Creating a Tableau Dashboard of My Strava Activities

Author: Ben Garrett
Date created: 2021-03-01
Last modified: 2021-03-01

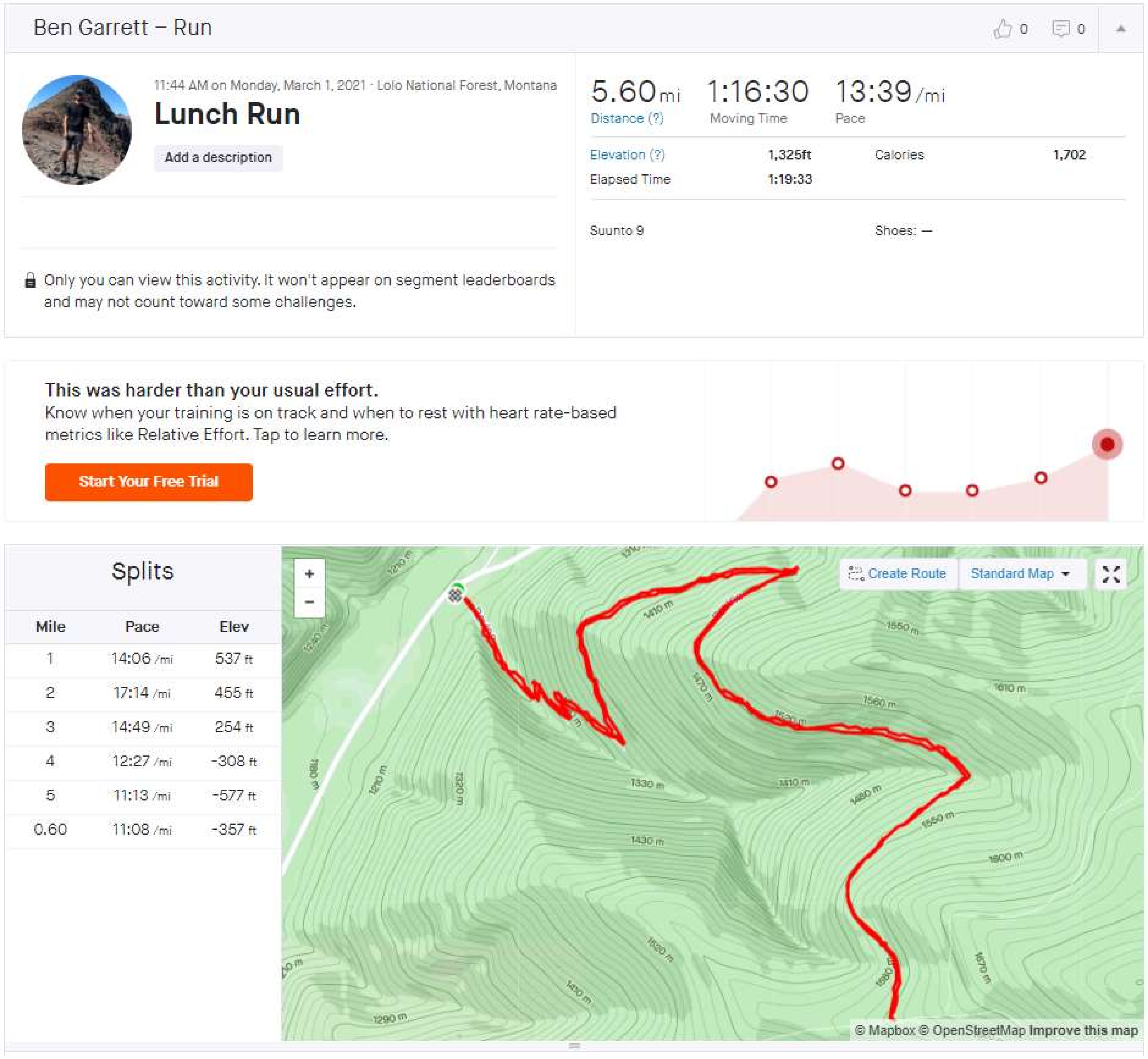
Introduction

The goal of this project is to extract tabular and spatial data from my Strava account and create a meaningful map and dashboard using python and Tableau. Strava is a social network for athletes to share their physical activities with their friends and local communities. I have been using Strava for a few years and have hundreds of hikes, runs and bike rides logged on the site. Although Strava makes it easy to see a list of your activities on their app, they provide few useful maps and charts to users. Luckily, I can fix that!

