daily_RFI_report_v2

October 17, 2018

In [1]:
```
import numpy as np
import datetime
import matplotlib.pyplot as plt
import os
import json

OUT_DAY = 'arr_day.npy' #the output files containing integrated power and m
OUT_NIGHT = 'arr_night.npy'
#DATA_PATH = #
SUNSET_TIMETABLE = np.genfromtxt('/lustre/aoc/projects/hera/ajosaiti/SDR_RF
STR_DAY = os.environ['which_day']
DEBUG = False

def file_flush():
        if DEBUG: print('flushing temporary and output files...')
        open(OUT_DAY, 'w').close()
        open(OUT_NIGHT, 'w').close()
        if DEBUG: print('done.')

def delta_hours_minutes(td):
    return td.seconds//3600, (td.seconds//60)%60

def recursive_key_search(dat, key): #https://stackoverflow.com/questions/98
#sum_dbm_recursive_key_search(dat, key): #https://stackoverflow.com/questio
    if key in dat:
        yield dat[key]
    for k in dat:
        if isinstance(dat[k], list):
            for i in dat[k]:
                for j in recursive_key_search(i):
                    yield j

def time_in_rids_fmt(datetime_time): # convert datetime.datetime.now() time
        str_iso = datetime_time.isoformat(' ')
        str_time_rids = str( str_iso[0:4] + str_iso[5:7] + str_iso[8:10] +
        return str_time_rids
```

```
def sum_array_of_dbm(arr):
        val_sum = 10.*np.log10(np.sum(np.power(10,np.array(arr)/10.)))
        return val_sum



def day_night_initial_calculation(arr_day,arr_night):
    ## connect to librarian
    #cl = LibrarianClient(connection_name)

    ## search for unprocessed sessions
    #files = cl.search_files(search)['results']

    #if not len(sessions):
    #    return # Nothing to do.

    #plot_script = os.path.join(plots_dir, 'run_notebook.sh')

    ## check these sessid aren't in the processed_sessid.txt file
    #processed_fileid = np.loadtxt(os.path.join(plots_dir, 'processed_filei

    # filter out sessions already processed
    unprocessed_files = []
    for file in os.listdir(os.environ['STAGING_DIR']):
        name=str(file["name"])
        fileid = int(name[int(name.find('.')+1):name.find('-')])
        if fileid not in processed_fileid:
            unprocessed_files.append(sess)
    return unprocessed_files
    #cwd = os.getcwd()
    for filename in unprocessed_files: #os.listdir(cwd):
        if DEBUG: print('considering filename: '+str(filename))
        if filename.endswith('.ridz') and (STR_DAY in filename): # NOTE: In
            fname_uzip = str(filename.replace('.ridz','.rids'))
            os.system(str('zipr.py ' +str(filename)))
            dat = json.loads(open(str(fname_uzip)).read())
            for dset in np.array(dat['feature_sets'].keys()):
                t_spectra = datetime.datetime.strptime(str(dset[5:20]), '%Y
                int_pwr =  sum_array_of_dbm(dat['feature_sets'][str(dset)]['
                if DEBUG:
                    print('int_pwr: '+str(int_pwr))
                    print('t_spectra: '+str(t_spectra))
                    print('sunrise: '+str(sunrise)+', sunset: '+str(sunset)
                if bool( (t_spectra >=sunrise) and (t_spectra<=sunset) ): a
                else: arr_night.append(np.array([t_spectra,int_pwr]))
            dat.clear()
            os.system(str('zipr.py ' +str(fname_uzip))) # re-zip the file
```

```
    if DEBUG:
        print('arr_day: '+str(arr_day))
        print('arr_night: '+str(arr_night))

    return arr_day, arr_night

#For the day in question, figure out when sunset and sunrise were.
col_date,col_key = np.where(SUNSET_TIMETABLE == STR_DAY[4:]) #find mmdd, bu
sunrise = datetime.datetime.strptime( str(str(STR_DAY[:4])+str(STR_DAY[4:])
sunset = datetime.datetime.strptime( str(str(STR_DAY[:4])+str(STR_DAY[4:])+
delta_hour, delta_minute = delta_hours_minutes((sunset-sunrise))
#Arrays containing data from each sweep measurement, sorted by whether the
arr_day=[] # [datetime.datetime object], [float(integrated power)]
arr_night=[]


file_flush() # Make sure output files aren't reused with old data in them.
arr_day, arr_night = day_night_initial_calculation(arr_day,arr_night)

#Turn the array of arrays into a 2D array
arr_day=np.array(arr_day)
arr_night = np.array(arr_night)

np.save(OUT_DAY,arr_day)
np.save(OUT_NIGHT,arr_night)


if DEBUG:
    print('arr_day: '+str(arr_day))
    print('arr_day[0,0]: '+str(arr_day[0,0]))
    print('arr_day[:,0]: '+str(arr_day[:,0]))

arr_full_day = np.concatenate((np.array(arr_day), np.array(arr_night)),axis

plt.figure(1, figsize=(20,10))
plt.subplot(211)
plt.plot(arr_day[:,0], arr_day[:,1],'bo')
plt.plot(arr_night[:,0], arr_night[:,1],'go')
plt.subplot(211).set_title(str('Daytime ('+str(delta_hour)+' Hours, '+str(d
plt.subplot(212)
plt.plot(arr_full_day[:,0], arr_full_day[:,1],'ro')
plt.subplot(212).set_title(str('Full 24 Hours, Average Integrated Power: '


plt.show()
```

```
    TypeErrorTraceback (most recent call last)

    <ipython-input-1-dc077a3d9391> in <module>()
    101
    102 file_flush() # Make sure output files aren't reused with old data in th
--> 103 arr_day, arr_night = day_night_initial_calculation(arr_day,arr_night)
    104
    105 #Turn the array of arrays into a 2D array


    <ipython-input-1-dc077a3d9391> in day_night_initial_calculation(arr_day, ar
     61     unprocessed_files = []
     62     for file in os.listdir(os.environ['STAGING_DIR']):
---> 63         name=str(file["name"])
     64         fileid = int(name[int(name.find('.')+1):name.find('-')])
     65         if fileid not in processed_fileid:


    TypeError: string indices must be integers, not str
```