Enabling Fast Bayesian Exoplanet Atmospheric Retrievals Using Amazon Web Services

Matthew Bourque, Kevin Stevenson, Joseph Filippazzo, and the ExoCTK team

AAS 235 HONOLULU, HAWAI'I 4 - 8 JANUARY 2020

Abstract

Bayesian atmospheric retrievals will be critical to the robust determination of exoplanetary atmospheric properties in the JWST era and beyond. However, some groups may be challenged by the need of programming expertise and/or computational resources required to perform such retrievals, thus limiting their ability to participate scientifically. The Exoplanet Characterization Toolkit (ExoCTK), which is an open-source data analysis software package and web application focused on the atmospheric characterization of exoplanets and time-series observation planning, aims to address these challenges by developing a module that performs atmospheric retrievals using GPU-enabled Amazon Web Services (AWS) EC2 instances. Here we present the design, usage, and results of this software.

1. ExoCTK

The Exoplanet Characterization Toolkit (ExoCTK) is a python-based, open-source software package and web application focused on the time-series observation planning and data analysis of exoplanets. ExoCTK is available on the web at https://exoctk.stsci.edu and on GitHub at https://github.com/ExoCTK/exoctk/.

2. Atmospheric Retrievals

Atmospheric Retrievals are algorithms used to determine the physical parameters of an exoplanetary atmosphere. The `atmospheric_retrievals` subpackage within the `exoctk` repository contains methods and tools for performing retrievals with PLATON (PLanetary Atmospheric Transmission for Observer Noobs)[1]. Users can choose to performing retrievals on a local machine, or on virtual machines located in the cloud on the Amazon Web Services (AWS) Elastic Computing (EC2) platform by properly configuring an EC2 instance and invoking the `use_aws()` method. A simple example using WASP-19b is provided below.

```
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
```

```
# Define initial guesses
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
```

```
# Instantiate PlatonWrapper object and set parameters
pw = PlatonWrapper()
pw.set_parameters(params)
```

```
# Set priors
T_guess = 0.04*M_jup
R_guess = 1.392 * R_jup
pw.fit_info.add_gaussian_fit_param('Mp', T_guess)
pw.fit_info.add_uniform_fit_param('Rp', 0.9*R_guess, 1.1*(R_guess)
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)
```

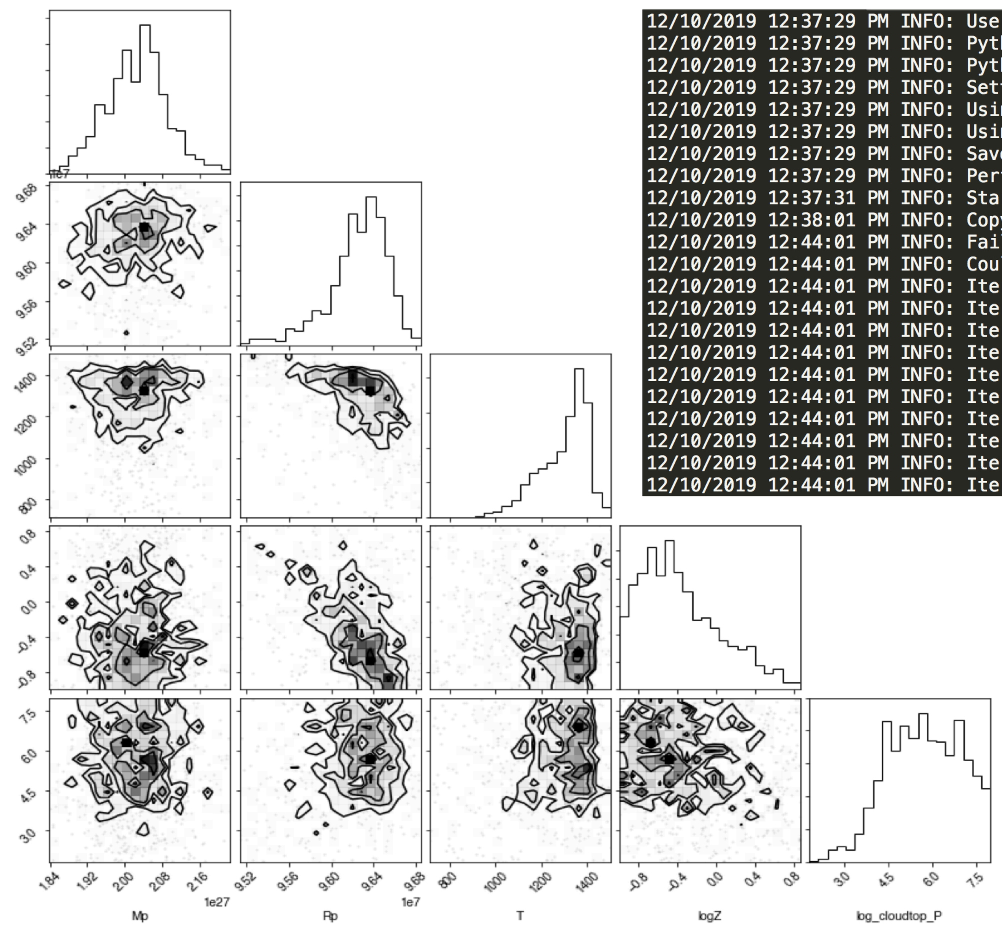
```
# Define wavelength bins, transit depths, and their uncertainties
pw.bins, pw.depths, pw.errors = get_example_data('wasp-19b')
```

```
# Use AWS EC2 instance for processing
pw.use_aws('</path/to/ssh_key>', '<ec2_id>')
```

```
pw.retrieve('multinest')  # Multinested sampling
pw.retrieve('emcee')  # MCMC
```

```
pw.results  # Object containing results
pw.save_results()  # Save results to file
pw.make_plot()  # Create and save corner plot
```

