
             if df['Total Infections'][i] >= threshold:
                 dat1 = df.index[i]
                 break
         for i in range(len(df)):
             if df['Total Infections'][i] >= peak_value - threshold:
                 dat2 = df.index[i]
         return (dat2 - dat1).days
     # ==============================================================================
     # ==============================================================================

     years = np.arange(1997, 2020)
     rise = [rise_time(int(np.average(total)) / 10, df[i]) for i in range(len(df))]

     fig, ax = plt.subplots()
     dat = ax.bar(years, rise)
     ax.set_ylabel('Rise Time (days)', fontsize=18)
     ax.set_xlabel('Flu Season End Year', fontsize=18)
     fig.tight_layout()
     plt.show()
     plt.close()
     print('')
     print('{}{:7.2f}'.format('Average Influenza Rise Time (days): ', int(np.
      ↪average(rise))))
     print('{}{:7.2f}'.format('Maximum Influenza Rise Time (days): ', np.max(rise)))
     print('{}{:7.2f}'.format('Minimum Influenza Rise Time (days): ', np.min(rise)))
```

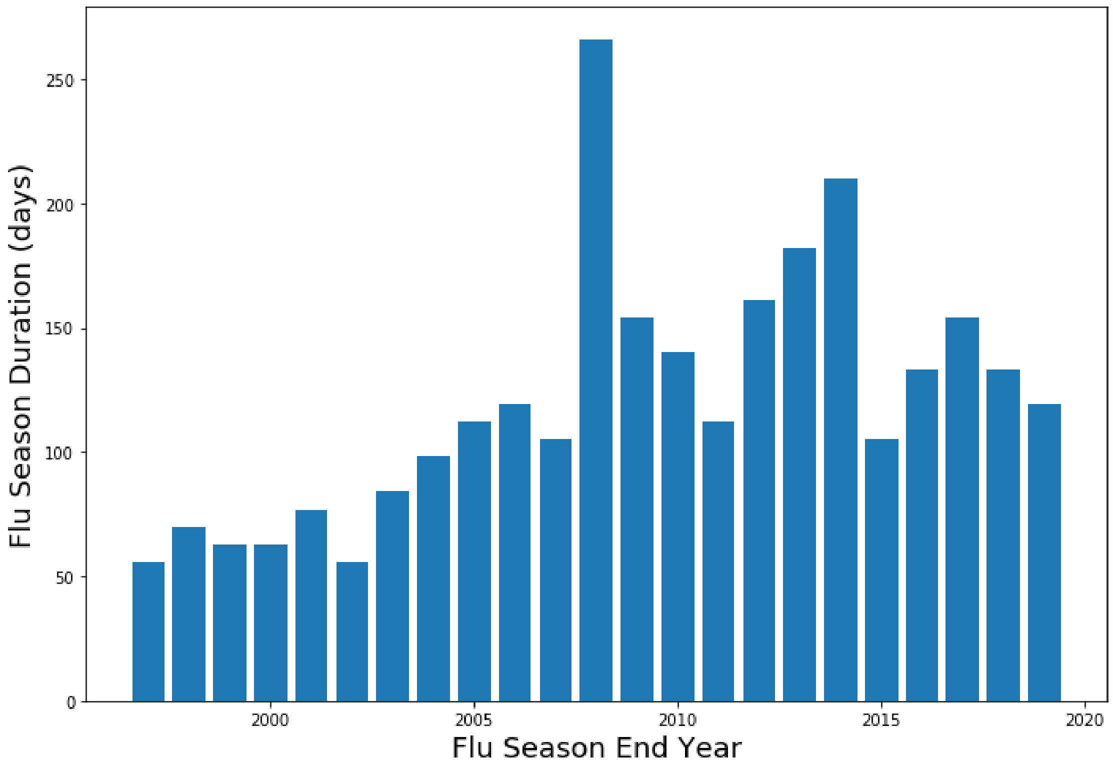


```
Average Influenza Rise Time (days):   63.00
Maximum Influenza Rise Time (days):  161.00
Minimum Influenza Rise Time (days):   14.00
```

1.7.4 Duration of Influenza Outbreaks

The following code will determine the length of each influenza outbreak (t_d) which is defined as the time that elapses between the 10% pulse height on the rise side and the decay side of the pulse. As can be seen in the code output, historical flu outbreaks have lasted anywhere from 56 days, to 266 days, where 120 days is the historical average duration. Interestingly, the outbreak length for each flu season appears to be increasing as a function of time. These numbers can be important in making data based estimates for the duration of future flu strains or flu like illnesses. **While future strains of influenza or other illnesses can be very different, these numbers are helpful for societal planning during an outbreak.**

[7]:
```
def outbreak_length(threshold: int, df: pd.DataFrame) -> int:
    """

    :param threshold: The threshold number of infections in a one
                      week timeframe that can be used as the estimate
                      for the outbreak beginning date and end date
    :param df: A pandas dataframe containing a date index and
               a 'Total Infections' column
    :return days: The number of days that an outbreak lasts based
                  on a threshold infection rate
    """
    for i in range(len(df)):
        if df['Total Infections'][i] >= threshold:
            dat1 = df.index[i]
            break
    j = len(df) - 1
    for i in range(len(df)):
        if df['Total Infections'][j] >= threshold:
            dat2 = df.index[j]
            break
        j -= 1
    return (dat2 - dat1).days
# ===============================================================================
# ===============================================================================

# Execute the function with the flu infection database
flu_length = [outbreak_length(int(np.average(total)) / 10, df[i]) for i in␣
 ↪range(len(df))]

fig, ax = plt.subplots()
dat = ax.bar(years, flu_length)
ax.set_ylabel('Flu Season Duration (days)', fontsize=18)
ax.set_xlabel('Flu Season End Year', fontsize=18)
#ax.set_yticks()
fig.tight_layout()
plt.show()
plt.close()

