First science with interoperable data School. SVO. February 2021

EXPLORING KEPLER-1661 DATA USING VO TOOLS AND FITS2OSC

Adrián García Riber
PhD Student. Music and Technology Program
Polytechnic University and Royal Conservatory of Madrid
adrian.griber@alumnos.upm.es

RESEARCH FRAME

- ✓ Developing an astronomical data automatic Music composition system based on Deep Learning.
- ✓ Allowing the interactive exploration of astronomical databases through sound in user-friendly non-time-consuming environments.
- ✓ Expanding the possibilities in the use of astronomical information as sound source for Music composition.
- ✓ Identifying the fields in which Sonification and multimodal displays can complement the tradicional graphic representations for a better understanding of astronomical phenomena.

USING VO TOOLS

- ✓ Exploring the possibilities of interoperable VO tools in the case study of Kepler 1661 star.
- ✓ Using Aladin, Vizzier and Pyvo to access astronomical catalogs.
- ✓ Using VOSA to estimate the Mass and Temperature of the star and comparing the results with the information found in the reference literature.
- ✓ Integrating the VO tools and the FITS2OSC pipeline in a Jupiter notebook to provide a compact interface for easily displaying graphic and auditory representations of the Kepler 1661 case study.
- ✓ Providing an auditory representation of Kepler 1661 light curves.

THE CASE STUDY. Kepler 1661 (Socia et al. 2020)

- ✓ KIC 6504534 was discovered and cataloged as a ~28.2 day eclipsing binary system in the second revision of the Kepler Eclipsing Binary Catalog (Prsa et al. 2011; Slawson et al. 2011).
- ✓ The Kepler Input Catalog (KIC) provides the following estimates for the stellar parameters: Kepmag = 14.216, T_{eff} = 4748 K, log g = 4.46, metallicity = -0.10, and a contamination = 0.00 for all four Kepler Seasons.
- ✓ The system hosts a Neptune-size (R_p = 3.87 +/- 0.06 R_Earth) transiting circumbinary planet, Kepler-1661 b, found in the Kepler photometry. The planet has a period of ~175 days and its orbit precesses with a period of only 35 years.