3. Outputs

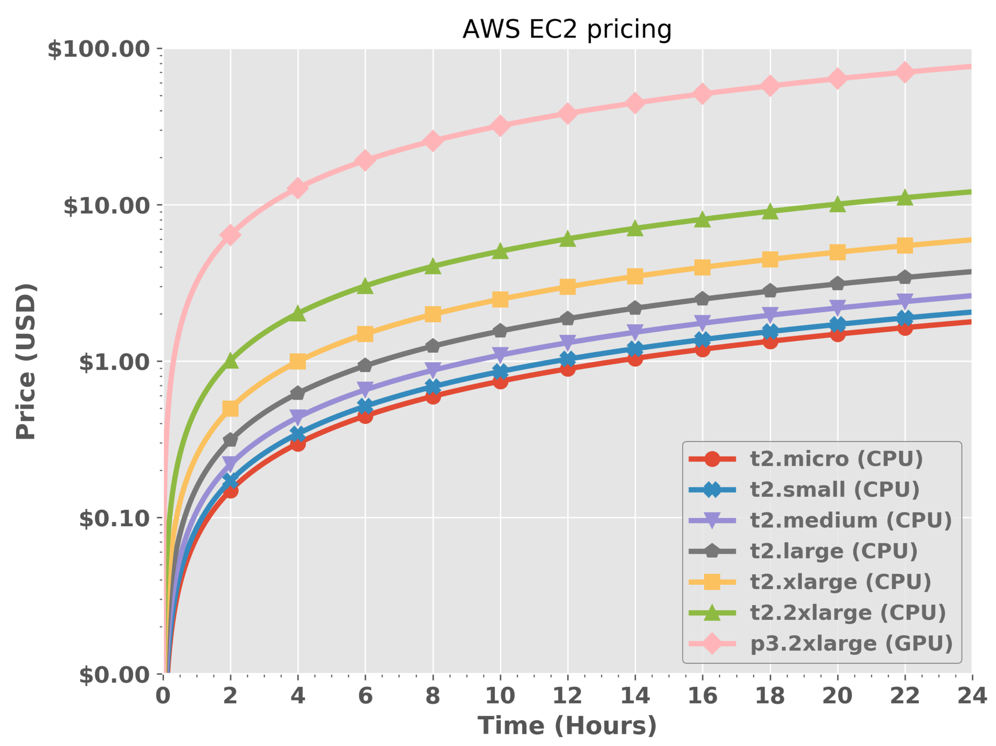


```
12/10/2019 12:37:29 PM INFO: User: bourque
12/10/2019 12:37:29 PM INFO: Python Version: 3.6.7 |Anaconda, Inc.| (default, Oct 23 2018, 19:16:44) [GCC 7.3.0]
12/10/2019 12:37:29 PM INFO: Python Executable Path: /home/bourque/miniconda3/envs/exoctk-3.6/bin/python
12/10/2019 12:37:29 PM INFO: Setting parameters: {'Rs': 1.0, 'Mp': 1.069, 'Rp': 1.392, 'T': 2100.42, 'logZ': 0, 'CO_
12/10/2019 12:37:29 PM INFO: Using data for wasp-19b
12/10/2019 12:37:29 PM INFO: Using AWS for processing
12/10/2019 12:37:29 PM INFO: Saved PlatonWrapper object to pw.obj
12/10/2019 12:37:29 PM INFO: Performing atmopsheric retrievals via multinest
12/10/2019 12:37:31 PM INFO: Started EC2 instance i-04cbc730a05ccd39f
12/10/2019 12:38:01 PM INFO: Copying pw.obj to EC2
12/10/2019 12:44:01 PM INFO: Failed to import gnumpy; not using GPU
12/10/2019 12:44:01 PM INFO: Could not import batman. Functionality may be limited.
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: Iteratio
12/10/2019 12:44:01 PM INFO: Iteratio
12/10/2019 12:44:01 PM INFO: Iteratio
12/10/2019 12:44:01 PM INFO: Iteratio
```

```
   niter: 689
   ncall: 3986
    logz: 13.186056861810265
 logzerr: 0.18480503182703512
       h: 3.415289978859146
 samples: array([[ 7.93283233e+08,  1.20539488e+27,  1.05347600e+08, ...,
        -3.85500310e-01,  4.00164895e+00,  5.96837861e-01],
       [ 8.18204480e+08,  1.42452815e+27,  1.01389527e+08, ...,
         1.04445666e+00, -9.73327495e-02,  5.97957645e-01],
       [ 7.99259458e+08,  1.29297630e+27,  1.04594722e+08, ...,
        -3.30562260e-01,  4.89164386e+00,  8.55853100e-01],
       ...,
       [ 8.11037426e+08,  1.34339841e+27,  9.38804579e+07, ...,
         3.40720798e-02,  2.12452042e-02,  5.94946287e-01],
       [ 8.13470192e+08,  1.53736333e+27,  9.47623922e+07, ...,
         1.51963804e+00,  9.81672649e-01,  8.55397925e-01],
       [ 8.28819268e+08,  1.38056170e+27,  9.87897439e+07, ...,
         2.85851060e+00,  4.28623929e-01,  1.00282809e+00]])
```

