salter Documentation

Release 0.0.dev076

Brett Morris, Jim Davenport

Sep 06, 2018

Contents

This is the documentation for `salter`.

Part I

Source Code

. . . is available on GitHub.

Part II

Installation

Install via pip (via GitHub):

```
pip install https://github.com/bmorris3/salter/archive/master.zip
```

Part III

Dependencies

- numpy
- scipy
- matplotlib
- astropy
- astroquery
- batman-package
- kplr
- h5py

Part IV

Prep your local data caches

1. First open and run the notebook `choose_targets.ipynb`.
2. In the repo, run this to construct and cache the database of kepler light curves (runtime = ~1 hr on conference wifi):

```
python -c "from salter import cache_light_curves; cache_light_curves()"
```

3. Run this to cache a local copy of the joined tables from the NASA Exoplanet Archive and the Exoplanet Orbit Database:

```
python -c "from salter import cache_planets_table; cache_planets_table()"
```

Part V

salter Package

CHAPTER 1

Functions

cache_light_curves()	Run this after running choose_targets.ipynb in order to cache light curves into a local HDF5 archive.
cache_planets_table()	Cache a joined table containing data from the NASA Exoplanet Archive and the Exoplanet Orbit Database.
concatenate_light_curves(light_curve_list[, ...])	Combine multiple transit light curves into one TransitLightCurve object.
concatenate_transit_light_curves(...[, name])	Combine multiple transit light curves into one TransitLightCurve object.
get_planets_table()	Get the joined planets table from the NASA Exoplanet Archive and the Exoplanet Orbit Database.
kic_to_params(kic)	For a KIC number kic, return a TransitParams object for that star-planet system.
q2u(q1, q2)	Convert the two parameter quadratic terms of the Kipping 2013 limb- darkening parameterization q_1 and q_2 to the standard linear and quadratic terms of the quadratic limb-darkening parameterization of Claret et al.
quad(Teff, logg, filt)	Quadratic limb-darkening law.
subtract_add_divide(whole_lc, transits)	Normalize transit light curves in transits with the "subtract-add-divide" method.
test([package, test_path, args, plugins, ...])	Run the tests using py.test.
transit_model(kic, times)	Compute a transit model for KIC kic at times times
u2q(u1, u2[, warnings])	Convert the linear and quadratic terms of the quadratic limb-darkening parameterization – called u_1 and u_2 in Kipping 2013 or a and b in Claret et al.

1.1 cache_light_curves

salter.**cache_light_curves**()

Run this after running choose_targets.ipynb in order to cache light curves into a local HDF5 archive.

Examples

```
>>> from salter import cache_light_curves; cache_light_curves()
```

1.2 cache_planets_table

salter.**cache_planets_table**()

Cache a joined table containing data from the NASA Exoplanet Archive and the Exoplanet Orbit Database.

To get the table, run the `get_planets_table()` function.

1.3 concatenate_light_curves

salter.**concatenate_light_curves**(*light_curve_list*, *name=None*)

Combine multiple transit light curves into one `TransitLightCurve` object.

Parameters

light_curve_list : list

List of `TransitLightCurve` objects

name : str

Name of new light curve

Returns

tlc : `TransitLightCurve`

Concatenated transit light curves

1.4 concatenate_transit_light_curves

salter.**concatenate_transit_light_curves**(*light_curve_list*, *name=None*)

Combine multiple transit light curves into one `TransitLightCurve` object.

Parameters

light_curve_list : list

List of `TransitLightCurve` objects

name : str

Name of new light curve

Returns

tlc : `TransitLightCurve`

Concatenated transit light curves

1.5 get_planets_table

salter.**get_planets_table**()

Get the joined planets table from the NASA Exoplanet Archive and the Exoplanet Orbit Database.

Returns
table : Table

Table of exoplanet properties

1.6 kic_to_params

salter.**kic_to_params**(*kic*)
For a KIC number kic, return a TransitParams object for that star-planet system.

