Dated_Stacked_Plots

June 26, 2023

[248]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import matplotlib.dates as mdates

# File paths
file_paths = [
    "PSP_FLD_L2_MAG_RTN_1MIN_145910.txt",
    "PSP_SWP_SPC_L3I_72861.txt",
    "PSP_SWP_SPC_L3I_209776.txt",
    "PSP_SWP_SPI_SF0A_L3_MOM_167546.txt"
]

# Header and footer line numbers
header_lines = [59, 75, 76, 44]
footer_lines = [3,3,3,3]


# Date and time information
date_str = '11-20-2018'
start_time = pd.to_datetime(date_str + ' 00:57')
end_time = pd.to_datetime(date_str + ' 11:03')

# Create a list to store the dataframes
dataframes = []

# Iterate over the files
for i, file_path in enumerate(file_paths):
    # Import the file skipping the header and footer lines
    df = pd.read_csv(file_path, skiprows=header_lines[i]+1,␣
 ↪skipfooter=footer_lines[i], engine='python', delim_whitespace=True)
    df = df.iloc[:, 1:]  # Remove the first column
    time_range = pd.date_range(start=start_time, end=end_time, periods=len(df))
    df.insert(0, 'Datetime', time_range)  # Add the Datetime column
    dataframes.append(df)

# Access each dataframe individually
```

```
df1 = dataframes[0]  # First file
df2 = dataframes[1]  # Second file
df3 = dataframes[2]  # Third file
df4 = dataframes[3]  # Fourth file
```

[249]:
```
A = df1
B = df2
C = df3

# Read the specific lines from the file into DataFrame D
D = pd.read_csv('PSP_SWP_SPI_SF0A_L3_MOM_167546.txt', skiprows=44, nrows=3,␣
 ↪delim_whitespace=True, header=None, names=['Year', 'Secs-of-year', 'km'])

# Convert the 'Year' and 'Secs-of-year' columns to datetime format
D['Datetime'] = pd.to_datetime(D['Secs-of-year'], unit='s', origin='2018-11-20')

# Set the 'Datetime' column as the index
D = D.set_index('Datetime')

#convert km to AU
dist_AU = D.iloc[:,2] * (6.68459e-9)
D_values = dist_AU
d = np.linspace(0, len(D_values) - 1, len(A))
D_rs = np.interp(d, np.arange(len(D_values)), D_values)
```

[250]:
```
# Interpolate values of the second column of B with the first column of A
B_interpolated = B.copy()  # Create a copy of dataframe B

# Linear interpolation of the second column of B to match the length of A
B_interpolated.iloc[:, 1] = B_interpolated.iloc[:, 1].interpolate()

# Adjust the length of B_interpolated to match A
B_interpolated = B_interpolated.iloc[:len(A)]
```

[251]:
```
y3_1 = C.iloc[:, 2] #Radial velocity

m_p = (1.67 * 10**-27) # proton mass in kg
N = (A.iloc[:,2] * 10**-9)/ (10**3 * np.sqrt(1.25663e-6 * 10**6 * m_p *␣
 ↪B_interpolated.iloc[:,2])) #convert nT to T (numerator) and cm^-3 to m^-3␣
 ↪then km (denominator)

T = (m_p * y3_1**2)/(1.380e-23) #Convert proton bulk velocity to temperature
C_s = np.sqrt(((5/3) * 1.380e-23 * T)/ m_p) #Determine the speed of sound
b = np.linspace(0, len(C_s) - 1, len(A))
C_rs = np.interp(b, np.arange(len(C_s)), C_s)
```

```
C:\Users\jowar\anaconda3\lib\site-packages\pandas\core\arraylike.py:397:
RuntimeWarning: invalid value encountered in sqrt
```