www.strava.com

About the Data

The data originated from any number of apps or smarwatches. I use a Suunto GPS watch to record each activity and the watch's app sends data to Strava when I sync it to my phone. The information of interest is average speed, elevation gain, distance, duration, ect. On Strava, a typical activity upload looks like the following:



Ben Garrett – Run

11:44 AM on Monday, March 1, 2021 · Lolo National Forest, Montana

Lunch Run

Add a description

Only you can view this activity. It won't appear on segment leaderboards and may not count toward some challenges.

5.60mi Distance (?) | 1:16:30 Moving Time | 13:39/mi Pace

Elevation (?)	1,325ft	Calories	1,702
Elapsed Time	1:19:33		
Suunto 9		Shoes: —	

This was harder than your usual effort.

Know when your training is on track and when to rest with heart rate-based metrics like Relative Effort. Tap to learn more.

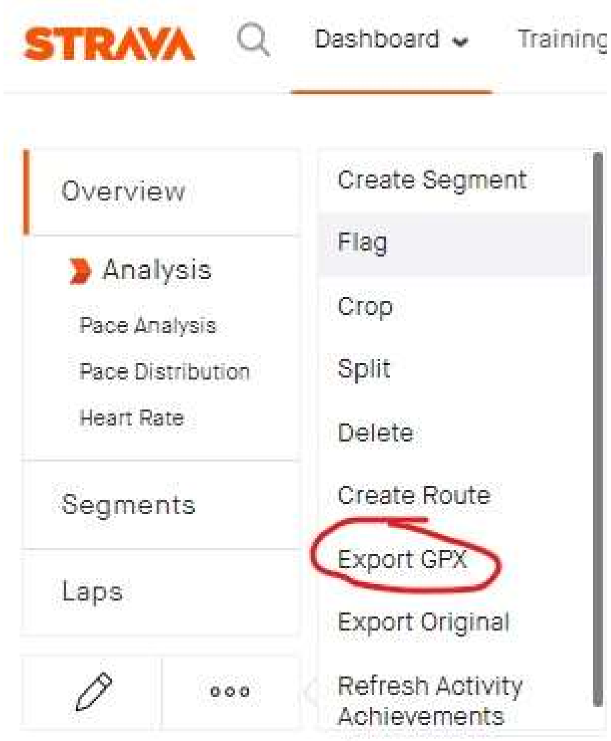
Splits

Mile	Pace	Elev
1	14:06 /mi	537 ft
2	17:14 /mi	455 ft
3	14:49 /mi	254 ft
4	12:27 /mi	-308 ft
5	11:13 /mi	-577 ft
0.60	11:08 /mi	-357 ft

Fortunately, Strava also allows you to bulk download your spatial and tablular data. The data comes in 40+ csv's and the spatial data comes as a single .fit file for each activity. The end goal is a single table with all of the important tabular data and spatial data all in one place, not hundreds of files in different formats. While you can use open source tools to convert .fit files to a more typical spatial format like `.gpx` or `.shp`, I decided to tackle the problem with python in a more interesting way...

Getting Activity Spatial Data With Python

On Strava's webpage for each activity, there is a link to download the activity track as a easy-to-use `.gpx` file. Conveniently, the url for each activity page is predictable and reproducable if one know the activity name they want to recieve. You can use Strava's API to routinely get these files with authentication, but I found that process no simpler or quicker than making a simple request. Thus, I decided to use python to direct my browser (already logged in to my account) to download each file via the link on the activity webpage.



