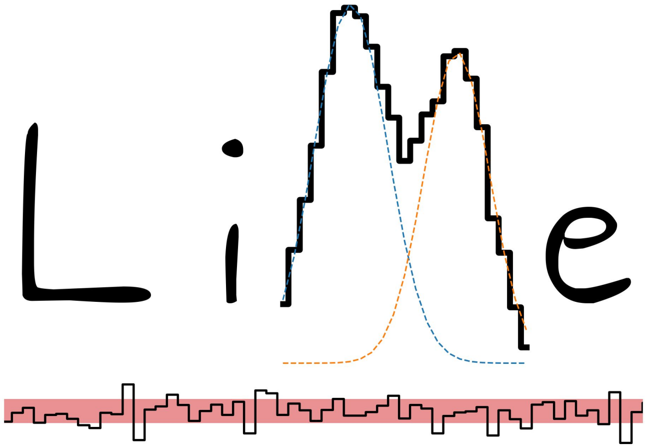
A Line Measuring library:



Vital Fernández
Ricardo Amorín, Christophe Morisset
Verónica Firpo

ESTALLIDOS XIII

UNIVERSIDAD
DE LA SERENA
CHILE

Why do we need another package?

- There are already (better) tools available:

Py Spec Kit

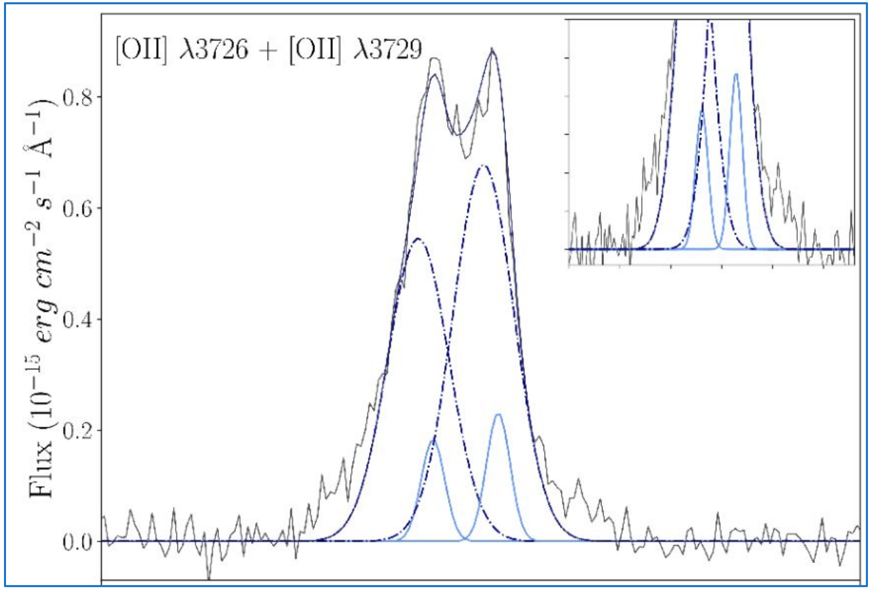
I R A F
Image Reduction and Analysis Facility

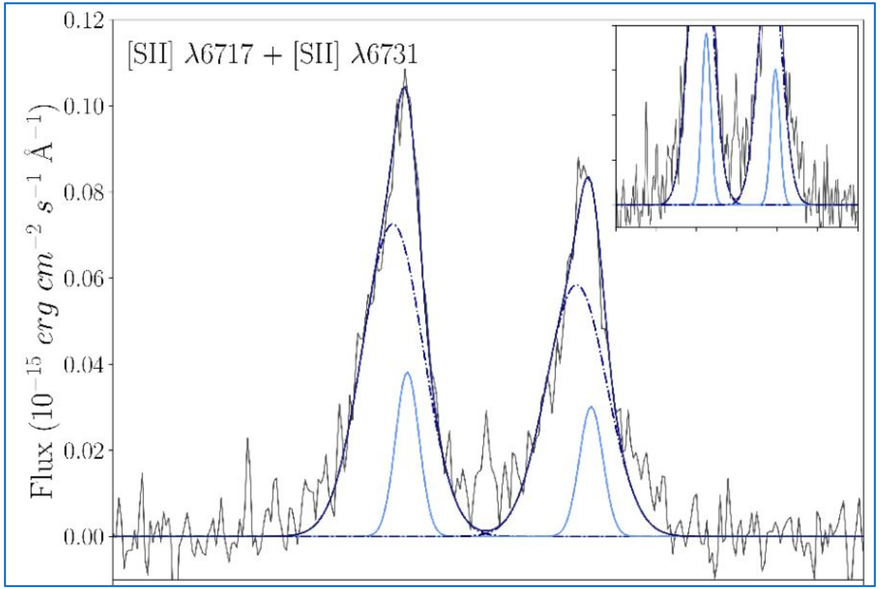
specutils
An Astropy package for spectroscopy

kapteyn astronomical institute

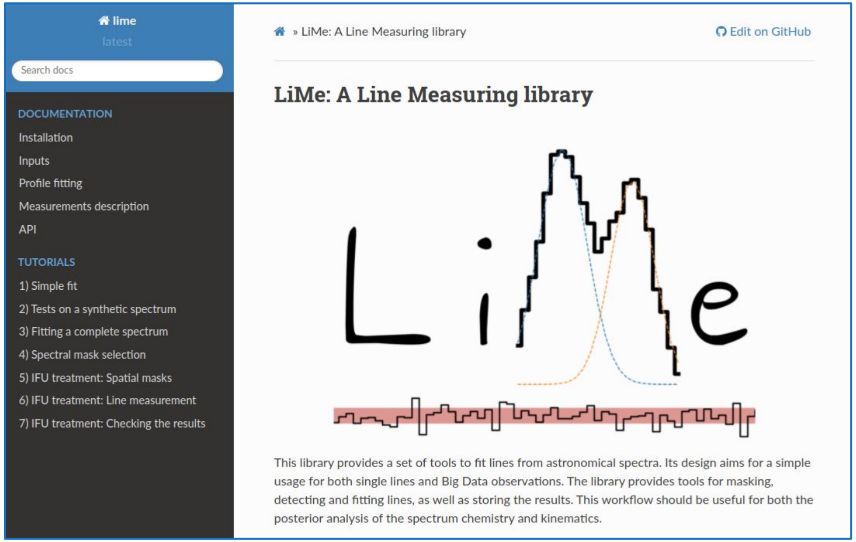
Chemodynamics analysis

- *Hogarth et al. 2020* compare the distinctive gas chemistry and kinematics from the emission of Green Pea galaxy GPJ1429.
- A successful analysis requires a careful track of the gaussian components to avoid conflicting results.



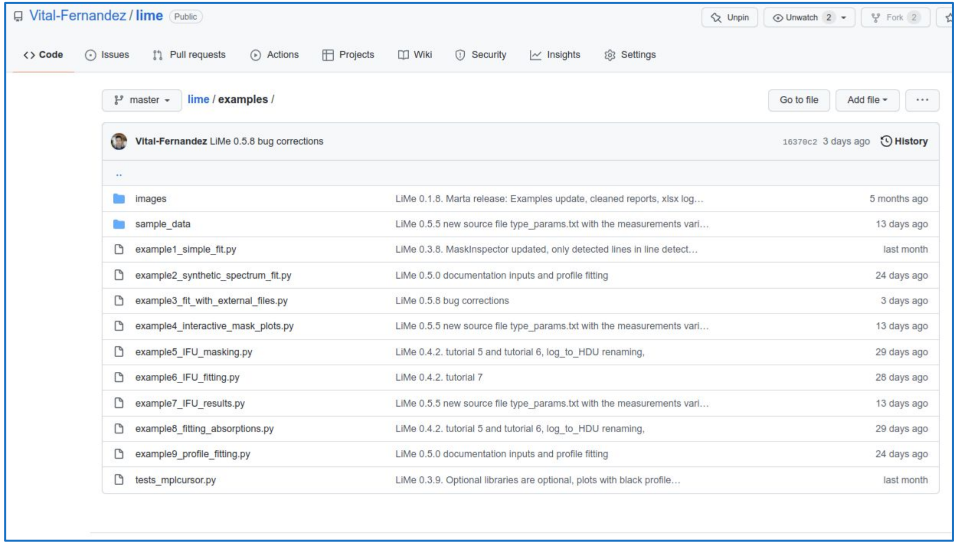


Where to start?


lime
latest
Search docs
DOCUMENTATION
Installation
Inputs
Profile fitting
Measurements description
API
TUTORIALS
1) Simple fit
2) Tests on a synthetic spectrum
3) Fitting a complete spectrum
4) Spectral mask selection
5) IFU treatment: Spatial masks
6) IFU treatment: Line measurement
7) IFU treatment: Checking the results
» LiMe: A Line Measuring library
Edit on GitHub
LiMe

LiMe: A Line Measuring library

This library provides a set of tools to fit lines from astronomical spectra. Its design aims for a simple usage for both single lines and Big Data observations. The library provides tools for masking, detecting and fitting lines, as well as storing the results. This workflow should be useful for both the posterior analysis of the spectrum chemistry and kinematics.



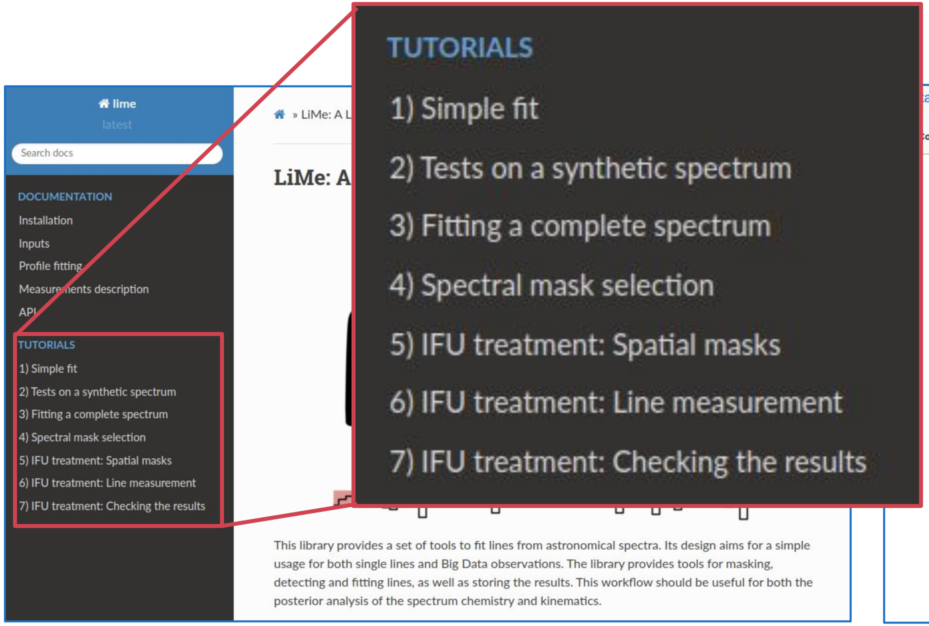
Vital-Fernandez LiMe 0.5.8 bug corrections 16370c2 3 days ago

..		
images	LiMe 0.1.8. Marta release: Examples update, cleaned reports, xlsx log…	5 months ago
sample_data	LiMe 0.5.5 new source file type_params.txt with the measurements vari…	13 days ago
example1_simple_fit.py	LiMe 0.3.8. MaskInspector updated, only detected lines in line detect…	last month
example2_synthetic_spectrum_fit.py	LiMe 0.5.0 documentation inputs and profile fitting	24 days ago
example3_fit_with_external_files.py	LiMe 0.5.8 bug corrections	3 days ago
example4_interactive_mask_plots.py	LiMe 0.5.5 new source file type_params.txt with the measurements vari…	13 days ago
example5_IFU_masking.py	LiMe 0.4.2. tutorial 5 and tutorial 6, log_to_HDU renaming,	29 days ago
example6_IFU_fitting.py	LiMe 0.4.2. tutorial 7	28 days ago
example7_IFU_results.py	LiMe 0.5.5 new source file type_params.txt with the measurements vari…	13 days ago
example8_fitting_absorptions.py	LiMe 0.4.2. tutorial 5 and tutorial 6, log_to_HDU renaming,	29 days ago
example9_profile_fitting.py	LiMe 0.5.0 documentation inputs and profile fitting	24 days ago
tests_mplcursor.py	LiMe 0.3.9. Optional libraries are optional, plots with black profile…	last month

Readthedocs: lime-stable.readthedocs.io/

Github: github.com/Vital-Fernandez/lime

Where to start?



lime

latest

Search docs

DOCUMENTATION

Installation

Inputs

Profile fitting

Measurements description

API

TUTORIALS

1) Simple fit

2) Tests on a synthetic spectrum

3) Fitting a complete spectrum

4) Spectral mask selection

5) IFU treatment: Spatial masks

6) IFU treatment: Line measurement

7) IFU treatment: Checking the results

» LiMe: A L

LiMe: A

This library provides a set of tools to fit lines from astronomical spectra. Its design aims for a simple usage for both single lines and Big Data observations. The library provides tools for masking, detecting and fitting lines, as well as storing the results. This workflow should be useful for both the posterior analysis of the spectrum chemistry and kinematics.

