CRIRES-planning-tool documentation

Version 15. September 2020

Introduction:

CRIRES-planning-tool is a software tool developed over the course of my Master's project under the supervision of Prof. Dr. Nikolai Piskunov and Dr. Andreas Korn, in collaboration with Dr. Alexis Lavail

The CRIRES-planning-tool is intended to be used to plan transit observations of exoplanets for CRIRES+, the new cross-dispersed high-resolution infrared spectrograph for the ESO VLT CRIRES+. Observaton of exoplanets can be planed in two ways. Single candidate by name in a given timespan or constraints for observable candidates by CRIRES+ can be loaded from a file: Nasa_Archive_Selection.txt (see section: **Constraints for Candidates**). The known exoplanets fulfilling these constraints are downloaded from Nasa Exoplanet Archive and each candidate is checked for its observability from Cerro Paranal, Chile for its observability during a given time frame. Each observable candidate is checked for a minimum signal-to-noise ratio (S/N)≥100 during 20 exposures. Each exposure is related to its total exposure time, calculated from the detector integration time times the number of detector integrations: (TEXP = DIT x NDIT) and NDIT is optimized to be withing 16≤NDIT≤32 for each exposure (see section: **Exposure Time Calculator**). Candidates reaching 20 exposures during the complete transit are added to the list of observable candidates and further informations can be found in the output excel files of accepted candidates (see section: **Result files**). The tool comes with plotting tools and a commandline window to access its functionalities. This document shall give an overview about the functionalities, accessibility, structure, installation and further development possibilities of the CRIRES-planning-tool.

Installation:

1. Navigate to your chosen directory to install the CRIRES-planning-tool.
2. Download github repository:
   `git clone https://github.com/jonaszubindu/CRIRES-planning-tool`
3. Setup a virtual environment to install the correct packages to run the planning tool

   `cd CRIRES-planning-tool/python`

   `pip install virtualenv`

   `virtualenv --python python3.7 [name of your venv]`

   activate your virtual environment:

   `source [name of your venv]/bin/activate`

   install the requirements to run CRIRES-planning-tool stored in requirements.txt `pip  install -r requirements.txt`

    after you are done with running CRIRES-planning-tool use

    `deactivate`

    to deactivate the virtual environment.

