ObservationTools Documentation

Release 0.1

IA

Nov 12, 2017

Contents:

A set of tools to plan astronomical observations.

CHAPTER 1

Installation

1.1 With git

If you have git installed (or if you want to install git and use it for the first time), then the tools can be installed with the following few commands in the terminal:

```
git clone https://github.com/iastro-pt/ObservationTools
cd ObservationTools
pip install -r requirements.txt  # You may need to use sudo here
```

1.2 Without git

If you do not have git installed, you can just download the entire directory here:

```
unzip ObservationTools-master.zip
cd ObservationTools-master
pip install -r requirements.txt  # You may need to use sudo here
```

1.3 Updates

If you want to update your tools and installed it with *git*, simply change the directory to this folder and do a *git pull*. If you don't used git, you have to do the installation again as described above.

CHAPTER 2

Visibility

The script `visibility.py` is used to plot the observability of objects to aid the planning of astronomical observations. It is inspired by STARALT and PyAstronomy's Transit Visibility tools.

2.1 Modes

Currently there are two user modes of visibility. `staralt` (default) and `starobs`. The usage of these is outlined in the following sections.

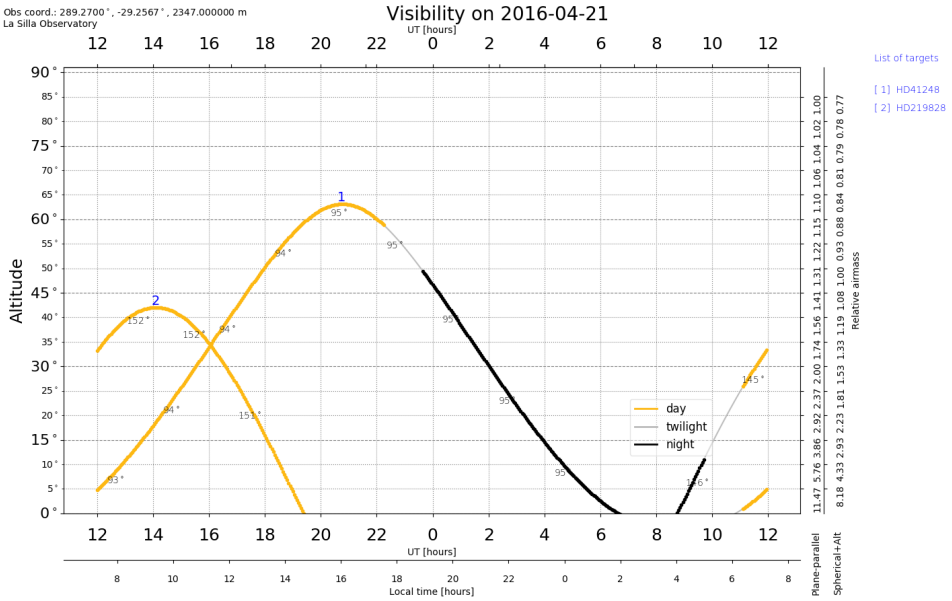
2.1.1 staralt

The `staralt` mode displays the altitude verse time of targets for a particular night.

For example:

```
python visibility.py HD41248,HD219828 -s esolasilla -d 2016-04-21
```

Results in the following image.



It is the default mode if no mode is specified. If the `-d`, or `--date` is not provided with the `YYYY-MM-DD` format is then it defaults to today/.