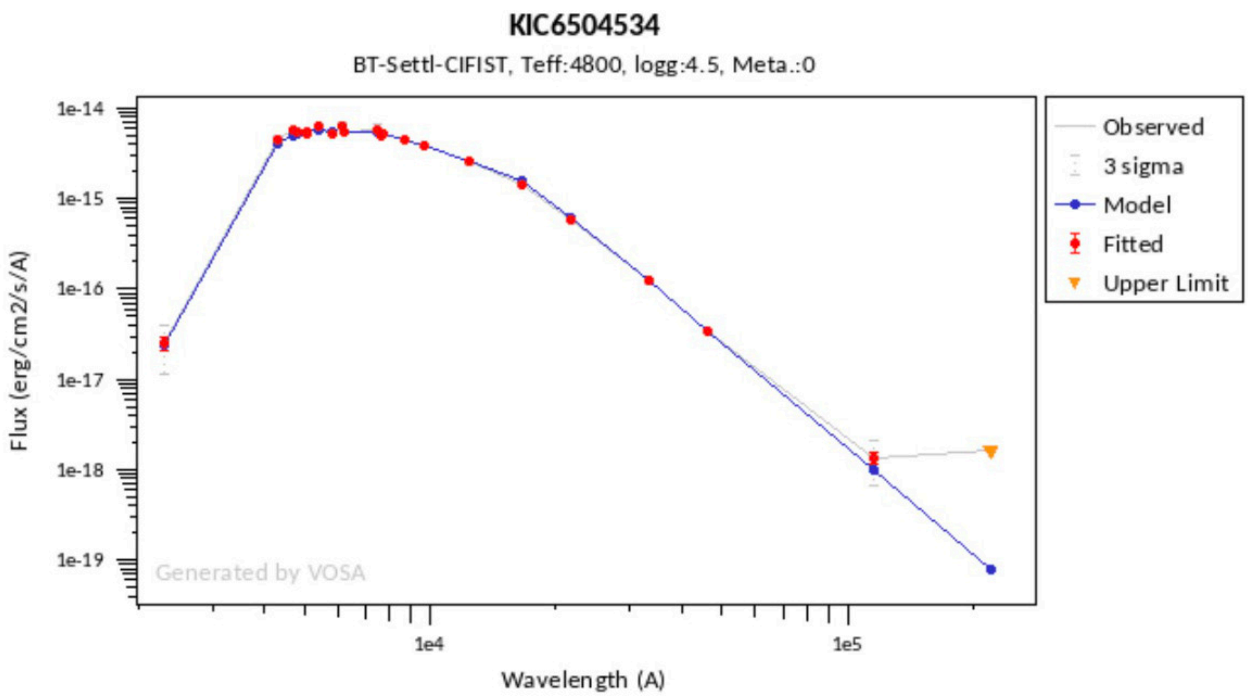
VOSA RESULTS

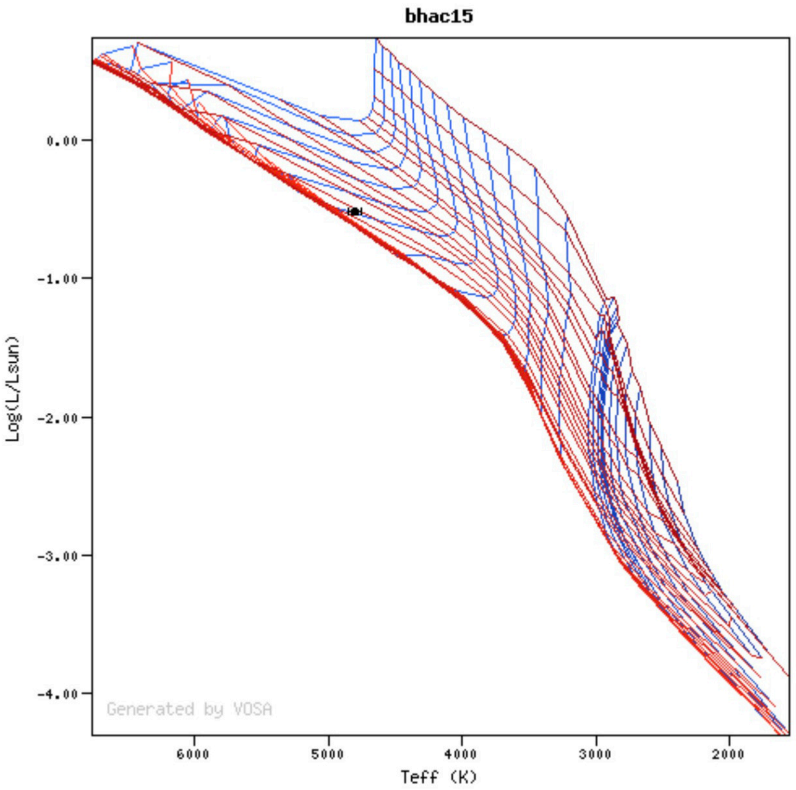
✓ Consistent with literature.

