**Appendices**

## A Code Retrieving Historical Weather Data

```
'''
title: Get Historical Weather Data
author: Dharma Hoy
date: 08/04/2022

This program gets weather data from the following NOAA api. It
   requires a token that can be requested.
https://www.ncdc.noaa.gov/cdo-web/webservices/v2#gettingStarted

The code written was inspired by the code in the following link
https://towardsdatascience.com/getting-weather-data-in-3-easy-steps
   -8dc10cc5c859

Tucson Airport Station ID GHCND:USW00023160

Variables measured at this station to know about:
average tempreature 1998-present TAVG
average tempreature 1946-present TMAX
minimum tempreature 1946-present TMIN
average wind speed 1984-present AWND
Direction of fastest 2-minute wind 1996-present WDF2
Precipitation 1946-present PRCP
Average cloudiness sunrise to sunset from manual observations
   1965-1995 ACSH
units: https://www.ncei.noaa.gov/pub/data/cdo/documentation/
   GHCND_documentation.pdf

code that looks at the body of a request output:
import requests
resp = requests.get(
    'https://www.ncdc.noaa.gov/cdo-web/api/v2/data?datasetid=GHCND&
       datatypeid=TAVG&limit=1000&stationid=GHCND:USW00023160&
       startdate=2010-01-01&enddate=2010-12-31',
    headers={'token': 'yourTokenHere'}
)
print(resp.json())

Versions:
  Python 3.8.12
  requests 2.27.1
  pandas 1.4.2
  json5 0.9.5
  numpy 1.21.5
```

```
'''
import requests
import pandas as pd
import json
import numpy as np
from datetime import datetime

## Global variables
# access token from NOAA
Token = 'yourTokenHere'
#Tucson Airport station
station_id = 'GHCND:USW00023160'

def makeAPICall(variable):
    '''
    This function takes the variable to get the results for, and
       returns
    a list of the dates the data was found on and the value of the
    specified variable on that date
    '''
    # initialize lists
    dates = []
    results = []
    #for each year from 2000-2020, end is exclusive
    for year in range(2000, 2021):
        year = str(year)
        print('working on year '+year)

        #make the api call
        r = requests.get('https://www.ncdc.noaa.gov/cdo-web/api/v2/
           data?datasetid=GHCND&datatypeid='+
        variable+'&limit=1000&stationid=GHCND:USW00023160&startdate
           ='+year+'-01-01&enddate='+year+
        '-12-31&units=standard', headers={'token':Token})
        #load the api response as a json
        d = json.loads(r.text)

        # not all years have responses
        if not d:
            continue

        #get all items in the response which are the correct
           variable
        temp_results = [item for item in d['results'] if item['
           datatype']==variable]
        #get the date field from the readings
        dates += [item['date'] for item in temp_results]
```

```
        #get the actual value from all the readings
        results += [item['value'] for item in temp_results]

    return dates, results


def main():
    #initialize dataframes
    df_temp = pd.DataFrame()
    df_windSpeed = pd.DataFrame()
    df_windDirection = pd.DataFrame()

    # get data
    dates_temp, temps = makeAPICall('TMAX')
    dates_windSpeed, wdsp = makeAPICall('AWND')
    dates_windDirection, wddr = makeAPICall('WDF2')

    #populate date and temperature fields (cast string date to
       datetime)
    df_temp['date'] = [datetime.strptime(d, "%Y-%m-%dT%H:%M:%S") for
        d in dates_temp]
    df_temp['highTemp'] = temps    # units degrees fahrenheit

    # populate date and average wind speed fields
    df_windSpeed['date'] = [datetime.strptime(d, "%Y-%m-%dT%H:%M:%S
       ") for d in dates_windSpeed]
    df_windSpeed['avgWindSpeed'] = wdsp     # units miles per hour

    # populate date and wind direction fields
    df_windDirection['date'] = [datetime.strptime(d, "%Y-%m-%dT%H:%M
       :%S") for d in dates_windDirection]
    df_windDirection['windDirection'] = wddr    # units degrees (
       clockwise from N, that what other wind measurments are)

    # create csvs, these will be combined in R
    df_temp.to_csv("Using ML on Monsoon Data/Data/clean_temp_data.
       csv")
    df_windSpeed.to_csv("Using ML on Monsoon Data/Data/
       clean_windSpeed_data.csv")
    df_windDirection.to_csv("Using ML on Monsoon Data/Data/
       clean_windDirection_data.csv")


main()
```