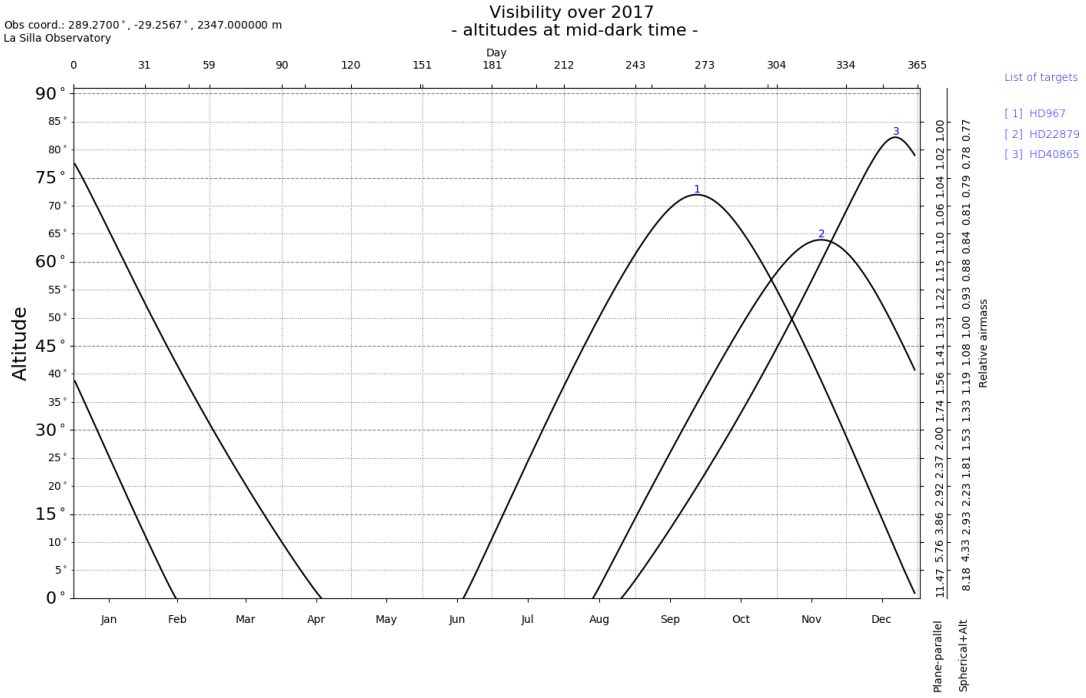
The observatory location can be specified using the `-s` or `--site` flag. The defualt observatory is ESO To find the list of available observatories and name codes run:

```
from __future__ import print_function, division
from PyAstronomy import pyasl
# List all available observatory data
pyasl.listObservatories()
```

2.1.2 starobs

The `starobs` mode shows how the altitude, at the *mid-dark time*, of each target changes over the course of the year. e.g.

```
python visibility.py HD967,HD22879,HD40865 -m starobs
```



Only the year *YYYY* is to be specified for the `--date` flag in this mode.

2.2 Other options

2.2.1 coordinates

The *-c* flag can be used to just return the coordinates of the targets in STARALT format then exit.

CHAPTER 3

Radial Velocity

The `rv.py` script allows you to perform radial velocity (rv) calculations and create radial velocity plots to aid in planning of radial velocity observations. The radial velocity calculations are handled by an *RV* class, while a *JulianDate* class is used to handle date/time conversions.

3.1 Parameter file

`rv.py` requires a parameter file to specify the orbital parameters of the system you wish to analysis. A template is provided in `data/template_params.txt` to help you get started. Comment lines starting with `#` and in-line comments are ignored.

For a basic rv calculations the standard rv parameters are required, `k1` [km/s], `omega` [deg], `eccentricity`, `tau` [days], `period` [days], as well as the `name` parameter.

If the mean system rv offset, `mean_val` (usually referred to as gamma), is not provided in the parameter file it is set to 0 km/s. The `ignore_mean` keyword in some functions can also be used to use a 0 km/s mean_val.

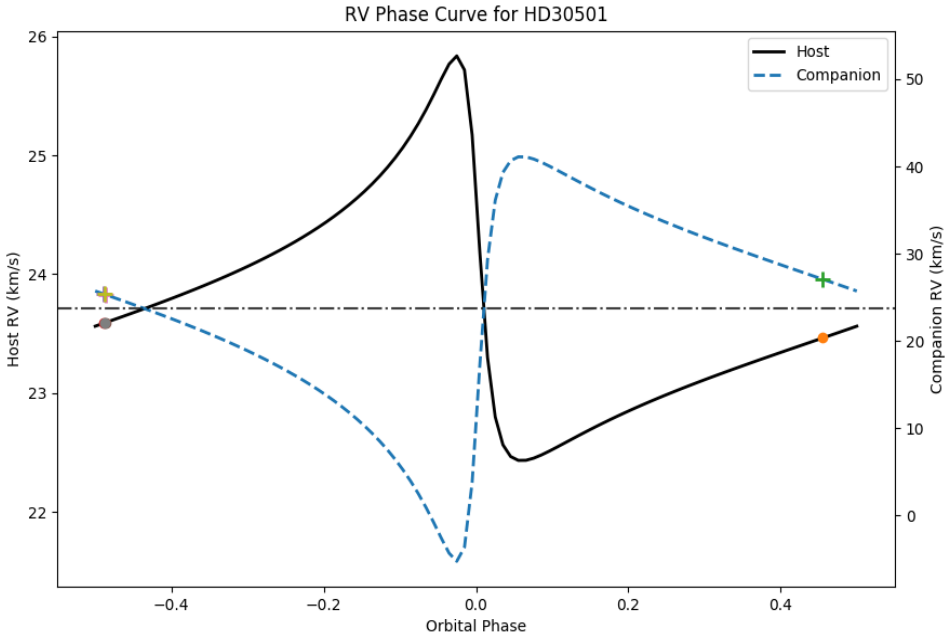
To include the rv of a companion the parameters `m_star` (star mass) and `msini` or `m_true` (companion mass) are required or `k2` the semi-major amplitude of the companion. If `k2` is not provided it is calculated from `k1` and the star and companion masses.