Object	Model	T_{eff}		LogL		Age		Mass		
KIC6504534	BHAC15	4800	(4750,4850)	-0.5195	(-0.5348,-0.5048)	0.0428	..(0.0397,0.0500)	0.8271	(0.8048,0.8595)	[1]

[1] The distance to one of the closer curves has been estimated as the one to the closest point in the curve

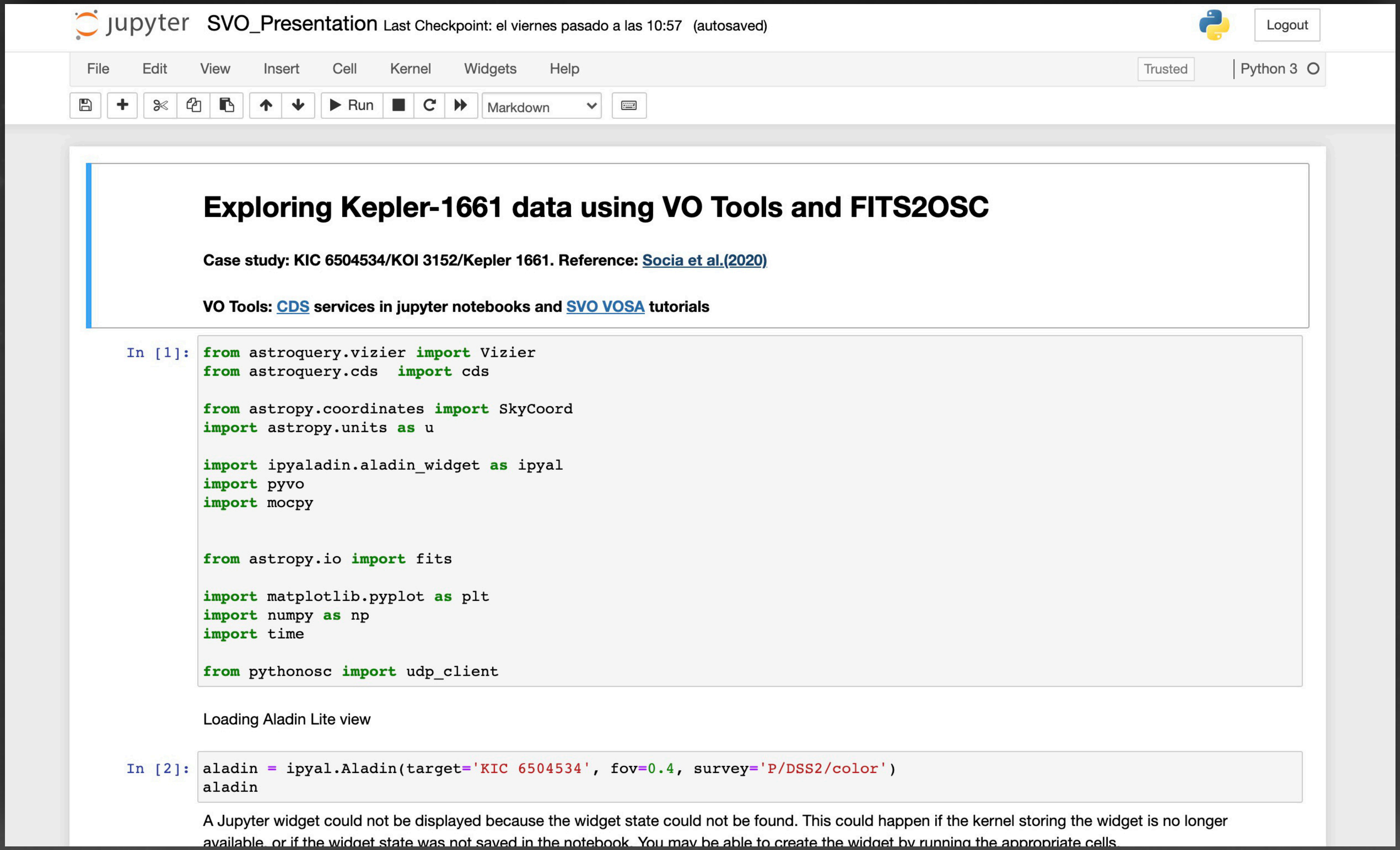
Object	RA	DEC	D (pc)	Model	A_V	T_{eff}	logg	Meta.	more
KIC6504534	285.16713903	+41.96705206	409.110	BT-Settl-CIFIST	---	4800	4.5	0	---