print('')
print('{}{:7.2f}'.format('Average Flu Outbreak Duration (days): ', int(np.
 ↪average(flu_length))))
print('{}{:7.2f}'.format('Maximum Flu Outbreak Duration (days): ', np.
 ↪max(flu_length)))
print('{}{:7.2f}'.format('Minimum Flu Outbreak Duration (days): ', np.
 ↪min(flu_length)))
```



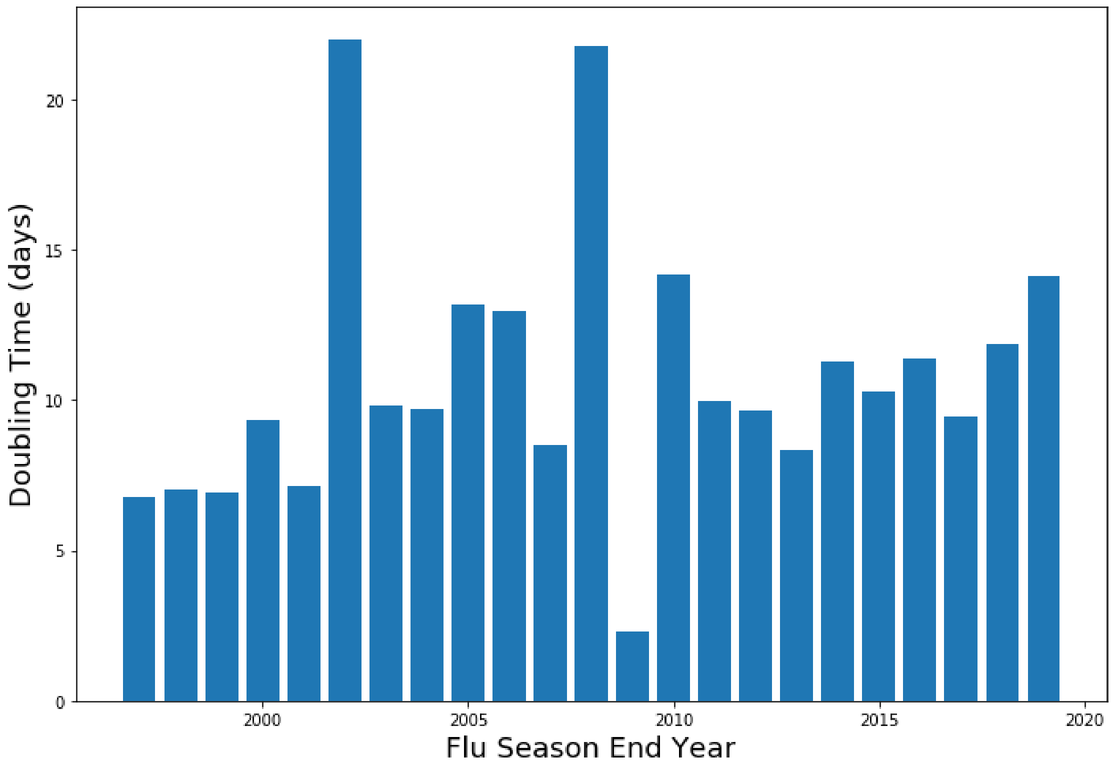
```
Average Flu Outbreak Duration (days):  120.00
Maximum Flu Outbreak Duration (days):  266.00
Minimum Flu Outbreak Duration (days):   56.00
```

1.8 Influenza Exponential Constant and Doubling Time

The code below is used to determine the values of λ and t_2 for influenza outbreaks, which helps us understand tha rate at which a pathogen reproduces and infects hosts. On average the influenza virus is capable in doubling the number of infected patients once every 8 days; however, history shows that it can double the number of diagnosed patients in as little as 2 days. The doubling time presented in this analysis is just for those who were formally diagnosed. If the true number of infected people were known along with the date they were infected, it is likely that the doubling time would be shorted than what is presented in this analysis.

```
[8]: def exp_growth(threshold: int, df: pd.DataFrame) -> float:
         """

         :param threshold: The threshold number of infections in a one
                           week timeframe that can be used as the estimate
                           for the outbreak beginning date and end date
         :param df: A pandas dataframe containing a date index and
```

```
                a 'Total Infections' column
    :return _lambda: A metric that describes the rate at which the
                     number of known infections doubles
    """
    peak_value = df['Total Infections'].max()
    for i in range(len(df)):
        if df['Total Infections'][i] >= threshold:
            date1 = df.index[i]
            break
    for i in range(len(df)):
        if df['Total Infections'][i] >= peak_value - threshold:
            date2 = df.index[i]
            break
    new_df = df[(df.index >= date1) & (df.index <= date2)]
    upper = new_df['Total Infections'].sum() - threshold
    lower = threshold
    _lambda = np.log((upper- threshold) / lower) / (date2 - date1).days
    return _lambda
# ==============================================================================
# ==============================================================================

# Execute function
_lambda = [exp_growth(int(np.average(total)) / 10, df[i]) for i in␣
 ↪range(len(df))]

fig, ax = plt.subplots()
dat = ax.bar(years, np.log(2) / _lambda)
ax.set_ylabel('Doubling Time (days)', fontsize=18)
ax.set_xlabel('Flu Season End Year', fontsize=18)
#ax.set_yticks()
fig.tight_layout()
plt.show()
plt.close()

print('')
print('{}{:6.3f}{}'.format('Average Influenza Doubling Rate: ', np.log(2) / np.
 ↪average(_lambda), ' days'))
print('{}{:6.3f}{}'.format('Peak Influenza Doubling Rate:    ', np.log(2) / np.
 ↪max(_lambda), ' days'))
print('{}{:6.3f}{}'.format('Minimum Influenza Doubling Rate: ', np.log(2) / np.
 ↪min(_lambda), ' days'))
```



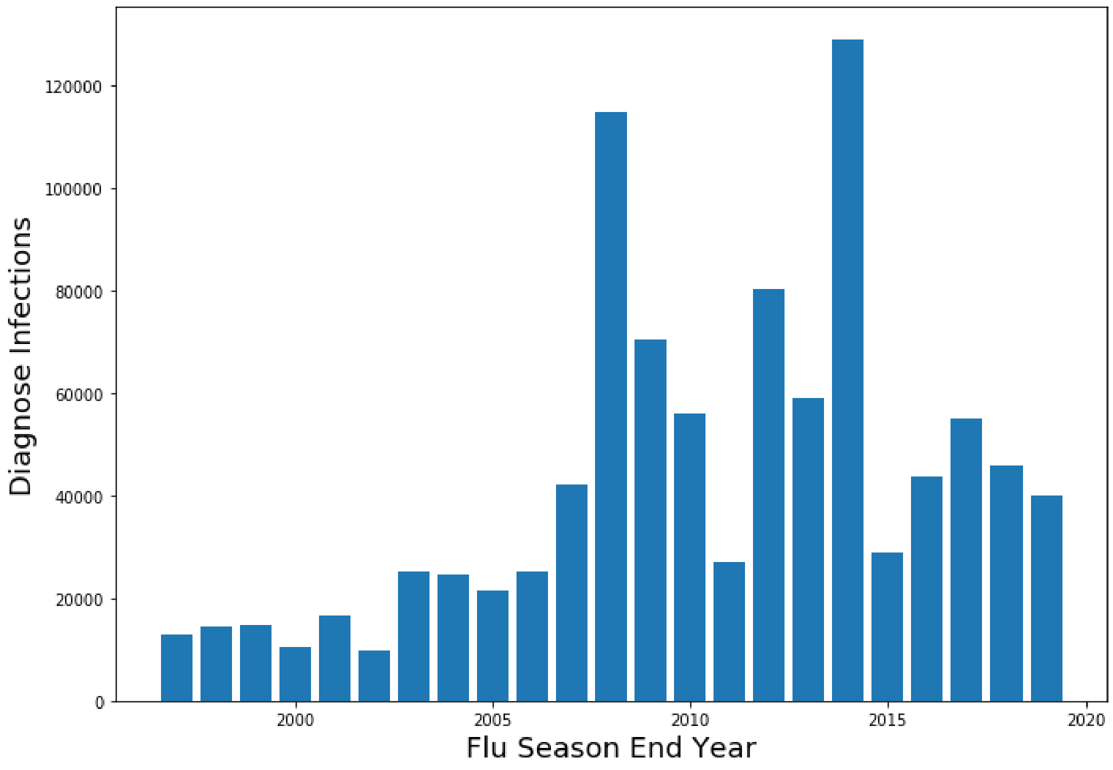
```
Average Influenza Doubling Rate:   8.790 days
Peak Influenza Doubling Rate:      2.279 days
Minimum Influenza Doubling Rate: 21.996 days
```

1.8.1 Cumulative Distribution Function

Instead of generating the time dependant cumulative distribution, the total number of diagnosed infections in a flu seas is determined with the `total_numbers()` python function below and plotted as a function of time. The data is plotted as a bar chart as a function of flu season, where each year represents the year that the plotted flu season ended. The data indicates that on average approximately 42,000 American are diagnosed with influenza every year, with a peak number of 128,000 diagnosed infections in the 2014-2015 flu season. The author of this report strongly urges the reader to remember that these are only the diagnosed infections. Each flu season likely has many un-diagnosed cases for each diagnosed case.