4. Create directories for data storage:

    `mkdir Plots csv_files picklefiles`

5. To run CRIRES-planning-tool run go into the p

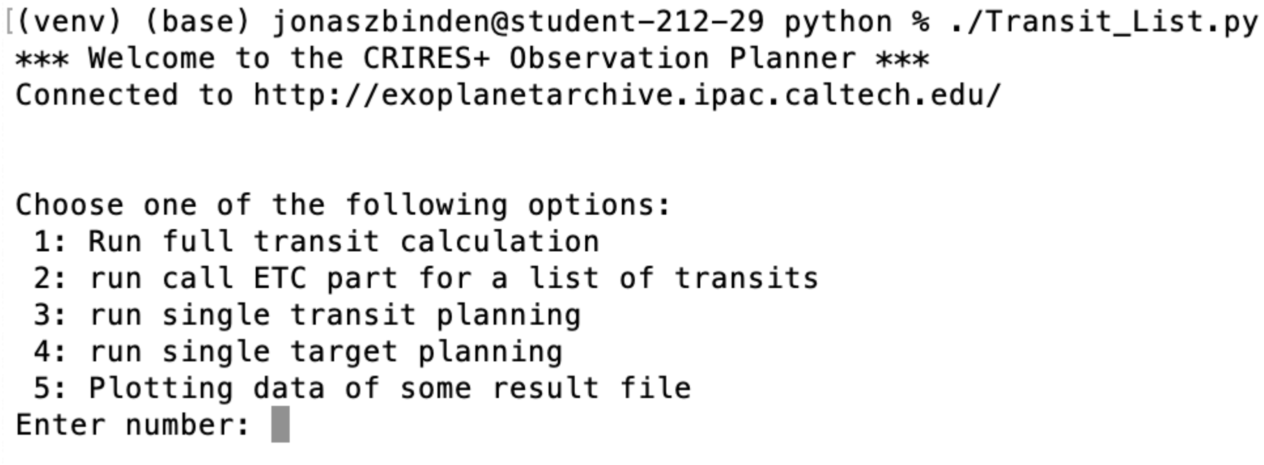
    `./Transit_List.py`

    If ./Transit_List.py has not the proper rights to be run, use

    `chmod +x Transit_List.py`

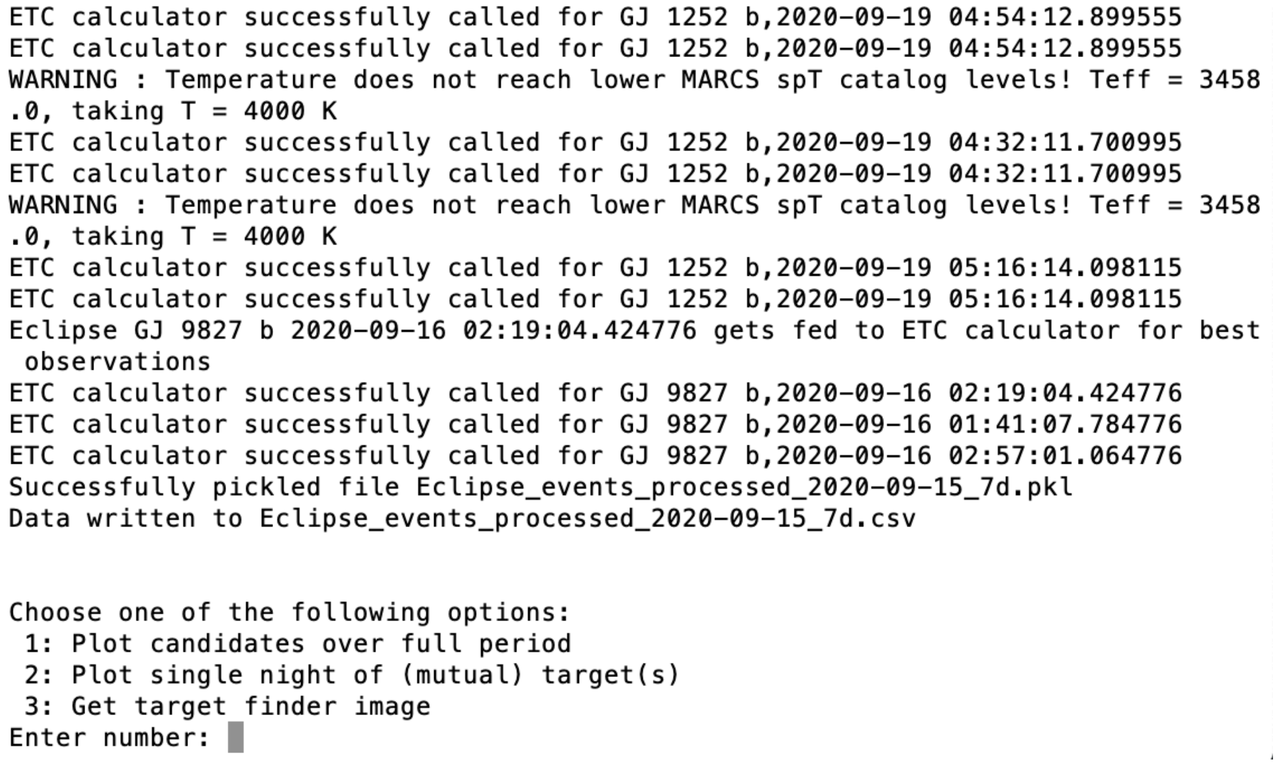
Commandline Menu

Running Transit_List.py presents the following commandline window with options:



```
[(venv) (base) jonaszbinden@student-212-29 python % ./Transit_List.py
*** Welcome to the CRIRES+ Observation Planner ***
Connected to http://exoplanetarchive.ipac.caltech.edu/


Choose one of the following options:
 1: Run full transit calculation
 2: run call ETC part for a list of transits
 3: run single transit planning
 4: run single target planning
 5: Plotting data of some result file
Enter number: █
```

