TACO documentations: pipeline.py

Final results can be found in the following files: filtered.csv, data.csv, pds.csv, oversampled_pds.csv, pds_bgr.csv, peaks_mle.csv. A brief description of each file is given in this documentation, and a longer description of each column of data.csv can be found in TACO_outputs.pdf

1 Filter light-curves (filter.R)

Filter the lightcurves with a high-pass filter (e.g. 40 days) in the form of a triangular smooth of the provided width. For that, it uses two rectangular smooths with half the provided width. Additionally it interpolates the single-point missing values by taking the mean of the two adjacent points. It also calculates the mean, variance and filling factor of the lightcurve and saves them in the data summary file. I assume the lightcurve is as provided by the APOKASC group which is an ascii table with three columns representing the time (in days), the observed ppm and the error (which is ignored from now on).

Input: lc, data, width, remove_gaps

lc:	ASCII table with three columns. First column must be time in days, second column must be the observed flux (ppm) and third column (optional) are the flux uncertainties (ppm).
data:	DataFrame as a summary file to save the results. **Columns:** KIC, raw_data, git-rev-hash.
width	Width of the high-pass triangular filter. **Default:** 40.
remove_gaps:	Gap size threshold in days. Gaps larger than this value are removed from the lightcurve. **Default:** -1 (do not remove gaps).

Output: filtered.csv, data

filtered.csv:	File containing the filtered lightcurve. **Columns:** time (days), time_raw (days), flux (ppm).
data:	DataFrame as a summary file. **New columns:** mean, var, start_date (days), end_date (days), fill_factor.

2 Lomb-Scargle periodogram (pds.py)

Calculate the Lomb-Scargle periodogram of a time series normalized using the spectral window function as described by Kallinger et al. 2014. The input lightcurve is assumed to have time stamps given in days, the flux in ppm and the periodogram is given in units of micro-Hertz. The functionality for computing the periodogram comes from the lightkurve package.

Input: ts, oversample_factor

ts:	Time series data in the shape of a DataFrame. **Default:** filtered.csv
oversample_factor:	int value for the oversample factor. **Default:** 1.

Output: pds

pds:	DataFrame containing the Periodogram of the input ts. **Columns:** frequency (μHz), power.
pds.csv:	CSV file of pds.

3 Numax estimation (numax_estimate.R)

Make three independent possible estimates of the frequency of maximum oscillation power (ν_{max} [μHz]) and compares them to get a final guess. The estimates are based on the following:

- **Variance of the time series:** It was shown by Hekker et al. 2012 that the variance of the time series has an exponential relation with nu_max.
- **Peak detections with a tree-map of the mexican-hat continuous wavelet transform:** Using method by Du et al. 2006 to detect significant peaks in a data set.
- **Morlet wavelet transfroms:** The maximum of a Morlet wavelet transform usually occurs near numax.

Updates the summary file with new columns: numax_var, numax_CWTMexHat, numax_Morlet and numax0 which are the 3 different nu_max estimations and the final guess respectively. It also adds a column numax0_flag. If true, the three estimates are different.

Input: pds, data, filter_width

pds:	Power density spectrum from the ts obtained with pds.py.
data:	DataFrame as a summary file. *Important cols for this function: "var" and "nuNyq".*
filter_width:	The width of the log-median filter used to remove the background for the wavelet ν_{max} estimation. (This is a first approximation of the background. The PDS used to estimate ν_{max} is then pds/bkg)

Output: data

data:	DataFrame as a summary file. **New columns:** numax0_flag, numax_var, numax_CWTMexHat, numax_Morlet, numax0.

4 Background fitting (background_fit.py)

Use an MCMC algorithm (emcee) to make a background estimation for the power-spectrum density normalized according with the above prescription. Note that this script uses some initial guesses that critically depend on the normalization of the power-spectrum density so, if you use a different normalization, this script might need some changes before it's able to work. You need to have installed the emcee python package (version 3 or greater).

The background model is similar to the one proposed by Kallinger et al. 2015 albeit with some differences. We use by default a unbinned version of the PDS to do the fit but this can be changed with the

command-line arguments.

Input: pds, oversampled_pds, data, bins, maxsteps, minsteps, nwalkers, nwarmup

pds:	Power density spectrum.
oversampled_pds:	PDS obtained in step 3 of pipeline.py
data:	DataFrame as a summary file.
bins:	The PDS will be binned by this number of bins for the MCMC. Setting it to 1 will, not bin it. **Default:** 300.
maxsteps:	Maximum number of steps for the whole MCMC run. **Default:** 5000.
minsteps:	Minimum number of steps for the MCMC estimation. **Default:** 2000.
nwalkers:	Number of walkers (chains) for the MCMC fit. **Default:** 50.
nwarmup:	Number of steps for the MCMC warmup. **Default:** 1000.

Output: pds_bgr, oversampled_pds_bgr, data

pds_bgr:	DataFrame Periodogram, background corrected. **Columns:** frequency (μHz), power.
oversampled_pds_bgr:	DataFrame Oversampled periodogram, background corrected. **Columns:** frequency (μHz), power. This data is saved as pds_bgr.csv
data:	DataFrame as a summary file. **New columns:** Hmax, Bmax, HBR, Pn, A1, b1, A2, b2, A3, b3, Pg, numax, sigmaEnv, lnprob.

