Homework 3 Part 1 Appendix

Abdulwahab, Fatma Barrett, Alexis McDonough, Jared Miller, Matthew Solares, Nicky

March 5, 2019

1 Journey Length

The total distance was 6425 miles or 10340.166 kilometers.

2 Trip Map

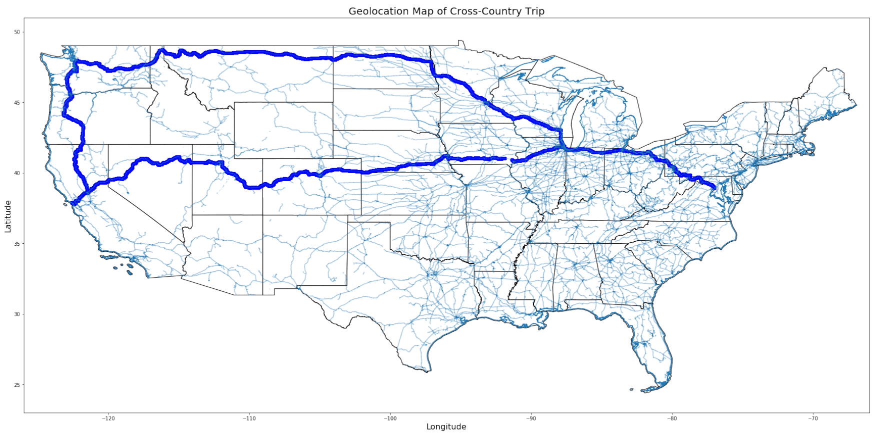

Figure 1: Trip Map

3 Visited Cities

The returned coordinates for the two cities are [92757, 47.608964, -122.33316299999998, 649.7] and [139309, 37.805606, -122.416288, 236.18333333333334]. This indicates that the cities were Seattle Washington and San Francisco, California. The most likely/closest hotels for each are The Hilton Seattle and the Pier 2620 Hotel, respectively.

4 Total Stops and the Definition of a Stop

If we define stops as maintaining a speed of 0 for 10 minutes, the number of total stops amounts to 37. If the duration is 5 minutes, the number of total stops amounts to 75.

5 Plotted Stops

6 Data Between Stops

The average speed between stops is 50.82 mph, with a standard deviation of 24.14. In kilometers, it is 81.79kmh with a standard deviation of 38.85.

7 Visited States

The states visited include Virginia, West Virginia, Pennsylvania, Ohio, Maryland, California, Nevada, Indiana, Illinois, Wisconsin, Minnesota, North Dakota, Montana, Idaho, Washington, Oregon, Utah, Colorado, Kansas, and Nebraska.

8 Landmarks

We divided the total map into four separate sections. Within Section 1, notable landmarks include Glacier Park, Pike Place Market, Mt. Olympia, Fort Vancouver, The Pearl District, Mount Shista, the Golden Gate Bridge, and Circus Circus. Within Section 2, notable landmarks include Roosevelt Park, the Fort Buford Historical Site, the Cedar Wilderness Area, Indian Head Peak, the Colorado National Monument, and Crystal Peak.

9 Appendix Code: Tidying

```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
import geopandas
import geoplot
import geoplot.crs as gcrs
from shapely.geometry import Point, Polygon
import math
'''
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
'''
plt.style.use('dark_background')

CC_Data = pd.read_csv("data/Dataset1_combined.csv")

CC_Data.head()

#Tidy the timestamp data
time_list = []
```

```
for i in CC_Data["timestamp"]:
'''This loop separates the clock time from the rest of the string'''
start = 0 #index for slicing
stop = 0 #index for slicing
flag = 0
for j in range(len(i)):
if (i[j] == " " and flag == 0):
start = j+1
flag = 1
elif (i[j] == " " and flag == 1):
stop = j
time_list.append(i[start:stop])

CC_Data["time"] = time_list

#####################################

second_list = []
for i in range(len(CC_Data)):
'''this will create a column of all the times in seconds, relative to the start time....th
if (i == 0):
x = CC_Data["time"].iloc[i]
x = x.split(":")
prev_hour = int(x[0])
prev_minute = int(x[1])
prev_second = int(x[2])
prev_time = 0 #previous value stored since init
second_list.append(0)
else:
x = CC_Data["time"].iloc[i]
x = x.split(":")
hour = int(x[0])
minute = int(x[1])
second = int(x[2])

if ((prev_hour == hour) and (prev_minute == minute)):
#if the hour and minute are the same
sec = second - prev_second #finds how many seconds since previous instance
second_list.append(prev_time + sec)
prev_time += sec
elif ((prev_hour == hour) and (prev_minute != minute)):
#if the hour is the same, but different minutes
sec = (second + (60-prev_second)) + ((minute-prev_minute-1) * 60) #finds how many seconds s
#the ((minute-prev_minute-1) * 60) is there in case the jump is larger than a minute
second_list.append(prev_time + sec)
prev_time += sec
else:
#if the hour has changed -- assuming the change is never more than 1 hour
sec = (minute*60 + second)
second_list.append(prev_time + sec)
prev_time += sec
```

```
prev_hour = hour
prev_minute = minute
prev_second = second

len(CC_Data)

len(second_list)

CC_Data["seconds"] = second_list

#remove points where the accuracy >10
CC_Data = CC_Data[CC_Data["accuracy"] <= 10]

CC_Data.head()
```

