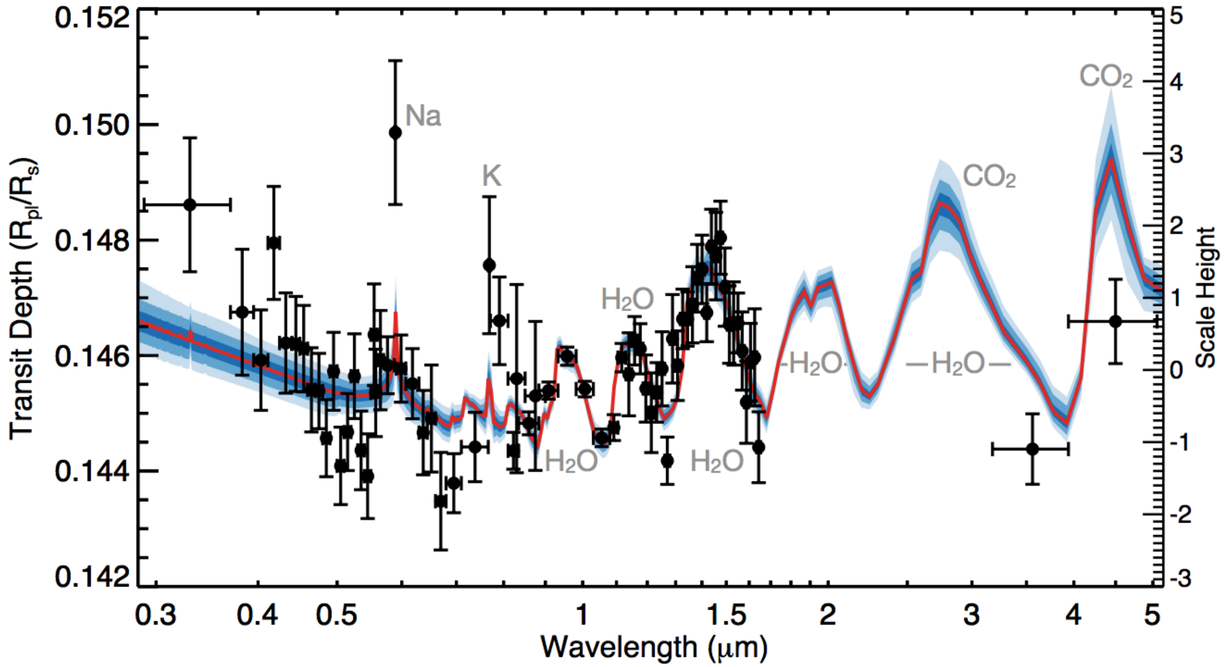
ATMOSPHERIC RETRIEVALS

- Atmospheric retrievals is an attempt to determine the physical parameters of an exoplanet atmosphere
- Given a set of initial guesses and a defined parameter space, retrieval algorithms will explore this parameter space to determine which physical parameters result in the best fit
- Important physical parameters include:
  - Mass of the planet
  - Radius of the planet
  - Temperature of the planet's atmosphere
  - Metallicity
  - C/O ratio
  - Cloud abundance



Wakeford et al., 2017

ATMOSPHERIC RETRIEVALS WITH PLATON AND AWS

The `atmospheric_retrievals/` subpackage in `exoctk` currently provides a way to perform atmospheric retrievals via PLATON

- New feature as of this workshop!
- PLATON – Planetary Atmospheric Transmission for Observer Noobs (https://platon.readthedocs.io)
- Support for multinested sampling (`multinest`) and MCMC (`emcee`) methods
- Users may perform the retrievals on a local machine or using Amazon Web Services (AWS) Elastic Computing (EC2) instances

Notable modules:

- `platon_wrapper.py` – wrapper module for PLATON
- `examples.py` – contains examples of how to use the code, with accompanying example data

Feature development expected to continue. Possible future work includes:

- More methods for storing, visualizing, and interacting with results
- Support for other retrieval algorithms such as CHIMERA (Caltech Inverse Modeling and Retrieval Algorithms)
- A web interface for performing retrievals, akin to other ExoCTK web tools

ATMOSPHERIC RETRIEVALS WITH PLATON AND AWS: WASP 19B EXAMPLE

Example: wasp 19b

```
from exoctk.atmospheric_retrievals.platon_wrapper import PlatonWrapper
from exoctk.atmospheric_retrievals.examples import get_example_data
from platon.constants import R_sun, R_jup, M_jup
import numpy as np
```

```
params = {
    'Rs': 1.000,  # Required
    'Mp': 1.069,  # Required
    'Rp': 1.392,  # Required
    'T': 2100.42,  # Required
    'logZ': 0,  # Optional
    'CO_ratio': 0.53,  # Optional
    'log_cloudtop_P': 4,  # Optional
    'log_scatt_factor': 0,  # Optional
    'scatt_slope': 4,  # Optional
    'error_multiple': 1,  # Optional
    'log_cloudtop_P': 4}  # Optional
```