By default TACO does NOT save out the chains/posterior distributions from the background fit (since the file is 50-100 MB per star), only the summary statistics. If you explicitly want to save the MCMC chains then please add the –save_posteriors flag when running the background fit.

5 Peak-detections (peakFind.R)

Takes the background-removed power-spectrum density and identifies the relevant oscillations. The oscillations are all modelled as lorentzians. This script estimates their position, width and height which can be later used to construct a prior in a more precise analysis. You need to have installed the R packages: wmtsa, argparser and dplyr. The auxiliary file peakFind_lib.R must be present in the same directory as peakFind.R.

Input: pds_bgr, oversampled_pds_bgr, data, peaks, snr, prob, maxlwd, removel02, minAIC, navg.

pds_bgr:	Background substracted power density spectrum.
oversampled_pds:	Background substracted oversampled power density spectrum.
data:	DataFrame as a summary file.
peaks:	File with identified peaks. It must contain the $l = 0, 2$ modes already identified. **Columns:** frequency (μHz), linewidth, height, snr, AIC, amplitude **Default:** None.
snr:	Minimum signal to noise ratio (on CWT space) for resolved peaks. **Default:** 1.2
prob:	Minimum (frequentist) probability threshold for unresolved peaks. **Default:** 0.0001.
maxlwd:	Maximum search linewidth for resolved peaks in the CWT search. **Default:** None.
removel02:	Whether or no the l02 peaks should be divided out before running the CWT search. **Default:** False
minAIC:	Minimum AIC value for a peak to have in orfer to be considered significant. **Default:** 2.
navg:	Number of power spectra averaged to create current power spectrum. **Default:** 1.

Output: peaks

peaks:	DataFrame of the identified peaks. It must contain the $l = 0, 2$ modes already identified. **Columns:** frequency, linewidth, amplitude.

6 Peak optimisations using MLE (peaksMLE.R)

MLE optimization for the peaks found by peakFind.R (or another way) It takes a csv file with columns named frequency, height and linewidth and makes an MLE optimization on a PDS. If linewidth is NA, the modes are assumed to be sinc^2 functions, otherwise they are assumed as Lorentzians.

Input: pds_bgr, peak, mixed_peaks, data, maxlwd, removel02, minAIC, navg, finalfit

pds_bgr:	Background substracted power density spectrum.
peaks:	DataFrame of the identified peaks. It must contain the $l = 0, 2$ modes already identified with peakFind.R
mixed_peaks:	DataFrame containing the mixed mode peaks from peak finding. **Default:** None.
data:	DataFrame as a summary file.
maxlwd:	Maximum search linewidth for resolved peaks in the CWT search. **Default:** None.
removel02:	Whether or no the l02 peaks should be divided out before running the CWT search. **Default:** False.
minAIC:	Minimum AIC value for a peak to have in orfer to be considered significant. **Default:** 2.
navg:	Numver of power spectra averaged to create current power spectrum. **Default:** 1.
finalfit:	Whether or not this is the final MLE optimisation. **Default:** False.

Output: peaks_mle, data

peaks_mle:	DataFrame of the identified peaks with peaksMLE.R
data:	DataFrame as a summary file. **New columns:** npeaks.

7 Spherical degree identification (peakBagModeId02.R)

Takes the oscillations found and tags them according to their spherical degree. It also calculate the large frequency separation $\Delta\nu$.

Input: pds_bgr, peaks_mle, data

pds_bgr:	Background substracted power density spectrum.
peaks_mle:	DataFrame of the identified peaks with peaksMLE.R
data:	DataFrame as a summary file.

Output: peaks_mle, data

peaks_mle:	DataFrame of the identified peaks and mixed peaks.
data:	DataFrame as a summary file. **New columns:** Deltanu, DeltaNu_sd, dNu02, eps_p, eps_sd, alpha, alpha_sd, Cenrtal_DeltaNu, Central_DeltaNu_sd, Central_eps_p, Central_eps_p_sd, Central_alpha, Central_alpha_sd, gamma0, modelDFlag.

8 Period spacing and coupling determination (peakBagPeriodSpacing.py)

Computes the period spacing and coupling using the power spectrum of the stretched power spectrum from Vrard et al. (2016). This will also require the sloscillations python package for the fast mixed mode calculations.

Input: pds_bgr, peaks_mle, mixed_peaks, data, maxiters, niters, dpi_only, ncores

pds_bgr:	Background substracted power density spectrum.
peaks_mle:	DataFrame of the identified peaks with peaksMLE.R
mixed_peaks:	DataFrame containing the mixed mode peaks from peak finding. **Default:** None.
data:	DataFrame as a summary file.
maxiters:	Maximum number of iterations to use in the self-consistent calculation of $\Delta\Pi_1$. **Default:** 10.
niters:	Number of iterations to repeat the calculation of $\Delta\Pi_1$, q and ϵ_g. **Default:** 5.
dpi_only:	Only infer the period spacing and don't calculate τ or q. **Default:** False.
ncores:	Number of cores to use for a parallel calculation. **Default:** 1.

Output: pds_bgr, mixed_peaks, data

pds_bgr:	Background substracted power density spectrum.
mixed_peaks:	DataFrame of the identified peaks and mixed peaks.
data:	DataFrame as a summary file. **New columns:** visibility_ratio, DeltaPi1 coupling, eps_g, DeltaPi1_sig.