JUPITER NOTEBOOK

- ✓ Aladin widget
- ✓ Vizier and Pyvo access
- ✓ Target and data visualization
- ✓ Light curve graphic display
- ✓ Light curve auditory display



jupyter SVO_Presentation Last Checkpoint: el viernes pasado a las 10:57 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Markdown

Exploring Kepler-1661 data using VO Tools and FITS2OSC

Case study: KIC 6504534/KOI 3152/Kepler 1661. Reference: Socia et al.(2020)

VO Tools: CDS services in jupyter notebooks and SVO VOSA tutorials

In [1]:

```
from astroquery.vizier import Vizier
from astroquery.cds  import cds

from astropy.coordinates import SkyCoord
import astropy.units as u

import ipyaladin.aladin_widget as ipyal
import pyvo
import mocpy


from astropy.io import fits

import matplotlib.pyplot as plt
import numpy as np
import time

from pythonosc import udp_client
```

Loading Aladin Lite view

In [2]:

```
aladin = ipyal.Aladin(target='KIC 6504534', fov=0.4, survey='P/DSS2/color')
aladin
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

SVO_Presentation

February 24, 2021

1 Exploring Kepler-1661 data using VO Tools and FITS2OSC

Case study: KIC 6504534/KOI 3152/Kepler 1661. Reference: Socia et al.(2020)

VO Tools: CDS services in jupyter notebooks and SVO VOSA tutorials

[ ]:
```
from astroquery.vizier import Vizier
from astroquery.cds  import cds

from astropy.coordinates import SkyCoord
import astropy.units as u

import ipyaladin.aladin_widget as ipyal
import pyvo
import mocpy


from astropy.io import fits

import matplotlib.pyplot as plt
import numpy as np
import time

from pythonosc import udp_client
```

Loading Aladin Lite view

[ ]:
```
aladin = ipyal.Aladin(target='KIC 6504534', fov=0.4, survey='P/DSS2/color')
aladin
```

Listing Vizier results for Kepler Eclipsing Binary Catalog (Prsa et al.2011) and (Slawson et al.2011)

[ ]:
```
catalog_list_KOI = Vizier.find_catalogs('Slawson')
for k, v in catalog_list_KOI.items():
    print(k, ': ', v.description)
```

Searching for the catalogue with pyvo

[ ]:
```
tap_vizier = pyvo.dal.TAPService('http://tapvizier.u-strasbg.fr/TAPVizieR/tap')
query = """SELECT  *  FROM tap_schema.tables
               WHERE table_name LIKE '%J/AJ/142/160%' """
catalog_list_KOI = tap_vizier.search(query).to_table()
catalog_list_KOI['table_name', 'description']
```

Loading and printing the catalog

[ ]:
```
Vizier.ROW_LIMIT = -1
catalogs_KOI = Vizier.get_catalogs('J/AJ/142/160/v3')
```

[ ]:
```
print(catalogs_KOI)
```

[ ]:
```
table_KOIlist = catalogs_KOI[0]
table_KOIlist
```

[ ]:
```
tap_vizier = pyvo.dal.TAPService('http://tapvizier.u-strasbg.fr/TAPVizieR/tap')
query = """SELECT  *  FROM "J/AJ/142/160/v3" """
table_KOIlist = tap_vizier.search(query).to_table()
table_KOIlist
```

Uploading the data into Aladin widget

[ ]:
```
aladin.add_table(table_KOIlist)
```

Loading time series FITS files from the STScI MAST archive

[ ]:
```
fits_file = "https://archive.stsci.edu/missions/kepler/lightcurves/0065/
 ↪006504534/kplr006504534-2012088054726_llc.fits"

fits.getdata(fits_file, ext=1).columns

with fits.open(fits_file, mode="readonly") as hdulist:
    k2_bjds = hdulist[1].data['TIME']
    sap_fluxes = hdulist[1].data['SAP_FLUX']
    pdcsap_fluxes = hdulist[1].data['PDCSAP_FLUX']
```

Adjusting the range and speed for the graphic and auditory representation of the light curve

[ ]:
```
length = len(pdcsap_fluxes)