```
[9]: def total_numbers(df: pd.DataFrame) -> int:
         """

         :param df: A pandas dataframe containing total number
                    of infections
         :return summation: The total number of infections
```

```
    """
    return df['Total Infections'].sum()
# ==============================================================================
# ==============================================================================

# Execute function
cumulative = [total_numbers(df[i]) for i in range(len(df))]
years = np.arange(1997, 2020)

# Plot data
fig, ax = plt.subplots()
dat = ax.bar(years, cumulative)
ax.set_ylabel('Diagnose Infections', fontsize=18)
ax.set_xlabel('Flu Season End Year', fontsize=18)
#ax.set_yticks()
fig.tight_layout()
plt.show()
plt.close()
print('')
print('{}{:6.3f}'.format('Average Diagnosed Flu Infections: ', np.
 ↪average(cumulative)))
print('{}{:6.3f}'.format('Peak Diagnose Flue Infections:    ', np.
 ↪max(cumulative)))
print('{}{:6.3f}'.format('Minimum Diagnose Flu Infections : ', np.
 ↪min(cumulative)))
```



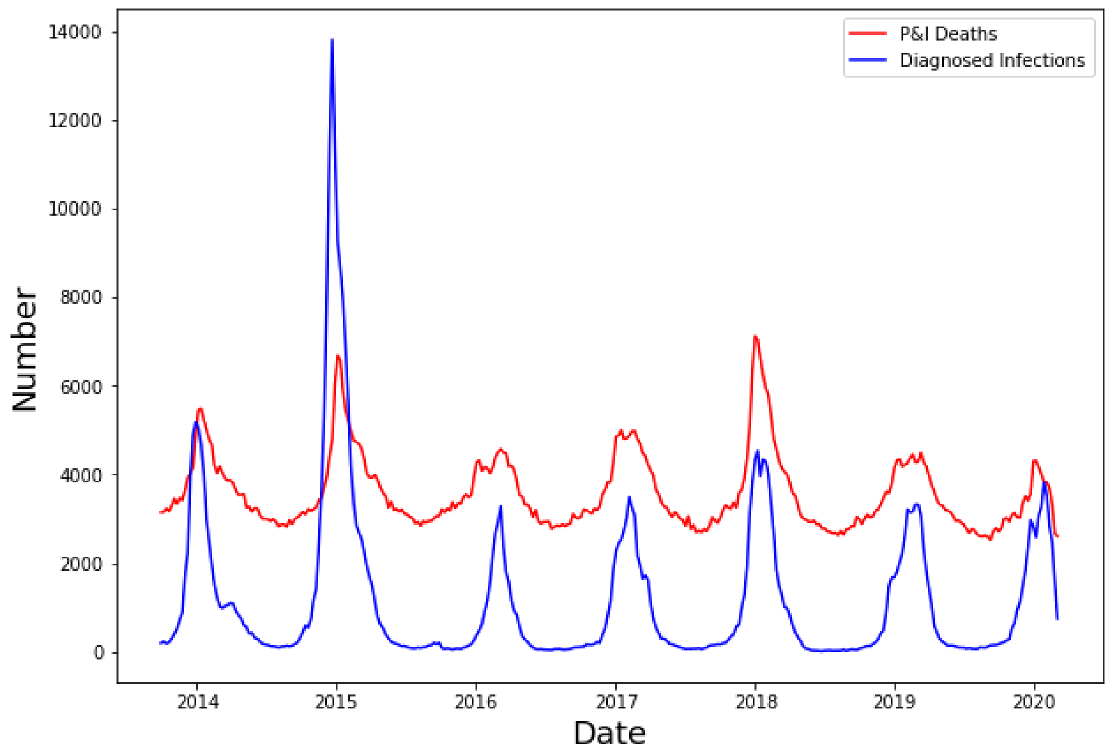
```
Average Diagnosed Flu Infections: 42016.826
Peak Diagnose Flue Infections:    128915.000
Minimum Diagnose Flu Infections : 9841.000
```

1.8.2 Mortality Analysis

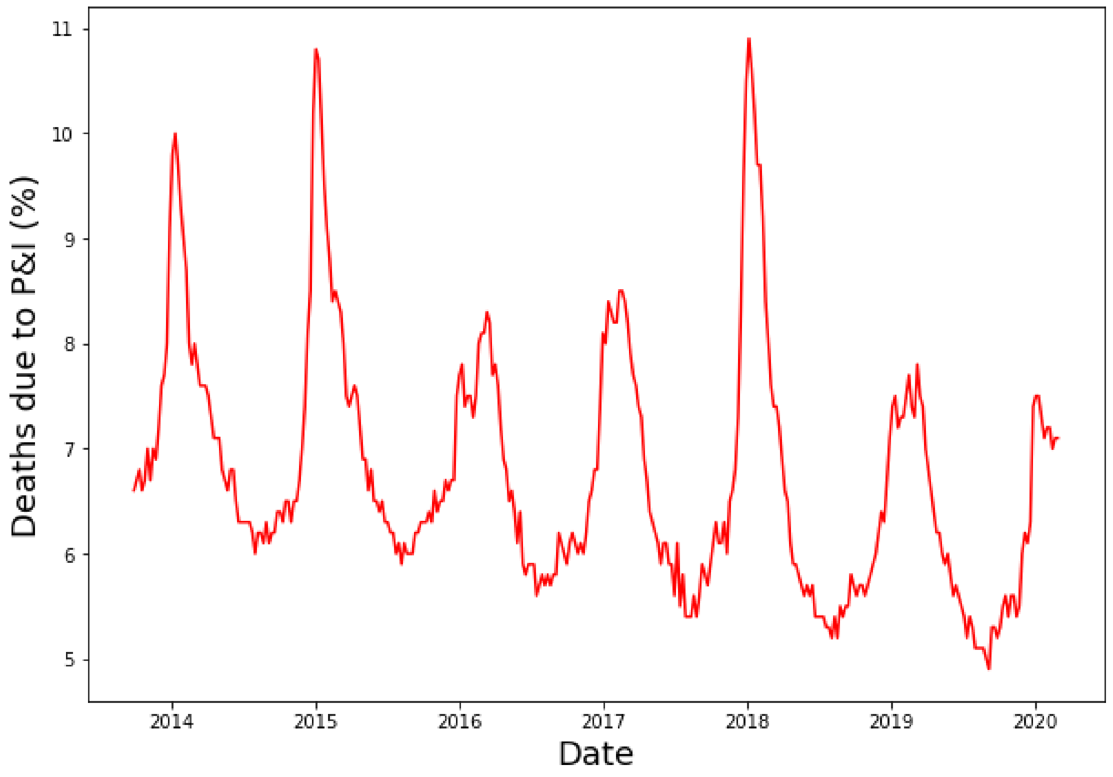
The CDC maintains the influenza database with the number of deaths that are known to have been caused by the flu as well as the total number of pneumonia related deaths, ostencibly because some fraction of the pneumonia deaths occured as a side effect to an influenza infection. The plot below shows an overlay of the number of diagnose influenza patients on a per week basis with the number of pneumonia and influenza (P&I) related deaths, also on a per week basis. The plot below the overlaid plot demonstrates the fraction of all deaths that occur in the United States due to P&I related illnesses on a per wekk basis.