TUTORIALS

1) Simple fit

2) Tests on a synthetic spectrum

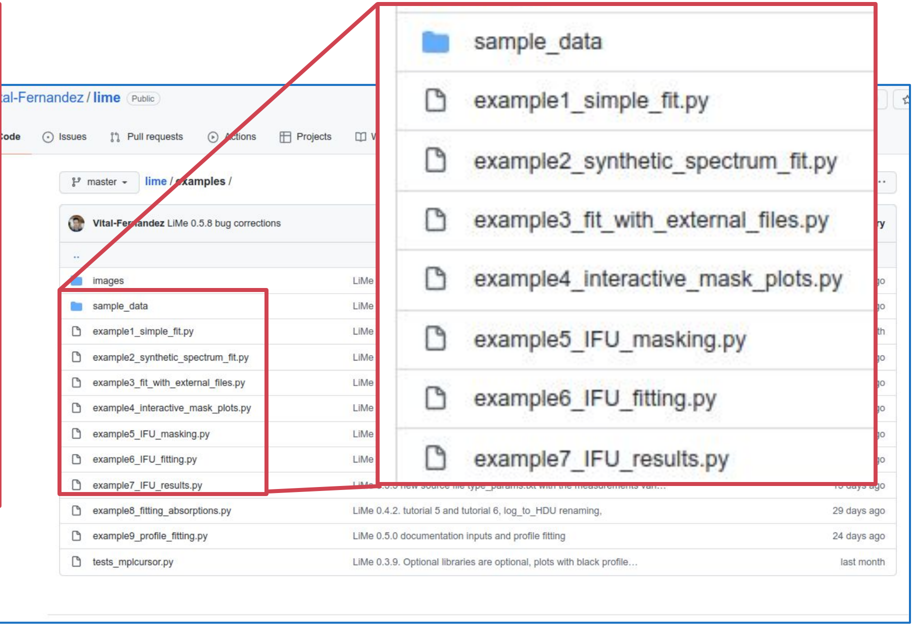
3) Fitting a complete spectrum

4) Spectral mask selection

5) IFU treatment: Spatial masks

6) IFU treatment: Line measurement

7) IFU treatment: Checking the results



lime / examples /

Vital-Fernandez LiMe 0.5.8 bug corrections

..

images LiMe

sample_data LiMe

example1_simple_fit.py LiMe

example2_synthetic_spectrum_fit.py LiMe

example3_fit_with_external_files.py LiMe

example4_interactive_mask_plots.py LiMe

example5_IFU_masking.py LiMe

example6_IFU_fitting.py LiMe

example7_IFU_results.py

example8_fitting_absorptions.py LiMe 0.4.2. tutorial 5 and tutorial 6, log_to_HDU renaming, 29 days ago

example9_profile_fitting.py LiMe 0.5.0 documentation inputs and profile fitting 24 days ago

tests_mplcursor.py LiMe 0.3.9. Optional libraries are optional, plots with black profile… last month

sample_data

example1_simple_fit.py

example2_synthetic_spectrum_fit.py

example3_fit_with_external_files.py

example4_interactive_mask_plots.py

example5_IFU_masking.py

example6_IFU_fitting.py

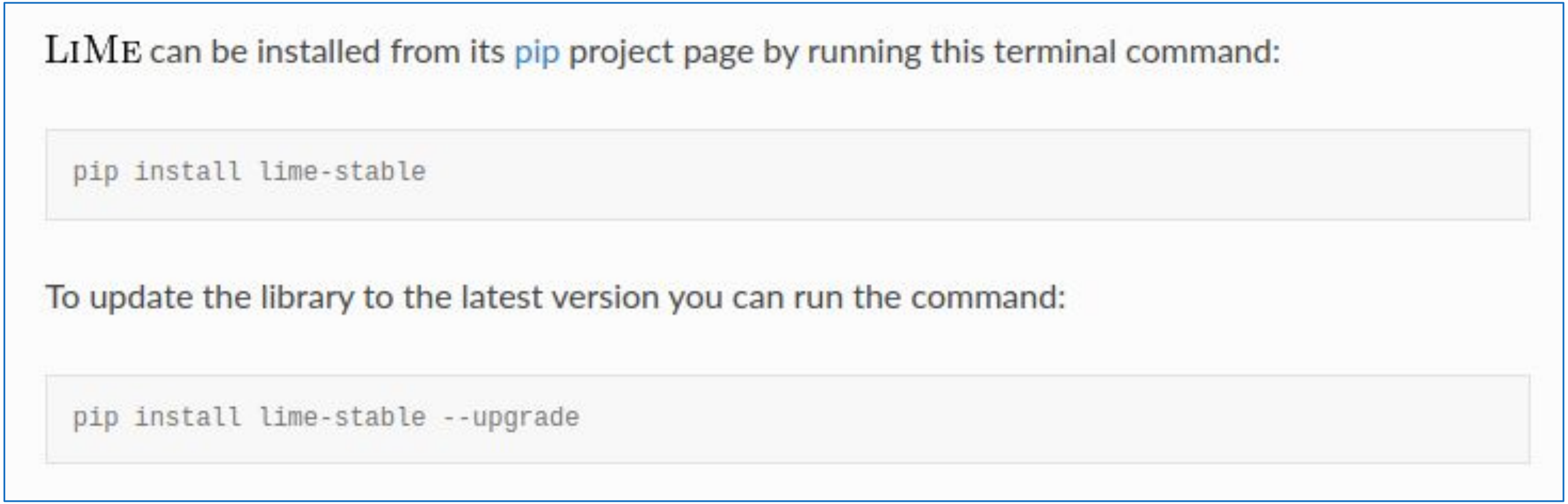
example7_IFU_results.py

Readthedocs: lime-stable.readthedocs.io/
step by step explanations

Github: github.com/Vital-Fernandez/lime
tutorials scripts and data

How to install?

lime-stable.readthedocs.io/en/latest/documentation/installation.html



LIME can be installed from its pip project page by running this terminal command:

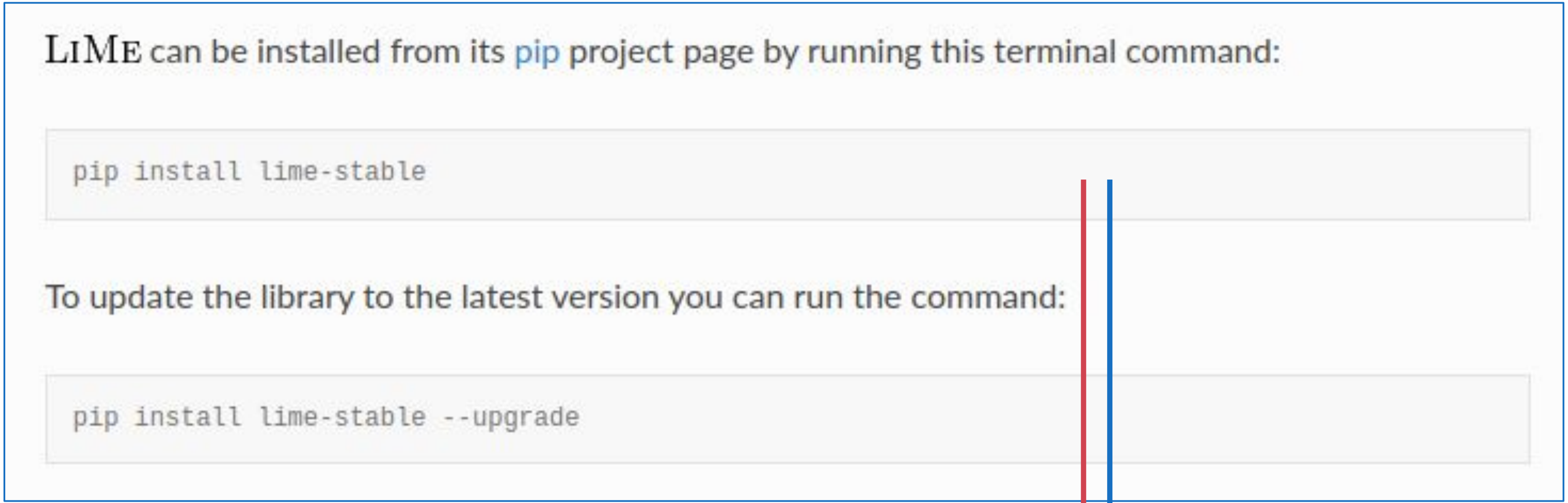
```
pip install lime-stable
```

To update the library to the latest version you can run the command:

```
pip install lime-stable --upgrade
```

How to install? (Dependencies)

lime-stable.readthedocs.io/en/latest/documentation/installation.html

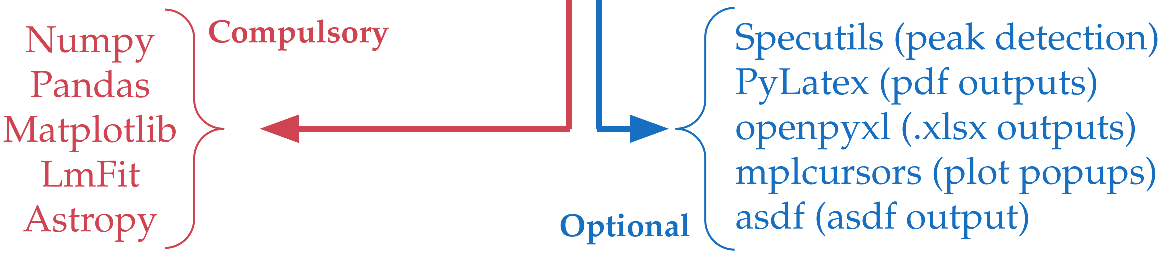


LiMe can be installed from its pip project page by running this terminal command:

```
pip install lime-stable
```

To update the library to the latest version you can run the command:

```
pip install lime-stable --upgrade
```