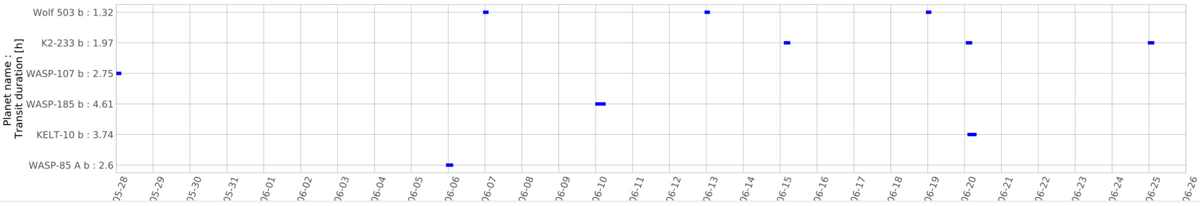
1. Runs a complete check of all available candidates fulfilling the constraints from Nasa_Archive_Selection.txt for a certain timescale. The tool asks for the starting date and the number of days to run the candidate list for and asks if the ETC part should also be run. Final results can only be optained by running the ETC part as well.
2. Runs the ETC part, where each observation of each observable candidate is checked for the possibiliy of 20 exposures with each one of them S/N≥100 from a stored picklefile. This can be used for instance if something during the ETC part running option 1 goes wrong, and one wants to continue from where the problem occured in the first place.
3. Checks the observability of a single candidate by name for a certain timeframe.
4. Other targets can be run in the same way as exoplanetary candidates. However, this feature is not included yet.
5. Make plots from stored datafile, this option is also presented at the end of running 1, 2 or 3.