```
[10]: # Read in the mortality database
flu2 = ProcessFluMortality('Data/Processed_Data/US_Flu_Mortality.csv')
total_df2 = flu2.read_data()

# Prepare relevant aspects of infection and mortality databases for analysis
total_df2['Total P&I Deaths'] = total_df2['NUM INFLUENZA DEATHS'] +␣
 ↪total_df2['NUM PNEUMONIA DEATHS']
```

```
total_df['Total Infections'] = total_df['A (H1+H3)'] + total_df['H3N2v'] +␣
 ↪total_df['A (2009 H1N1)'] + \
                               total_df['B (B+BVic+BYam)']

# Create dataframes of the same size
mask = (total_df.index > '2013-09-22') & (total_df.index <= '2020-03-01')
new_df = total_df[mask]

# Combine relevant aspects of both dataframes into one
data = {'Date': new_df.index, 'Infections': new_df['Total Infections'],
        'Deaths': total_df2['Total P&I Deaths']}
final_df = pd.DataFrame(data)
final_df['Date'] = pd.to_datetime(final_df['Date'])
final_df = final_df.set_index('Date')

# Plot data
plt.rcParams["figure.figsize"] = (10, 7)
fig, td_plot = plt.subplots()
td_plot.set_xlabel('Date', fontsize=18)
td_plot.set_ylabel('Number', fontsize=18)
td_plot.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
td_plot.plot(final_df.index, final_df['Deaths'], color='red',
             linestyle='-', label='P&I Deaths')
td_plot.plot(final_df.index, final_df['Infections'], color='blue',
             linestyle='-', label='Diagnosed Infections')
plt.legend()
plt.show()
plt.close()

plt.rcParams["figure.figsize"] = (10, 7)
fig, td_plot = plt.subplots()
td_plot.set_xlabel('Date', fontsize=18)
td_plot.set_ylabel('Deaths due to P&I (%)', fontsize=18)
td_plot.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
td_plot.plot(total_df2.index, total_df2['PERCENT P&I'], color='red',
             linestyle='-')
plt.show()
plt.close()
```



P&I Deaths
Diagnosed Infections
Number
Date



Deaths due to P&I (%)
Date

There is obviously a correlation between the number of diagnosed influenza patients and the number of deaths that occur to pneumonia and influenza related symptoms. Below we run a correlation on the two trends and find that they are firmly correlated with a correlation value of 0.766. This correlation is largely driven by the inflection points of the two trends, however, the geometry is not identical, which is why the value is firm, but still only 0.766. As we see the number of diagnosed flu patients essentially reaches zero near the end of each year, which helps us understand that number of P&I deaths during the same time frame probably has little to do with influenza. The inflection point in P&I deaths at the end of each year ranges from approximately 5% to 6.5% of all fatalities, and the value seems to be decreasing as a function of time. For this study we will assume the average value of 5.46%, which means that on average 5.46% of all P&I deaths are unrelated to Influenza and all deaths above that percentage will be assumed as being related to the flu.

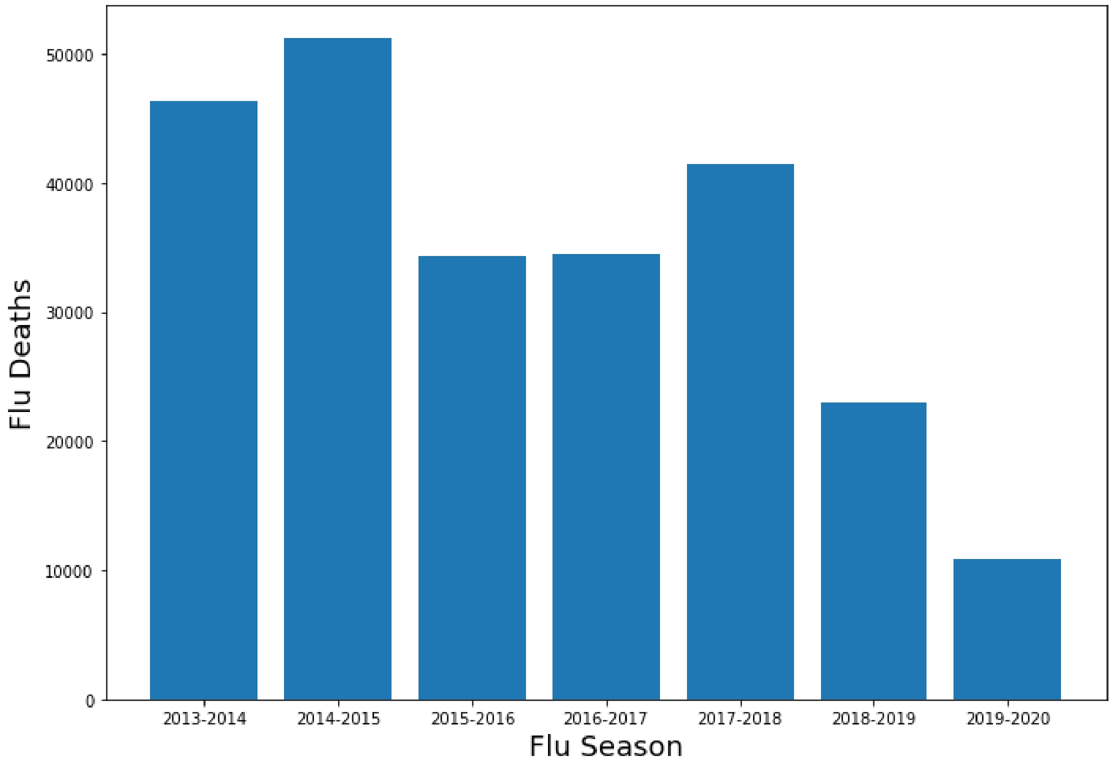
```
[11]: print('The correlation factor between infections and deaths is shown below')
display(final_df.corr())
# Determine which seasons are in the database
seasons2 = list(np.unique(total_df2['SEASON']))

# Create mortality dataframe for each flu season
mort = [flu2.filter_by_season(seasons) for seasons in seasons2]

# Determine minimum percentage in each season
minimum = [dataframe['PERCENT P&I'].min() for dataframe in mort]

print('{}{:4.2f}{}'.format("The Average Minima is: ", np.average(minimum), "␣
 ↪%"))
print('{}{:4.2f}{}'.format("The Minima sigma is. : ", np.std(minimum), " %"))
# Find local peaks
```

```
The correlation factor between infections and deaths is shown below

            Infections    Deaths
Infections    1.000000  0.765705
Deaths        0.765705  1.000000


The Average Minima is: 5.46 %
The Minima sigma is. : 0.37 %
```

The following code subtracts 5.46% from all P&I deaths as a representation of deaths that have a high probability of having been caused by influenza. The numbers range from a low value in the 2018-2019 flu season of 23,017 fatalities to a high value in the 2014-2015 season of 51,241 deaths. The 2019-2020 flu season is still in progress so it is not considered in the totals. The figures presented in the figure below only represent nominal figures, in reality the actualy death toll could vary by values that range from 9,525 to 10,444 deaths, depending on the year. Unfortunately since many pneumonia related deaths were not correctly attributed to influenza, it is impossible to put together a meaningful case fatality rate P_d and the rates appear to range from 20% to 119%, which obviously cannot be correct. All we can draw from this data is that there are between 13,000 to

61,000 deaths annualy in the United States due to influenza, which accounts for approximately 5.09 to 5.83 % of all deaths in the country each year.

[12]:
```python
def influenza_deaths(df: pd.DataFrame, threshold: int,
                     uncertainty: int) -> pd.DataFrame:
    """