ATMOSPHERIC RETRIEVALS WITH PLATON AND AWS: WASP 19B EXAMPLE

Example: wasp 19b

```
pw = PlatonWrapper()
pw.set_parameters(params)
```

```
pw.fit_info.add_gaussian_fit_param('Mp', 0.04*M_jup)
pw.fit_info.add_uniform_fit_param('Rp', 0.9*(1.392 * R_jup), 1.1*(1.392 * R_jup))
pw.fit_info.add_uniform_fit_param('T', 300, 3000)
pw.fit_info.add_uniform_fit_param("logZ", -1, 1)
pw.fit_info.add_uniform_fit_param("log_cloudtop_P", 0, 8)
```

```
pw.bins, pw.depths, pw.errors = get_example_data('wasp-19b')
```

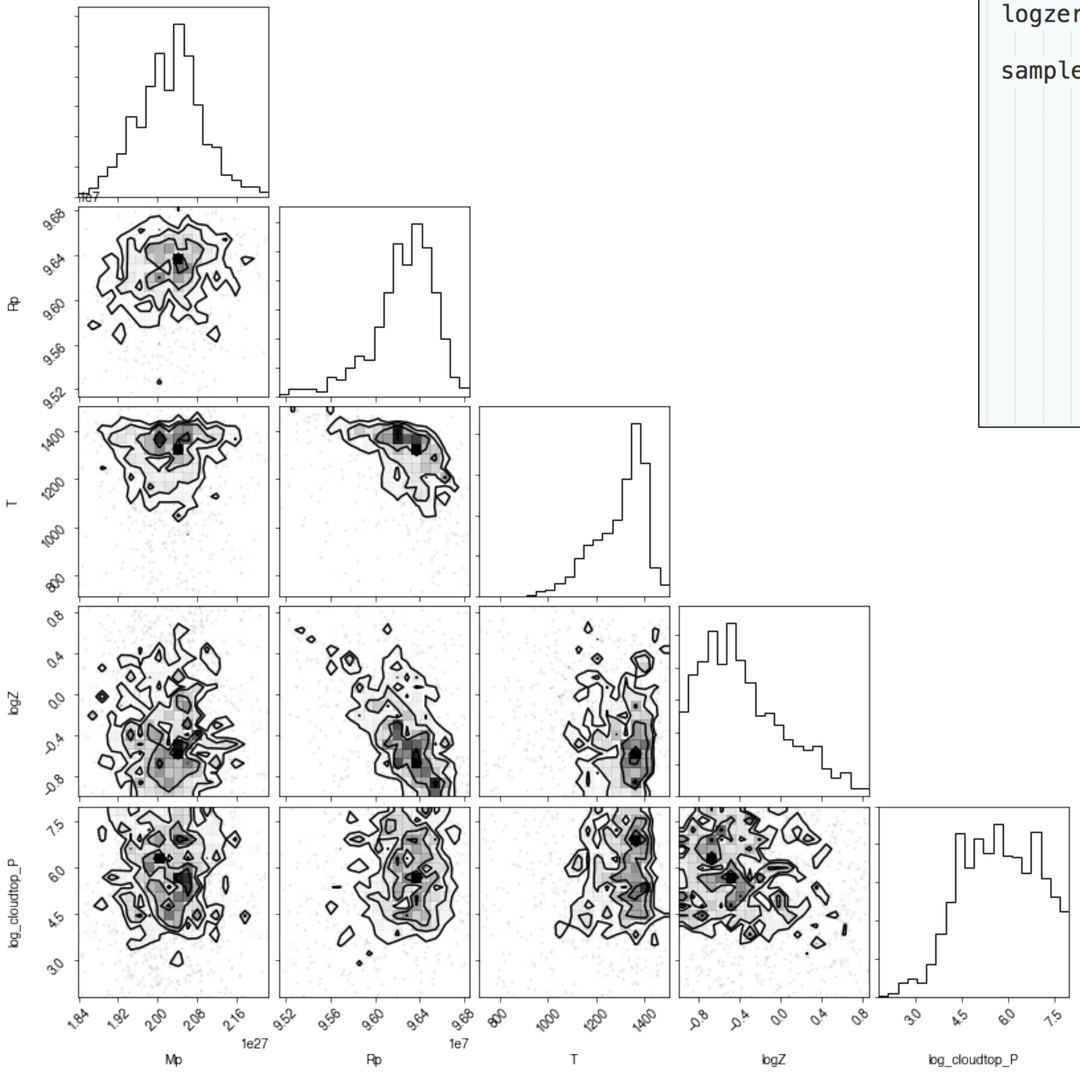
```
pw.retrieve('multinest')
pw.retrieve('emcee')
```

```
pw.save_results()
```

```
pw.make_plot()
```

