List and description of the existing Python scripts

by Dr. Lothar Schanne April

1, 2024

This collection of Python scripts was created in the last few years, in which I repeatedly wanted to evaluate - sometimes extensive - astronomical spectrum series from a celestial object in the optical wavelength range, but repeatedly realized that
the usual spectra processing programs cannot process time series. Therefore, the focus of these self-written Python scripts is clearly on the evaluation of time series of 1d spectra, mostly in ts format. All evaluation methods that I have encountered in my practice are available. Starting with normalization to the continuum, heliocentric correction, determination of equivalent widths of individual lines, determination of radial velocities by cross-correlation or from the displacement of individual lines, calculation of theoretical spectra and their adaptation to the own spectrograph, graphical representation of spectra, etc.
As I said, not only for individual spectra but for a large number of spectra (time series of an object). There are many comments within the scripts,
so that even those with rudimentary knowledge of Python can understand how a script works or adapt the scripts to their own needs if necessary.
(e.g. the entry of the geographic coordinates of your own observatory) is easily accomplished.

I have explained how to install a Python distribution on my website
https://lotharschanne.wordpress.com/
explains.

To be able to use the scripts, the Python modules that are imported at the beginning of the scripts with the import commands must be installed in the Python distribution used.

You can start the scripts in different ways:

The easiest way is to go to a console, change to the directory containing the spectrum files (ts or ascii) that you want to edit and start the respective script with the command:

```
python3 path/to/the/script/scriptname.py
```

Another method is to start a script within a development environment. I prefer Spyder for this, which offers the advantage that, in addition to a console in which the

script is processed, there is also a window with the Python code of the script (so that you can quickly adjust parameters or comment out individual lines, for example).
can be changed). In addition, variables created in the program sequence are displayed in the variable manager window. This can be helpful if something goes wrong and you can see, for example, when which spectrum file the program was interrupted because, for example, a header entry is missing. And in the file window you can see the files that are currently in the working directory (including those that have been created by the running script in the meantime).
were). You can also set the working directory in Spyder and include the paths to the scripts in the PYTHONPATH environment variable so that the scripts are automatically searched for in these path directories.

Most scripts that generate graph windows (e.g. images of spectra) switch to an interactive backend (QT5Agg) at the beginning of the script. I.e.
The saved graphics windows can be actively changed when the focus is placed on a window. This means that the window can be enlarged or reduced in size, or a section of the displayed content (the spectrum) can be selected, the section can be moved, the graph can be saved, etc. This allows you to take a closer look at the details of a spectrum.

Folder ascii

1d_dat_read_plot_csv_save.py

The script reads an ASCII file and plots the spectrum. The ASCII file has 2 columns with the column headings 'WAVE' and 'FLUX'. The extension of the file is .dat. The separator is determined automatically with guess=True. Plot and save as csv without column headings (script lines 40 and 41).

1d_txt_plotting.py

The script reads a spectrum that is stored in an ASCII file. The data are in 2 columns with float numbers (wavelength and flux) without caption. Path/name are queried. The spectrum is plotted.

comparison_plot_2spectra_ascii.py

The script is intended to plot a measured spectrum and a template (e.g. a theoretical spectrum) against each other and thus compare them.

It reads in the template and the measured spectrum (enter path/name). The 1d spectra must be available as tab-separated ASCII tables with the column headings 'WAVE' and 'FLUX'. The plot is saved as a PDF (line 38).

PlotAndCrop_1d_ascii.py

The script has the task of plotting a 1d spectrum in ASCII format with the two columns 'WAVE' and 'FLUX'. Then a wavelength range to which the spectrum is reduced (cropping). This section is then plotted and saved as an ASCII file, whereby the wavelength range is specified in the name of the file.

Folder astroplanner

astroplanner.py

Calculates the height of the object above the horizon for a point in time at an observation location and plots it over a period of 24 hours. Input of the star, the observer and the date is required.

astroplanner_for_object_list.py

Calculates the heights of the objects in an object list (ascii file (csv) with the object names in a column) above the horizon for a geolan time at an observation location and plots them over a period of 24 hours. The graphs are saved as pdf in the working directory. Input of the list of stars, the observer and the date/time is required.