Parameters
kic : int

KIC number

Returns
params : TransitParams

Transit parameter object

Examples

```
>>> from salter import kic_to_params
>>> params = kic_to_params(9705459)
```

1.7 q2u

salter.**q2u**(*q1*, *q2*)
Convert the two parameter quadratic terms of the Kipping 2013 limb- darkening parameterization q_1 and q_2 to the standard linear and quadratic terms of the quadratic limb-darkening parameterization of Claret et al. 2013 – called u_1 and u_2 in Kipping 2013 or a and b in Claret et al. 2013:

http://adsabs.harvard.edu/abs/2013A%26A...552A..16C

Parameters
q1 : float

First component of Kipping 2013 quadratic limb-darkening

q2 : float

Second component of Kipping 2013 quadratic limb-darkening

Returns
(u1, u2) : tuple of floats

Claret et al. (2013) style quadratic limb-darkening parameters

1.8 quad

salter.**quad**(*Teff*, *logg*, *filt*)
Quadratic limb-darkening law. All grid points have metallicities Z=0 (solar), microturbulence xi=2.

Parameters

filt : string

Name of the filter used.

logg : float

Logarithm of the stellar surface gravity.

Teff : float

Host star effective temperature.

method : string, optional. Either 'F' or 'L'

Method of computation: least-square or flux conservation

Returns

a, b : float, float

The linear and quadratic limb-darkening terms for a quadratic law: I(mu)/I(1) = 1 - a*(1 - mu) - b*(1 - mu)**2

1.9 subtract_add_divide

salter.**subtract_add_divide**(*whole_lc*, *transits*)

Normalize transit light curves in `transits` with the "subtract-add-divide" method.

Parameters

whole_lc : `LightCurve` or subclass

Light curve over the whole kepler mission

transits : list of `LightCurve` or subclasses

Transit light curves

1.10 test

salter.**test**(*package=None*, *test_path=None*, *args=None*, *plugins=None*, *verbose=False*, *pastebin=None*, *remote_data=False*, *pep8=False*, *pdb=False*, *coverage=False*, *open_files=False*, ***kwargs*)

Run the tests using py.test. A proper set of arguments is constructed and passed to pytest.main.

Parameters

package : str, optional

The name of a specific package to test, e.g. 'io.fits' or 'utils'. If nothing is specified all default tests are run.

test_path : str, optional

Specify location to test by path. May be a single file or directory. Must be specified absolutely or relative to the calling directory.

args : str, optional

Additional arguments to be passed to pytest.main in the `args` keyword argument.

plugins : list, optional

Plugins to be passed to pytest.main in the `plugins` keyword argument.

verbose : bool, optional

Convenience option to turn on verbose output from py.test. Passing True is the same as specifying `'-v'` in `args`.

pastebin : {'failed','all',None}, optional

Convenience option for turning on py.test pastebin output. Set to `'failed'` to upload info for failed tests, or `'all'` to upload info for all tests.

remote_data : bool, optional

Controls whether to run tests marked with @remote_data. These tests use online data and are not run by default. Set to True to run these tests.

pep8 : bool, optional

Turn on PEP8 checking via the pytest-pep8 plugin and disable normal tests. Same as specifying `'--pep8 -k pep8'` in `args`.

pdb : bool, optional

Turn on PDB post-mortem analysis for failing tests. Same as specifying `'--pdb'` in `args`.

coverage : bool, optional

Generate a test coverage report. The result will be placed in the directory htmlcov.

open_files : bool, optional

Fail when any tests leave files open. Off by default, because this adds extra run time to the test suite. Requires the psutil package.

parallel : int, optional

When provided, run the tests in parallel on the specified number of CPUs. If parallel is negative, it will use the all the cores on the machine. Requires the pytest-xdist plugin installed. Only available when using Astropy 0.3 or later.

kwargs

Any additional keywords passed into this function will be passed on to the astropy test runner. This allows use of test-related functionality implemented in later versions of astropy without explicitly updating the package template.