Note: A future version could maybe have the option to obtain parameters from planetary databases such as exoplanet.eu. Although this functionality would be limited to the stars/planets of the databases.

3.2 Usage examples

Simple usage cases:

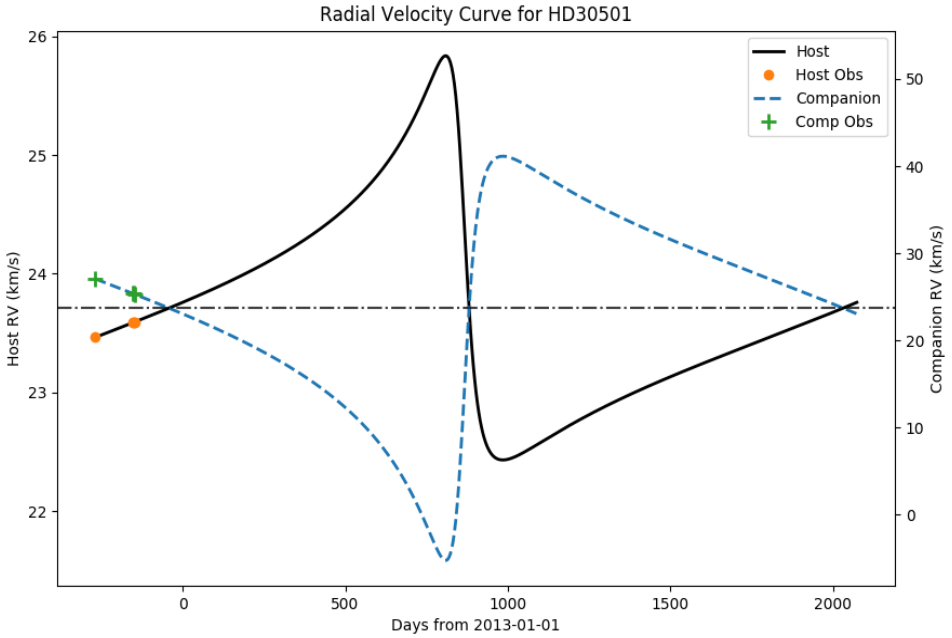
```
python rv.py data/HD30501_params.txt
```



displays the phase curve of HD30501 and its companion.

```
python rv.py data/HD30501_params.txt -l data/HD30501_obs.txt -m time -d 2013-01-01
```

Will create a temporal RV curve, marking the locations of the observations provided in the observation list file.



More usage examples can be found here: Modes ======= Different functionally can be accessed from the modes flag. `-m`, or `--mode`. The dmoefault mode is `phase`.

3.2.1 phase

Produces the RV phase curve with `cycle_fraction` and `phase_ceter` configurable. If the `k2` parameter is provided or the mass of the host (`m_host`) and companion (`msini` or `m_true`), then the RV for the companion is plotted on the second y-axis.

3.2.2 time

Produces a temporal RV curve of the system. The reference day is today but can be changed using the `-d`, `--date` option.

3.2.3 Marking observations

To mark the position of past and/or future observations you the dates can be provided with the `-o` (times passed to command line), or `-l` (file with list of observations) options. The observation times are marked/coloured as `past` or `future`, relative to the reference date.