10 Appendix Code: Distance

Note that this code is a continuation of the code from Tidying, and thus the same imports and data cleaning apply.

```
def Haversine(lat1, lon1, lat2, lon2):
R = 6372.8 # Earth radius in kilometers

dLat = np.radians(lat2 - lat1)
dLon = np.radians(lon2 - lon1)

lat1 = np.radians(lat1)
lat2 = np.radians(lat2)

a = np.sin(dLat/2)**2 + np.cos(lat1)*np.cos(lat2)*np.sin(dLon/2)**2
c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))

return R * c

Total_Dist = 0.0
for i in range(len(CC_Data)-1):
Total_Dist += Haversine(lat1=CC_Data["latitude"].iloc[i], lon1=CC_Data["longitude"].iloc[i

print("The total distance covered in this Cross Country was %0.3f Km or %0.3f miles" %(Tot
```

11 Appendix Code: Trip Map

```
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point
import matplotlib.pyplot as plt
from shapely.geometry import Point

dataset1 = pd.read_csv('dataset1_combined.csv', ',')
```

```
dataset1['Coordinates'] = list(zip(dataset1.longitude, dataset1.latitude))

dataset1['Coordinates'] = dataset1['Coordinates'].apply(Point)

geofile1 = gpd.GeoDataFrame(dataset1, geometry='Coordinates')

usa = gpd.read_file('tl_2018_us_state/tl_2018_us_state.shp')
coastline = gpd.read_file('tl_2018_us_coastline/tl_2018_us_coastline.shp')
rails = gpd.read_file('tl_2018_us_rails/tl_2018_us_rails.shp')
urban_zones = gpd.read_file('tl_2018_us_uac10/tl_2018_us_uac10.shp')

california_lm = gpd.read_file('tl_2018_06_arealm/tl_2018_06_arealm.shp')
colorado_lm = gpd.read_file('tl_2018_08_arealm/tl_2018_08_arealm.shp')
dc_lm = gpd.read_file('tl_2018_11_arealm/tl_2018_11_arealm.shp')
idaho_lm = gpd.read_file('tl_2018_16_arealm/tl_2018_16_arealm.shp')
illinois_lm = gpd.read_file('tl_2018_17_arealm/tl_2018_17_arealm.shp')
indiana_lm = gpd.read_file('tl_2018_18_arealm/tl_2018_18_arealm.shp')
iowa_lm = gpd.read_file('tl_2018_19_arealm/tl_2018_19_arealm.shp')
kansas_lm = gpd.read_file('tl_2018_20_arealm/tl_2018_20_arealm.shp')
maryland_lm = gpd.read_file('tl_2018_24_arealm/tl_2018_24_arealm.shp')
minnesota_lm = gpd.read_file('tl_2018_27_arealm/tl_2018_27_arealm.shp')
montana_lm = gpd.read_file('tl_2018_30_arealm/tl_2018_30_arealm.shp')
nebraska_lm = gpd.read_file('tl_2018_31_arealm/tl_2018_31_arealm.shp')
nevada_lm = gpd.read_file('tl_2018_32_arealm/tl_2018_32_arealm.shp')
northdakota_lm = gpd.read_file('tl_2018_38_arealm/tl_2018_38_arealm.shp')
ohio_lm = gpd.read_file('tl_2018_39_arealm/tl_2018_39_arealm.shp')
oregon_lm = gpd.read_file('tl_2018_41_arealm/tl_2018_41_arealm.shp')
pennsylvania_lm = gpd.read_file('tl_2018_42_arealm/tl_2018_42_arealm.shp')
utah_lm = gpd.read_file('tl_2018_49_arealm/tl_2018_49_arealm.shp')
virginia_lm = gpd.read_file('tl_2018_51_arealm/tl_2018_51_arealm.shp')
washington_lm = gpd.read_file('tl_2018_53_arealm/tl_2018_53_arealm.shp')
westvirginia_lm = gpd.read_file('tl_2018_54_arealm/tl_2018_54_arealm.shp')
wisconsin_lm = gpd.read_file('tl_2018_55_arealm/tl_2018_55_arealm.shp')

fig, ax = plt.subplots(figsize = (30,30))
ax.set_xlim([-126, -66])
ax.set_ylim([23, 51])

usa.plot(ax = ax, color='white', edgecolor='black')
coastline.plot(ax = ax)
rails.plot(ax = ax, alpha = 0.4)
#urban_zones.plot(ax = ax, color='green', edgecolor='green', alpha = 0.3)
#cali_landmarks.plot(ax = ax, color = 'orange')
#lm1.plot(ax = ax, color = 'orange', alpha = 0.5)

geofile1.plot(ax=ax, color='blue')

plt.title('Geolocation Map of Cross-Country Trip', fontsize=48)
plt.xlabel('Longitude', fontsize=36)
plt.ylabel('Latitude', fontsize=36)
plt.tick_params(axis='both', which='major', labelsize=32)
plt.tick_params(axis='both', which='minor', labelsize=32)
plt.show()
```