1.11 transit_model

salter.**transit_model**(*kic*, *times*)

Compute a transit model for KIC `kic` at times `times`

Parameters

kic : int

KIC number

times : `Time`

Times to compute model

Returns

flux : `ndarray`

Model fluxes at `tiems`

1.12 u2q

salter.**u2q**(*u1*, *u2*, *warnings=True*)

Convert the linear and quadratic terms of the quadratic limb-darkening parameterization – called u_1 and u_2 in Kipping 2013 or a and b in Claret et al. 2013 – and convert them to q_1 and q_2 as described in Kipping 2013:

http://adsabs.harvard.edu/abs/2013MNRAS.435.2152K

Parameters

u1 : float

Linear component of quadratic limb-darkening

u2 : float

Quadratic component of quadratic limb-darkening

Returns

(q1, q2) : tuple of floats

Kipping (2013) style quadratic limb-darkening parameters

CHAPTER 2

Classes

LightCurve([times, fluxes, errors, ...])	Container object for light curves.
Residuals([residuals, buffer_duration, ...])	Transit light curve residuals.
TransitLightCurve([times, fluxes, errors, ...])	Container for a single transit light curve.

2.1 LightCurve

class salter.**LightCurve**(*times=None*, *fluxes=None*, *errors=None*, *quarters=None*, *name=None*, *params=None*)

Bases: object

Container object for light curves.

Parameters

times : ndarray

Times in JD

fluxes : ndarray

Fluxes (normalized or not)

errors : ndarray

Uncertainties on the fluxes

quarters : ndarray (optional)

Kepler Quarter for each flux

name : str

Name this light curve (optional)

params : TransitParams

Planet transit parameters

Attributes Summary

times_jd	Get the times in this light curve in JD.

Methods Summary

delete_outliers()	
from_hdf5(kic)	Load a light curve from the HDF5 light curve archive with KIC number kic.
get_available_quarters()	Get which quarters are available in this LightCurve
get_quarter(quarter)	Get a copy of the data from within LightCurve during one Kepler quarter.
get_transit_light_curves([plots])	For a light curve with transits only (i.e.
mask_in_transit([oot_duration_fraction])	Mask out the in-transit light curve based on transit parameters
mask_out_of_transit([oot_duration_fraction, ...])	Mask out the out-of-transit light curve based on transit parameters
normalize_each_quarter([rename, ...])	Use polynomial fit to each quarter to normalize the data.
phases()	
plot([transit_params, ax, quarter, show, ...])	Plot light curve.
split_at_index(index)	Split the light curve into two light curves, at index
transit_model([short_cadence])	

Attributes Documentation

times_jd
Get the times in this light curve in JD.

Returns
t_jd : ndarray

Julian dates.

Methods Documentation

delete_outliers()

classmethod from_hdf5(*kic*)
Load a light curve from the HDF5 light curve archive with KIC number kic.

Parameters
kic: float

KIC number

get_available_quarters()
Get which quarters are available in this LightCurve

Returns
qs : list

List of unique quarters available.

get_quarter(*quarter*)
Get a copy of the data from within `LightCurve` during one Kepler quarter.

Parameters
quarter : int

Kepler Quarter

Returns
lc : `LightCurve`

Light curve from one Kepler Quarter

get_transit_light_curves(*plots=False*)
For a light curve with transits only (i.e. like one returned by `LightCurve.mask_out_of_transit`), split up the transits into their own light curves, return a list of `TransitLightCurve` objects.

Parameters
plots : bool

Make diagnostic plots.

Returns
transit_light_curves : list

List of `TransitLightCurve` objects

mask_in_transit(*oot_duration_fraction=0.25*)
Mask out the in-transit light curve based on transit parameters

Parameters
oot_duration_fraction : float (optional)

Fluxes from what fraction of a transit duration of the out-of-transit light curve should be included in the mask?

Returns
d : dict

Inputs for a new `LightCurve` object with the mask applied.

mask_out_of_transit(*oot_duration_fraction=0.25*, *flip=False*)
Mask out the out-of-transit light curve based on transit parameters

Parameters
oot_duration_fraction : float (optional)

Fluxes from what fraction of a transit duration of the out-of-transit light curve should be included in the mask?

flip : bool (optional)

If `True`, mask in-transit rather than out-of-transit.