    :param df: A pandas dataframe containing mortality
               related information
    :param threshold: The trheshold for what deaths should
                      not be considered
    :param uncertainty: The uncertainty in the mortality
                        threshold
    :return df: A dataframe containing the number of
                flu related deaths and the uncertainty
                in the mortality estimate
    """
    total = []
    unc = []
    for i in range(len(df)):
        if df["PERCENT P&I"][i] - threshold<= 0.0:
            percent = 0
        else:
            percent = df["PERCENT P&I"][i] - threshold
        total.append(int(percent / 100 * df["TOTAL DEATHS"][i]))
        unc.append(int(uncertainty/100 * df["TOTAL DEATHS"][i]))
    total = np.array(total, int)
    unc = np.array(unc, int)
    df["Influenza Deaths"] = total
    df["Uncertainty"] = unc
    return df
# ==============================================================================
# ==============================================================================

mortality = [influenza_deaths(mortality, 5.46, 0.37) for mortality in mort]

# Determine how many influenza related deaths occur each season
total_deaths = [dat["Influenza Deaths"].sum() for dat in mortality]
uncertainty = [dat["Uncertainty"].sum() for dat in mortality]

fig, ax = plt.subplots()
dat = ax.bar(seasons2, total_deaths)
ax.set_ylabel('Flu Deaths', fontsize=18)
ax.set_xlabel('Flu Season', fontsize=18)
#ax.set_yticks()
fig.tight_layout()
plt.show()
plt.close()

print('')
print('{}{:8.2f}'.format("Average Num Flu Deaths Per Year:  ", np.
 ↪average(total_deaths)))

# Develop estimate of case fatality rate
cases = np.array(cumulative[16:], int)
total = np.array(total_deaths, int)
data = {'Season': seasons2, 'Case Fatality Rate': total_deaths / cases,
        'Deaths': total_deaths}
display(pd.DataFrame(data))
```



```
Average Num Flu Deaths Per Year:  34531.43

      Season  Case Fatality Rate  Deaths
0  2013-2014            0.785791   46353
1  2014-2015            0.397479   51241
2  2015-2016            1.194516   34328
3  2016-2017            0.791945   34490
4  2017-2018            0.752846   41393
5  2018-2019            0.501635   23017
```

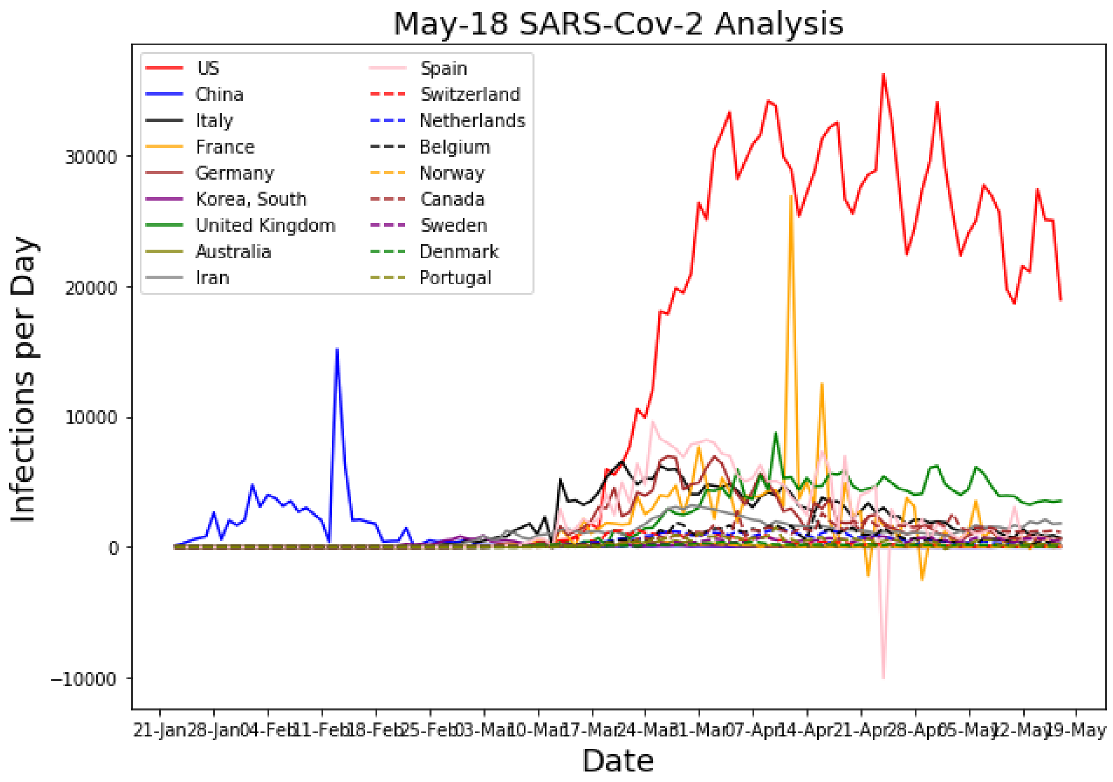
```
6   2019-2020              0.272062    10898
```

1.9 SARS-Cov-2 Analysis

1.10 SARS Infection Rate Probability Distribution Function

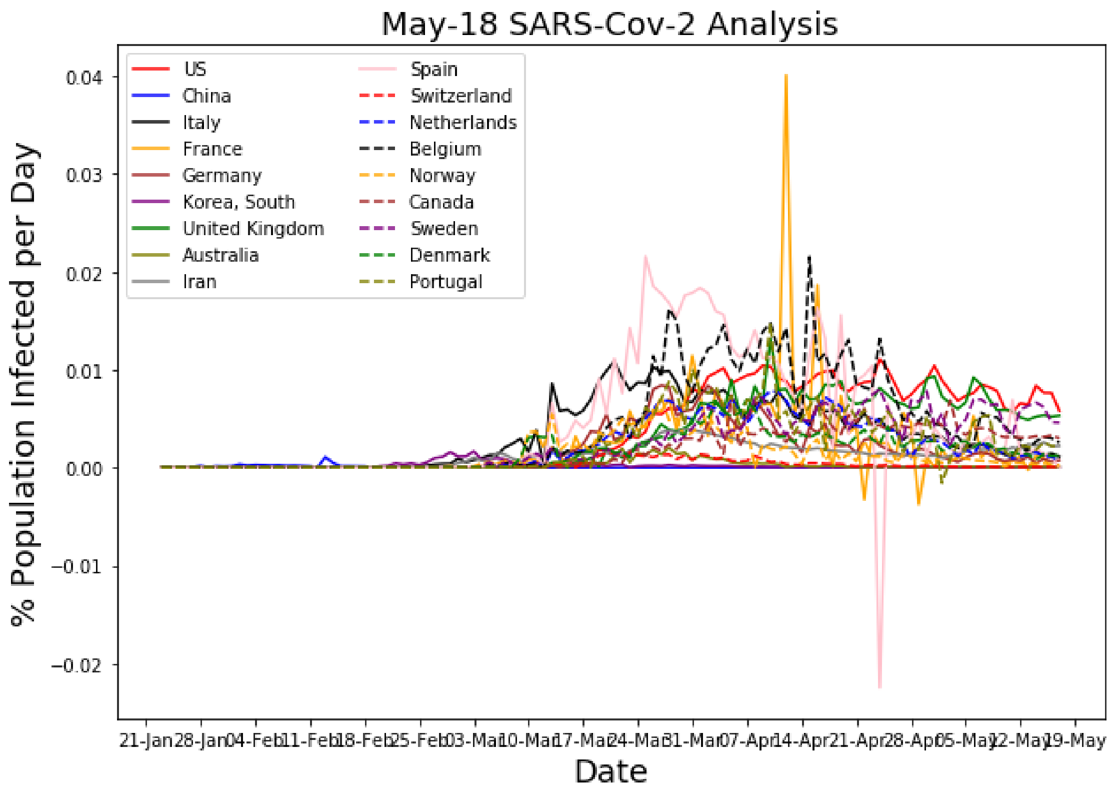
The SAR-Cov-2 virus is still in the early phases of its infectious profile, which makes it imposible to build a complete PDF from the data. However, what we can do is display the number of new infections on a day by day basis. Once the infection progresses further we can start to view it on a weekly basis in order to better compare it with influenza. However, there are some factors that will inherently make the analysis different. Early on in the infection profile of SARS-Cov-2, there was very limited testing capability around the globe, so it is likely that the early number of diagnosed infections per day will be below that of Influenza. However, SARS-Cov-2 has become headline news across the globe, so it is likely that once testing begins in full swing, a larger fraction of the population will undergo testing, which will likely lead to a much higher rate of diagnosed cases than for influenza. This does not nessarily mean that there are more or less SARS-Cov-2 infections than influenza, just that a larger fraction of the infected population is getting tested. However, early results do indicate that SARS-Cov-2 is more than twice as infectuous of the common strains of influenza, as is discussed in this NPR article. The plot shown below shows the number of daily infections for the countries most affected by the virus, which are mostly in Europe and the United States. At present, the United States is diagnosing approximately 20,000 new SARS-Cov-2 patients per day. Within the United States the number of new infections per day seems to be decreasing with a linear trend, where the number of daily infections varies stochasticaly around the mean value. As with Europe, the number of new daily infections in the United States seems to have plateaued and is fluctuating within statistical bounds on a daily basis.