BC folder

BC_1d_spectra_ ts_series.py

The script calculates the barycentric correction BC and the corrected JD as HJD of a series of 1d spectra in the fits format of an object. Writes an ascii file (format tab) with the BC's and HJD's to the working directory. If desired, corrects the spectra read in by the calculated barycentric correction BC and saves them as ts (and after commenting out lines 206 to 213 also as an asci table). The coordinates of the observer and the object must first be entered in the script or commented out/ commented in. B y default, the coordinates of the object are read in from the Internet (Internet connection required), but can also be entered or commented out. must be entered manually in the script (line range 71 to 107).

Folder binaries (period determination and calculation of orbital elements)

This folder contains scripts which are used to determine the orbital elements from the radial velocities of SB1 or SB2 star systems measured at de ned times (in JD) and heliocentrically corrected. For this purpose, the user summarizes the points in time (JD) and the radial velocities (RV in km/s) in an ascii file, separated in 2 columns without a caption with a defined uniform delimiter. E.g. with a comma (csv file) or

a tab (dat file). This file with the experimental data is read by the scripts discussed here.
First, the period is determined independently using the scripts PDM_Periodenanalyse.py and Periodenanalyse.py. Both use different methods to determine the period.
The best result can be judged on the basis of the scattering of the measured values in the phase plots.
In addition to the period P, a time To must also be defined, the reference time with which the phase plot is calculated. The point in time is arbitrary; it is practical to take the time of the first RV measurement, which is done automatically by the scripts.
will.

Calculation_Phase_plot_withKnownPeriod

The script reads in an ascii file with the two columns (without column caption!) JD (observation times) and RV (radial velocities). The time points (JD) are convolved with the period to be entered (assumed to be known) and the corresponding RV's are plotted with the phases. The plot is saved as a pdf. The results are saved in a csv file with the columns labeled JD, RV and Phase.

PDM_Period_Analysis.py

It is a tool for detecting (possibly non-sinusoidal) periodic fluctuations in time series data by minimizing the scatter of the convolved data set. The scatter is determined bin-wise.
Reads in a file (in csv or tab format) with the two columns JD and associated value (e.g. RV's), but without column headings. Then performs a period analysis (PDM analysis) with both columns, displays the periodogram graphically and outputs the period (in days).
The period range for which the calculation is to be carried out must be adapted to the respective case in line 38 !!!!

Period analysis.py

Reads in a file (in csv or tab format) with the two columns JD and associated value (e.g. RV's), but without column headings. Then performs a period analysis with both columns (Lomb-Scargle method and Stringl ngen method), graphically displays the RV values against time, the periodogram and the phase plot and outputs the period (in days).
The stringlength method is a technique for searching for potentially non-sinusoidal, periodic fluctuations in adata set. Theidea i s to convolve the data with a number of trial periods. Consecutive points in the folded data set are connected by hypothetical lines and the total length (the string length) is calculated. The length statistic takes a minimum when the convolution results in a well-ordered set.

The range used to calculate the frequencies or periods must be adjusted in lines 51 and 85 !!!!

Phase calculation.py

The script reads in a csv file with the columns JD and RV (without column headings). With the period and T0 parameters specified in the script (adjust !), the observation times are then convolved, the phase plot is displayed and the results are saved in an ascii file (format tab, columns 'JD', 'Phase', 'RV').

Orbitelements_optimize.py

The script reads an ascii file with two columns without caption with the time points (JD, first column) and the RV's of a binary component (SB1 or SB2) (RV, second column). The script calculates the phase plot with the orbital parameters P and T, which are assumed to be known (independently determined). P and T must be updated in lines 41 and 42. For the remaining orbital elements, the estimated values in lines 35 to 38 are generally sufficient. Then the remaining orbital parameters (of the observed binary component) K1, e, w1 and gamma are optimized using a curve tting routine, so that the theoretical RVs are calculated and displayed in the phase plot for comparison with the measured RVs. The optimized orbital parameters K1, e, w1, and gamma and their standard deviations are printed out.

RV_Calculation.py

The script calculates the theoretical radial velocities from the orbital parameters assumed to be known (to be adjusted in the script!) and a series of time points (in JD, are read in from an ascii table). Write the results into a csv file named RVs.csv and plot the phase diagram.

Convolution folder

convol_dat.py

The script is used to broaden a high-resolution (usually a theoretical) 1d spectrum with a standard deviation (determined from the apparatus profile of the spectrograph) so that the line widths correspond to the measured spectra.