Returns
d : dict

Inputs for a new `LightCurve` object with the mask applied.

normalize_each_quarter(*rename=None*, *polynomial_order=2*, *plots=False*)
Use polynomial fit to each quarter to normalize the data.

Parameters
rename : str (optional)

New name of the light curve after normalization

polynomial_order : int (optional)

Order of polynomial to fit to the out-of-transit fluxes. Default is 2.

plots : bool (optional)

Show diagnostic plots after normalization.

phases()

plot(*transit_params=None*, *ax=None*, *quarter=None*, *show=True*, *phase=False*, *plot_kwargs={u'color': u'b', u'lw': 0, u'marker': u'o'}*)

Plot light curve.

Parameters

transit_params : TransitParams (optional)

Transit light curve parameters. Required if phase is True.

ax : Axes (optional)

Axis to make plot on top of

quarter : float (optional)

Plot only this Kepler quarter

show : bool

If True, call matplotlib.pyplot.show after plot is made

phase : bool

If True, map times in JD to orbital phases, which requires that transit_params be input also.

plot_kwargs : dict

Keyword arguments to pass to matplotlib calls.

split_at_index(*index*)

Split the light curve into two light curves, at index

transit_model(*short_cadence=False*)

2.2 Residuals

class salter.**Residuals**(*residuals=None*, *buffer_duration=0.15*, *params=None*, *phases=None*)

Bases: object

Transit light curve residuals.

Methods Summary

anderson(attrs1, attrs2)	k-sample Anderson test from anderson_ksamp.
from_rms(times, residuals, star, planet[, ...])	Load residuals from an rms simulation.

Continued on next page

Table 4 – continued from previous page

from_transits(transits, params[, ...])	Load transit residuals from a list of TransitLightCurve objects.
ks(attrs1, attrs2)	Two-sample KS test from ks_2samp
plot()	Generate a quick plot of the transit residuals.
ttest(attrs1, attrs2)	Independent two-sample T test from ttest_ind.

Methods Documentation

anderson(*attrs1*, *attrs2*)
k-sample Anderson test from anderson_ksamp.

Parameters
attrs1 : list of attributes

List of conditions in first sample

attrs2 : list of attributes

List of conditions in second sample

Returns
sig : float

significance level (see anderson_ksamp)

Examples

```
>>> import numpy as np
>>> import batman
>>> from salter import LightCurve
>>> # Create example transiting planet properties
>>> params = batman.TransitParams()
>>> params.t0 = 0.5
>>> params.rp = 0.1
>>> params.per = 1
>>> params.duration = 0.3
>>> params.inc = 90
>>> params.w = 90
>>> params.ecc = 0
>>> params.a = 10
>>> params.limb_dark = 'quadratic'
>>> params.u = [0.2, 0.1]
>>> # Create example transit light curves:
>>> transits = [LightCurve(times=i + np.linspace(0, 1, 500),
>>>                        fluxes=np.random.randn(500),
>>>                        params=params) for i in range(10)]
>>> r = Residuals(transits, params)
>>> # How significant is the difference between the distributions of the fluxes in and out-
↪of-transit?
>>> r.anderson('out_of_transit', 'in_transit')
1.1428634099527666
>>> # How significant is the difference between the distributions of the in-transit fluxes␣
↪before and after midtransit?
>>> r.anderson(['in_transit', 'before_midtransit'], ['in_transit', 'after_midtransit'])
0.2792395871784852
```

classmethod from_rms(*times*, *residuals*, *star*, *planet*, *buffer_duration=0.15*)
Load residuals from an rms simulation.

Parameters
times : ndarray

Times of each flux

residuals : ndarray

Flux residual measurements

star : Star

Stellar properties

planet : Planet

Transiting planet parameters

buffer_duration : float

fraction of transit duration to ignore centered on ingress and egress.

classmethod from_transits(*transits*, *params*, *buffer_duration=0.15*)
Load transit residuals from a list of TransitLightCurve objects.