12 Appendix Code: Visited Cities

Note that this code is a continuation of the code from Tidying and Distance, and thus the same imports and data cleaning apply.

```
##speed should be zero for "n" seconds
def GetStops(n = 10, data = CC_Data):
'''This function will gather a list of latitude/longitude pairs where each pair represents
Stops = []
#Stops = [[index in CC_data,"lat", "lon", "duration (minutes)"]]
n = n*60 #time, in seconds of how long a stop is
start = 0 #index  the last stop started
stop = 0 #index the last stop stopped

for i in (range(0,len(data)-1)):
#for each element in the data
if ((data["speedmph"].iloc[i] == 0) and (i >= stop)):
#if (speed at location i is zero) and (we are after the last stop ended)
start = i #index of stop's start time


j = i+1 #start at next index to continue check
while ((data["speedmph"].iloc[j] == 0)):
#while the speed remains 0
stop = j
j += 1

if ((data["seconds"].iloc[stop] - data["seconds"].iloc[start]) >= n):
#if (the duration of the stop is long enough)
Stops.append([i, data["latitude"].iloc[i], data["longitude"].iloc[i], (data["seconds"].iloc


return Stops

print(GetStops(n = 600)) #10 hours

print(GetStops(n = 200))
```

13 Appendix Code: Between Stop Data

```
import pandas as pd

df = pd.read_csv("dataset1_combined.csv")
dfclean = df[df.accuracy <= 20]

nonstopmphdf = df[(df['speedmph']) != 0]
print(nonstopmphdf['speedmph'].mean())
print(nonstopmphdf['speedmph'].std())

nonstopkmhdf = df[(df['speedkmh']) != 0]
print(nonstopkmhdf['speedkmh'].mean())
print(nonstopkmhdf['speedkmh'].std())
```