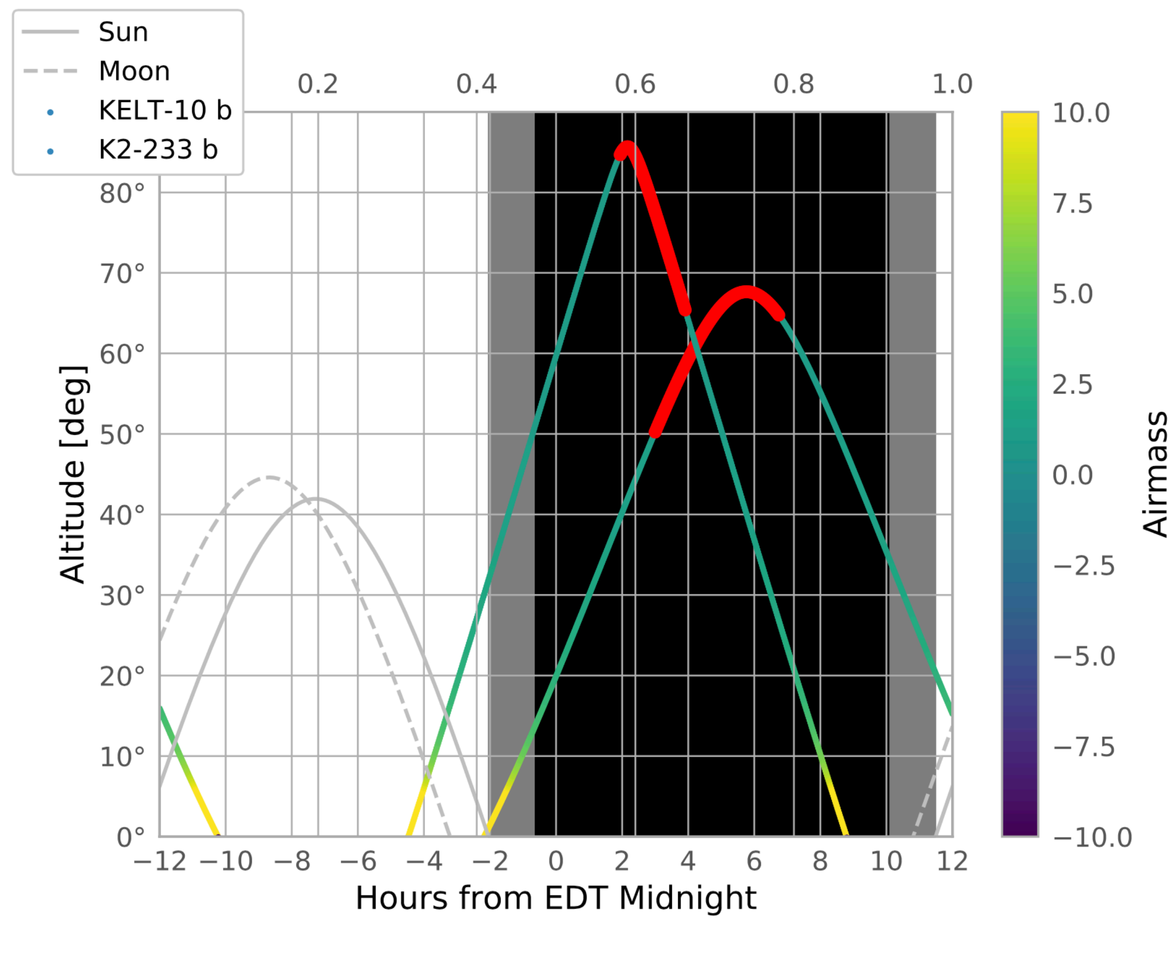
```
ETC calculator successfully called for GJ 1252 b,2020-09-19 04:54:12.899555
ETC calculator successfully called for GJ 1252 b,2020-09-19 04:54:12.899555
WARNING : Temperature does not reach lower MARCS spT catalog levels! Teff = 3458
.0, taking T = 4000 K
ETC calculator successfully called for GJ 1252 b,2020-09-19 04:32:11.700995
ETC calculator successfully called for GJ 1252 b,2020-09-19 04:32:11.700995
WARNING : Temperature does not reach lower MARCS spT catalog levels! Teff = 3458
.0, taking T = 4000 K
ETC calculator successfully called for GJ 1252 b,2020-09-19 05:16:14.098115
ETC calculator successfully called for GJ 1252 b,2020-09-19 05:16:14.098115
Eclipse GJ 9827 b 2020-09-16 02:19:04.424776 gets fed to ETC calculator for best
 observations
ETC calculator successfully called for GJ 9827 b,2020-09-16 02:19:04.424776
ETC calculator successfully called for GJ 9827 b,2020-09-16 01:41:07.784776
ETC calculator successfully called for GJ 9827 b,2020-09-16 02:57:01.064776
Successfully pickled file Eclipse_events_processed_2020-09-15_7d.pkl
Data written to Eclipse_events_processed_2020-09-15_7d.csv


Choose one of the following options:
 1: Plot candidates over full period
 2: Plot single night of (mutual) target(s)
 3: Get target finder image
Enter number: 
```

The following plots can be produced:

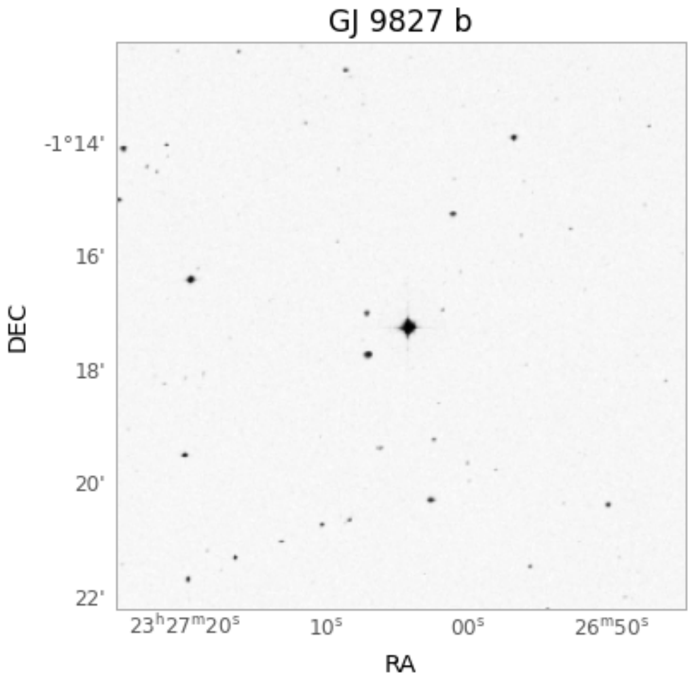
Schedule for entire period:



Graphic depiction of single night:



and target finder image by name:

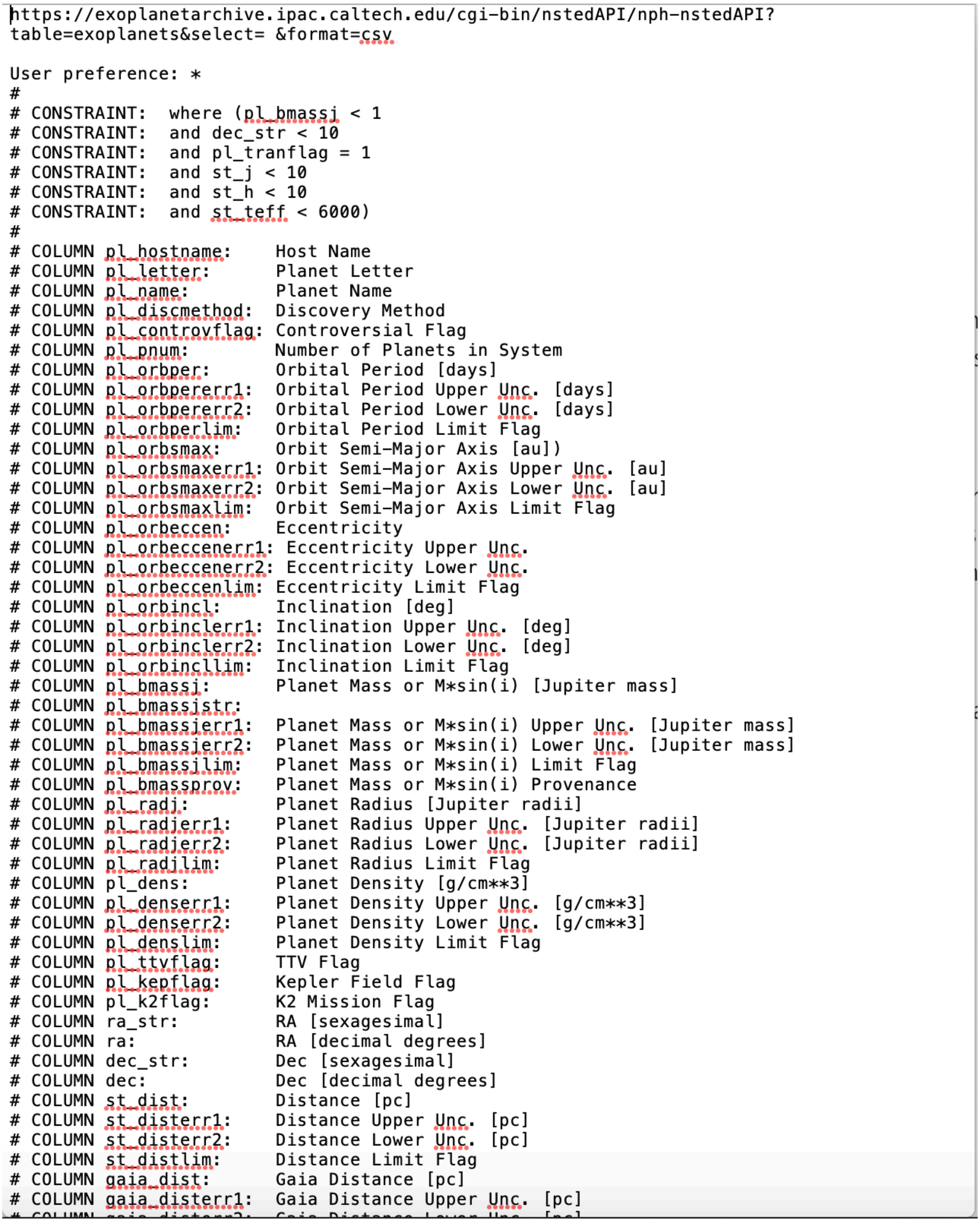


The tool should guide one self-explanatory through each step.