Using the `activities.csv` downloaded from the bulk download, it is easy to generate a url to make the request to get each file. Unfortunately, each file is named as the non-unique activity name, so the tricky part of the following code is finding the actively downloading file and logging its name.

```
In [ ]:
```

```
import webbrowser
import pandas as pd
import time
import numpy as np
from os import listdir
import os.path

save_path = r'C:\garrett_workspace\tableau\strava_dashboard\gpx'
chrome_path = r'C:\Program Files\Google\Chrome\Application\chrome.exe'
webbrowser.register('chrome', None,webbrowser.BackgroundBrowser(chrome_path))

activity_data = pd.read_csv(r'C:\garrett_workspace\Strava_Backup_Data\activities.csv')
activity_data['gpx_filepath'] = np.nan

full_file_names = []

for i in range(len(activity_data)):
    if activity_data['Filename'][i] != np.nan:
        activity_id = int(activity_data['Activity ID'][i])
        url = 'https://www.strava.com/activities/{}/export_gpx'.format(activity_id)
        webbrowser.get('chrome').open_new_tab(url)
        time.sleep(15)
        '''the following loop checks to see if the file has finished downloading
        to ensure only one file gets downloaded at a time.'''
        for z in range(1, 30000):
            files = listdir(save_path)
            for x in files:
                if 'Unconfirmed' in x:
                    time.sleep(1)
                else:
                    pass

        for file in files:
            full_file_names.append(save_path+'\{}'.format(file))
        latest_file = max(full_file_names, key=os.path.getctime)
        activity_data.loc[i, 'gpx_filepath'] = str(latest_file)

activity_data.to_csv(r'C:\garrett_workspace\Strava_Backup_Data\activities.csv')
```

Combining Data Into One Format

Now that I have all of the tabluar activity data and their associated .gpx files, the next step is to combine them into one format where all attributes can live. The best file format for this is the shapefile, which contains a database table and spatial information in a format that all GIS software and many database tools can read (PostgreSQL via PostGIS, Tableau, many others). To convert the files, the python library `gpxpy` can parse the gpx files and `geopandas` (includes `shapely`) can write the shapefile. The general workflow is this:

- open each gpx
- pull out each point (coordinate pair of vertice on line), add all points to a shaply linestring object, append object into geopandas geodataframe.
- save shapefile via geopandas (and manually write a .prj file containing spatial projection information)

In [ ]:

```
import gpxpy
import pandas as pd
import geopandas as gpd
from shapely.geometry import LineString

tabular = pd.read_csv(r'C:\garrett_workspace\tableau\strava_dashboard\activities.csv')

geo_dataframe = gpd.GeoDataFrame(tabular)
geo_dataframe['geometry'] = None

for index in range(len(geo_dataframe)):
    filepath = geo_dataframe['gpx_filepath'][index]
    file = open(filepath, 'r')
    gpx = gpxpy.parse(file)

    points = []

    for track in gpx.tracks:
        for segment in track.segments:
            for point in segment.points:
                points.append(tuple([point.longitude, point.latitude]))

    line = LineString(points)
    geo_dataframe.loc[index, 'geometry'] = line
    print(index+1,'files parsed.')


geo_dataframe.to_file(r'C:\garrett_workspace\tableau\strava_dashboard\strava_dashboard.shp')

crs_to_write = """GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137,298.2572235

with open(r'C:\garrett_workspace\tableau\strava_dashboard\{}.prj'.format('strava_dashboard'), 'w')
    file.write(crs_to_write)
```