```
[13]: # Read in the confirmed infection database
cov = ProcessCovidData('Data/Raw_Data/time_series_19-covid-Confirmed.csv')

countries = ['US', 'China', 'Italy', 'France', 'Germany', 'Korea, South',
             'United Kingdom', 'Australia', 'Iran', 'Spain', 'Switzerland',
             'Netherlands', 'Belgium', 'Norway', 'Canada', 'Sweden',
             'Denmark', 'Portugal']
infect = [cov.country_cases_per_day(country, start_date, end_date) for country␣
 ↪in countries]
current_date = pd.datetime.now().strftime('%b-%d')

# Create parameters required for plotting
colors = ['red', 'blue', 'black', 'orange', 'brown',
          'purple', 'green', 'olive', 'gray', 'pink',
          'red', 'blue', 'black', 'orange', 'brown',
          'purple', 'green', 'olive', 'gray', 'pink',
          'red', 'blue', 'black']
linestyles = np.repeat('-', 10)
lines = np.array(['-.', '-.', '-.'])
linestyles = np.concatenate((linestyles, np.repeat('--', 10)))
linestyles = np.concatenate((linestyles, lines))
ttle = str(current_date) + " SARS-Cov-2 Analysis"
plt.rcParams["figure.figsize"] = (10, 7)
fig, td_plot = plt.subplots()
td_plot.set_xlabel('Date', fontsize=18)
td_plot.set_ylabel('Infections per Day', fontsize=18)
td_plot.set_title(ttle, fontsize=18)
td_plot.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
for i in range(len(infect)):
    td_plot.plot(infect[i].index, infect[i], color=colors[i],
                 linestyle=linestyles[i], label=countries[i])
td_plot.xaxis.set_major_locator(mdates.WeekdayLocator(interval=1))
plt.legend(loc=2, ncol=2)
plt.show()
plt.close()
```



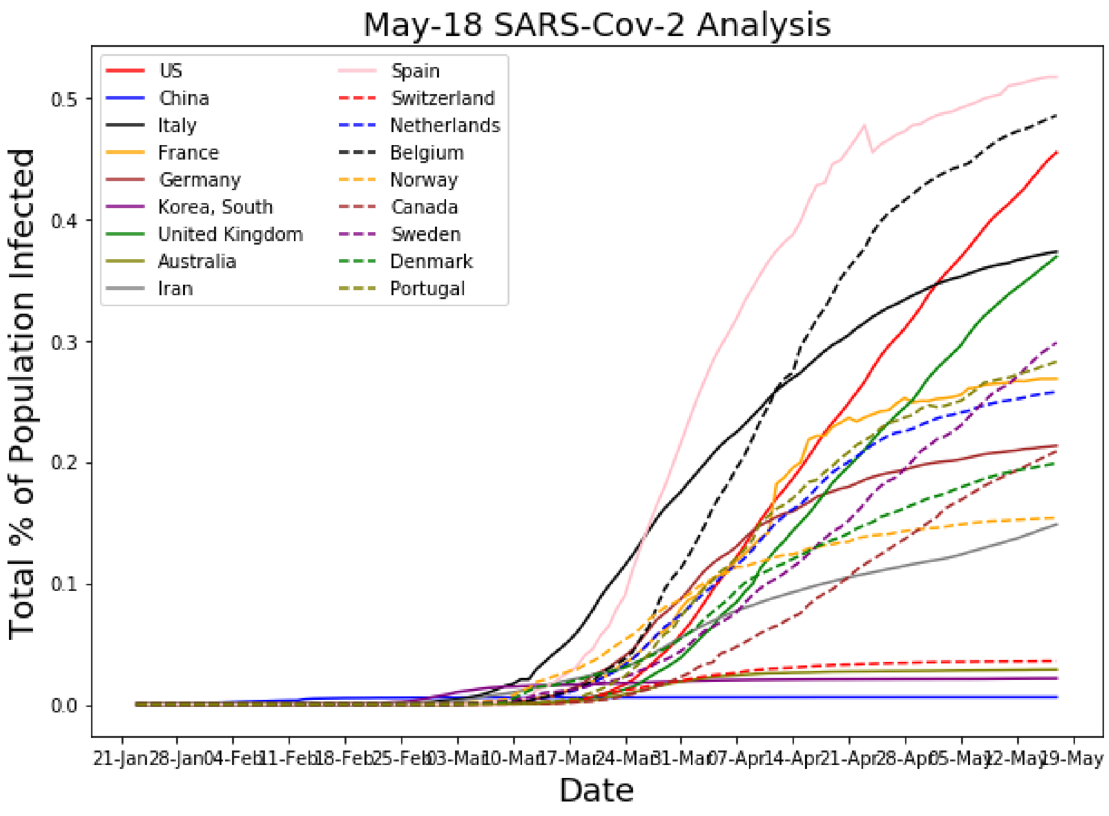
The above plot confirms that the time required for the diseases to spread from China to the rest of the world was approximately 3 months. The plot seems to indicate that there was a drop in the infection rate in late January; however, this is unlikely, and the drop most likely indicates a testing shortage during that time frame. Countires should not be compared on the total number of daily infections, but on the fraction of the population infected, as larger countries with high total populations and higher population densities will understandably have higher values. In the plot below we compare infection rates in countries normalized to their populations. The United States does appear to be one of the worst affected countries when normalized to population, which worse rates only in France, Spain and Belgium. If the average outbreak duration for a typical flu season is used as the standard, it implues that we may have about another 30 days of values existing at their plateau before we see large drops in numbers.

[14]:
```
population = {'US': 327200000.0, 'China': 1386000000.0, 'Italy': 60480000.0,␣
↪'France': 66990000.0,
              'Germany': 82790000.0, 'Korea, South': 51470000.0, 'United␣
↪Kingdom': 66440000.0,
              'Australia': 24600000.0, 'Iran': 81160000.0, 'Spain': 44660000.0,
              'Switzerland': 85700000.0, 'Netherlands': 17180000.0, 'Belgium':␣
↪11400000.0,
              'Norway': 5368000.0, 'Canada': 37590000.0, 'Sweden': 10120000.0,␣
↪'Denmark': 5603000.0,
              'Portugal': 10290000.0}