Read in a 1d spectrum in the form of an ASCII table with the extension .dat, 2 columns with the columns labeled 'WAVE' and 'FLUX'. The spectrum is convolved with a standard deviation (stddev in the unit of the step). The convolved flux is stored together with the wavelength in a two-column ASCII table (spacer = tab). The columns are labeled 'WAVE' and 'FLUX'. In addition, the two spectra are plotted and the plots are saved as PDF and PNG.

convol_ ts.py

Read in a 1d spectrum in the form of a ts file.
The spectrum can be convolved (folded) with a standard deviation, expressed by the expression R. The convolved ts file is saved with the R in the name of the file.

Pollux_cutout_convol.py

Read in a synthetic spectrum in the form of a table (as available as .spec in the Pollux database, http://pollux.graal.univ-montp2.fr/). Start wavelength and step are given in the accompanying Pollux- le Spektrum.txt. Calculation of an optional section (Panda's dataframe labeled 'range'). The column 'NFLUX' = normalized flux for the entire spectrum and for the range is plotted. The range can be convolved with a standard deviation (stddev in the unit of the step). The convolved flux is stored together with the wavelength in a two-column ASCII table (spacer = tab, column names WAVE and FLUX).

rotation widening_ ts.py

Read in a series of 1d spectra in ts format (intended for synthetic spectra to calculate the rotational broadening and the limb-darkening effect).
The spectra are rotationally broadened by entering the limb-darkening coefficient and the rotation speed. The convolved ts files are saved, whereby the file names are supplemented with the term 'red-broadened'.

Folder RepresentationSpeed space

tsSeries_line_in_speed_space

Read in a series of heliocentrically corrected spectra in ts format, display an optional line in velocity space in two plots: First, all spectra
plotted on top of each other. Secondly, all spectra are plotted with an optional o set. The plots can be saved as pdf.

tsSeries_line_in_speed_space_JDrange.py

Read in a series of heliocentrically corrected spectra in ts format, enter a time period as JD start and JD end. Display of an optional line in velocity space in two plots:
First, all spectra plotted together.
Secondly, all spectra are plotted with an optional o set. The plots can be saved as pdf and png.

tsSeries_lines_in_speed_space_JDrange.py

Read in a series of heliocentrically corrected spectra in ts format, enter a time period as JD start and JD end. Representation of several optional lines in velocity space in one plot, plotted one above the other with an optional o set. The plots can be saved as pdf and png.

Order EW

Time_series_EW.py

Calculated for a time series ON THE CONTINUUM OF NORMAL SPECTRES in the t format the equivalent width of a line. The integration limits can be entered interactively or manually. The EW calculation assumes that the same wavelength interval can be used to calculate the integral for all spectra in the series, i.e. that there are no significant RV changes, or if there are, that the line is isolated (i.e. the flux = 1 in the environment).[1]. It is best to use heliocentrically corrected spectra. The EW's are stored in an ASCII table with the columns 'Spectrum' and 'EW'.

Time_series_EW_line_depth_line_wave_length.py

Calculated for a time series ON THE CONTINUUM OF NORMAL SPECTRES in the ts format, the equivalent width of a line, as well as the line depth and the wavelength of the line minimum. Enter the integration limits manually or graphically.

Normalization errors are compensated for by a renormalization routine in the integration area.

The EW calculation assumes that the same wavelength interval can be used to calculate the integral for all spectra in the series, i.e. no significant RV-
If there are no changes, or if there are, the line is isolated (i.e. the flux = 1 in the environment). It is best to use barycentrically corrected spectra.

The first spectrum is plotted and displayed for 20 seconds. During this period, the graph window is interactive so that the spectrum can be enlarged and the integration wavelength range for the EW calculation can be visually selected can. This reaction time window of 205 seconds can be changed in line 78 (plt.pause(TimeInSeconds).

TimeSeries_EW_RangesInOneLine.py

Calculates the equivalent distance of a line divided into areas for a time series normalized to the continuum in ts format. Input of the integration limits (in angstr m) via a list that must be adapted. The EW calculation assumes that the same wavelength interval is used to calculate the integral for all spectra in the series.