14 Section Maps

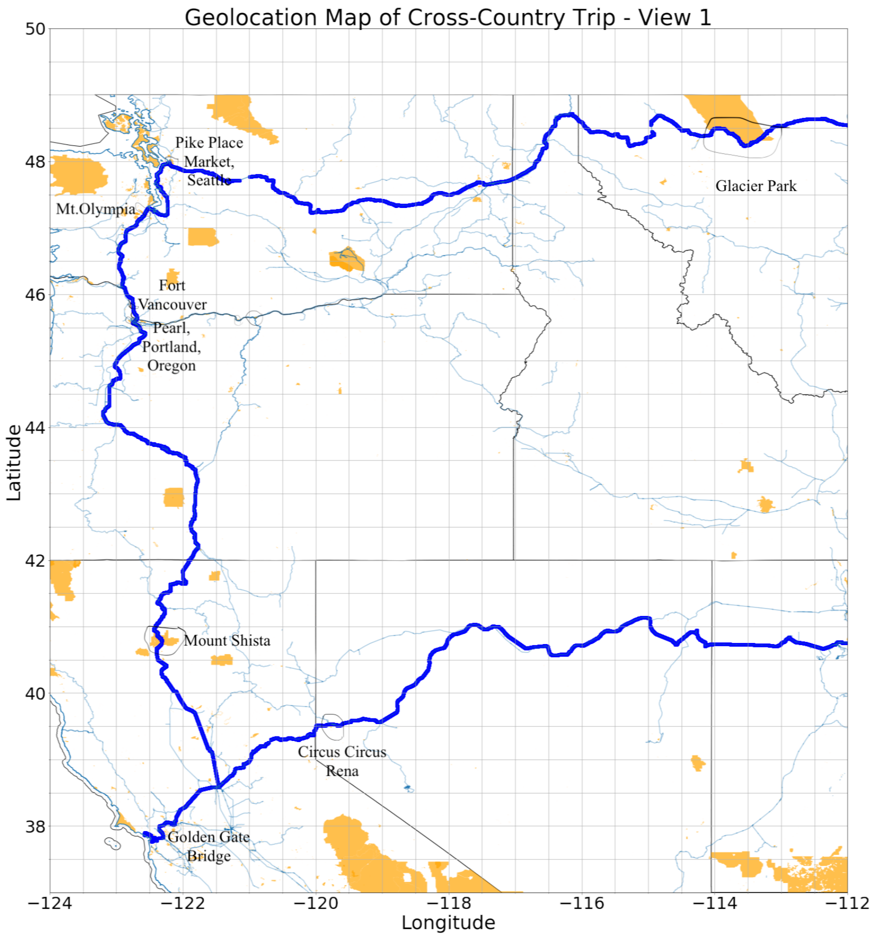

Figure 2: Notable Landmarks Section 1

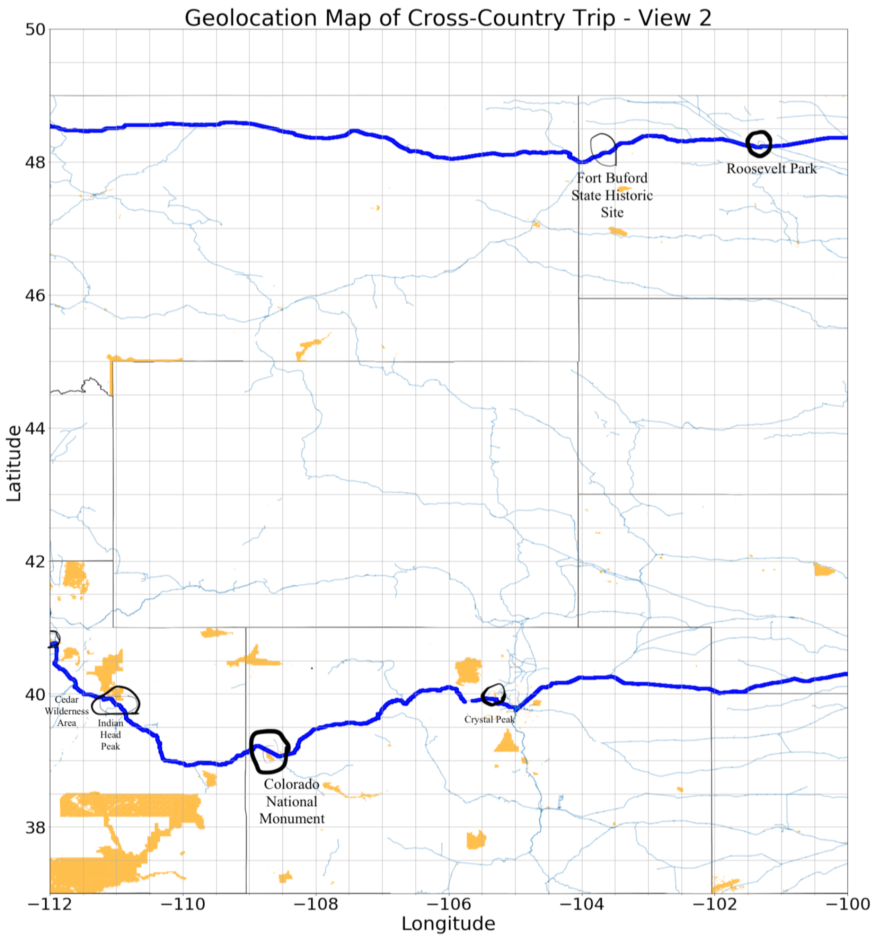

Figure 3: Notable Landmarks Section 2