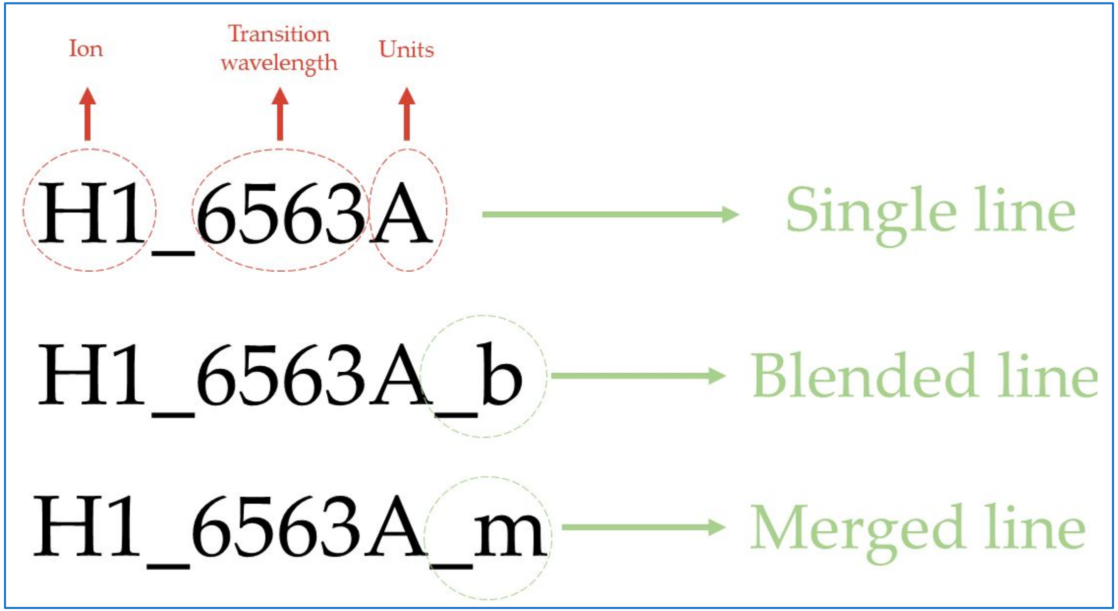
Compulsory

Numpy
Pandas
Matplotlib
LmFit
Astropy

Optional

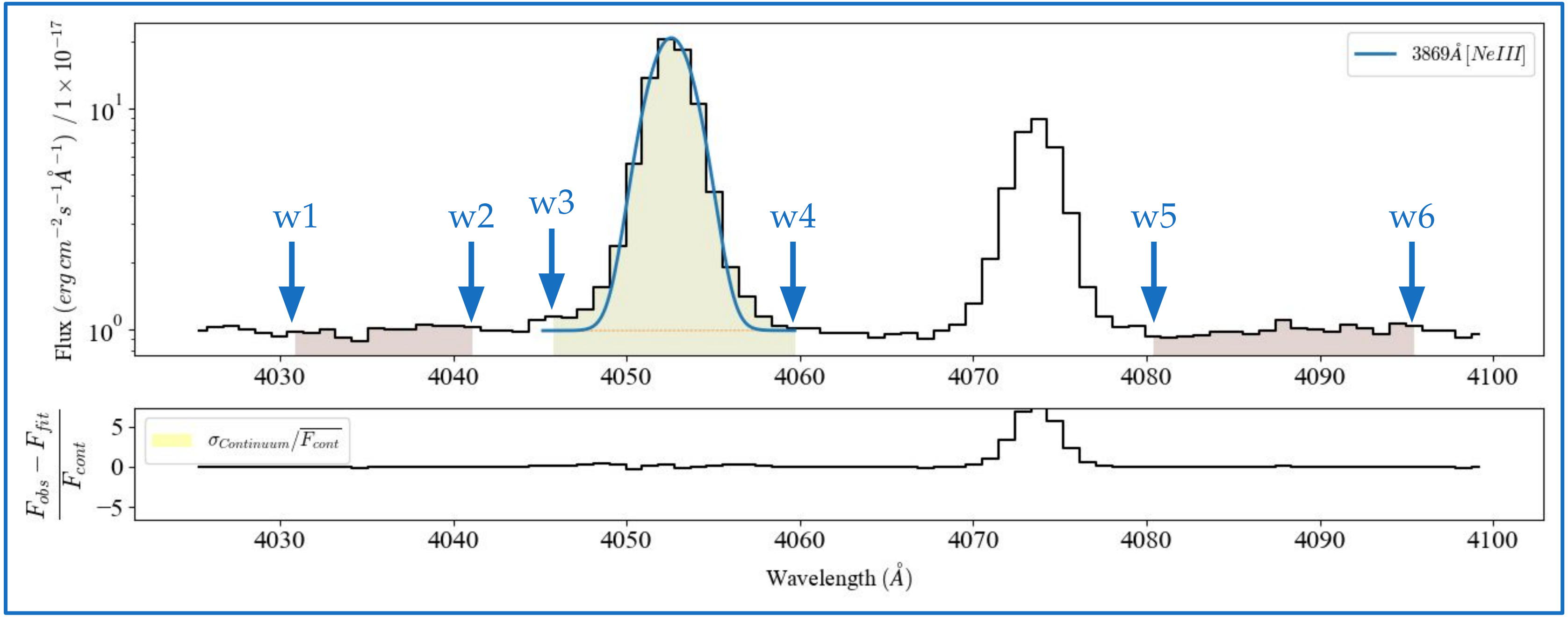
Specutils (peak detection)
PyLatex (pdf outputs)
openpyxl (.xlsx outputs)
mplcursors (plot popups)
asdf (asdf output)

LiMe inputs (1): Line name



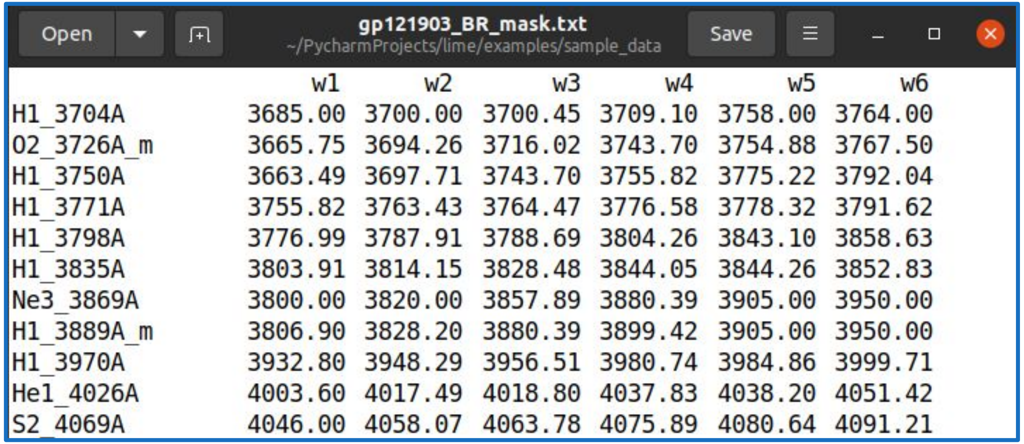
- The user **must include** a line label for the fitting. The notation follows the style in PyNeb by *Luridiana and Morisset (2014)*. This simplifies a posterior chemical analysis via the direct method or fitting from photoionization grids.
- The blended **(_b)** and merged **(_c)** suffixes provide the profile fitting policy.

LiMe inputs (2): Line mask

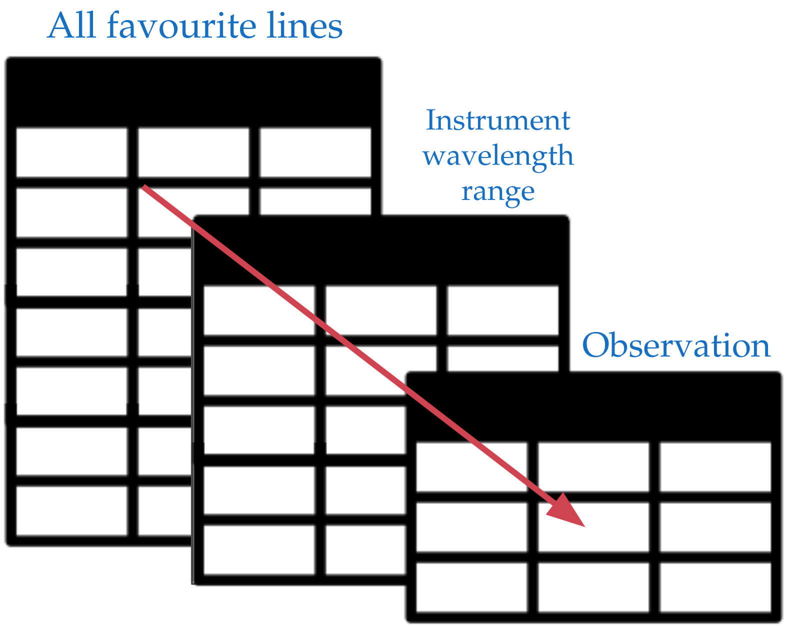


- The user **must include** a spectral mask for each line fitting. The mask design follows the Lick indices (see *Zhang et al 2005*): The line and two adjacent continuum bands.
- The mask consists in an **sorted** wavelength array (w1, w2, …, w6) in the **rest frame**.

Loading the spectral masks



```
                     w1      w2      w3      w4      w5      w6
H1_3704A        3685.00 3700.00 3700.45 3709.10 3758.00 3764.00
O2_3726A_m      3665.75 3694.26 3716.02 3743.70 3754.88 3767.50
H1_3750A        3663.49 3697.71 3743.70 3755.82 3775.22 3792.04
H1_3771A        3755.82 3763.43 3764.47 3776.58 3778.32 3791.62
H1_3798A        3776.99 3787.91 3788.69 3804.26 3843.10 3858.63
H1_3835A        3803.91 3814.15 3828.48 3844.05 3844.26 3852.83
Ne3_3869A       3800.00 3820.00 3857.89 3880.39 3905.00 3950.00
H1_3889A_m      3806.90 3828.20 3880.39 3899.42 3905.00 3950.00
H1_3970A        3932.80 3948.29 3956.51 3980.74 3984.86 3999.71
He1_4026A       4003.60 4017.49 4018.80 4037.83 4038.20 4051.42
S2_4069A        4046.00 4058.07 4063.78 4075.89 4080.64 4091.21
```



All favourite lines

Instrument wavelength range

Observation

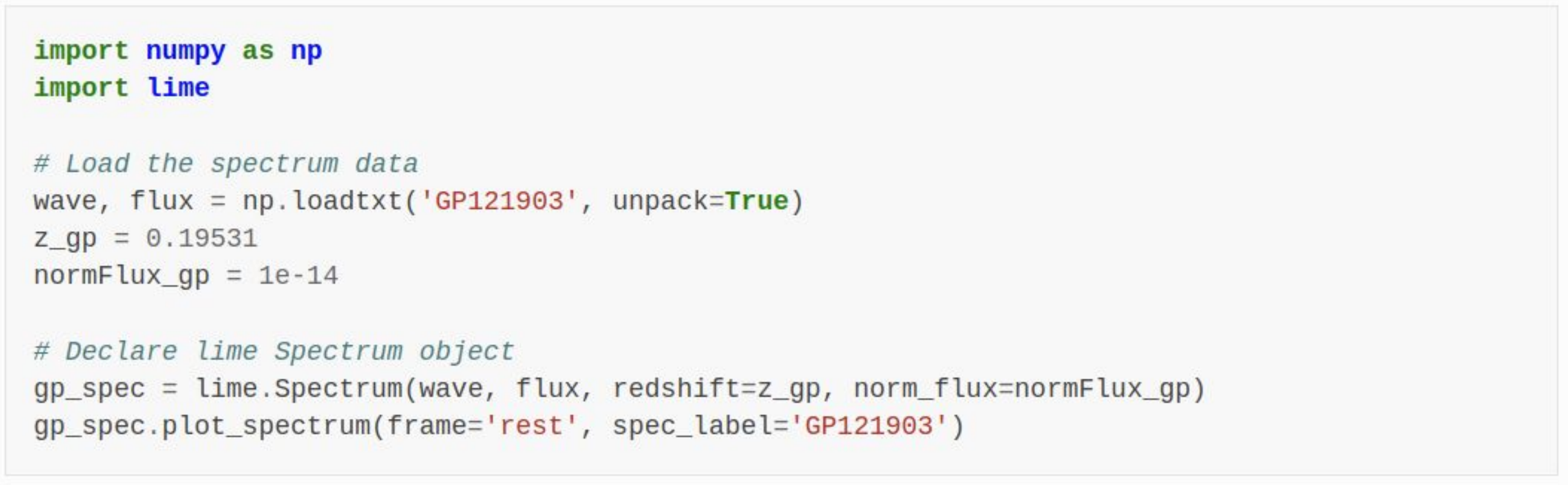
- The spectral mask are stored as tables with the line name its transition and contina bands

- These can be loaded using the *lime.load_lines_log()* function

- In the ideal scenario, each object has its own spectral mask.

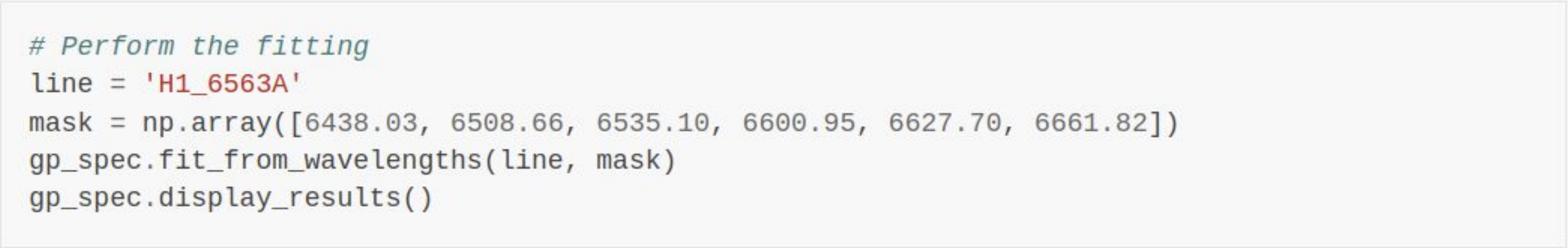
- You can use the *lime.MaskInspector()* class to manually them (tutorial 4)

LiMe inputs: Example (tutorial 1)