## B Code Retrieving Historical Flood Data

```
# title: Get Historical Flood Data
# author: Dharma Hoy
# date: 08/07/2022
# R version 4.1.0

# This program cleans up the csv of the storm search result data
   from NOAA.
# The data was found at the cite below and resulted from a search
   for floods
# and flash floods in Pima county between 01-01-2000 and 12-31-2020.
# https://www.ncdc.noaa.gov/stormevents/

# load data
# install.packages("tidyverse")
library(tidyverse) # version: 1.3.1
searchResults <- read_csv("Using ML on Monsoon Data/Data/
   storm_data_search_results.csv")

cleanSearchResults <- searchResults %>%
  select(BEGIN_LOCATION, BEGIN_DATE) %>%
  separate(BEGIN_DATE, sep = '/', into = c("month", "day", "year"))
    %>%
  relocate(year, month, day, BEGIN_LOCATION) %>%
  unite(year, month, day, col = "date", sep = '-') %>%
  rename(floodBeginLocation = BEGIN_LOCATION)

write_csv(cleanSearchResults, "Using ML on Monsoon Data/Data/
   clean_storm_data_search_results.csv")
```

## C Code Retrieving Historical Sensor Precipitation Data

```
'''
title: Get Historical Sensor Precipitation Data
author: Dharma Hoy
date: 08/24/2022

This program gets daily rainfall totals from the stations in pima
   county. It uses the Monsoon API that accesses the UArizona
   Environment monsoon dataset. This dataset is visually represented
    in their monsoon plotter found at  monsoon.environment.arizona.
   edu More information about this API, the plotter and its creation
    read the research article published by frontiers titled "
   Curating  and Visualizing Dense Networks of Monsoon Precipitation
    Data: Integrating  Computer Science Into Forward Looking Climate
    Services Development". The authors are Ben McMahan, Rey L.
   Granillo III, Benni Delgado, Mauricio Herrera, and Michael A.
   Crimmins

Versions:
  Python 3.8.12
  pandas 1.4.2
  numpy 1.21.5
'''
import monsoon
import pandas as pd
import numpy as np
import datetime
from dateutil.relativedelta import relativedelta

# input api credentials
username = "yourUsername"
key = "yourKey"

monsoonOccurance = monsoon.MonsoonAPI(username, key)

def getData(startYear, endYear):
  # initalize dates
  startDate, endDate = datetime.date(startYear, 1, 1), datetime.date
     (endYear, 12, 31),
  delta = relativedelta(years = 1)
  # find last day of start year
  # use delta and account for response end date being inclusive
  nextDate = datetime.date(startYear-1, 12, 31) + delta
  # make lists to store a data frame for each year in the date range
  yearDataFrames = list()
```

```
# make calls for each year in given period
while startDate < endDate:
  # make call
  response = monsoonOccurance.sensor_readings("pima_fcd", str(
     startDate), str(nextDate))
  # not all years have responses
  if len(response) == 0:
    startDate = nextDate
    nextDate = nextDate + delta
    continue

  # df for current year of data
  yearDF = pd.DataFrame(response)
  # add to dataframes list
  yearDataFrames.append(yearDF)

  # iterate date values
  startDate = startDate + delta
  nextDate = nextDate + delta

# concatinate all the yearly data frames
allYearsDF = pd.concat(yearDataFrames)

# remove duplicate sensor readings
allYearsDF.drop_duplicates()
# unique sensors list
sensorNames = allYearsDF['sensor_name'].unique()

# for each sensor get difference in daily rainfall readings
allSensorDF = list()
for sensor in sensorNames:
  # seperate out current sensor
  sensorDF = allYearsDF[allYearsDF['sensor_name'] == sensor]

  # drop sensor column so groupby operation works
  sensorDF = sensorDF.drop(columns=['sensor_name'])
  # convert datatime to datetime type and set it as index column
  sensorDF['datetime'] = pd.to_datetime(sensorDF['datetime'])
  sensorDF = sensorDF.set_index('datetime')
  sensorDF = sensorDF.sort_index()

  # column with difference between each reading
  sensorDF = sensorDF.diff()
  sensorDF = sensorDF.resample('1D').sum()
  sensorDF = sensorDF.where(sensorDF['reading_value'] >= 0, 0)
```

```
        # add back in sensor column and rename readings to
            total_rainfall
        sensorDF['sensor_name'] = sensor
        sensorDF = sensorDF[['sensor_name', 'reading_value']]
        sensorDF = sensorDF.rename(columns = {"reading_value":"
            total_rainfall"})

        # add the sensorDF to the list of all sensorDF
        allSensorDF.append(sensorDF)

    # concat all the sensor dataframes together
    finalDF = pd.concat(allSensorDF)
    finalDF.fillna(0)

    #return finalDF
    return finalDF

def main():
    sensorData = getData(2000, 2020)
    sensorData.to_csv('Using ML on Monsoon Data/Data/clean_sensor_data
        .csv')
    # this takes less than two minutes to run for 20 years of data

main()
```