For the case, the user gets asked to enter the name of the appropriate picklefile, refer to section: **Result files**.

Constraints for Candidates

The constraints can be found in Nasa_Archive_Selection.txt and can be adequately changed. The file also contains the column names that should be loaded. Only few of these columns are at the end actually called, however, as the tool evolves for other applications, columns can be added or deleted, as desired:



```
https://exoplanetarchive.ipac.caltech.edu/cgi-bin/nstedAPI/nph-nstedAPI?
table=exoplanets&select= &format=csv

User preference: *
#
# CONSTRAINT:  where (pl_bmassj < 1
# CONSTRAINT:  and dec_str < 10
# CONSTRAINT:  and pl_tranflag = 1
# CONSTRAINT:  and st_j < 10
# CONSTRAINT:  and st_h < 10
# CONSTRAINT:  and st_teff < 6000)
#
# COLUMN pl_hostname:    Host Name
# COLUMN pl_letter:      Planet Letter
# COLUMN pl_name:        Planet Name
# COLUMN pl_discmethod:  Discovery Method
# COLUMN pl_controvflag: Controversial Flag
# COLUMN pl_pnum:        Number of Planets in System
# COLUMN pl_orbper:      Orbital Period [days]
# COLUMN pl_orbpererr1:  Orbital Period Upper Unc. [days]
# COLUMN pl_orbpererr2:  Orbital Period Lower Unc. [days]
# COLUMN pl_orbperlim:   Orbital Period Limit Flag
# COLUMN pl_orbsmax:     Orbit Semi-Major Axis [au])
# COLUMN pl_orbsmaxerr1: Orbit Semi-Major Axis Upper Unc. [au]
# COLUMN pl_orbsmaxerr2: Orbit Semi-Major Axis Lower Unc. [au]
# COLUMN pl_orbsmaxlim:  Orbit Semi-Major Axis Limit Flag
# COLUMN pl_orbeccen:    Eccentricity
# COLUMN pl_orbeccenerr1: Eccentricity Upper Unc.
# COLUMN pl_orbeccenerr2: Eccentricity Lower Unc.
# COLUMN pl_orbeccenlim: Eccentricity Limit Flag
# COLUMN pl_orbincl:     Inclination [deg]
# COLUMN pl_orbinclerr1: Inclination Upper Unc. [deg]
# COLUMN pl_orbinclerr2: Inclination Lower Unc. [deg]
# COLUMN pl_orbincllim:  Inclination Limit Flag
# COLUMN pl_bmassj:      Planet Mass or M*sin(i) [Jupiter mass]
# COLUMN pl_bmassjstr:
# COLUMN pl_bmassjerr1:  Planet Mass or M*sin(i) Upper Unc. [Jupiter mass]
# COLUMN pl_bmassjerr2:  Planet Mass or M*sin(i) Lower Unc. [Jupiter mass]
# COLUMN pl_bmassjlim:   Planet Mass or M*sin(i) Limit Flag
# COLUMN pl_bmassprov:   Planet Mass or M*sin(i) Provenance
# COLUMN pl_radj:        Planet Radius [Jupiter radii]
# COLUMN pl_radjerr1:    Planet Radius Upper Unc. [Jupiter radii]
# COLUMN pl_radjerr2:    Planet Radius Lower Unc. [Jupiter radii]
# COLUMN pl_radjlim:     Planet Radius Limit Flag
# COLUMN pl_dens:        Planet Density [g/cm**3]
# COLUMN pl_denserr1:    Planet Density Upper Unc. [g/cm**3]
# COLUMN pl_denserr2:    Planet Density Lower Unc. [g/cm**3]
# COLUMN pl_denslim:     Planet Density Limit Flag
# COLUMN pl_ttvflag:     TTV Flag
# COLUMN pl_kepflag:     Kepler Field Flag
# COLUMN pl_k2flag:      K2 Mission Flag
# COLUMN ra_str:         RA [sexagesimal]
# COLUMN ra:             RA [decimal degrees]
# COLUMN dec_str:        Dec [sexagesimal]
# COLUMN dec:            Dec [decimal degrees]
# COLUMN st_dist:        Distance [pc]
# COLUMN st_disterr1:    Distance Upper Unc. [pc]
# COLUMN st_disterr2:    Distance Lower Unc. [pc]
# COLUMN st_distlim:     Distance Limit Flag
# COLUMN gaia_dist:      Gaia Distance [pc]
# COLUMN gaia_disterr1:  Gaia Distance Upper Unc. [pc]
# COLUMN gaia_disterr2:  Gaia Distance Lower Unc. [pc]
```