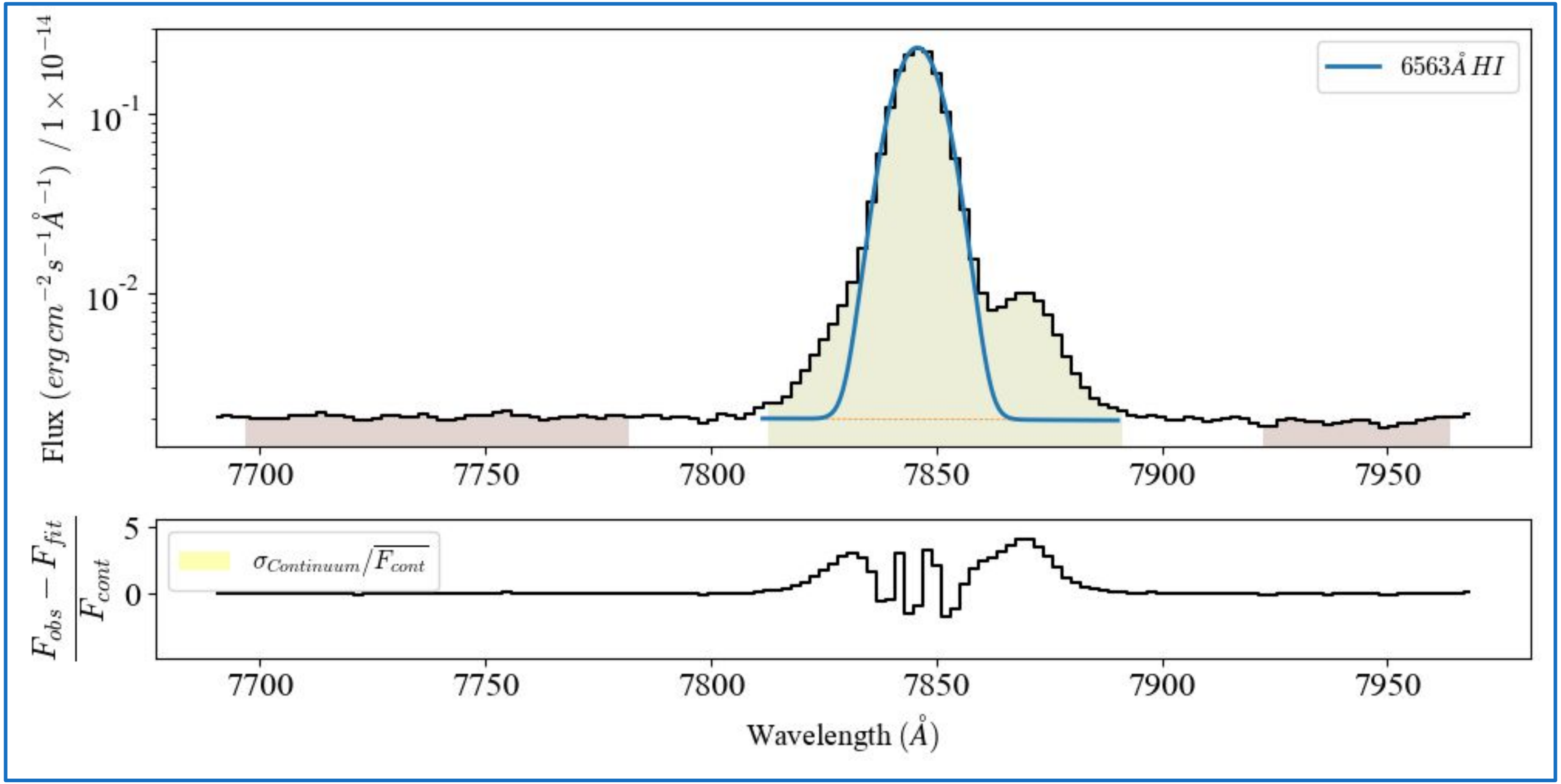
```python
import numpy as np
import lime

# Load the spectrum data
wave, flux = np.loadtxt('GP121903', unpack=True)
z_gp = 0.19531
normFlux_gp = 1e-14

# Declare lime Spectrum object
gp_spec = lime.Spectrum(wave, flux, redshift=z_gp, norm_flux=normFlux_gp)
gp_spec.plot_spectrum(frame='rest', spec_label='GP121903')
```

LiMe inputs: Example (tutorial 1)

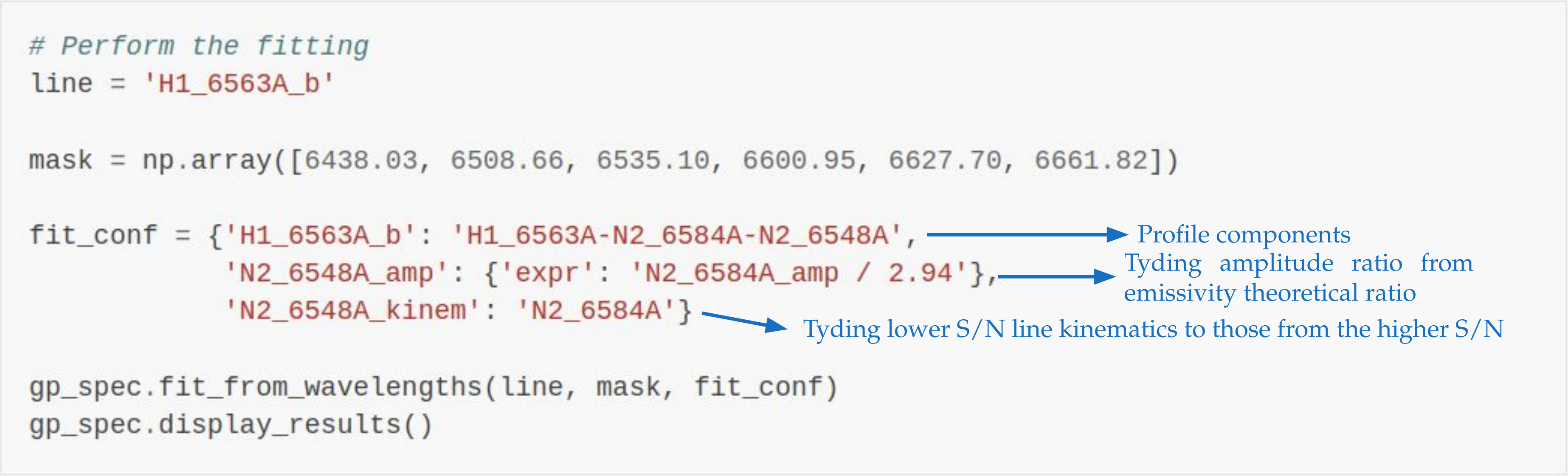


```
# Perform the fitting
line = 'H1_6563A'
mask = np.array([6438.03, 6508.66, 6535.10, 6600.95, 6627.70, 6661.82])
gp_spec.fit_from_wavelengths(line, mask)
gp_spec.display_results()
```

LiMe inputs: Example (tutorial 1)



LiMe inputs (3): Fitting configuration



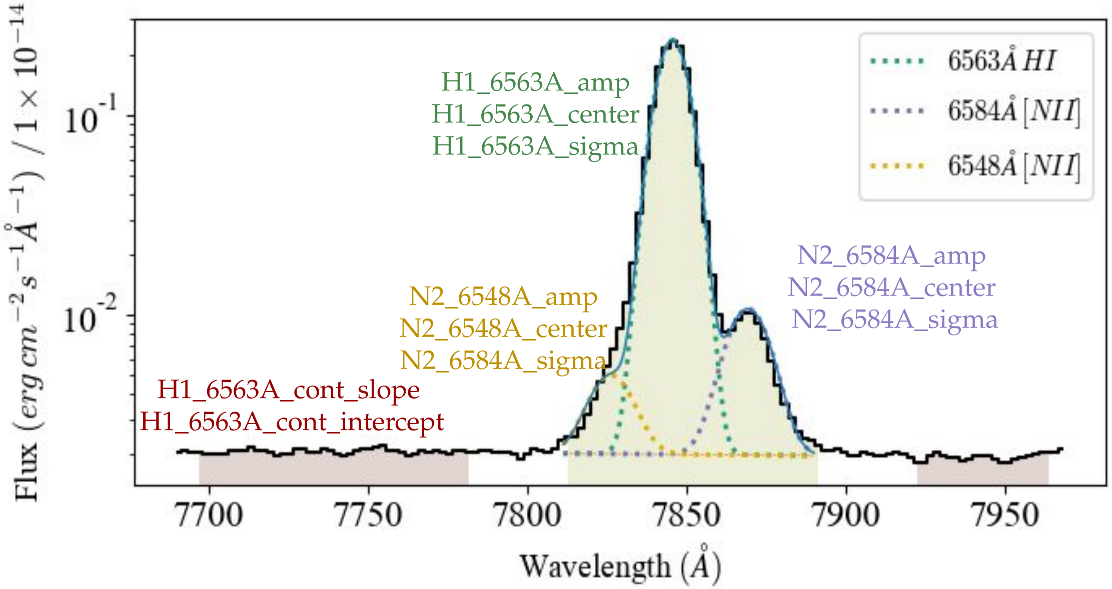
```python
# Perform the fitting
line = 'H1_6563A_b'

mask = np.array([6438.03, 6508.66, 6535.10, 6600.95, 6627.70, 6661.82])

fit_conf = {'H1_6563A_b': 'H1_6563A-N2_6584A-N2_6548A',
            'N2_6548A_amp': {'expr': 'N2_6584A_amp / 2.94'},
            'N2_6548A_kinem': 'N2_6584A'}

gp_spec.fit_from_wavelengths(line, mask, fit_conf)
gp_spec.display_results()
```

Profile components

Tyding amplitude ratio from emissivity theoretical ratio

Tyding lower S/N line kinematics to those from the higher S/N

Profile fitting configuration: Parameters label



- Each Gaussian profile contributes with 3 parameters (amplitude, μ_g, σ_g). Additionally, the continuum slope and intercept are free unless otherwise specified

- Each parameter can be constrained with these arguments:
  - value: initial guess
  - min: minimum value
  - max: maximum value
  - expr: mathematical expression as a function of other parameter
  - vary: fixed at the initial value (True, default)

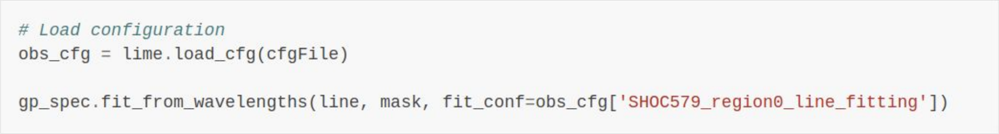
- LiMe can provide reasonable initial conditions which respect your constraints, so you don't need to specify it.

- If one line parameter depends on another line parameter make sure to put it afterwards in the list:

H1_6563A_b = H1_6563A-N2_6584A-N2_6548A

Profile fitting configuration: Loading from file

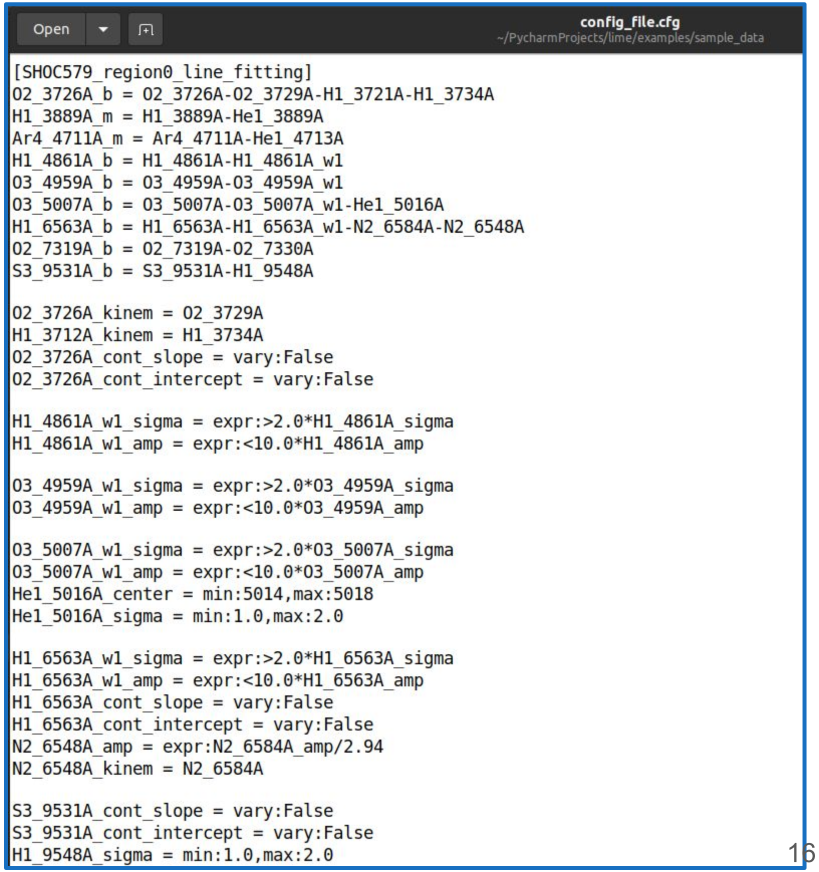
- The profile fitting configuration can be loaded from a text file using the *lime.load_cfg()* function.



```
# Load configuration
obs_cfg = lime.load_cfg(cfgFile)

gp_spec.fit_from_wavelengths(line, mask, fit_conf=obs_cfg['SHOC579_region0_line_fitting'])
```

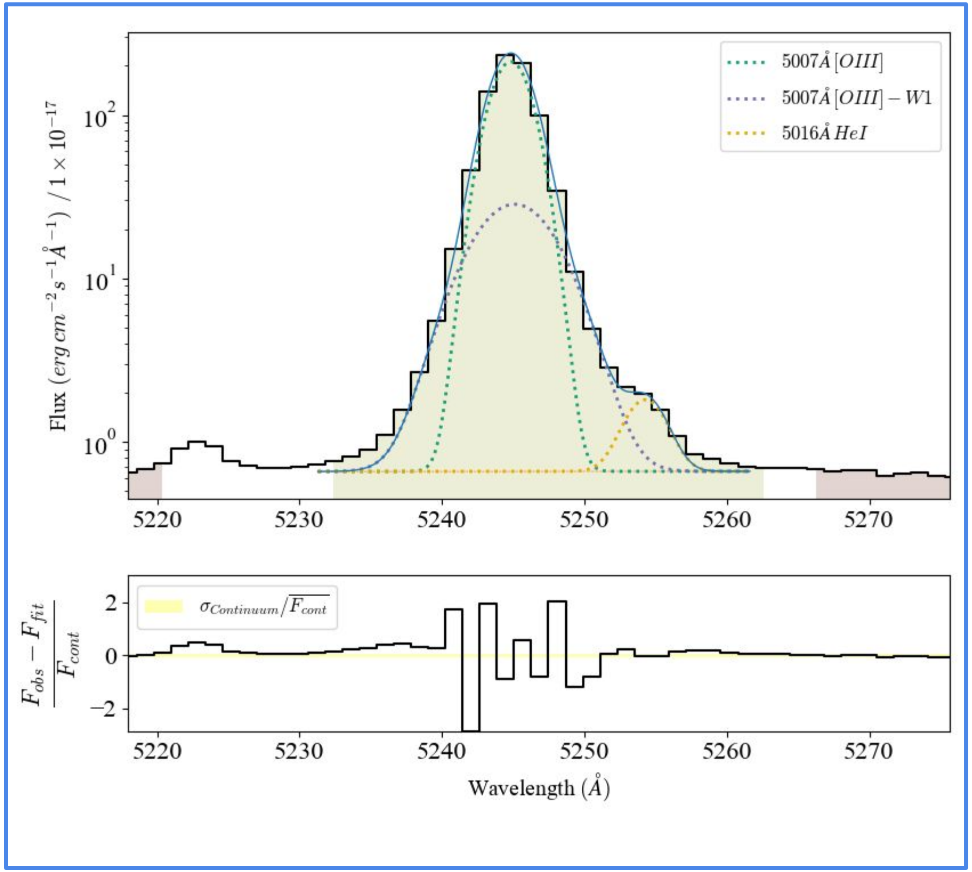
- Those sections with the "_line_fitting" keyword are converted to the variable format expected by LiMe.