```
result = getattr(ufunc, method)(*inputs, **kwargs)
```

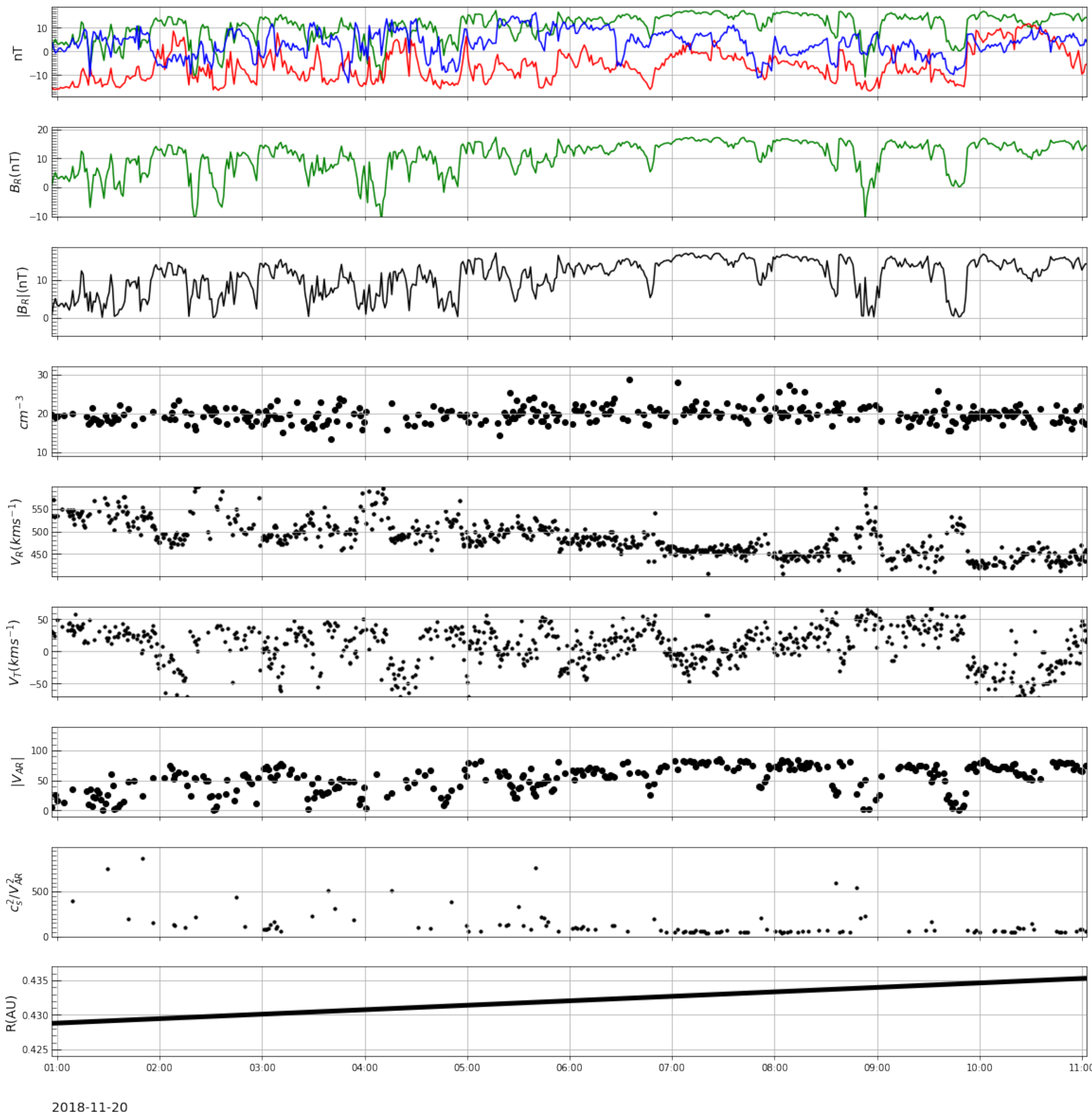
[252]:
```
x1 = A.iloc[:,0] #Magnetic flux time values
y1_1 = A.iloc[:,2] #B_R
y1_2 = A.iloc[:,3] #B_T
y1_3 = A.iloc[:,4] #B_N

x2 = A.iloc[:, 0] #Number density time values
y2 = B_interpolated.iloc[:, 2] #Number density