[1]If this cannot be assumed, the spectra must first be corrected with regard to their RVs, which can be done with the script RVKorrekturPerTemplate_1d_spectra_ ts_series.py in the RV_Korrektur folder using a template.

can be used, i.e. no significant RV changes take place. It is therefore best to use barycentrically corrected spectra. The determined EW's and the flux minima and maxima of the line areas are saved in an ascii file.

Folder ts

1d_ tSpektrum_ansehen.py

Read out and display the header data of a 1d spectrum in t-format. Plotting the spectrum.

1d_ tSpectrum_view_with_line_identi cation.py

Read in a 1d spectrum in ts format, plot the spectrum with the option of marking one or more laboratory wavelengths of spectral lines in the form of vertical lines in the plot.

1d_ tSpectrum_view_with_line_identi cationPerElement

View an optional 1d spectrum in ts format, enter the ions/elements whose lines are to be marked in the spectrum. The graph can be saved as a pdf if desired.

1d_ tSpectrum_view_with_Subplots.py

Viewing a wavelength-calibrated 1d spectrum, reading out and displaying the header data. Plotting of the entire spectrum and a division into a selectable number of sections, which are then displayed in corresponding subplots. In addition, any wavelength range can be selected and plotted.

Folder format conversions

Format conversions.py

Conversion from

Julian date in calendar dates

Calendar date in Julian date

RA and DEC in hexagesimal form in oat

RA and DEC of oat in hexagesimal form

KK folder (cross-correlations)

CrossCorrelation_dat_Series.py

A cross-correlation of a series of target spectra with respect to a template spectrum is performed. All are available as an ascii file (.dat). The calculated RV's are printed and saved in a file.

CrossCorrelation_ ts_Series.py

A cross-correlation of a series of target spectra with respect to a template spectrum is performed. of a template spectrum. Both are available as ts. The RVs and optionally barycentrically corrected RVs are printed out and saved in a file. If the barycentrically corrected RVs are to be calculated, the observer and object coordinates must be adjusted in the script. By default, however, the star coordinates are taken from the Internet by entering the object name (e.g. alp Ori). However, this requires an active internet connection.

CrossCorrelation_ ts_Series_individualLines.py

A cross-correlation of a series of target spectra with respect to a template spectrum is performed. All are available as ts. A spectrum section is queried (surrounding a line or a center wavelength), which is used for the KK.

The RV and optionally the baryc. corrected RV are calculated. The data is written to an ascii file. If the barycentric corrected RV's are to be calculated, the observer and object coordinates must be adjusted in the script.
can be used. By default, however, the star coordinates are taken from the Internet by entering the object name (e.g. alp Ori).

Light curves folder

AAVSO.py

The program takes the light curve data from a csv file (which has been imported from the AAVSO database, for example). You enter the period to be evaluated (JD start and end) and the program calculates the daily mean values of the mag values. These are written to a csv file and displayed in gra sch.

Folder Lines tting

FitLine.py

The script is used for modeling an absorption or emission line process. Various models (Gauss, Voigt, SplitLorentz) can be selected.

The script inserts a series of spectra normalized to the continuum in ts format.

The first spectrum is then plotted and you can either e n l a r g e and select the line to be modeled for a time defined and changeable in line 89 with a mouse click or alternatively enter the wavelength range and the wavelength of the line extremum manually.

When graphically selecting the line, you have to click three times with the mouse in the plot, first to the left of the line (with sufficient continuum), then to the right of the line and thirdly the approximate line minimum/maximum. This selects the wavelengths in question.

The model to be used is selected. You can choose between Gauss, Doppelgauss, Lorentz, Voigt, Doppelvoigt. The results of the fitting are saved in an Excel file.

For emission lines, the amplitude should be set to positive in the initial values for the models. Negative for absorption lines.

For double peaks (blends of two lines), the distance between the peaks in the initial values of the models must be adjusted so that the initial t (=initial t in the left part of the generated graph) roughly matches the measured line pro l: vary the number in the lines center=dict(value=extremum-10).

The graphics can be saved if desired.

FitLine_automaticParameterized.py

The script is used to model an absorption or emission line pro l. The pro l can have one or two maxima or minima.

Various models (Gauss, Voigt, SplitLorentz for asymmetric profiles) can be selected.

The script l dts a series on the continuum of normalized spectra in ts format. If the line pro l in a series is Doppler shifted or completely different from the first spectrum, the tting may not work. Such spectra must then be treated individually.