- The algorithm will only use the profile constraints involving the current line.

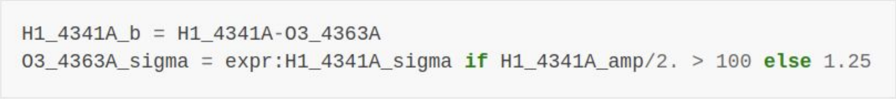


```
[SHOC579_region0_line_fitting]
O2_3726A_b = O2_3726A-O2_3729A-H1_3721A-H1_3734A
H1_3889A_m = H1_3889A-He1_3889A
Ar4_4711A_m = Ar4_4711A-He1_4713A
H1_4861A_b = H1_4861A-H1_4861A_w1
O3_4959A_b = O3_4959A-O3_4959A_w1
O3_5007A_b = O3_5007A-O3_5007A_w1-He1_5016A
H1_6563A_b = H1_6563A-H1_6563A_w1-N2_6584A-N2_6548A
O2_7319A_b = O2_7319A-O2_7330A
S3_9531A_b = S3_9531A-H1_9548A

O2_3726A_kinem = O2_3729A
H1_3712A_kinem = H1_3734A
O2_3726A_cont_slope = vary:False
O2_3726A_cont_intercept = vary:False

H1_4861A_w1_sigma = expr:>2.0*H1_4861A_sigma
H1_4861A_w1_amp = expr:<10.0*H1_4861A_amp

O3_4959A_w1_sigma = expr:>2.0*O3_4959A_sigma
O3_4959A_w1_amp = expr:<10.0*O3_4959A_amp

O3_5007A_w1_sigma = expr:>2.0*O3_5007A_sigma
O3_5007A_w1_amp = expr:<10.0*O3_5007A_amp
He1_5016A_center = min:5014,max:5018
He1_5016A_sigma = min:1.0,max:2.0

H1_6563A_w1_sigma = expr:>2.0*H1_6563A_sigma
H1_6563A_w1_amp = expr:<10.0*H1_6563A_amp
H1_6563A_cont_slope = vary:False
H1_6563A_cont_intercept = vary:False
N2_6548A_amp = expr:N2_6584A_amp/2.94
N2_6548A_kinem = N2_6584A

S3_9531A_cont_slope = vary:False
S3_9531A_cont_intercept = vary:False
H1_9548A_sigma = min:1.0,max:2.0
```

Profile fitting configuration: Inequality expressions

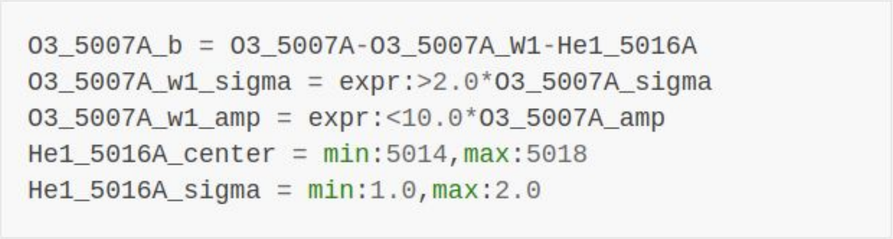


- The “expr” attribute allows the user to constrain parameter using a mathematical and/or logical expression:



```
H1_4341A_b = H1_4341A-O3_4363A
O3_4363A_sigma = expr:H1_4341A_sigma if H1_4341A_amp/2. > 100 else 1.25
```

- The “expr” can also be used to constrain a parameter as an inequality of another parameter:



```
O3_5007A_b = O3_5007A-O3_5007A_W1-He1_5016A
O3_5007A_w1_sigma = expr:>2.0*O3_5007A_sigma
O3_5007A_w1_amp = expr:<10.0*O3_5007A_amp
He1_5016A_center = min:5014,max:5018
He1_5016A_sigma = min:1.0,max:2.0
```

- Values for the gaussian center are assumed to be in the rest frame and converted to the observed.

Profile fitting configuration: Importing kinematics

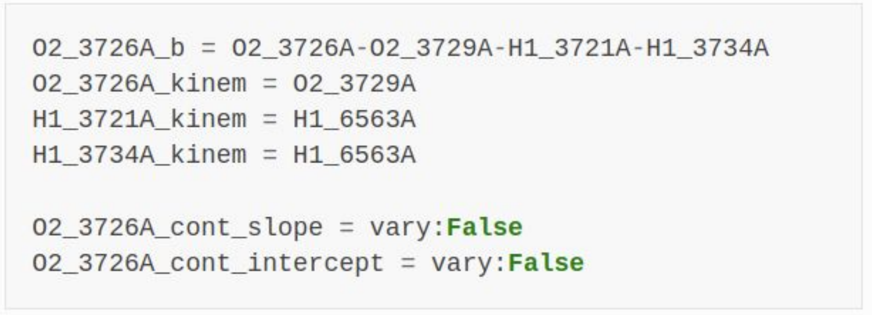
- The "_kinem" suffix will import line kinematics from another line (both v_r, σ_{vel}). Respecting the wavelength frame units:

$$\sigma_A = \sigma_B \, (km/s) \qquad\qquad \sigma_A = \sigma_B \cdot \frac{\lambda_A}{\lambda_B} \, (\mathring{A})$$

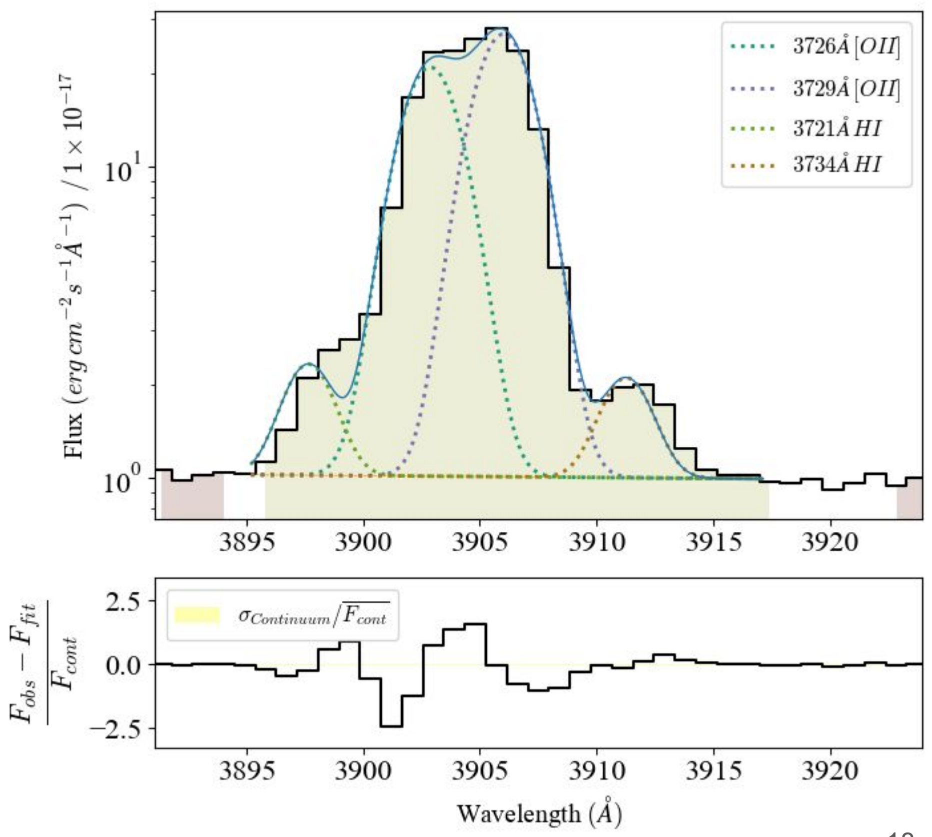
$$\Longrightarrow$$

$$v_{r,\,A} = v_{r,\,B} \, (km/s) \qquad\qquad \mu_A = \mu_B \cdot \frac{\lambda_A}{\lambda_B} \, (\mathring{A})$$

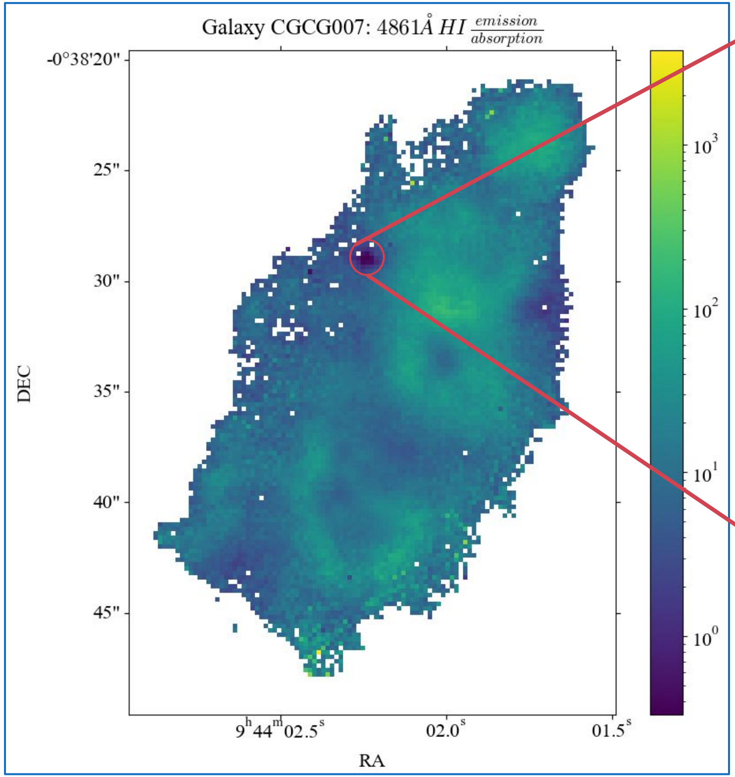
- If the parent line is within the current blended group, the kinematics are **binded**. If the parent line is outside the group the kinematics are **fixed.**

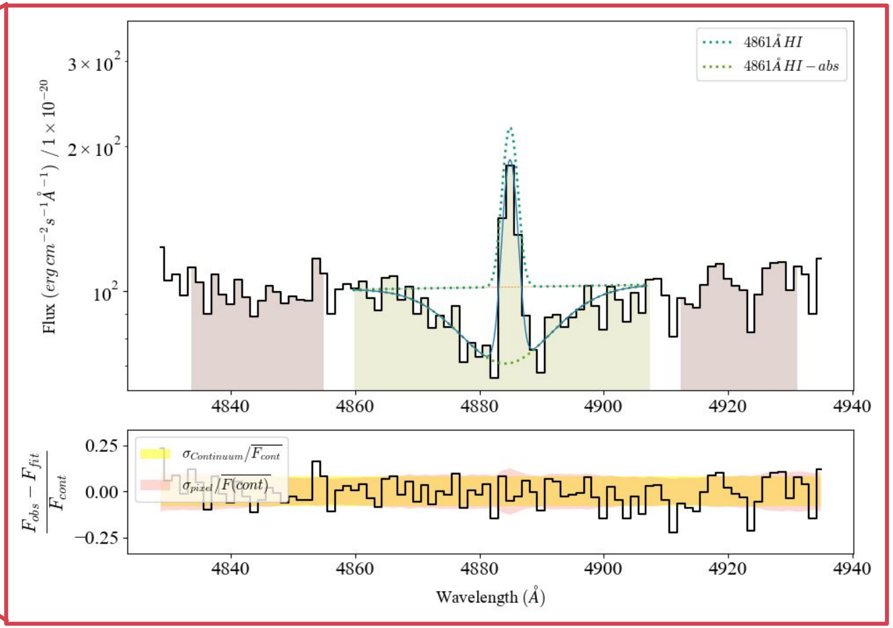


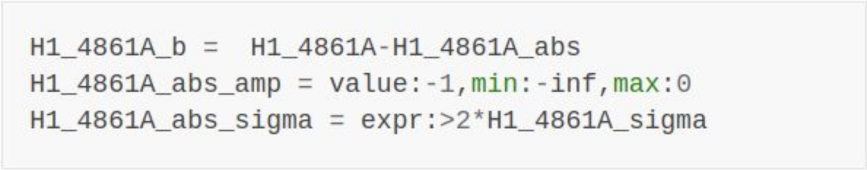
```
O2_3726A_b = O2_3726A-O2_3729A-H1_3721A-H1_3734A
O2_3726A_kinem = O2_3729A
H1_3721A_kinem = H1_6563A
H1_3734A_kinem = H1_6563A

O2_3726A_cont_slope = vary:False
O2_3726A_cont_intercept = vary:False
```