The constraints are loaded with the script Request_Table_NasaExoplanetArchive.py: (excerpt from code documentation)

"*Request Confirmed Exoplanets Table from Nasa Exoplanet Archive*

This script opens a file with constraints and columns that should constrain the Nasa exoplanets archive data.
The script contains two important information:

which columns do you want to import in your Exoplanet table

and

with which contraints should the table be filtered.

The script looks automatically for constraints and columns in a file called Nasa_Archive_Selection.txt.
It is important that columns are defined as COLUMN and constraints as CONSTRAINT for the script to find them.
Please do not add any special characters to a column or constraint. Write the constraint in the format

constraint < value

explicitly with spaces similar to the other constraints. The logic symbol < and > are inclusive(>=, <=).
Like this the module will find the details of the constraint. Make sure that the defined constraints are also columns of the table you request. Otherwise the constraints are not applicable.
The script creates a URL to request for the exoplanet table and filters the initial table after the constraints.
It stores a .csv file of that table that can be imported to Transit_List.py via csv_file_import.py"

The retrieved data are stored in a csv-file which has a default name PlanetList.csv and is stored in /CRIRES-planning-tool. Running option 1 in Transit_List.py will import the name column of PlanetList.csv. Running the script will yield the following

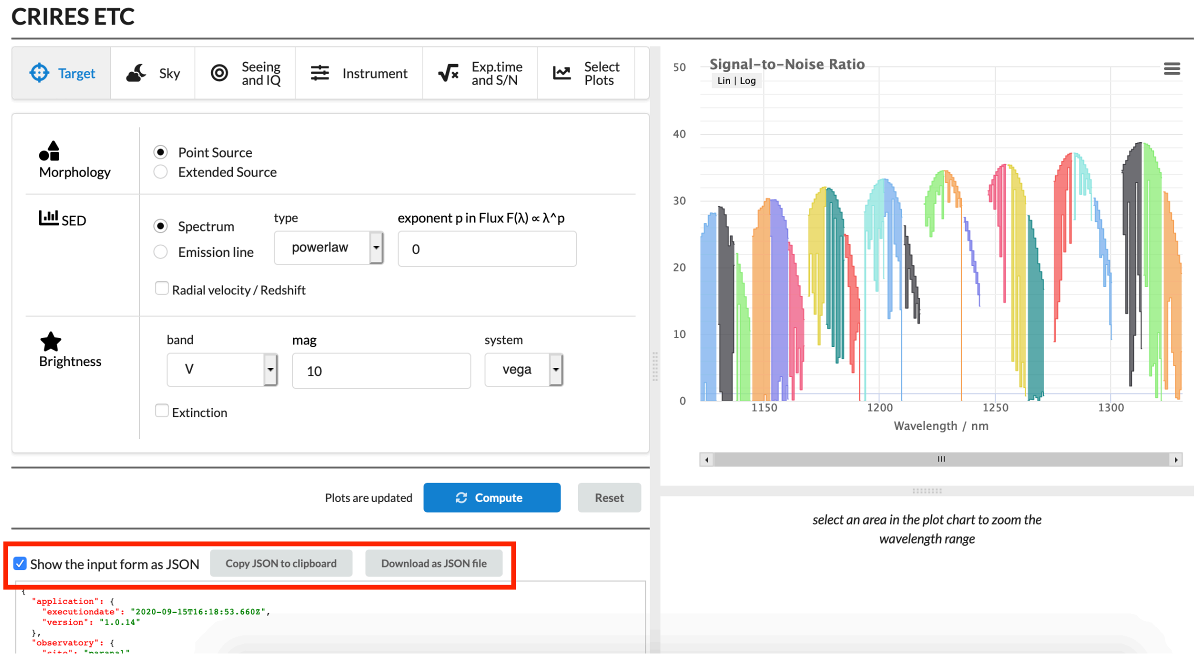
```
Write name to store file: [PlanetList.csv]
```