#User selectable range for the graphic representation:
t2= np.nanmax(k2_bjds) #Horizontal axis Right limit           #Star spot at:␣
 ↪1101.5  #Transit at: 1142.50
t1= np.nanmin(k2_bjds) #Horizontal axis Left limit            #Star spot at:␣
 ↪1101     #Transit at: 1141.73
max = np.nanmax(pdcsap_fluxes)+20 #Vertical axis Max.limit   #""                   ␣
 ↪    #28900
```

```
min = np.nanmin(pdcsap_fluxes)-20 #Vertical axis Min.limit    #""
↪     #28750

#User selectable range for the auditory representation:
i2= length #Equivalent index for "t2"                       #Star spot at: 104
↪     #Transit at: 2110
i1= 0 #Equivalent index for "t1"                            #Star spot at: 79
↪     #Transit at: 2072

#Speed control for the auditory representation:
s=100                                                       #2
↪#2
```

Representing the Lightcurve

[ ]:
```
fig, ax = plt.subplots()
fig.set_size_inches(12., 8.)
ax.plot(k2_bjds, pdcsap_fluxes, 'r.')

ax.set_xlim(t1, t2)
ax.set_ylim(min, max)

fig.suptitle(fits_file)
ax.set_ylabel("PDCSAP Flux (e-/s)")
ax.set_xlabel("Time (BKJD)")

plt.savefig('Lightcurve.png')
```

Lauching udp clients

[ ]:
```
client1 = udp_client.SimpleUDPClient("127.0.0.1", 9999) #x
client2 = udp_client.SimpleUDPClient("127.0.0.1", 9998) #y
client3 = udp_client.SimpleUDPClient("127.0.0.1", 9997) #length
client4 = udp_client.SimpleUDPClient("127.0.0.1", 9996) #speed
client5 = udp_client.SimpleUDPClient("127.0.0.1", 9995) #index
client6 = udp_client.SimpleUDPClient("127.0.0.1", 9994) #play/stop
```

Sending Light curve data via OSC and printing final information

[ ]:
```
client4.send_message("/s", s)
client6.send_message("/p", 1)

for i in range (i1, i2):
    print("index:", i)
    print("Time (BKJD):", k2_bjds[i]);
    print("PDCSAP Flux (e-/s):", pdcsap_fluxes[i]);

    x=float(k2_bjds[i])
```



```
    y=float(pdcsap_fluxes[i])

    client5.send_message("/i", i)
    client1.send_message("/x", x)
    client2.send_message("/y", y)
    time.sleep(1/s)

client1.send_message("/x", 0)
client2.send_message("/y", 1)
client6.send_message("/p", 0)
#-----------------------------------------------------------------------
print ("Time represented:",float(t2-t1), "BKJD");
print ("Number of samples represented:",int(i2-i1));
```

MUSICAL PRECEDENTS TO ILLUSTRATE FUTURE WORK

Quadrivium Soundscapes is a quadraphonic musical composition of four movements that explore the creative possibilities of astronomical data sonification. Mainly extracted from the light curve database of the *Planet Hunters TESS* project, *Astronomie, Musique, Géométrie*, and *Arithmétique*, offer an auditory exploration of selected synthesized light curves in four different degrees of abstraction and signification. The piece includes a sonification of *Trappist-1* exoplanetary system generated with data from *NASA Exoplanet Archive* and has been created using *Cabbage Audio* and the sonification software prototypes *Sonifigrapher* and *Planethesizer,* both available for free download at: https://archive.org/details/@agriber.

Thank you.

Feedback: adrian.griber@.alumnos.upm.es