Profile fitting configuration: Absorptions





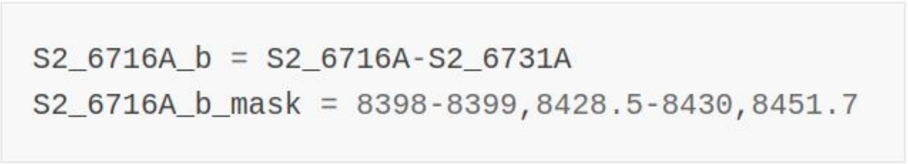


```
H1_4861A_b =  H1_4861A-H1_4861A_abs
H1_4861A_abs_amp = value:-1,min:-inf,max:0
H1_4861A_abs_sigma = expr:>2*H1_4861A_sigma
```

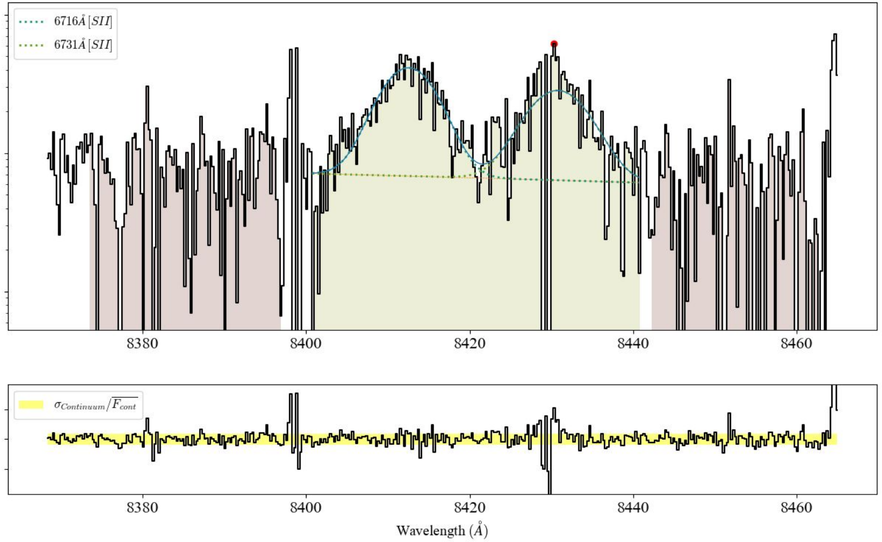
- The default emission line configuration can be switch to False in the *.fit_from_wavelengths* function

Profile fitting configuration: Pixel masking

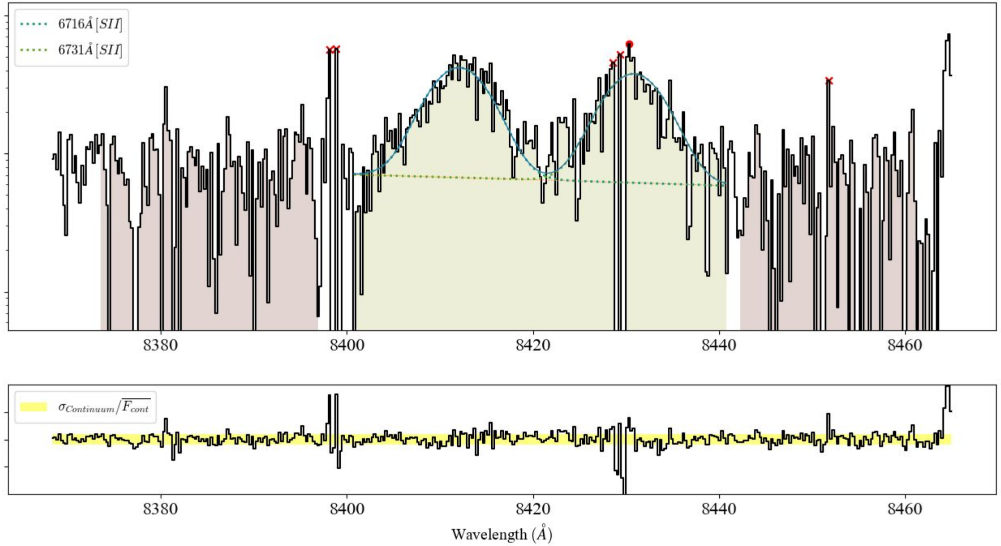
- Adding the **“_mask”** suffix to the line label allows the user to define regions to exclude from the fitting



```
S2_6716A_b = S2_6716A-S2_6731A
S2_6716A_b_mask = 8398-8399,8428.5-8430,8451.7
```



Without mask



With mask

LiMe measurements

No-Profile parametrization

Line fluxes
Peak flux
Peak wavelength
FWHM
Line redshift

F_{cont}, m_{cont}, n_{cont}, σ_{cont}
Velocity percentiles
λ_i, λ_f
FWZI
Pixel velocity width

Eqw

Gaussian

Amplitude
Profile center
Profile sigma
FWHM

Line Fluxes
Radial velocity
Velocity dispersion
$\sigma_{instr,}$ $\sigma_{thermal}$

Identity

Transition ion
Transition label latex
mask (rest frame)

Transition wavelength
blended/merged group

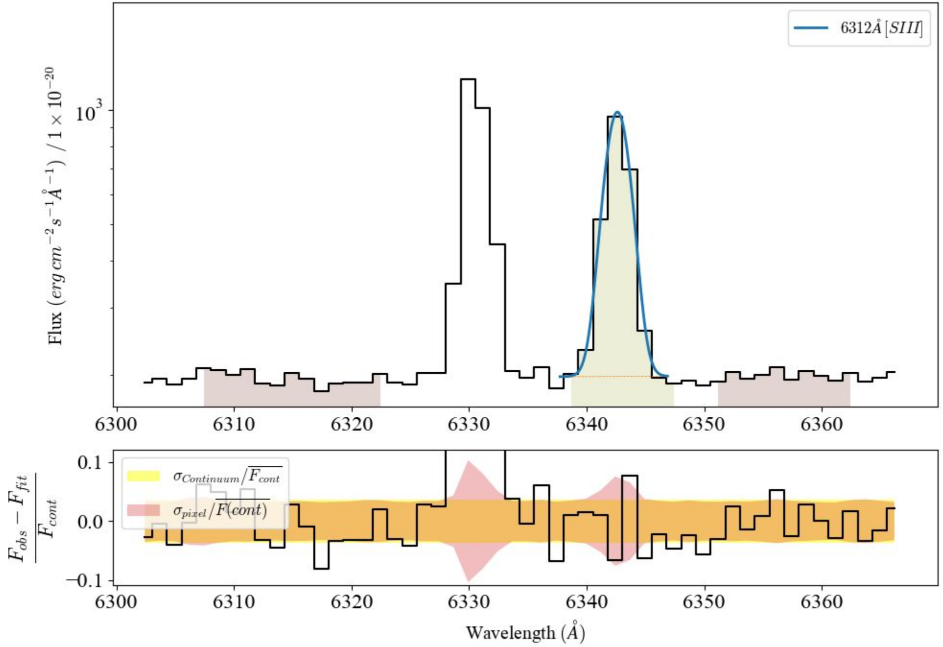
Diagnostic

S/N line
S/N continuum
χ^2
$\chi_\nu{}^2$

AIC
BIC
observations (errors)
comments (user)

- The measurements physical/mathematical description can be found at the online documentation

Line fluxes calculation



- A 1000 steps Monte Carlo integration.
- The pixel uncertainty is taken from the error spectrum provided by the user. Otherwise, a uniform value is computed from the adjacent continua bands flux standard deviation.
- The array mean integrated flux is taken as the nominal value and the standard deviation as the uncertainty.

Integrated

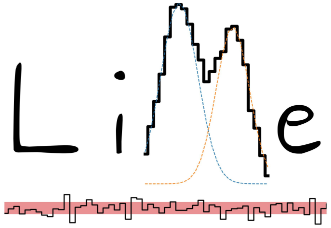
- A multi-Gaussian profile minimization .
- The Gaussian flux is calculated from the theoretical relation:

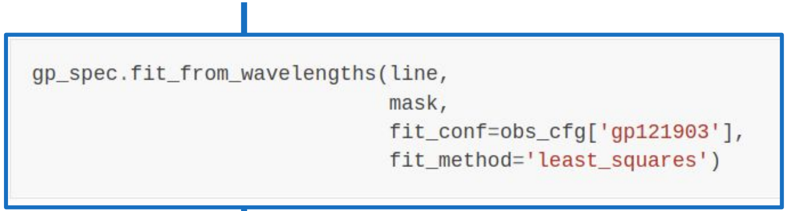
$$F_{i,g} = A_i \cdot \sqrt{2\pi} \cdot \sigma_i$$

- The minimizing algorithm models the uncertainty of the parameters. In most cases this is the 1σ error

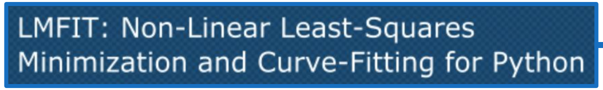
Gaussian

Minimizing algorithm





```
gp_spec.fit_from_wavelengths(line,
                             mask,
                             fit_conf=obs_cfg['gp121903'],
                             fit_method='least_squares')
```



LMFIT: Non-Linear Least-Squares Minimization and Curve-Fitting for Python

- LmFit by *Newville et al. (2014)* manages the minimizing algorithm call.

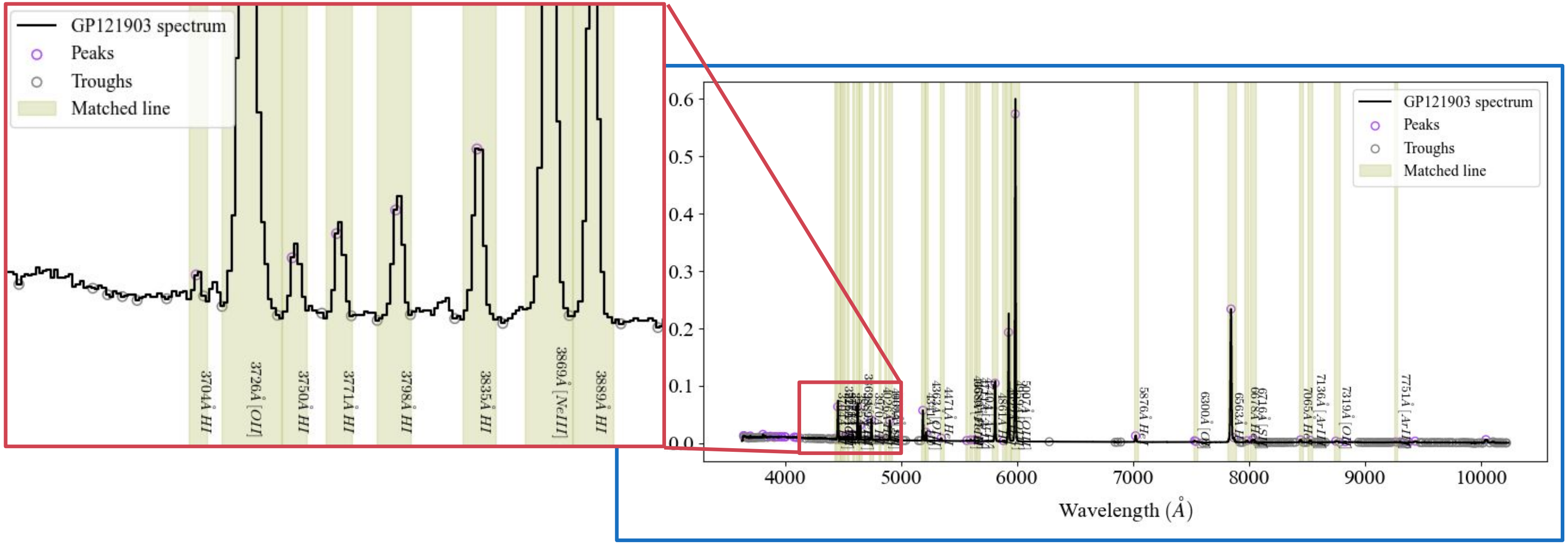
- ’leastsq’: Levenberg-Marquardt (default) → Most reliable
- ’least_squares’: Least-Squares minimization, using Trust Region Reflective method → Better for fittings with inequalities
- ’differential_evolution’: differential evolution
- ’brute’: brute force method
- ’basinhopping’: basinhopping
- ’ampgo’: Adaptive Memory Programming for Global Optimization
- ’nelder’: Nelder-Mead
- ’lbfgsb’: L-BFGS-B
- ’powell’: Powell
- ’cg’: Conjugate-Gradient
- ’newton’: Newton-CG
- ’cobyla’: Cobyla
- ’bfgs’: BFGS
- ’tnc’: Truncated Newton
- ’trust-ncg’: Newton-CG trust-region
- ’trust-exact’: nearly exact trust-region
- ’trust-krylov’: Newton GLTR trust-region
- ’trust-constr’: trust-region for constrained optimization
- ’dogleg’: Dog-leg trust-region
- ’slsqp’: Sequential Linear Squares Programming
- ’shgo’: Simplicial Homology Global Optimization
- ’dual_annealing’: Dual Annealing optimization