where one can choose a different name (without suffix .csv) to store the data and press enter, if one would like to use the data from the Nasa exoplanet archive in a different manner. Pressing enter will choose the default name.

Exposure Time Calculator

The exposure time calculator is called through a client and requires a json input file containing all the input data to compute the exposure time or the signal-to-noise ratio for a particular observation. The exposure time calculator is provided by ESO and maintained by Jakob Vinther. The theory behind the ETC can be looked up in my thesis: *Planning observations of terrestrial Exoplanetsaround M type Stars with CRIRES+*, section *2.8 Signal-to-noise ratio and the Exposure Time Calculator*. The public interface can be accessed here. Any updates of the etc conflicting with the CRIRES-planning tool should be checked in correspondence with Jakob Vinther. Here are a few reasons why CRIRES-planning-tool might not be able to access the ETC anymore and strategies to solve it:

1. The baseurl to call the ETC with the cli has changed. You can change the baseurl in the file: /python/classes_methods/etc_cli.py
2. The structure of the input json file has changed. There are several ways to fix this. The easiest way is by accessing the api version of the etc and plugging in standard inputs. Clicking on the box Show json input file:



CRIRES ETC

Morphology: ● Point Source ○ Extended Source

SED: ● Spectrum ○ Emission line; type: powerlaw; exponent p in Flux F(λ) ∝ λ^p: 0

□ Radial velocity / Redshift

Brightness: band: V; mag: 10; system: vega

□ Extinction

Plots are updated

☑ Show the input form as JSON

```
{
  "application": {
    "executiondate": "2020-09-15T16:18:53.660Z",
    "version": "1.0.14"
  },
  "observatory": {
    "site": "paranal"
```

select an area in the plot chart to zoom the wavelength range

one can download the jsonfile and depending on the desired input method as one of the following. Before you store you store the file, make sure that you make a copy of the old json file(s).

calculating S/N, using spectral templates -> store file as:

> etc-form-default-snr-Templ.json

calculating S/N, using effective temperature of the target -> store file as:

> etc-form-default-snr-Teff.json

or calculating exposure time for minimum S/N, using spectral templates: -> store file as:

> etc-form-default-ndit-Templ.json

or calculating exposure time for minimum S/N, using effective temperature of the target: -> store file as:

> etc-form-default-ndit-Teff.json

Check for differences between the old and the new file. Check in the script Etc_form_class.py if the function update_etc_form is still following the right structure to write input data into the replaced json file and adjust the structure adequately. To test the structure of the input json file, navigate to CRIRES-planning-tool/python, open an iPython console and type the following:

```
from classes_methods import Etc_form_class
etc_form = Etc_form_class.etc_form('[jsonfile-type]')
```

and write the desired type of json input file at [jsonfile-type]: snr-Templ, edit-Templ, snr-Teff, ndit-Teff. Now you can investigate the structure of etc_form by writing [etc_form] + [.] + [Tab] and navigate through the file...

If none of these two strategies solve the problem, you need to contact Jakob Vinther.

Result files

Result files are available as follows:

Code documantation

Dependencies