The first spectrum is plotted and you can either e n l a r g e and select the line to be modeled for a time defined and changeable in line 89 with a mouse click or alternatively enter the wavelength range and the wavelength of the line extremum manually.

When graphically selecting the line, you have to click four times with the mouse in the plot, first to the left of the line (with as much continuum as possible), then to the right (with as much continuum as possible) and thirdly the approximate line minimum/maximum and

The fourth is the heavier line minimum/maximum (or again the line minimum/maximum). This selects the wavelengths and flows to be used.

You can choose which model you want to use. You can choose between Gauss, Doppelgauss, Lorentz, Voigt and Doppelvoigt. The results of the fitting are saved in an Excel file.

The graphics can be saved if desired.

FitLine_PositiveAndNegativeGauss.py

The script is used for Gaussian modeling of an emission line profile, which, as with beta Lyrae consists of a Gaussian emission line with a central absorption contained therein.

The script l dt a series on the continuum of normalized spectra in ts format. If the line pro l in a series is Doppler shifted or is completely different from the first spectrum, the tting may not work. Such spectra must then be treated individually.

The first spectrum is then plotted and the line to be modeled for a time defined and changeable in line 84 can either be e n l a r g e d and selected with a mouse click.

When graphically selecting the line, you have to click four times with the mouse in the plot, first to the left of the line (with as much continuum as possible), then to the right (with as much continuum as possible) and thirdly on the approximate line maximum and fourthly on the minimum of the embedded absorption. This selects the wavelengths and fluxes to be used.

The results of the fitting are saved in an Excel file. The graphs can be saved if desired.

FitLine_PositiveAndNegativeVoigt.py

The script is used for Gaussian modeling of an emission line profile, which, as with beta Lyrae consists of a Gaussian emission line with a central absorption contained therein.

The script l dt a series on the continuum of normalized spectra in ts format. If the line pro l in a series is Doppler shifted or is completely different from the first spectrum, the tting may not work. Such spectra must then be treated individually.

The first spectrum is then plotted and the line to be modeled for a time defined and changeable in line 84 can either be e n l a r g e d and selected with a mouse click.

When graphically selecting the line, you have to click four times with the mouse in the plot, first to the left of the line (with as much continuum as possible), then to the right (with as much continuum as possible) and thirdly on the approximate line maximum and fourthly on the minimum of the embedded absorption. This selects the wavelengths and fluxes to be used.

The results of the fitting are saved in an Excel file. The graphs can be saved if desired.

Folder Standardization

automaticNormalization_timeseries_20230615.py

Works for a single or a time series of 1d spectra in fits format. For the first spectrum, the parameters 'inter' and 'sm' are optimized. Takes an interval of 'inter' pixels from pixel to pixel and compares it with the neighboring intervals. Finds local maxima = center points. You can also exclude several wavelength ranges from the formation of normalization points, i.e. delete them (wide lines). The smoothing parameter 'sm' (decimal number) sets the sensitivity of the spline that is used at the end for normalization. sm = 0.0 means: The spline goes through all grid points. The larger sm (10.0, 100, 1000....), the stiffer the spline. An sm is optimal if the spline traces the curves of the continuum well, but does not protrude into the lines. You can then delete points that are l a r g e r or smaller than the spline by a certain percentage. Finally, all spectra of the time series are normalized with the optimized parameters. The normalized spectra (as well as the plots and the normalization functions) are saved if desired. In addition, all selected parameters are saved in an ascii file called Parameterlist.txt.

Folder Rebinning_averaged_spectrum_Di erence_spectra

newbinning_mediumSpectrum.py

Reads in a spectrum catalog (ts), rebinds the spectra and generates tab spectra (tab table with the column names WAVE and FLUX) as well as ts files, all with the same selectable step size and the same selectable wavelength range. The type of interpolation (linear or cubic spline) can be selected by commenting out (lines 100 to 106). In addition, an averaged spectrum is calculated from the common wavelength range and saved as a tab table with the column names WAVE and FLUX. The averaged spectrum is also plotted and saved as ts.

Di erence spectra.py

Reads in 1d spectra of a series (in the ascii format 'tab', created with the script newbinning_mittleresSpekt
all spectra used must have the same wavelength range and the same wavelength range).
step size) and forms density spectra for the specified reference spectrum, which are then saved as a tab file. All density spectra are displayed graphically in a plot, which is also saved.