x3 = C.iloc[:, 0] #Bulk velocity times
y3_1 = C.iloc[:, 2] #Radial velocity
y3_2 = C.iloc[:, 3] #Tangential velocity

x4 = x1
y4 = np.abs(N)

x5 = x1
y5 =( C_rs / y4)**2

x6 = x1 #Spacecraft times
y6 = D_rs #Distance from barycenter in AU
```

[274]:
```
# Plot time series of Magnetic flux density in RTN coordinates
fig, axes = plt.subplots(nrows=9, figsize=(16, 16))
for ax in axes:
    ax.grid(True)
fig.tight_layout()



# Subplot 0
ax0 = axes[0]
ax0.plot(x1, y1_1, color='g')
ax0.plot(x1, y1_2, color='r')
ax0.plot(x1, y1_3, color='b')
ax0.set_ylabel('nT', fontsize=14)
ax0.set_xlim([min(A.iloc[:, 0]), max(A.iloc[:, 0])])
major_ticks = np.arange(-10, 20, 10)
minor_ticks = np.arange(-19, 19, 1)
ax0.tick_params(axis='y', which='major', direction='in',length=10)
ax0.tick_params(axis='y', which='minor', direction='in',length=5)
ax0.set_yticks(major_ticks)
ax0.set_yticks(minor_ticks, minor=True)
ax0.set_xticklabels([])

#Magnetic flux in the radial direction
ax1 = axes[1]
```

```python
ax1.plot(x1, y1_1, color='g')
ax1.set_ylabel('$B_{R}$(nT)', fontsize=14)
ax1.set_xlim([min(A.iloc[:, 0]), max(A.iloc[:, 0])])
ax1.set_ylim([-10, 21])
major_ticks = np.arange(-10, 21, 10)
minor_ticks = np.arange(-10, 21, 1)
ax1.set_yticks(major_ticks)
ax1.set_yticks(minor_ticks, minor=True)
ax1.tick_params(axis='y', which='major', direction='in',length=10)
ax1.tick_params(axis='y', which='minor', direction='in',length=5)
ax1.set_xticklabels([])

#Absolute value of radial magnetic flux (magnitude is directionally invariant)
ax2 = axes[2]
ax2.plot(x1, abs(y1_1), color='k')
ax2.set_ylabel('$|B_{R}|$(nT)', fontsize=14)
ax2.set_xlim([min(A.iloc[:, 0]), max(A.iloc[:, 0])])
major_ticks = np.arange(0, 20, 10)
minor_ticks = np.arange(-5, 20, 1)
ax2.set_yticks(major_ticks)
ax2.set_yticks(minor_ticks, minor=True)
ax2.tick_params(axis='y', which='major', direction='in',length=10)
ax2.tick_params(axis='y', which='minor', direction='in',length=5)
ax2.set_xticklabels([])


#Bulk proton number density
ax3 = axes[3]
ax3.scatter(x2, y2, color='k')
ax3.set_ylabel('$cm^{-3}$', fontsize=14)
ax3.set_ylim([9,32])
ax3.set_xlim([min(A.iloc[:, 0]), max(A.iloc[:, 0])])
major_ticks = np.arange(10, 32, 10)
minor_ticks = np.arange(9, 32, 1)
ax3.set_yticks(major_ticks)
ax3.set_yticks(minor_ticks, minor=True)
ax3.tick_params(axis='y', which='major', direction='in',length=10)
ax3.tick_params(axis='y', which='minor', direction='in',length=5)
ax3.set_xticklabels([])

#Radial velocity of charged particles (solar wind)
ax4 = axes[4]
ax4.scatter(x3, y3_1, color='k',s=12)
ax4.set_ylabel('$V_{R}(kms^{-1})$', fontsize=14)
ax4.set_ylim([400, 600])
ax4.set_xlim([min(C.iloc[:, 0]), max(C.iloc[:, 0])])
major_ticks = np.arange(450, 600, 50)
```