Parameters
transits : list of (or single) TransitLightCurve objects

list of transits

params : TransitParams()

transiting planet parameters

buffer_duration : float

fraction of transit duration to ignore centered on ingress and egress.

ks(*attrs1*, *attrs2*)
Two-sample KS test from ks_2samp

Parameters
attrs1 : list of attributes

List of conditions in first sample

attrs2 : list of attributes

List of conditions in second sample

Returns
pvalue : float

p value.

Examples

```
>>> import numpy as np
>>> import batman
>>> from salter import LightCurve
>>> # Create example transiting planet properties
>>> params = batman.TransitParams()
```

(continues on next page)

(continued from previous page)

```
>>> params.t0 = 0.5
>>> params.rp = 0.1
>>> params.per = 1
>>> params.duration = 0.3
>>> params.inc = 90
>>> params.w = 90
>>> params.ecc = 0
>>> params.a = 10
>>> params.limb_dark = 'quadratic'
>>> params.u = [0.2, 0.1]
>>> # Create example transit light curves:
>>> transits = [LightCurve(times=i + np.linspace(0, 1, 500),
>>>                        fluxes=np.random.randn(500),
>>>                        params=params) for i in range(10)]
>>> r = Residuals(transits, params)
>>> # How significant is the difference between the distributions of the fluxes in and out-
↪of-transit?
>>> r.ks('out_of_transit', 'in_transit')
0.91710727901331124
>>> # How significant is the difference between the distributions of the in-transit fluxes␣
↪before and after midtransit?
>>> r.ks(['in_transit', 'before_midtransit'], ['in_transit', 'after_midtransit'])
0.39171715554793468
```

plot()

Generate a quick plot of the transit residuals.

Returns

fig : Figure

Figure object

ax : Axes

axis object

ttest(*attrs1*, *attrs2*)

Independent two-sample T test from ttest_ind.

Parameters

attrs1 : list of attributes

List of conditions in first sample

attrs2 : list of attributes

List of conditions in second sample

Returns

pvalue : float

p value.

Examples

```
>>> import numpy as np
>>> import batman
>>> from salter import LightCurve
```

(continues on next page)

(continued from previous page)

```
>>> # Create example transiting planet properties
>>> params = batman.TransitParams()
>>> params.t0 = 0.5
>>> params.rp = 0.1
>>> params.per = 1
>>> params.duration = 0.3
>>> params.inc = 90
>>> params.w = 90
>>> params.ecc = 0
>>> params.a = 10
>>> params.limb_dark = 'quadratic'
>>> params.u = [0.2, 0.1]
>>> # Create example transit light curves:
>>> transits = [LightCurve(times=i + np.linspace(0, 1, 500),
>>>                        fluxes=np.random.randn(500),
>>>                        params=params) for i in range(10)]
>>> # Create residuals object
>>> r = Residuals(transits, params)
>>> # How significant is the difference between the means of the fluxes in and out-of-
↪transit?
>>> r.ttest('out_of_transit', 'in_transit')
0.310504218041
>>> # How significant is the difference between the means of the in-transit fluxes before␣
↪and after midtransit?
>>> r.ttest(['in_transit', 'before_midtransit'], ['in_transit', 'after_midtransit'])
0.823997471194
```

2.3 TransitLightCurve

class salter.**TransitLightCurve**(*times=None*, *fluxes=None*, *errors=None*, *quarters=None*, *name=None*, *params=None*)

Bases: salter.LightCurve

Container for a single transit light curve. Subclass of LightCurve.