The temporal space ranges from the maximum extent of the `cycle_fraction` after the reference date and any marked observation times.

3.2.4 debug

You can turn on debugging information using the `--debug` flag, e.g.:

```
python rv.py data/HD30501_params.txt -l data/hd30501_obs.txt --debug
```

3.2.5 RV class

class utils.rv_utils.**RV**(*semi_amp=0.0*, *period=0.0*, *ecc=0.0*, *tau=0.0*, *gamma=0.0*, *omega=0.0*, ***other_params*)

Bases: object

1. Omega should be given in degrees. This function converts it to radians.
2. The units of mean_val and k1, k2 should be the same e.g.both km / s
3. Tau should be the julian date, and the period given in days.

create_companion(*mass_ratio=None*)

Create companion RV object.

It has 3 ways to determine the amplitude of the companion in order of priority: 1: using a mass_ratio m1/m2 passed into the method. 2: If "k2" is already a parameter use that. 3: Calculate the mass ratio from m1 and m2. (In same units)

Switches k1 and k2 and m1 an m2 parameters. (m1 refers to self, while m2 the other body in orbit.)

mass_ratio: float m_1 / m_2

companion: RV Rv object for the companion.

classmethod from_dict(*params*)

classmethod from_file(*filename*)

Parameters in key = val text file.

ignore_mean

max_amp()

static mean_anomaly(*times*, *t0*, *period*)
Calculate mean anomaly using period, tau and a time value.

times: array-like Times to compute mean anomaly.

t0: float Time of periastron passage. (Julian days)

period: float Period of orbit.

ma: array-like Mean anomaly.

orbit_dict()

static radial_velocity(*gamma*, *k*, *ta*, *omega*, *ecc*)
Radial velocity equation.

gamma: float Mean RV motion of system.

k: float RV amplitude.

ta: array-like, float True anomaly.

omega: float Argument of periastron. (radians)

ecc: float Eccentricity of orbit.

RV: array-like, float Radial velocity values

RV = gamma + k *(np.cos(ta + omega) + ecc * np.cos(omega)).

rv_at_phase(*phase*)

rv_at_times(*t*)
Evaluate RV at the provided times.

rv_full_phase(*center=0*, *points=100*)
Return RV curve evaluated one full phase.

to_dict()

static true_anomaly(*ma*, *ecc*, *niterationmax=10000*)
Compute the true anomaly using the Newton-Raphson method.

ma: array-like Mean anomaly.

ecc: float Orbital eccentricity.

niterationmax: int Maximum number of iterations for N-R method.

ta: array-like True anomaly

Adapted from Rodrigo Diaz.

3.2.6 JulianDate

Used to convert from datetime objects and strings into julian dates and back again. This was created because epehm.julain_date() only converted to julian date.

class utils.rv_utils.**JulianDate**(*jd*, *reduced=False*)

Bases: object

Handle julian dates.

classmethod from_datetime(*dt*, *reduced=False*)

Convert from a datetime to a jd object.

Test against pyehem.julian_date()

dt: datetime object Datetime for date to calculate jd.

reduced: bool Return reduced JD, (JD-2400000)

jd: JulianDate JulianDate object.

Inspiration from https://stackoverflow.com/questions/13943062/

classmethod from_str(*time_str*, *format='%Y-%m-%d %H:%M:%S'*)

Return JulianDate from a time string.

time_str: str format: str

Format of time string.

dt: datetime object Datetime of julian date.

julian_epoch_dt = datetime.datetime(2000, 1, 1, 12, 0)

julian_epoch_jd = datetime.timedelta(2451545)

classmethod now()

reduce()

reduce_jd = 2400000

strformat = '%Y-%m-%d %H:%M:%S'

strformat2 = '%Y-%m-%d'

to_datetime()

Return JulianDate as a datetime.datetime object.

dt: datetime object Datetime of julian date.

Inspiration from https://stackoverflow.com/questions/13943062/

to_str(*format=None*)

Return date string from a JulianDate.

format: str String datetime format.

datesting: str String with date.

CHAPTER 4

More Examples

Here are some more examples of rv.py usages.