Folder RV_Correction

RVCorrection_1d_spectra_ ts_series.py

The script corrects a series of 1d spectra in the fit format of an object by a manually entered RV [km/s]. Writes the fit files and ascii files of the RV-corrected spectra to the working directory. The respective RV is noted in the header of the generated fit.

RVCorrectionPerTemplate_1d_spectra_ ts_series.py

The script corrects a series of 1d spectra in the fit format of an object by an RV [km/s] calculated by KK in comparison to a template. Writes the fit files of the RV-corrected spectra to the working directory. The respective RV is noted in the header of the generated fit. An ASCII table with the RVs is also saved.

terrLinien_RVKorrektur_Regression_1d_spectra_ ts_series

Input: Read in a spectrum series in ts format. These must not be heliocentrically corrected.
The script corrects the calibration by determining the line minimum of known terrestrial lines by regression and recalculates the spectra with the determined mean RV, as far as it exceeds 3 km/s. Saves the corrected spectra as ts. Show the modeled terrestrial lines as gra k. Print out the determined RVs. Writing the RV's, the mean value and the standard deviation into an ascii- le.

terrLinien_RVKorrektur_RBF_1d_spectra_ ts_series.py

Input: Read in a spectrum series in ts format. These must not be heliocentrically corrected.
The script corrects the calibration by determining the line minimum of known terrestrial lines via RBF and converting the spectra with the determined mean RV. Saving the corrected spectra as ts. Showing the modeled
terrestrial lines as Gra k. Printout of the determined RVs.
The terrestrial lines must of course be present in the spectra. Currently 3 normally less disturbed water lines between 6543 and 6574 angstr m are taken into account.

Folder RV_Measurement

The RV_Messung folder contains scripts whose task is to carry out radial velocity measurements on individual spectra or on spectra series. Various methods are used for this purpose:

KK (cross-correlation) of spectrum sections

Minimum determination on individual lines.

For the minimum determination at defined lines, a module called Linienlisten.py exists in this directory, which is shared by many of the RV determination scripts and must therefore be imported by the scripts. Therefore, the module must be installed in the
must be in the same directory as the respective script or in a directory that is contained in the environment variable PYTHONPATH so that Python can find it. Additional lines can of course be included in this module. To do this, the module only needs to be edited accordingly.

RV_MessungAnLinie_ ts_interaktiv.py

Reads in a spectrum catalog (time series). Calculates the heliocentric correction from the observation time and the (to be adjusted) coordinates of the observer and object, shows the selected line in a plot, the minimum of the selected line is
calculates the heliocentrically corrected radial velocity RV_bc from the minimum. Outputs the calculated data as ASCII files (tab-separated).

RV_MessungAnLinie_Zeitserie_dat_perInteraktion.py

The script is used to determine the RV of a line in double star systems (SB2), which can be split into 2 lines.

Reads in the spectra catalog (time series of normalized 1d spectra in tab format, 2 columns WAVE and FLUX). Defines the line minimum by interaction and determines the radial velocity from the minimum. Plots all spectra to mark up to 2 line minima and outputs determined data (RV and apex) as ascii files (tab-separated, as .dat). The RV's are not barycentrically corrected.

RV_MeasurementOnLine_Time_Series_dat_perInteraction_regression_2Lines.py

Like the previous script, except that the minimum (the minima of the two lines) is determined mathematically by regression (degree 2, 4 or 6).

Reads in a spectra catalog (time series of normalized 1d spectra in tab format). Determines the line minimum by interaction and determines the radial velocity RV from the minimum calculated by regression. Plots all ttings for control purposes and outputs the determined data as ascii files (tab-separated, as .dat).

RV_MessungAnLinie_Zeitserie_dat_perRegression.py

Reads spectra catalog (time series of normalized 1d spectra in tab format). Fits the selected line by nth degree regression and determines the radial velocity RV from the minimum. Plots all ttings and outputs determined data as ascii files (tab-separated, as .dat).

RV_MessungAnLinie_Zeitserie_ ts_perGauss t.py

Reads in a spectra catalog (time series of normalized!!!)   in ts-format).
Calculates the barycentric correction from the time of observation and the (to be adjusted) coordinates of the observer and object, ttests the selected line three times (total and inner areas) using Gauss t and determines the heliocentrically corrected radial velocity RV from the heliocentrically corrected minimum. Plots
all ttings and outputs determined data as ascii files (tab-separated, as .dat).