Parameters

times : ndarray

Times in JD

fluxes : ndarray

Fluxes (normalized or not)

errors : ndarray

Uncertainties on the fluxes

quarters : ndarray (optional)

Kepler Quarter for each flux

name : str

Name this light curve (optional)

Methods Summary

chi2_lc_3param(p)	Compute chi^2 for a three-parameter light curve model with parameters in tuple p consisting of the planet-star radius ratio, and two limb-darkening parameters (Kipping 2013)
fit_lc_3param()	Fit three-parameter light curve model, and replace the transit parameters in self.params with the best fit planet-star radius ratio, and two limb-darkening parameters (Kipping 2013).
fit_linear_baseline([cadence, ...])	Find OOT portions of transit light curve using similar method to LightCurve.mask_out_of_transit, fit linear baseline to OOT.
fit_polynomial_baseline([order, cadence, ...])	Find OOT portions of transit light curve using similar method to LightCurve.mask_out_of_transit, fit linear baseline to OOT
remove_linear_baseline([plots, cadence])	Find OOT portions of transit light curve using similar method to LightCurve.mask_out_of_transit, fit linear baseline to OOT, divide whole light curve by that fit.
remove_polynomial_baseline([order, plots, ...])	Find OOT portions of transit light curve using similar method to LightCurve.mask_out_of_transit, fit linear baseline to OOT, divide whole light curve by that fit.
scale_by_baseline(linear_baseline_params)	
subtract_add_divide_without_outliers(..., ...])	
subtract_polynomial_baseline([plots, order, ...])	Find OOT portions of transit light curve using similar method to LightCurve.mask_out_of_transit, fit polynomial baseline to OOT, subtract whole light curve by that fit.

Methods Documentation

chi2_lc_3param(*p*)
Compute chi^2 for a three-parameter light curve model with parameters in tuple p consisting of the planet-star radius ratio, and two limb-darkening parameters (Kipping 2013)

fit_lc_3param()
Fit three-parameter light curve model, and replace the transit parameters in self.params with the best fit planet-star radius ratio, and two limb-darkening parameters (Kipping 2013).

fit_linear_baseline(*cadence=<Quantity 30. min>*, *return_near_transit=False*, *plots=False*)
Find OOT portions of transit light curve using similar method to LightCurve.mask_out_of_transit, fit linear baseline to OOT.

Parameters
cadence : Quantity (optional)
Length of the exposure time for each flux. Default is 1 min.

return_near_transit : bool (optional)
Return the mask for times in-transit.

Returns

linear_baseline : numpy.ndarray

Baseline trend of out-of-transit fluxes

near_transit : numpy.ndarray (optional)

The mask for times in-transit.

fit_polynomial_baseline(*order=2*, *cadence=<Quantity 30. min>*, *plots=False*, *mask=None*)

Find OOT portions of transit light curve using similar method to LightCurve.mask_out_of_transit, fit linear baseline to OOT

remove_linear_baseline(*plots=False*, *cadence=<Quantity 30. min>*)

Find OOT portions of transit light curve using similar method to LightCurve.mask_out_of_transit, fit linear baseline to OOT, divide whole light curve by that fit.

Parameters

cadence : Quantity (optional)

Length of the exposure time for each flux. Default is 1 min.

plots : bool (optional)

Show diagnostic plots.

remove_polynomial_baseline(*order=2*, *plots=False*, *cadence=<Quantity 30. min>*)

Find OOT portions of transit light curve using similar method to LightCurve.mask_out_of_transit, fit linear baseline to OOT, divide whole light curve by that fit.

Parameters

cadence : Quantity (optional)

Length of the exposure time for each flux. Default is 1 min.

plots : bool (optional)

Show diagnostic plots.

scale_by_baseline(*linear_baseline_params*)

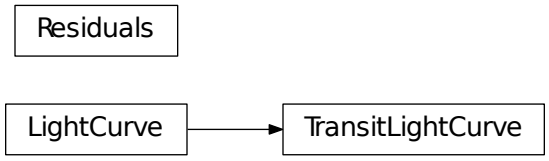
subtract_add_divide_without_outliers(*quarterly_max*, *order=2*, *cadence=<Quantity 30. min>*, *outlier_error_multiplier=50*, *outlier_tolerance_depth_factor=0.2*, *plots=False*)

subtract_polynomial_baseline(*plots=False*, *order=2*, *cadence=<Quantity 30. min>*)

Find OOT portions of transit light curve using similar method to LightCurve.mask_out_of_transit, fit polynomial baseline to OOT, subtract whole light curve by that fit.

CHAPTER 3

Class Inheritance Diagram



Python Module Index

S

Index

U