SciPy

- ’emcee’: Maximum likelihood via Monte-Carlo Markov Chain → emcee

Line detection

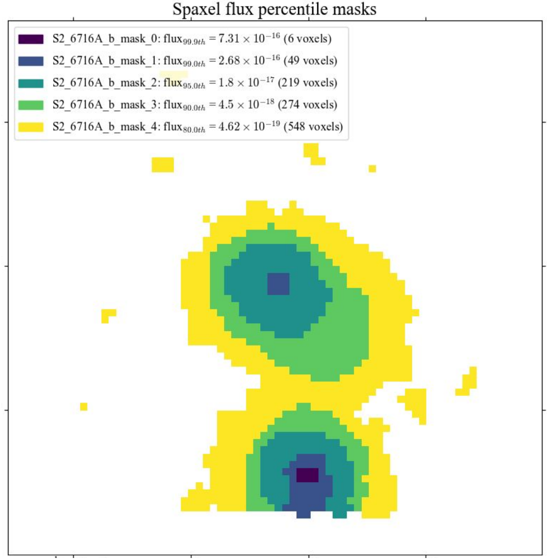
- For large datasets individual masks might not be an option. In this case you can use the *lime.Spectrum.match_line_mask()*

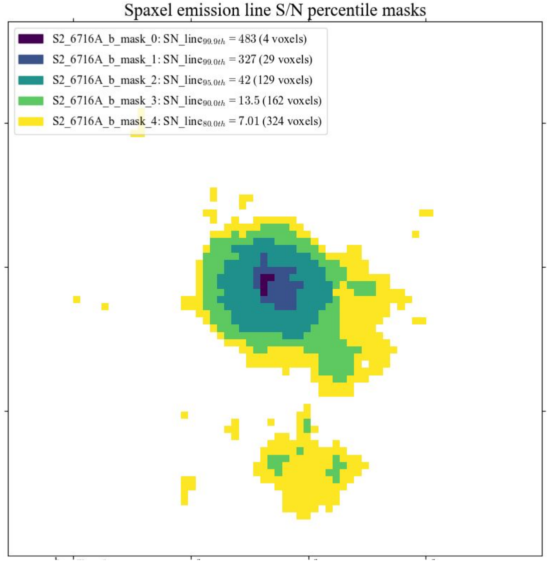


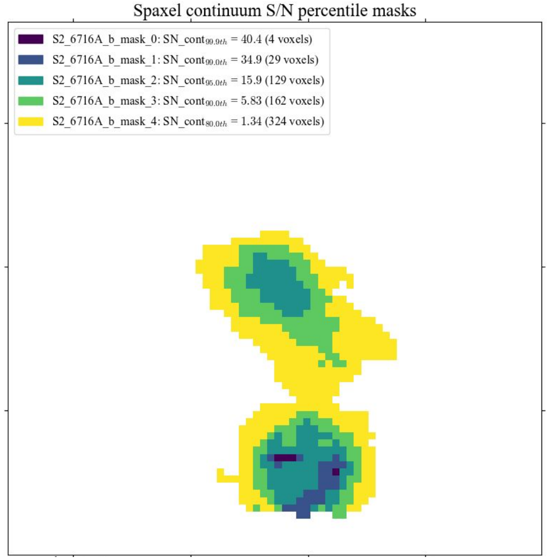
- The function fits a polynomial iteratively to normalize the continuum and determines the peaks location using the *find_lines_derivative()* function from SpecUtils by *Earl et al. (2022)*

Spatial masks (Tutorial 5)

- In IFU datasets, you may want to adjust your fittings according to the data spatial behaviour. You can use the *lime.spatial_mask_generator* to generate them:


Spaxel flux percentile masks

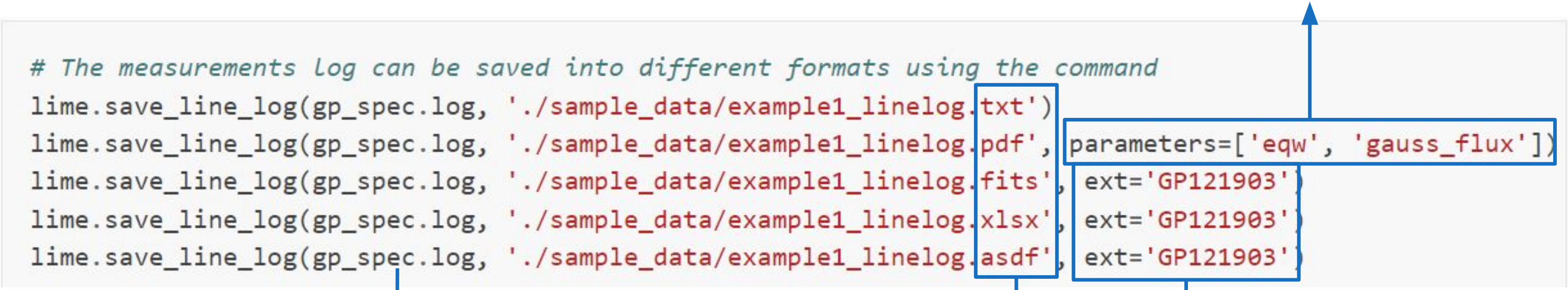

Spaxel emission line S/N percentile masks


Spaxel continuum S/N percentile masks

- For a manual mask spaxel adjustment you can use the *lime.CubeInspector()* class.

Saving/loading the log results

Restrict the log output measurements

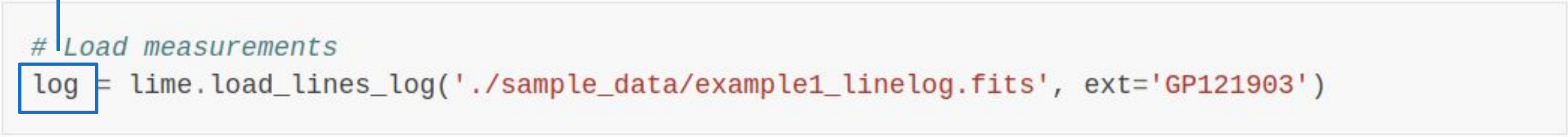


```
# The measurements log can be saved into different formats using the command
lime.save_line_log(gp_spec.log, './sample_data/example1_linelog.txt')
lime.save_line_log(gp_spec.log, './sample_data/example1_linelog.pdf', parameters=['eqw', 'gauss_flux'])
lime.save_line_log(gp_spec.log, './sample_data/example1_linelog.fits', ext='GP121903')
lime.save_line_log(gp_spec.log, './sample_data/example1_linelog.xlsx', ext='GP121903')
lime.save_line_log(gp_spec.log, './sample_data/example1_linelog.asdf', ext='GP121903')
```

Tables in *LiMe* are *Pandas* DataFrames

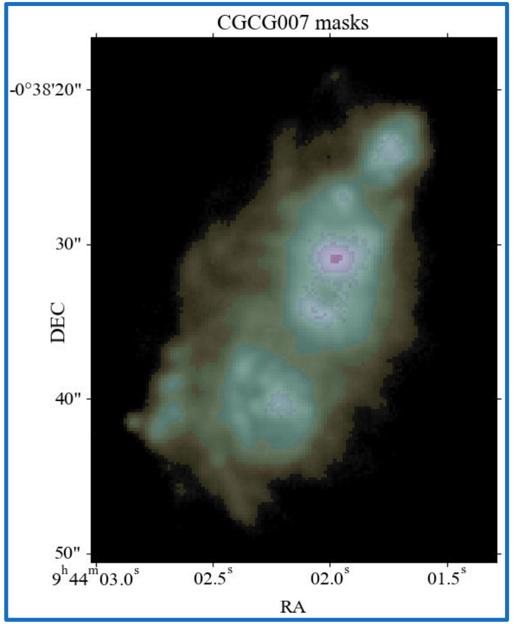
File type determined from the path extension

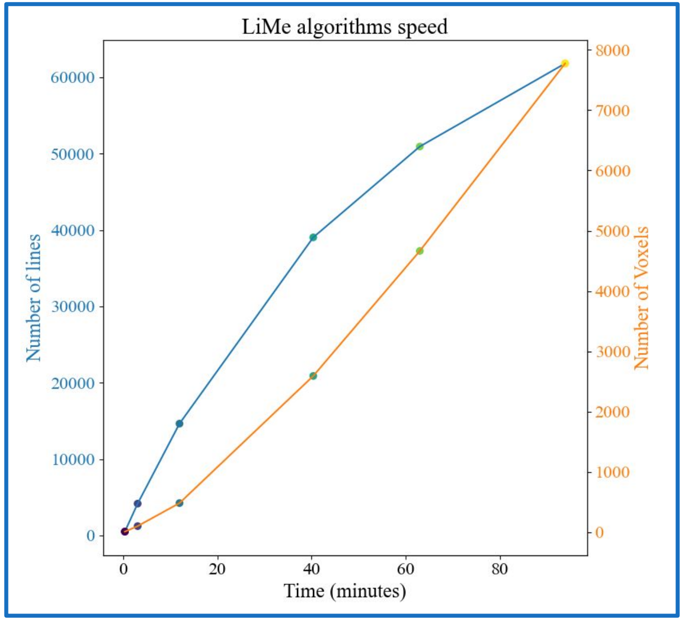
Label for “multi-page” files



```
# Load measurements
log = lime.load_lines_log('./sample_data/example1_linelog.fits', ext='GP121903')
```

Algorithm profiling



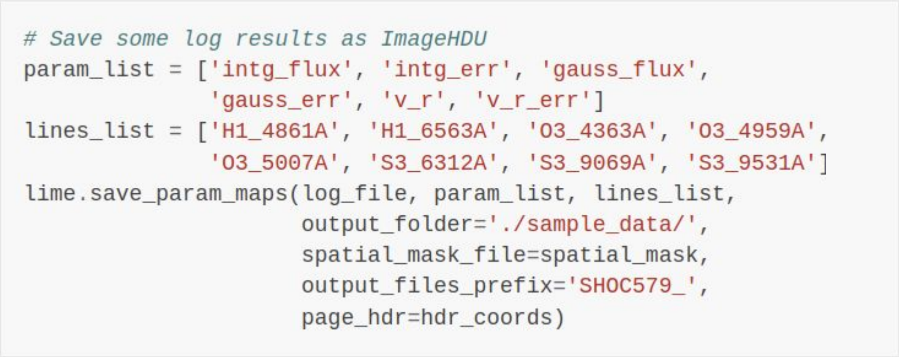


- This MUSE treatment consisted in 61800 lines and 7774 voxels.
- The mean line number goes from from 45 to 4 lines per voxel.
- Output (.fits) file size is 150MB, 19 kb per voxel

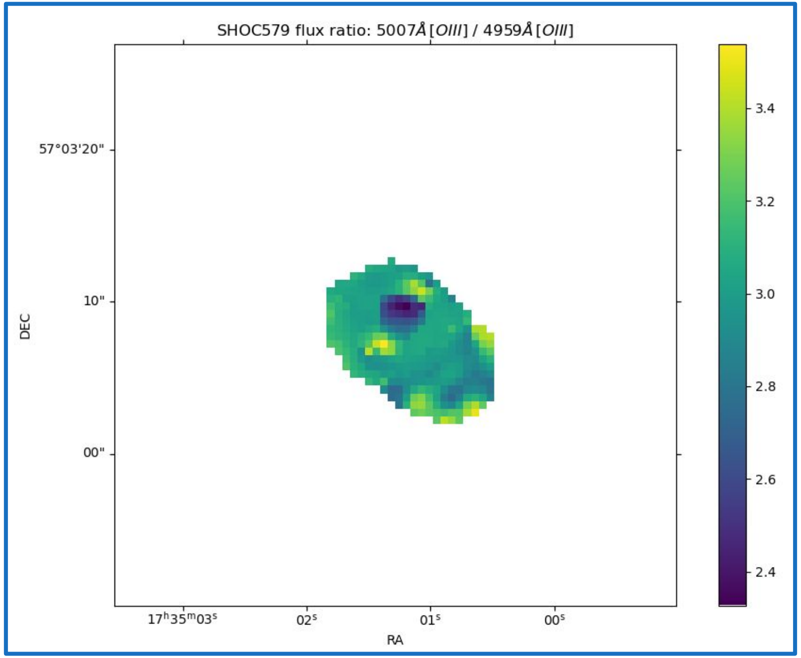
- The lines cube analysis took 93 minutes
- The averaged line measurement time is 0.08 seconds
- The averaged voxel measurement time is 1.2 seconds