These use these observation dates.

```
# obstimes.txt
2012-02-13
2012-04-15
2013-07-21
2013-09-30
2016-01-15
```

4.1 phase

Simple usage cases:

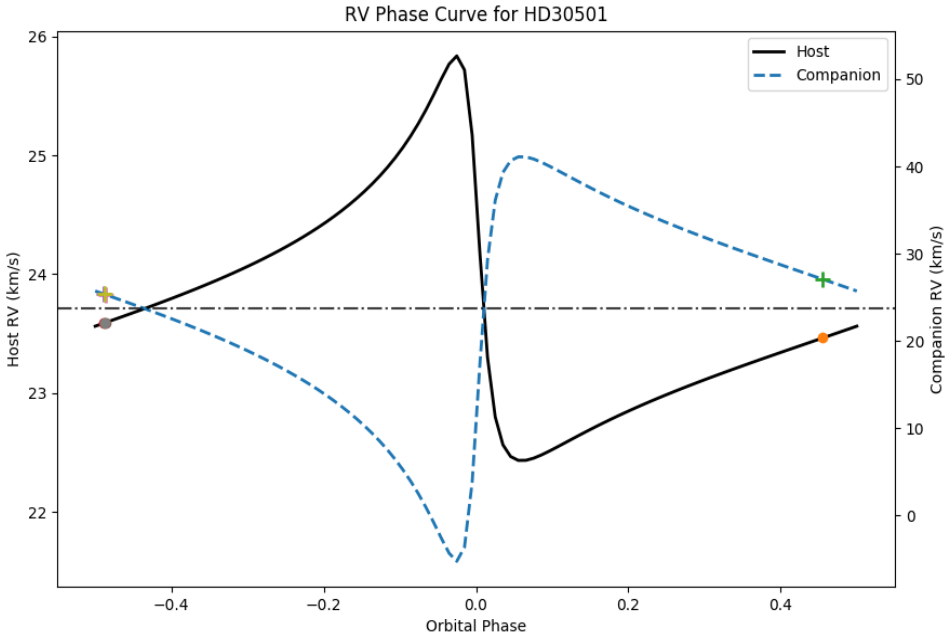
displays the phase curve of HD30501 and its companion.

4.2 time

More time mode examples

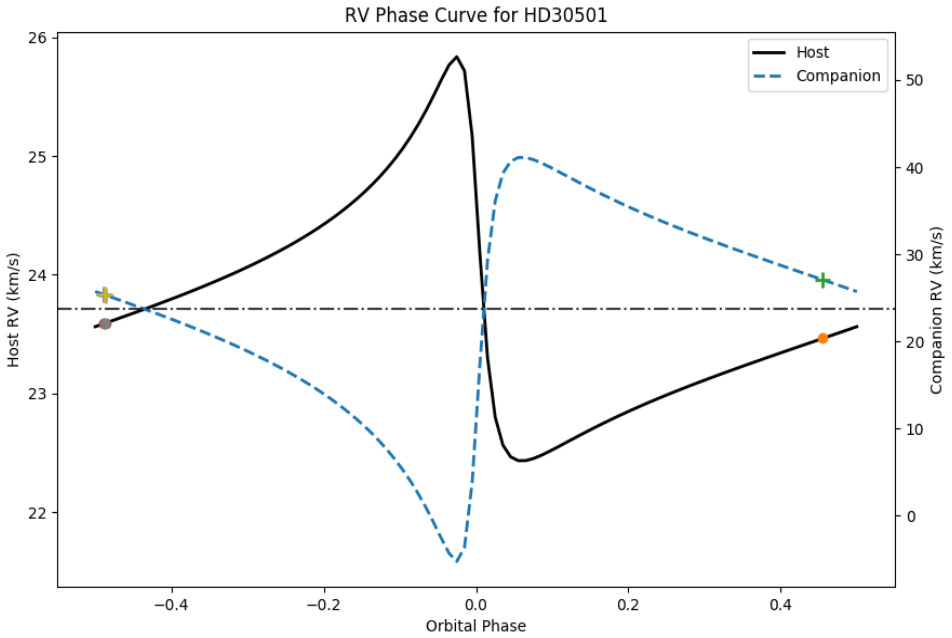
4.2.1 ::