ATMOSPHERIC RETRIEVALS WITH PLATON AND AWS: WASP 19B EXAMPLE

Example: wasp 19b



Corner plot

```
  niter: 681
  ncall: 3301
   logz: 12.773991835990161
logzerr: 0.18823412959796412
      h: 3.5432087545503155
samples: array([[ 8.19318796e+08,  1.34575447e+27,  1.04219425e+08, ...,
         1.86882126e+00,  4.99828816e+00,  5.43558853e-01],
       [ 8.45571358e+08,  1.46189276e+27,  1.07274439e+08, ...,
         2.04758809e+00,  2.86101833e+00,  5.33405567e-01],
       [ 8.19518172e+08,  1.46285492e+27,  1.07278561e+08, ...,
        -5.00664955e-01, -6.67623893e-01,  1.43144503e+00],
       ...,
       [ 8.24752719e+08,  1.69500736e+27,  9.80642732e+07, ...,
         2.39787715e+00,  1.67276391e+00,  1.21909720e+00],
       [ 8.05436626e+08,  1.48068159e+27,  9.40540810e+07, ...,
        -1.21912997e-01,  1.18414995e+00,  1.16629761e+00],
       [ 8.23303280e+08,  1.36251813e+27,  9.76609498e+07, ...,
        -8.33906515e-01,  2.80188501e+00,  9.65253374e-01]])
```

Results file

```
12/10/2019 12:43:29 PM INFO: User: bourque
12/10/2019 12:43:29 PM INFO: Python Version: 3.6.7 |Anaconda, Inc.| (default, Oct 23 2018, 19:16:44) [GCC 7.3.0]
12/10/2019 12:43:29 PM INFO: Python Executable Path: ~/miniconda3/envs/exoctk-3.6/bin/python
12/10/2019 12:43:29 PM INFO: Setting parameters: {'Rs': 1.0, 'Mp': 1.069, 'Rp': 1.392, 'T': 2100.42, 'logZ': 0, 'CO_
12/10/2019 12:43:29 PM INFO: Using data for wasp-19b
12/10/2019 12:43:29 PM INFO: Performing atmopsheric retrievals via multinest
12/10/2019 12:44:01 PM INFO: Failed to import gnumpy; not using GPU
12/10/2019 12:44:01 PM INFO: Iteration 1: ln_prob=-1.62e+03  Rs=0.98 R_sun  Mp=1.05 M_jup  Rp=1.48 R_jup  T=2984 K
12/10/2019 12:44:01 PM INFO: Iteration 2: ln_prob=-4.63e+02  Rs=1.00 R_sun  Mp=1.06 M_jup  Rp=1.49 R_jup  T= 854 K
12/10/2019 12:44:01 PM INFO: Iteration 3: ln_prob=-2.40e+03  Rs=1.02 R_sun  Mp=1.03 M_jup  Rp=1.50 R_jup  T=2144 K
12/10/2019 12:44:01 PM INFO: Iteration 4: ln_prob=-2.40e+02  Rs=0.97 R_sun  Mp=1.08 M_jup  Rp=1.27 R_jup  T=1913 K
12/10/2019 12:44:01 PM INFO: Iteration 5: ln_prob=-5.82e+02  Rs=0.97 R_sun  Mp=1.06 M_jup  Rp=1.26 R_jup  T=1440 K
12/10/2019 12:44:01 PM INFO: Iteration 6: ln_prob=-1.40e+02  Rs=0.99 R_sun  Mp=1.14 M_jup  Rp=1.39 R_jup  T= 992 K
12/10/2019 12:44:01 PM INFO: Iteration 7: ln_prob=-9.02e+02  Rs=0.98 R_sun  Mp=1.06 M_jup  Rp=1.37 R_jup  T=1854 K
12/10/2019 12:44:01 PM INFO: Iteration 8: ln_prob=-6.79e+02  Rs=1.01 R_sun  Mp=1.04 M_jup  Rp=1.39 R_jup  T=2838 K
12/10/2019 12:44:01 PM INFO: Iteration 9: ln_prob=-6.05e-01  Rs=0.98 R_sun  Mp=1.01 M_jup  Rp=1.31 R_jup  T=1117 K
12/10/2019 12:44:01 PM INFO: Iteration 10: ln_prob=-1.17e+03  Rs=0.97 R_sun  Mp=1.07 M_jup  Rp=1.39 R_jup  T=2467 K
```

Log file

Why use AWS?:

- Atmospheric retrievals can be computationally expensive
- Can be a burden to day-to-day research work
- Outsource computational effort to cloud-based machines for low costs
- For example, a simple AWS machine costs ~$0.023/hour

There will be an optional ~30 minute demo at the end of this workshop that shows how to perform retrievals using AWS!

```
pw.use_aws()
pw.retreive('multinest')
```

See also: *Enabling Fast Bayesian Exoplanet Atmospheric Retrievals using AWS* (poster 109.04; tomorrow 9:00-10:00)