## D Code Creating Final 2000-2020 Data Frame

```
# title: Creating Final Data Frame
# author: Dharma Hoy
# date: 09/13/2022
# R version 4.1.0

# This program combines the weather data from NOAA with the flood
    data from
# NOAA and the sensor data gathered through the AIRES API. This data
     can be
# used for ML analysis and examples of this use are shown in this
    repository.


# Import Libraries
    ------------------------------------------------------
library(tidyverse) # version 1.3.1
library(arrow) # version 9.0.0.1

# Load Data
    ------------------------------------------------------------
sensorPrcp <- read_csv("Using ML on Monsoon Data/Data/
    clean_sensor_data.csv")
storm <- read_csv("Using ML on Monsoon Data/Data/
    clean_storm_data_search_results.csv")
temp <- read_csv("Using ML on Monsoon Data/Data/clean_temp_data.csv
    ")
windDirection <- read_csv("Using ML on Monsoon Data/Data/
    clean_windDirection_data.csv")
windSpeed <- read_csv("Using ML on Monsoon Data/Data/
    clean_windSpeed_data.csv")


# Look at Data
    ------------------------------------------------------------
glimpse(sensorPrcp)
glimpse(storm)
glimpse(temp)
glimpse(windDirection)
glimpse(windSpeed)

# make sensor_name a factor
sensorPrcp$sensor_name <- as.factor(sensorPrcp$sensor_name)

# make floodBeginLocation a factor
```

```
storm$floodBeginLocation <- as.factor(storm$floodBeginLocation)


# Combine the Data
   ----------------------------------------------------
combinedData <- windDirection %>%
  full_join(windSpeed, by = c("date" = "date")) %>%
  full_join(temp, by = c("date" = "date")) %>%
  full_join(storm) %>% # duplicate days accounted for multiple
     floods one day
  full_join(sensorPrcp, by = c("date" = "datetime")) %>% # 65-100
     sensors, per day increase in rows reasonable
  select(date, windDirection, avgWindSpeed, highTemp,
     floodBeginLocation, sensor_name, total_rainfall) # disregard
     pandas indexes


# Saving the Data
   ----------------------------------------------------

# use feather so data can be read by R and Python
write_feather(combinedData, "Using ML on Monsoon Data/Data/
   masterDataFrame.feather")
```

## E Code Retrieving 2020 Data

````
---
title: "2020 Precipitation Data"
author: "Dharma Hoy"
date: "2022-09-23"
R version: "4.1.0"
output: html_document
---


```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

Load R libraries
```{r}
library(tidyverse) # version 1.3.1
library(tidymodels) # version 1.0.0
library(lubridate) # version 1.8.0
library(arrow) # version 9.0.0.1
library(gplots) # version 3.1.3
library(reticulate) # version 1.24
```

Import Python libraries
```{python}
import monsoon
import pandas as pd
```

Pull 2020 Data from precipitation API
```{python}
# input api credentials
username = "yourUsername"
key = "yourKey"

monsoonOccurance = monsoon.MonsoonAPI(username, key)

# Pull sensor readings from 2020
response = monsoonOccurance.sensor_readings("pima_fcd",
   "2020-01-01", "2020-12-31")

# Create data frame
sensorReadings2020 = pd.DataFrame(response)
````
````

```

Read in flood occurrences from 2020
```{r}
floods <- read_csv("Using ML on Monsoon Data/Data/
   clean_storm_data_search_results.csv") %>%
  filter(date >= "2020-01-01") %>%
  mutate(date = as_datetime(date))
```

Combine flood occurrences with data from API. Create feather file
```{r}
py$sensorReadings2020 %>%
  relocate(datetime, before = sensor_name) %>%
  mutate(datetime = as_datetime(datetime)) %>%
  mutate(date = as.Date(datetime, "%m/%d/%Y")) %>% # allows for
     easier join
  full_join(floods, by = c("date" = "date")) %>%
  rename("sensor_name" = "before") %>%
  select(datetime, sensor_name, reading_value, floodBeginLocation)
     %>%
  mutate(floodBeginLocation = as.factor(floodBeginLocation)) %>%
  mutate(sensor_name = as.factor(sensor_name)) %>%
  write_feather(., "Using ML on Monsoon Data/Data/2020Floods.feather
     ")
```
```