python rv.py data/HD30501_params.txt



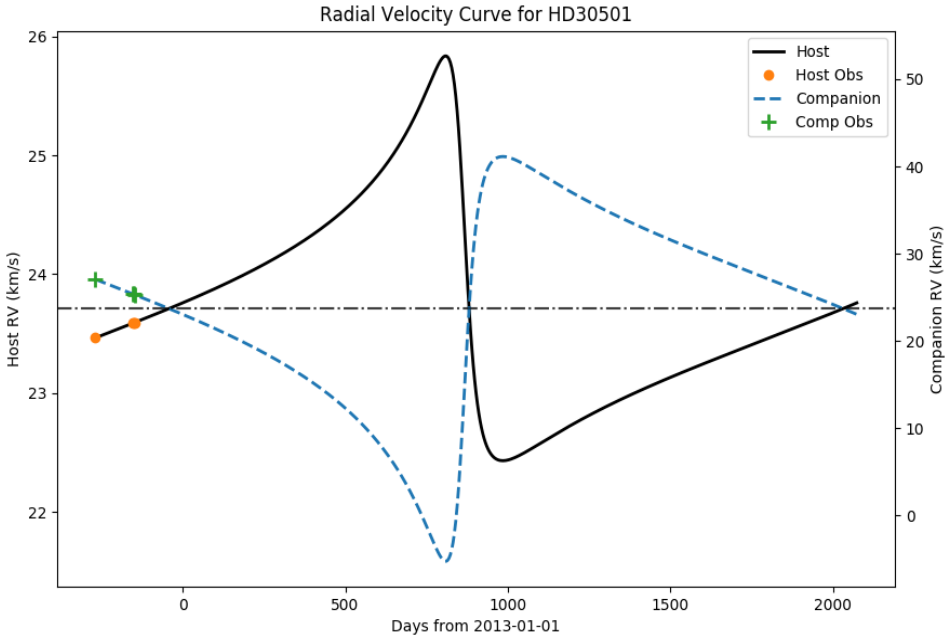
Show 1.5 phases.

```
python rv.py data/HD30501_params.txt -c 1.5
```



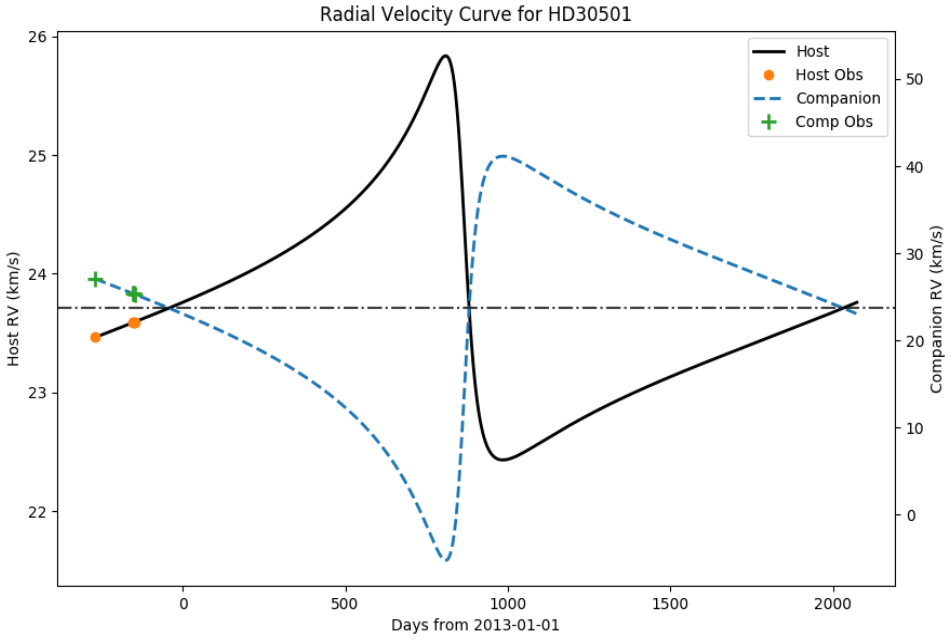
Center phase curve on 0.5

```
python rv.py data/HD30501_params.txt -p 0.5
```

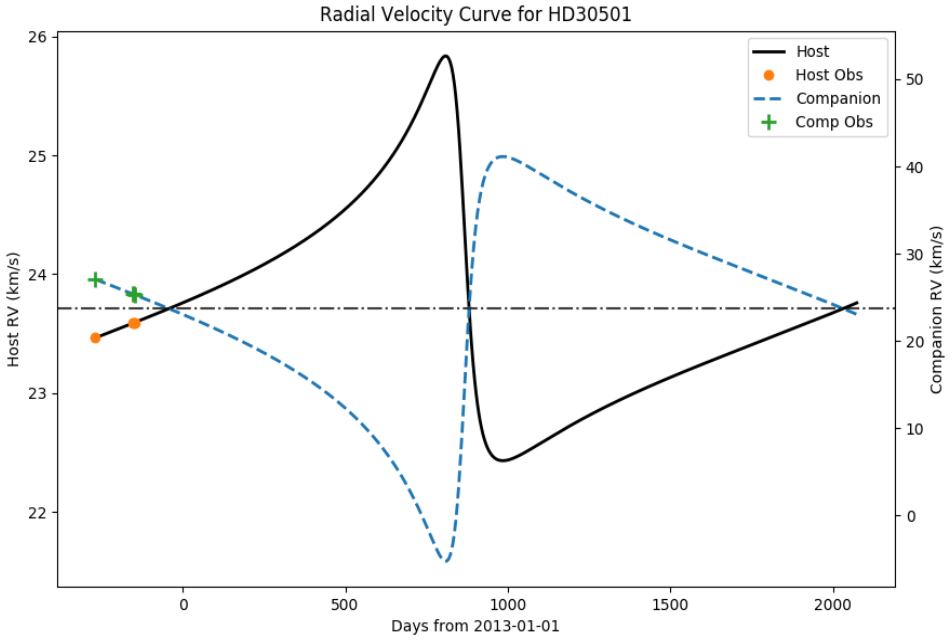


Using both `-l` and `-o`

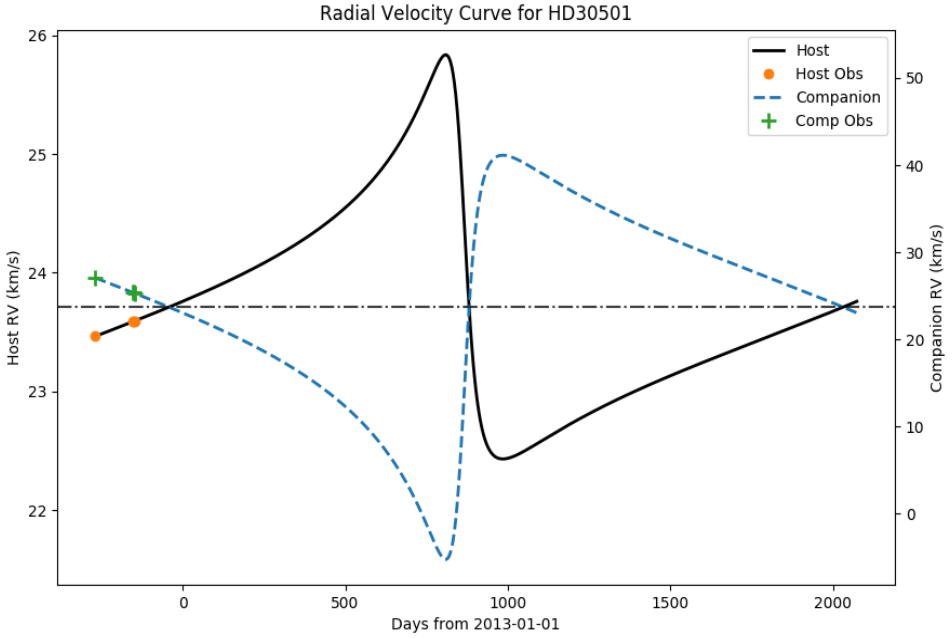
```
python rv.py data/HD30501_params.txt -l data/HD30501_obs.txt -m time -d 2013-01-01 -o␣
↪2014-03-12 2014-07-22
```