```
minor_ticks = np.arange(400, 600, 10)
ax4.set_yticks(major_ticks)
ax4.set_yticks(minor_ticks, minor=True)
ax4.tick_params(axis='y', which='major', direction='in',length=10)
ax4.tick_params(axis='y', which='minor', direction='in',length=5)
ax4.set_xticklabels([])

#Tangential velocity of charged particles (solar wind)
ax5 = axes[5]
ax5.scatter(x3, y3_2, color='k', s=10)
ax5.set_ylabel('$V_{T}(kms^{-1})$', fontsize=14)
ax5.set_ylim([-70, 70])
ax5.set_xlim([min(C.iloc[:, 0]), max(C.iloc[:, 0])])
major_ticks = np.arange(-50, 70, 50)
minor_ticks = np.arange(-70, 70, 10)
ax5.set_yticks(major_ticks)
ax5.set_yticks(minor_ticks, minor=True)
ax5.tick_params(axis='y', which='major', direction='in',length=10)
ax5.tick_params(axis='y', which='minor', direction='in',length=5)
ax5.set_xticklabels([])

#Absolute value of radial Alfvenic wave velocity
ax6 = axes[6]
ax6.scatter(x4, y4, color='k')
ax6.set_ylabel('$|V_{AR}|$', fontsize=14)
ax6.set_xlim([min(A.iloc[:, 0]), max(A.iloc[:, 0])])
major_ticks = np.arange(0, 150, 50)
minor_ticks = np.arange(-10, 150, 10)
ax6.set_yticks(major_ticks)
ax6.set_yticks(minor_ticks, minor=True)
ax6.tick_params(axis='y', which='major', direction='in',length=10)
ax6.tick_params(axis='y', which='minor', direction='in',length=5)
ax6.set_xticklabels([])

#The ratio of the speed of sound in the plasma to Alfvenic wave speed, squared␣
↪to guarantee positive values
ax7 = axes[7]
#ax7.set_yscale('log')
ax7.scatter(x5, y5, color='k', s=10)
ax7.set_ylabel('$c_{s}^{2}/V_{AR}^{2}$', fontsize=14)
ax7.set_ylim([1, 1000])
ax7.set_xlim([min(A.iloc[:, 0]), max(A.iloc[:, 0])])
major_ticks = np.arange(0, 1000, 500)
minor_ticks = np.arange(0, 1000, 50)
ax7.set_yticks(major_ticks)
ax7.set_yticks(minor_ticks, minor=True)
ax7.tick_params(axis='y', which='major', direction='in',length=10)
```

```python
ax7.tick_params(axis='y', which='minor', direction='in',length=5)
ax7.set_xticklabels([])


#Distance from spacecraft to barycenter.
ax8 = axes[8]
ax8.plot(x6, y6, color='k', linewidth=5)
ax8.set_ylabel('R(AU)', fontsize=14)
ax8.set_xlabel('2018-11-20', ha='left', fontsize=14)
ax8.xaxis.set_label_coords(0, -0.5)
ax8.set_xlim([min(A.iloc[:, 0]), max(A.iloc[:, 0])])
ax8.set_ylim([0.424, 0.437])
major_ticks = np.arange(0.425, 0.435, 0.005)
minor_ticks = np.arange(0.425, 0.436, 0.001)
ax8.set_yticks(major_ticks)
ax8.set_yticks(minor_ticks, minor=True)
ax8.tick_params(axis='y', which='major', direction='in',length=10)
ax8.tick_params(axis='y', which='minor', direction='in',length=5)
ax8.set_xticks(x6)
ax8.xaxis.set_major_locator(mdates.HourLocator(interval=1))
ax8.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
ax8.xaxis.set_minor_locator(mdates.HourLocator(interval=2))


plt.show()
```


nT
B_R(nT)
$|B_R|$(nT)
cm^{-3}
$V_R(kms^{-1})$
$V_T(kms^{-1})$
$|V_{AR}|$
c_s^2/V_{AR}^2
R(AU)
01:00 02:00 03:00 04:00 05:00 06:00 07:00 08:00 09:00 10:00 11:00
2018-11-20

[ ]: