File Format

Inside each .tar file is a .dat ASCII (plain-text) file.

In the .dat files:

- One file per day
- format:

```
* 2011    9   26
T  17   0   0
     52.000  -169.500     0.115
     52.500  -169.500     0.104
     53.000  -169.500     0.094
[…]
     -4.500   -33.000     0.044
     -4.500   -32.500     0.053
     -4.000   -33.000     0.035
     -3.500   -33.000     0.013
     -3.000   -33.000     0.015
T  17   5   0
     52.000  -169.500     0.182
     52.500  -169.500     0.163
     53.000  -169.500     0.149
     53.500  -169.500     0.127
     54.000  -169.500     0.114
     54.500  -169.500     0.125
     55.000  -169.500     0.132
     55.500  -169.500     0.135

[…]
```

Columns correspond to geographic latitude, longitude, then dTEC

Each time is separated by the `T HH  MM  SS` header, which makes the file a little tricker to read.

Here's a python function that I wrote a few years ago that **should** be able to read the data in. If it doesn't work and you can't figure out how to fix it, let me know. It will read the dTEC files into a python dictionary, which can then be put into a pandas dataframe or a 2D numpy array to make plotting easier.

I wrote this a few years ago, and it's not the best way to do this, but it can give you a jump start on reading these files. Feel free to modify however you want/need, I don't use it anymore so there's no need for backwards compatibility or anything.

```
def import_gps_dict(GPS_PREFIX, GPS_DATE_WORDS):
    '''
            This will return us a dictionary (not dataframe) of gps data.
                keys in dataframe is the datetime of measurement,

                dict [key = datetime] [lats , lons, dtecs]
    '''
    print('NOTE: we only have gps data for 3 days - couple hours each (curr
    try:
        fname1 = (GPS_PREFIX + 'New' + GPS_DATE_WORDS[1] + str(GPS_DATE_WOF
                 '-' + str(GPS_DATE_WORDS[0]) + '.dat')

        fname2 = (GPS_PREFIX + GPS_DATE_WORDS[1] + str(GPS_DATE_WORDS[2]).r
                 '-' + str(GPS_DATE_WORDS[0]) + '.dat')
        if os.path.isfile(fname1):
            fname = fname1
            print("using newer tec values :)")
        elif os.path.isfile(fname2):
            fname = fname2
            print('using older tec values')
        with open(fname) as file_in:
            dates = []
            datas = []
            times = []
            is_ = -1
            for line in file_in:
                line = line.strip('\n')
                if '*' in line:
                    dates.append(line)
```

```
                    print('new date', line)
                elif 'T' in line:
                    times.append(line)
                    is_ +=1
                    datas.append([])
                else:
                    datas[is_].append(line)

        dtimes = []
        for i in range(len(times)):
            l = [x for x in times[i].strip('T').split(' ') if x]
            if len(dates) == 1:
                dtimes.append(dt.strptime(dates[0].strip('*'), ' %Y %m %d')
            else:
                raise ValueError('The GPS file covers more than one day, lo
        dict_ = {}
        for i in range(len(datas)):
            data_list_local = []
            for j in range(len(datas[i])):
                inter_ = datas[i][j].split(' ')
                inter2 = []
                for gg in inter_:
                    if len(gg) > 2:
                        inter2.append(float(gg))
                data_list_local.append(inter2)
            dict_[str(dtimes[i])] = data_list_local

        new_dict = {}
        for key in dict_.keys():
            lats = []
            lons = []
            dns = []
            for i in range(len(dict_[key])):
                lats.append(dict_[key][i][0])
                lons.append(dict_[key][i][1])
                dns.append(dict_[key][i][2])
            new_dict[key] = {'lats':lats, 'lons':lons, 'dns':dns}
        return new_dict

    except:
        print("ERROR ERROR ERROR")
```