countries = ['US', 'China', 'Italy', 'France', 'Germany', 'Korea, South',
             'United Kingdom', 'Australia', 'Iran', 'Spain', 'Switzerland',
             'Netherlands', 'Belgium', 'Norway', 'Canada', 'Sweden',
             'Denmark', 'Portugal']

plt.rcParams["figure.figsize"] = (10, 7)
fig, td_plot = plt.subplots()
td_plot.set_xlabel('Date', fontsize=18)
td_plot.set_ylabel('% Population Infected per Day', fontsize=18)
td_plot.set_title(ttle, fontsize=18)
td_plot.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
for i in range(len(infect)):
    td_plot.plot(infect[i].index, (infect[i] / population[countries[i]]) * 100,␣
↪color=colors[i],
                 linestyle=linestyles[i], label=countries[i])
td_plot.xaxis.set_major_locator(mdates.WeekdayLocator(interval=1))
plt.legend(ncol=2)
plt.show()
plt.close()
```



The data shown for China seems to be highly Unusual. The disease started in China, and China has the largest population in the world, which would make one believe China should have the worst rates, which is contrary to the provided data. It is highly likely that the Chinese government is not being honest with the infection numbers, as well as the death rates.

[15]:
```
plt.rcParams["figure.figsize"] = (10, 7)
fig, td_plot = plt.subplots()
td_plot.set_xlabel('Date', fontsize=18)
td_plot.set_ylabel('Total % of Population Infected', fontsize=18)
td_plot.set_title(ttle, fontsize=18)
td_plot.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
for i in range(len(infect)):
    td_plot.plot(infect[i].index, (infect[i].cumsum() /␣
 ↪population[countries[i]]) * 100, color=colors[i],
                 linestyle=linestyles[i], label=countries[i])
td_plot.xaxis.set_major_locator(mdates.WeekdayLocator(interval=1))
plt.legend(ncol=2)
plt.show()
plt.close()
```



1.10.1 Duration of SARS-Cov-2 Outbreak

Thus far, China is the only country that appears to have completed an entire outbreak period; however, it is the author's opinion that the data coming from China may not be totally accurate. Regardless, we analyze the Chinese data to develop an estimate fo the outbreak duration t_d to find a value of 21 days. It should be noted that China completely locked down their country and put everyone in the Hebei province on quarantine to contain the outbreak. In the absense of a complete lockdown it is likely that an outbreak will last longer.

[16]:
```
def outbreak_length(threshold: int, df: pd.DataFrame) -> int:
    """

    :param threshold: The threshold number of infections in a one
                      week timeframe that can be used as the estimate
                      for the outbreak beginning date and end date
    :param df: A pandas dataframe containing a date index and
               a 'Total Infections' column
    :return days: The number of days that an outbreak lasts based
                  on a threshold infection rate
    """
    for i in range(len(df)):
        if df[i] >= threshold:
            dat1 = df.index[i]
            break
    j = len(df) - 1
    for i in range(len(df)):
        if df[j] >= threshold:
            dat2 = df.index[j]
            break
        j -= 1
    return (dat2 - dat1).days
# ==============================================================================
# ==============================================================================

duration = outbreak_length(infect[1].max() / 10.0, infect[1])
print('{}{}{}'.format('Chinese Outbreak Duration: ', duration, ' days'))
```

```
Chinese Outbreak Duration: 21 days
```

1.10.2 SARS-Cov-2 Exponential Factor and Doubling Time

In most countries the SARS-Cov-2 outbreak is still developing; however, there is enough data to start estimating the exponential factor λ and the doubling time t_2. The exponential factor and doubling time for each country are shown in the table below. The fastest influenza doubling time in the U.S. since 1997 was 2.27 days; however, the fraction of those infected with influenza and SARS-Cov-2 who were diagnosed is not known and it is likely that the fractions for the two illnesses is different. Until mid-March there was very limited testing capability in the United States; however, now that testing has began in the U.S. the number of people being diagnosed per day is hyper inflated compared to the number who were infected each day. This will make the current estimates for doubling time much lower than they should be, and indeed the doubling time in the United States has been getting longer for the past three days and this trend is expected to continue. Currently the doubling time for the United States is 2.52 days.

```
[17]: def exp_growth2(df: pd.DataFrame) -> float:
          """

          :param threshold: The threshold number of infections in a one
                            week timeframe that can be used as the estimate
                            for the outbreak beginning date and end date
          :param df: A pandas dataframe containing a date index and
                     a 'Total Infections' column
          :return a_lambda: A metric that describes the rate at which the
                            number of known infections doubles
          """
          peak_value = df.max()
          threshold = peak_value * 0.1
          for i in range(len(df)):
```

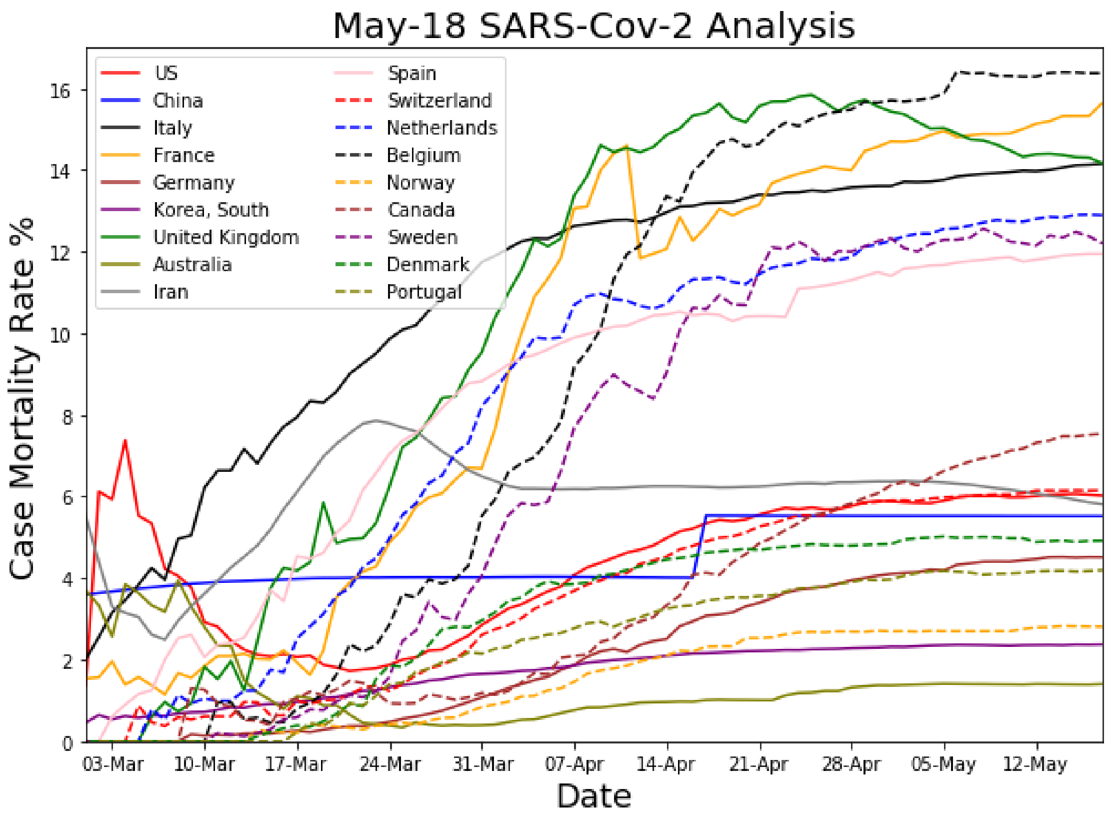
```
        if df[i] >= threshold:
            date1 = df.index[i]
            break
    for i in range(len(df)):
        if df[i] >= peak_value - threshold:
            date2 = df.index[i]
            break
    new_df = df[(df.index >= date1) & (df.index <= date2)]
    upper = new_df.sum() - threshold
    lower = threshold
    _lambda = np.log((upper- threshold) / lower) / (date2 - date1).days
    return _lambda