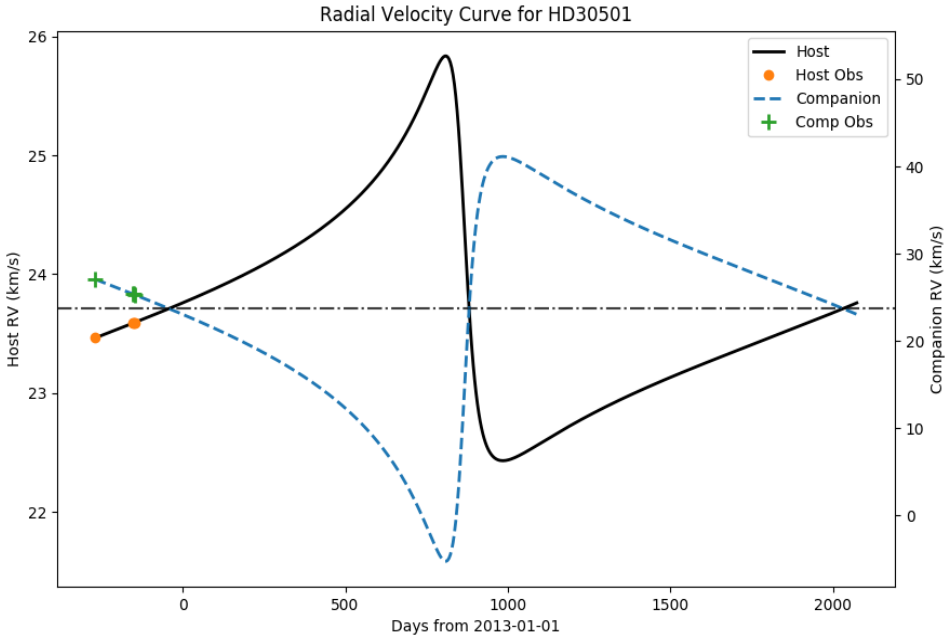

```
# All future points
python rv.py data/HD30501_params.txt -l obstimes.txt -m time -d 2012-01-01
```




```
python rv.py data/HD30501_params.txt -l obstimes.txt -m time -d 2013-08-01
```

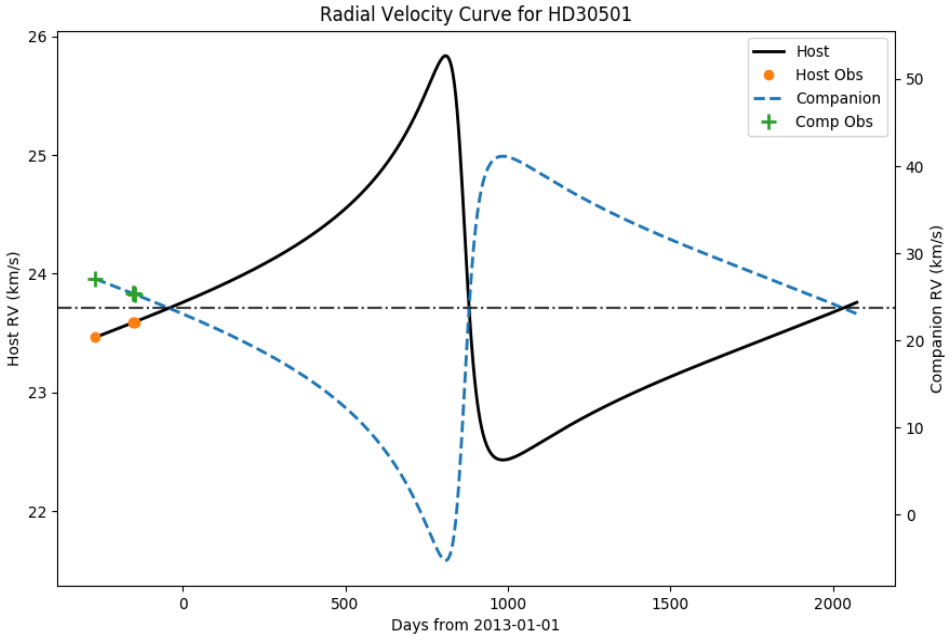



```
# All past observations
python rv.py data/HD30501_params.txt -l obstimes.txt -m time -d 2017-01-01
```



And with using the `cycle_ration`

```
# All past observations
python rv.py data/HD30501_params.txt -l obstimes.txt -m time -d 2017-01-01 -c 0.2
```



CHAPTER 5

Contributors

The list of contributors to this project so far are

- Daniel Andreasen
- João Faria
- Jason Neal

People are more than welcome to do pull requests, open issues, give suggestions, etc. on the github repo. Then your name could appear here also.

If you do not have a github user (or don't want to use github for some obscure reason), I can be contacted at daniel.andreasen@astro.up.pt.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Index