Cleaning up the Data

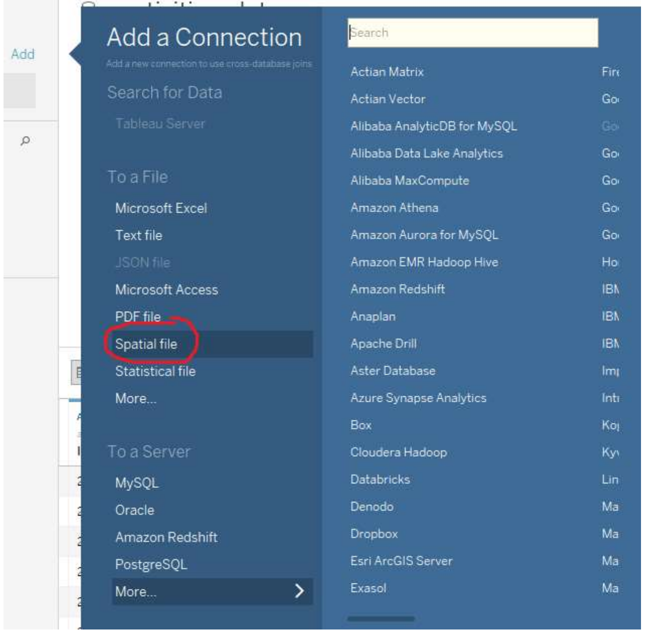
The data provided by Strava can use a bit of tidying. Needed transforms include converting units to imperial, time fields to hours, dealing with some null values.

In [ ]:

```
import geopandas as gpd
import pandas as pd

data = gpd.read_file(r'C:\garrett_workspace\tableau\strava_dashboard\strava_dashboard.shp')

data = data[['Activity I', 'Activity D', 'Activity N', 'Activity T', 'Elapsed Ti',
             'Distance', 'Moving Tim', 'Average Sp',
             'Elevation', 'Elevatio_1', 'Elevatio_2', 'Elevatio_3', 'Calories', 'geometry']]

data.columns = ['id', 'date', 'name', 'type', 'duration', 'distance', 'moving_time',
                'avg_speed', 'elev_gain', 'elev_1', 'elev_2', 'elev_3', 'calories', 'geometry']

# shapefiles need datetime's as strings
data['date'] = (pd.to_datetime(data['date'])).astype('str')
# seconds to decimal hours
data['duration'] = data['duration'] / 60 / 60
# kms to miles
data['distance'] = data['distance'] * 0.6213
# seconds to decimal hours
data['moving_time'] = data['moving_time'] / 60 / 60
# I can't figure out what units are in this field so I recalculated avg_speed
data['avg_speed'] = data['distance'] / data['moving_time']
# using my personal knowledge of these activities, I was able to figure out how
# to use these elev fields
data['elev_gain'] = data['elev_gain'].fillna(data['elev_3'] - data['elev_2']).fillna(0)

data = data.drop(['elev_1', 'elev_2', 'elev_3'], axis=1)
# meters to feet
data['elev_gain'] = (data['elev_gain'] * 3.28).astype('int')

filename = 'activities_data'

data.to_file(r'C:\garrett_workspace\tableau\strava_dashboard\{}.shp'.format(filename))
# could rename the old projection file but this is just as easy
crs_to_write = """GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137,298.2572235

with open(r'C:\garrett_workspace\tableau\strava_dashboard\{}.prj'.format(filename), 'w') as file:
    file.write(crs_to_write)
```

Load Data into Tabeau

Far too easy!



Build a Dashboard

The rest of the data manipulation can easily be done within Tableau. My activities contain millions of coordinate points which can slow down the queries in the final dashboard. To fix this, you can resample the points and remove x out of every z points to reduce the size. As long as the resampling rate isn't too large, the end user wouldn't notice much detail loss in the map. Below is a simple way to do that in python:

In [ ]:
```python
import geopandas as gpd

data = gpd.read_file(r'C:\garrett_workspace\tableau\strava_dashboard\activities_data.shp')

resample_factor = 4

for row in range(len(data)):
    '''the try - except block is because I added dummy null rows to represent weeks
    that I had no activities, thus these rows have no geometry...'''
    try:
        linestring = data['geometry'][row]
        simplified_points = []
        for i in range(len(linestring.coords)):
            coord = linestring.coords[i]
            if i % resample_factor == 0:
                simplified_points.append(coord)
        data['geometry'][row].coords = simplified_points
    except:
        pass

data.to_file(r'C:\garrett_workspace\tableau\strava_dashboard\{}.shp'.format('activities_data'))
```

The Dashboard!

The dashboard can be found on Tableau Public here:
https://public.tableau.com/profile/ben.garrett#!/vizhome/StravaVisualization/MyStravaData