countries = ['US', 'China', 'Italy', 'France', 'Germany', 'Korea, South',
             'United Kingdom', 'Australia', 'Iran', 'Spain', 'Switzerland',
             'Netherlands', 'Belgium', 'Norway', 'Canada', 'Sweden',
             'Denmark', 'Portugal']
growth = [exp_growth2(inf) for inf in infect]

data = {'Country': countries, 'Doubling Time (days)': np.log(2) / growth,
        'Lambda': growth}
display(pd.DataFrame(data).sort_values(by=['Doubling Time (days)']))
```

```
           Country  Doubling Time (days)    Lambda
5     Korea, South              1.575219  0.440032
10     Switzerland              1.763804  0.392984
7        Australia              1.901584  0.364510
9            Spain              2.159744  0.320939
4          Germany              2.215651  0.312841
2            Italy              2.490713  0.278293
0               US              2.522728  0.274761
14          Canada              2.640138  0.262542
13          Norway              3.062272  0.226351
1            China              3.102785  0.223395
11     Netherlands              3.287605  0.210837
6   United Kingdom              3.318759  0.208857
17        Portugal              3.332605  0.207990
16         Denmark              3.779670  0.183388
3           France              3.831111  0.180926
12         Belgium              3.852480  0.179922
8             Iran              3.933965  0.176196
15          Sweden              5.749290  0.120562
```

1.10.3 SARS-Cov-2 Mortality Analysis

The case fatality rate is defined as the number of confirmed deaths divided by the number of diagnosed patients. The case mortality rate is expected to change as a larger number of patients become diagnosed with SARS-Cov-2. Normally the case mortality rate is expected to decrease as a function of time; however, the SARS-Cov-2 database shows the opposite trend. Most countries, with exception of Austrialia and the United States indicate a mortality rate that is increasing with time. This is an unexpected trend; however, we are still early in the progression of this illness, and the trends for most countries are still expected to decrease. At present Belgium has the highest case mortality rate of 16.37%, and Australia has the lowest rate of 1.40%. Currently the United States has a mortality rate of 6.02% and that number seems to be holding steady over time. Spain and Italy are tourist destinations and it is possible the virus was firmly embedded within those societies before its existance was made widely known. Due to a low birth rate, it is possible that the age distribution in Spain and Italy also play a role in the high infection and mortality rates as well as genetic and behavioral aspects indicitive to those nations; however, at present this is speculative. As predicted yesterday, the case fatality rate in the United States has increased. This trend is expected to continue for up to a week. The trend of increasing case fatality rate with time would imply that there is a forward peaked distribution of time to death from date of diagnosis.

[18]:
```
countries = ['US', 'China', 'Italy', 'France', 'Germany', 'Korea, South',
             'United Kingdom', 'Australia', 'Iran', 'Spain', 'Switzerland',
             'Netherlands', 'Belgium', 'Norway', 'Canada', 'Sweden',
             'Denmark', 'Portugal']

infect = [cov.total_cases_by_country(country, start_date, end_date) for country␣
 ↪in countries]

# Create databases of total deaths by country
cov2 = ProcessCovidData('Data/Raw_Data/time_series_19-covid-Deaths.csv')
deaths = [cov2.total_cases_by_country(country, start_date, end_date) for␣
 ↪country in countries]

plt.rcParams["figure.figsize"] = (10, 7)
fig, td_plot = plt.subplots()
td_plot.set_xlabel('Date', fontsize=18)
td_plot.set_ylabel('Case Mortality Rate %', fontsize=18)
td_plot.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
td_plot.set_xlim(datetime(2020, 3, 1, 00), datetime(2020, 5, 17))
td_plot.set_title(ttle, fontsize=20)
td_plot.set_ylim(0.0, 17.0)
for i in range(len(infect)):
    td_plot.plot(infect[i].index, (deaths[i] / infect[i]) * 100,␣
 ↪color=colors[i],
                 linestyle=linestyles[i], label=countries[i])
td_plot.xaxis.set_major_locator(mdates.WeekdayLocator(interval=1))
plt.legend(ncol=2)
plt.show()
plt.close()

total_deaths = []
total_infections = []
total_mortality = []
for i in range(len(deaths)):
    infct = infect[i][len(infect[0]) - 1]
    dth = deaths[i][len(deaths[i]) - 1]
    total_infections.append(infct)
    total_deaths.append(dth)
    total_mortality.append((dth * 100)/ infct)
data = {'Country': countries, 'Infections': total_infections,
        'Deaths': total_deaths, 'Case Fatality Rate %': total_mortality}
mortality_rate = pd.DataFrame(data)
display(mortality_rate.sort_values(by=['Case Fatality Rate %']))
```



```
           Country  Infections  Deaths  Case Fatality Rate %
7        Australia        7054      99              1.403459
5     Korea, South       11065     263              2.376864
13          Norway        8249     232              2.812462
17        Portugal       29036    1218              4.194793
```

```
4          Germany      176369    7962              4.514399
16         Denmark       11125     547              4.916854
1            China       84054    4638              5.517881
8             Iran      120198    6988              5.813741
0               US     1486757   89562              6.023984
10     Switzerland       30587    1881              6.149671
14          Canada       78332    5903              7.535873
9            Spain      230698   27563             11.947655
15          Sweden       30143    3679             12.205155
11     Netherlands       44195    5699             12.895124
2            Italy      225435   31908             14.153969
6   United Kingdom      244995   34716             14.170085
3           France      179693   28111             15.643904
12         Belgium       55280    9052             16.374819
```

1.11 Conclusions

At this point in time there is too much fluctiation in the SARS-Cov-2 information to make comparisons against the influenza infection. What we do know is that the infection seems to spread apprxoimately twice as fast as the influenza virus and in the most optimistic case it has a case mortality rate roughly twice that of the influenza virus. Once the trends in the SARS-Cov-2 infection period begin to stabilize we will be able to calculate the statistical range of possible infections and fatalities in the United States as well as elsewhere in the world. Until that point, people should not panic and instead take precautions to minimize their potential for acquiring the virus.