RV_MessungAnLinie_Zeitserie_ ts_perRBF.py

Reads in a spectra catalog (time series of normalized 1d spectra in t-format). Calculates the heliocentric correction from the time of observation and the coordinates (to be adjusted) of the observer and object, ttts the selected line using the radial basis function (RBF) and determines the heliocentrically corrected radial velocity RV from the heliocentrically corrected minimum. Plots all ttings for checking and outputs the determined data as ascii files (tab-separated, as .dat).

RV_MessageTime_series_ ts_perRegression.py

Reads in a spectra catalog (time series of normalized 1d spectra in ts format). Calculates the heliocentric correction from the time of observation and the (to be adjusted) coordinates of the observer and object, tts the selected line in the minimum range by regression and determines the (heliocentrically corrected) radial velocities RV and RV_bc from the (heliocentrically corrected) minimum. Plots all ttings and outputs determined data as ascii files (tab-separated, as .dat).

RV_MessageTime_series_ ts_perRegression_AbsUndEmi.py

Reads in a spectra catalog (time series of normalized 1d spectra in ts format). Calculates the heliocentric correction from the time of observation and the (to be adjusted) coordinates of the observer and object, tts the selected line in the minimum (absorption) or maximum (emission) range by regression and determines the heliocentrically corrected radial velocity RV from the heliocentrically corrected minimum/maximum. Plots and saves all ttings and outputs the determined data as an ascii file (comma-separated, as .csv).

RV_MessungAnLinie_Zeitserie_ ts_perSpline.py

Reads in a spectra catalog (time series of normalized 1d spectra in ts format). Calculates the heliocentric correction from the time of observation and the (to be adjusted) coordinates of the observer and object, tts the selected line by spline and determines the (heliocentrically corrected) radial velocities RV and RV_bc from the (heliocentrically corrected) minimum. Plots all ttings and outputs determined data as ascii files (tab-separated, as .dat).

Folder SNR

SNR_betasigma.py

Calculates the SNR of each spectrum for a series of 1d-ts spectra using the beta*sigma method. If appropriate, adjust the parameters N and j in the script (lines 40 and 43).

SNR_ ts_severalRanges.py

Plots the selected 1d-ts spectrum. Select the ranges for the SNR calculation (continuum) by setting the limits with left mouse clicks and ending the respective input of the range by double pressing the Escape key. Any number of ranges can be entered.

noise.py

The script can be used to introduce additional noise into a spectrum.

Theoretical spectra folder

ascii_Spectrum_section_rebinned_convol

Reading of a synthetic spectrum in the form of an ascii table with 2 columns WAVE and FLUX labeled. Calculation of a selectable wavelength section. This section is then rebinned with an optional step size and then additionally convolved with an optional FWHM (apparatus sample). The
selected rebinned wavelength range and additionally the convolved spectrum. The rebinned flux section of the original ascii file and the rebinned and convolved flux are saved together with the wavelength in a two-column ascii table (spacer = comma) and in a ts file each.

ArtSpectrumGenerate.py

The script generates an artificial line spectrum broadened by gau broadening to a specific FWHM, expressed by an expression R. The lines, line widths, wavelength range and R are defined in lines 19 to 24.

Kurucz_Model_GaussBroadened.py

Loading a Kuruzc star atmosphere model (1d spectrum) after selecting T_e and log g. Convolution with a Gau function on spectrograph r e s o l u t i o n, selection of the wavelength range and adjustment to a desired step width. Storage
as an ASCII tab table and as . t with the filename to be entered. Saving the plot as a PDF.

To be able to use the script, a PyAData folder containing the model spectra must exist in the user directory. The folder is created automatically the first time the script is called. It is best to use the default values for the questions.

Pollux_cutout_rebinned_convol_spec.py

The Pollux database (http://pollux.graal.univ-montp2.fr/) allows you to download theoretical spectra for a wide range of physical stellar properties. We use the .spec format.

Import of a synthetic spectrum in the form of a table (as available as .spec in the Pollux database). The normalized flux is used. Calculation of a selectable wavelength section (Panda's dataframe labeled 'newtable'). This area is then rebinned with an optional step size and then additionally convolved with an optional FWHM (apparatus sample). The
selected rebinned wavelength range and additionally the convolved spectrum. The rebinned flux section of the original .spec and the rebinned and convolved flux are saved together with the wavelength in a two-column ascii table (spacer = tab) and in one ts file each.