Making the parameter maps (Tutorial 7)

- If the line logs are stored as a *.fits* file, you may use the function *lime.save_param_maps()* to export spaxel measurements as *.fits* images for a list of lines and parameters:

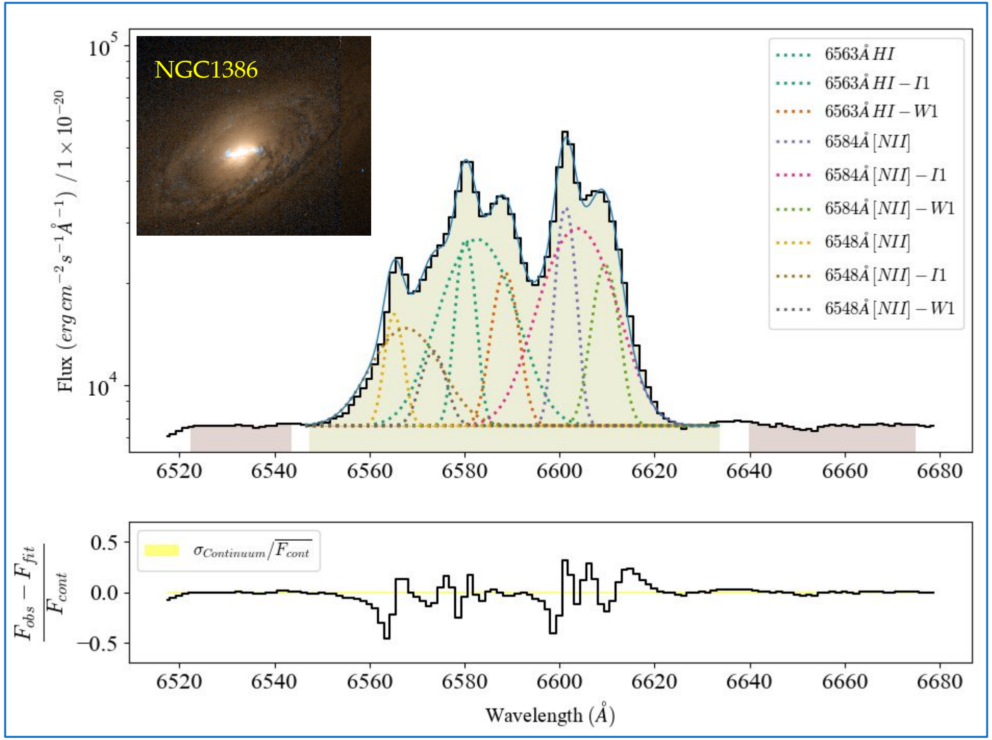


```
# Save some log results as ImageHDU
param_list = ['intg_flux', 'intg_err', 'gauss_flux',
              'gauss_err', 'v_r', 'v_r_err']
lines_list = ['H1_4861A', 'H1_6563A', 'O3_4363A', 'O3_4959A',
              'O3_5007A', 'S3_6312A', 'S3_9069A', 'S3_9531A']
lime.save_param_maps(log_file, param_list, lines_list,
                     output_folder='./sample_data/',
                     spatial_mask_file=spatial_mask,
                     output_files_prefix='SHOC579_',
                     page_hdr=hdr_coords)
```

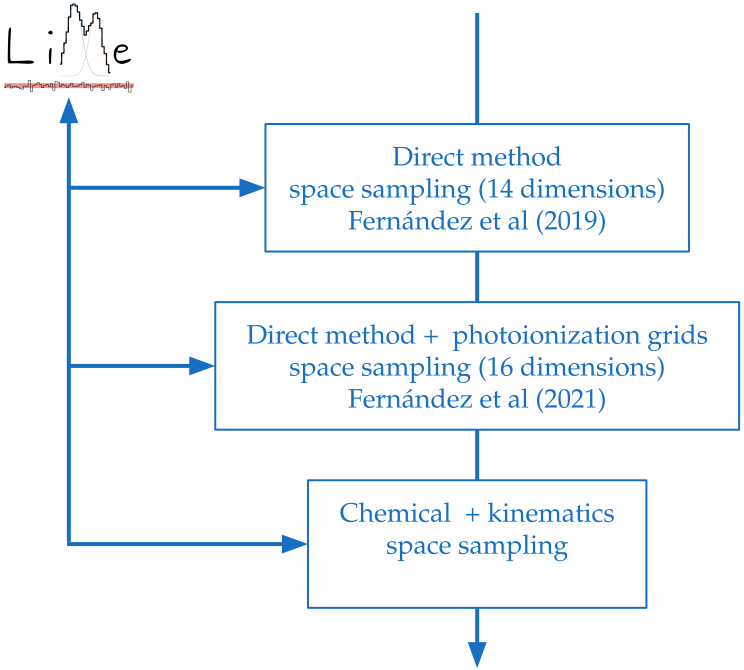


Final remarks

- As the number of dimensions increases beyond 10, traditional algorithms may not be efficient enough to explore the parameter space.



- We are working in new models and tools using neural network samplers to handle the situation. LiMe will provide a user friendly standard for the inputs.



Direct method
space sampling (14 dimensions)
Fernández et al (2019)

Direct method + photoionization grids
space sampling (16 dimensions)
Fernández et al (2021)

Chemical + kinematics
space sampling

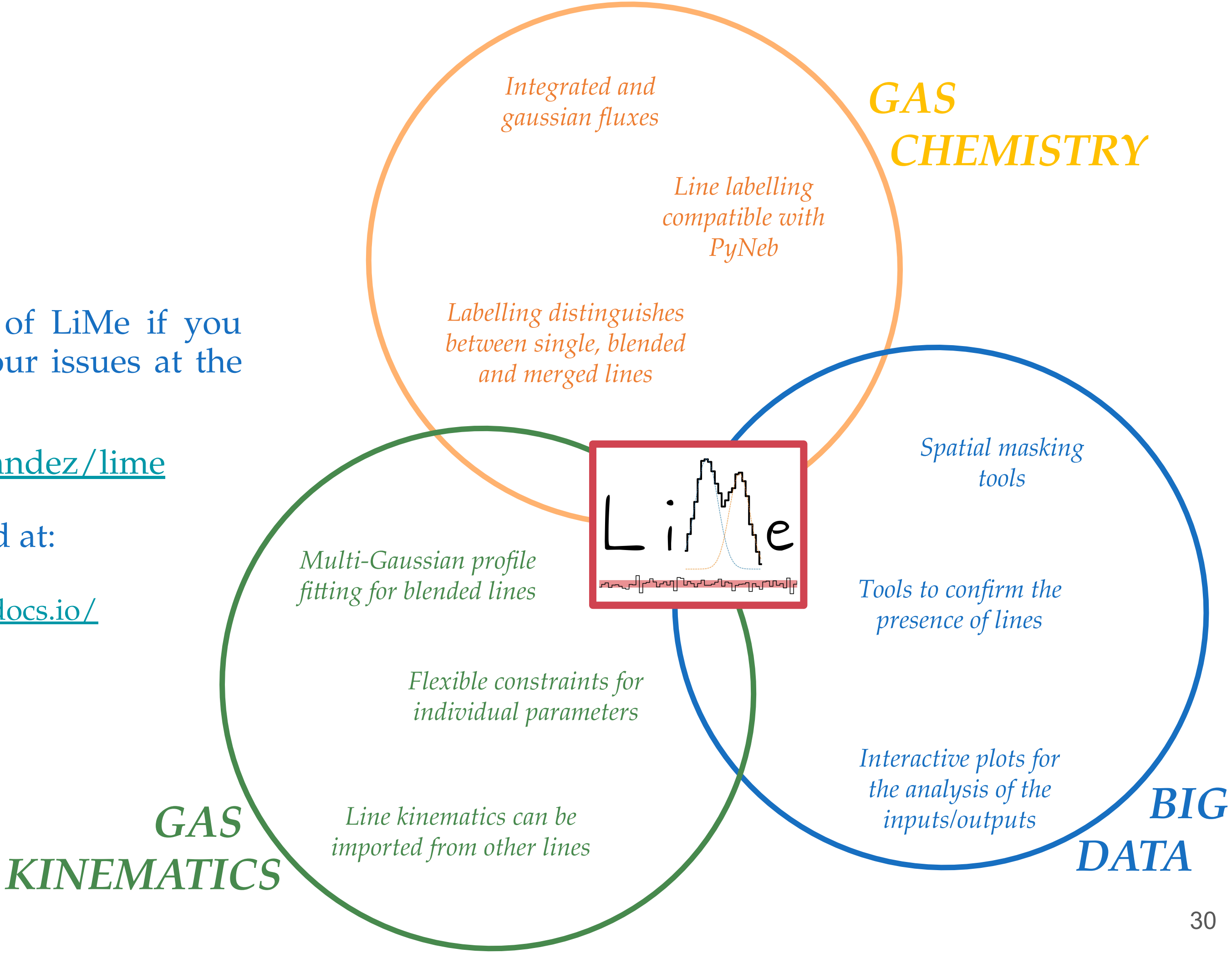
Conclusions

- This is the beta release of LiMe if you want to test please share your issues at the github:

  github.com/Vital-Fernandez/lime

- More details can be found at:

  https://lime-stable.readthedocs.io/



GAS CHEMISTRY

Integrated and gaussian fluxes

Line labelling compatible with PyNeb

Labelling distinguishes between single, blended and merged lines

GAS KINEMATICS

Multi-Gaussian profile fitting for blended lines

Flexible constraints for individual parameters

Line kinematics can be imported from other lines

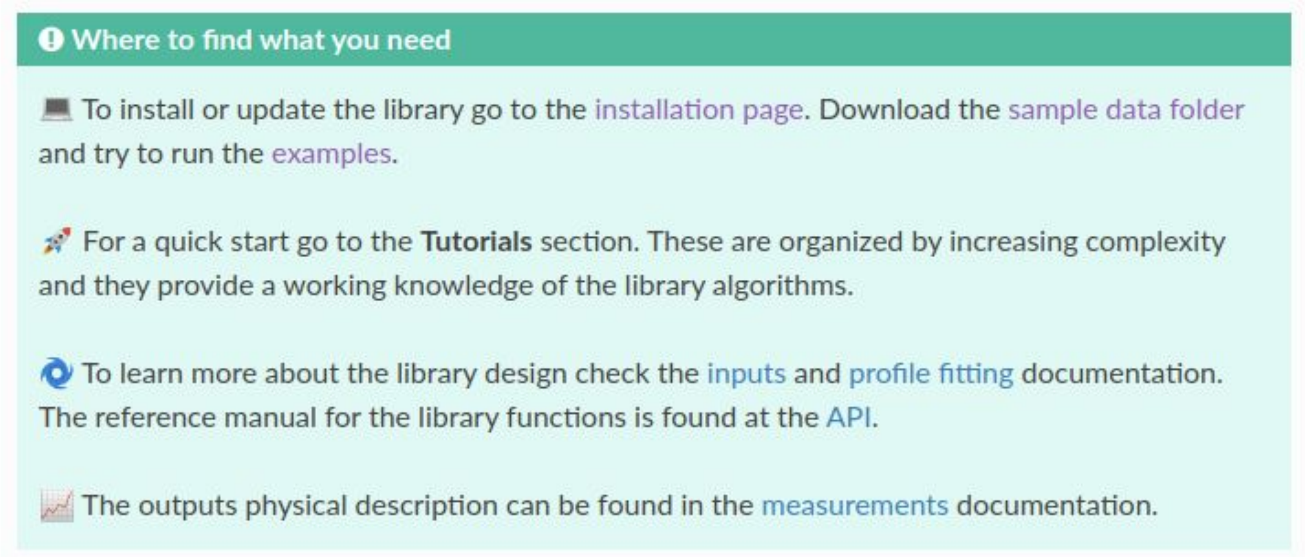
BIG DATA

Spatial masking tools

Tools to confirm the presence of lines

Interactive plots for the analysis of the inputs/outputs

Thank you for your time!



❗ Where to find what you need

💻 To install or update the library go to the installation page. Download the sample data folder and try to run the examples.

🚀 For a quick start go to the **Tutorials** section. These are organized by increasing complexity and they provide a working knowledge of the library algorithms.

🌀 To learn more about the library design check the inputs and profile fitting documentation. The reference manual for the library functions is found at the API.

📈 The outputs physical description can be found in the measurements documentation.

I am, gentlefolk,
yours faithfully
May 18, 2022