Time series folder

ascii_timeseries_aPlot_withO set

Reads in a time series of 1d spectra in tab format. Plots the spectra with an O set and saves the plot as .png and .pdf

badPixFilter.py

The script is used to remove single-pixel spikes (hot pixels) in 1d spectra.

Create file list for wavelength-calibrated 1d spectra of a series in t format. Replacement of individual pixel values that are above a limit value (badpixel) by the mean value of the neighboring pixels and saving of all corrected spectra as otherwise unchanged t. File name supplemented by '_badPixRemoved'.

badPixFilter_CompareNeighborPixel.py

Create file list for wavelength-calibrated 1d spectra of a series in t format. Replace individual pixel values that exceed the neighboring pixels by a factor with the mean value of the neighboring pixels and save all corrected spectra as an otherwise unchanged t. Name supplemented by _ badPixRemoved.

Dynamic spectra plot

Generates a dynamic spectrum plot from a sequence of (barycentrically corrected) ts-1d spectra, with the ordinates forming the observation times (a header entry named 'JD' must exist in the header of each spectrum !) The wavelength range shown is selected.

t_Series_in_csv_and_dat.py

Convert a series of wavelength calibrated 1d spectra in t-format to text format .csv (comma-separated) and .dat format (tab-separated). With column headings 'WAVE' and 'FLUX'.

HeaderDisplayAndCorrection_Series.py

Creation of a spectra list of the 1d spectra in t-format in a folder, correction/addition of the header of all spectra.

Headerprint_Series_csv.py

Reads the header data for a 1d spectra series in ts format and writes them to separate ascii files (.csv).

JD_EintragInHeader.py

The script reads in a series of ts-1d spectra and calculates the respective JD from different header entries for the observation date and enters it as JD i n the header of the respective spectrum.

wavelength_range_time_series_ ts

Create file list for wavelength-calibrated 1d spectra of a time series in t format. Save the file list with information on the start and end of the wavelength scale. Print out the common wavelength range.

Crop_section_time_series_ ts.py

Create file list for wave length calibrated 1d_spectra of a time series in t-format. Plotting of all spectra with selection of whether the graph should be saved. Printout of the jointly covered wavelength range in the console. Trim all spectra to a selectable wavelength range and save all trimmed spectra as t with wavelength range in the file name.

AveragingFrom_ ts.py

1d spectra in ts format. They must all have the same header['CRVAL1'], header['CDELT1'], header['NAXIS1'], header['CRPIX1']). These variables are printed out for each spectrum (check for equality). By entering 'y', all fluxes of the spectrum series are averaged and the average spectrum is saved as t.

timeseries_aPlot_withO set.py

Reads in all 1d- ts spectra of a series and plots them with an optional O set
each other. Saves the graph as a PNG and as a PDF.

timeseries_aPlot_withO set_JDBereich.py

Reads in a time series of 1d spectra in ts format. A JD range is selected that is to be plotted. Plots the spectra with a O set on top of each other and saves the plt as .png and .pdf.

timeseries_aPlot_withO set_JD_range_wavelength_range

Reads a time series of 1d spectra in ts format. A Julian date range can be selected, the spectra of which are plotted and a wavelength range. Plots the spectra sections with a O set a n d saves the plot as .png and .pdf.

timeseries_aPlot_withO set_wavelength_range

Reads a time series of 1d spectra in ts format. A wavelength range can be selected. Plots the spectra sections with a O set on top of each other and saves the plt as .png and .pdf.

timeseries_plot_JeEineGra k.py

Reads in all 1d- ts spectra of a series and plots them in separate graphs. The plots are saved as PDF files.

Verification_JD_inHeader.py

The script reads in a series of ts-1d spectra and checks whether there is an entry for the observation date (in JD) in the header. The findings are printed out for each spectrum.

Time_series_observation_times.py

Reads in the 1d-ts spectra of a spectrum series and prints out the observation times. In addition, the start and end of the wavelength range of each spectrum are output. The observation times are saved in an ASCII file.

Time_Series_Spectra_smoothen.py

Reading in a series of 1d ts files and smoothing the u x with a specific window width. Saving the smoothed ts.