The output products consist of a corner plot *(left)* describing the results of the fit, a log file *(top right)* that describes the execution of the software, and a results file *(bottom right)* containing the results of the fit.

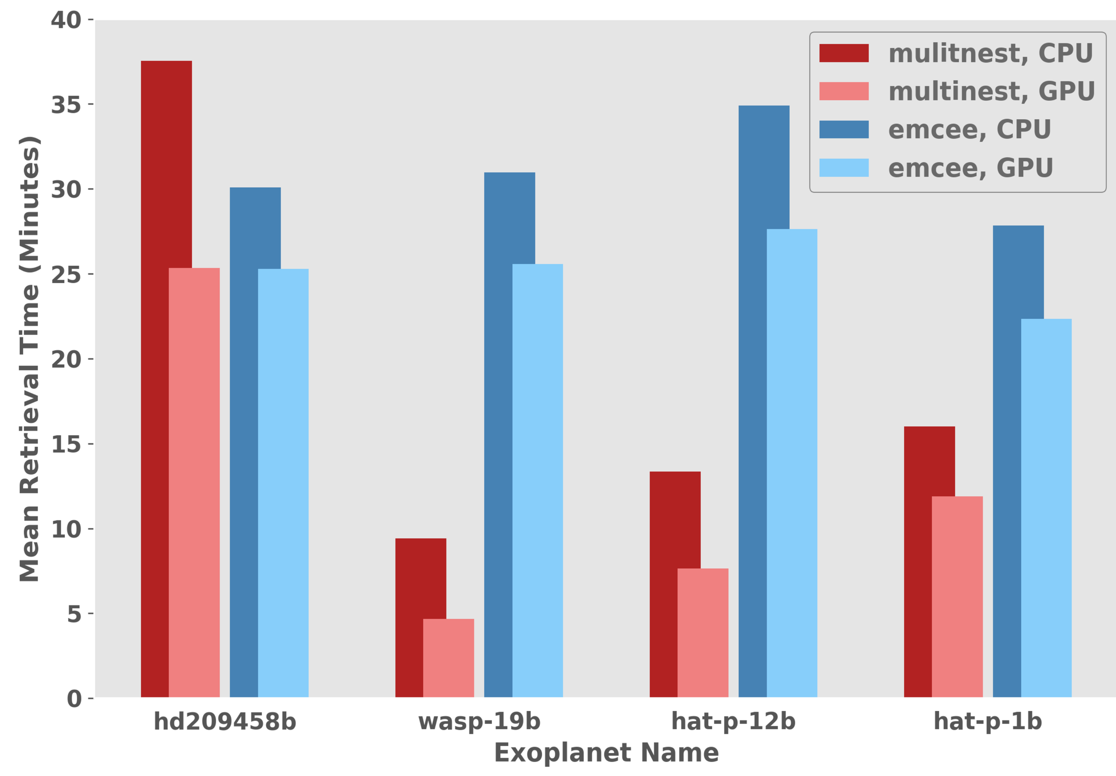
4. Amazon Web Services



	Machine 1	Machine 2
Amazon Machine Image ID	ami-098bb5d92c8886ca1	ami-098bb5d92c8886ca1
Operating System	RHEL v8.0.0	RHEL v8.0.0
EC2 Instance Type	t2.small	p3.2xlarge
Processor	1xCPU	8xGPU
Memory	2 GB	61 GB
Storage	20 GB SSD	20 GB SSD
Region	N. Virginia	N. Virginia
Hourly Cost	$0.023	$3.060

(Top Left) The total cost of running various AWS EC2 instances, as a function of time.

We conducted trial runs of the atmospheric retrieval software with transmission spectra from four exoplanets[2], using two different AWS EC2 instances (one CPU-enabled, and one GPU-enabled). The table in the *top right* provides the specifications of these instances, and the average results from five trial runs are given in the plot on the *bottom right*.



5. Future Work

- More methods for storing, visualizing, and interacting with results
- A web interface for performing retrievals, akin to other ExoCTK web tools
- Support for other retrieval algorithms, namely CHIMERA (Caltech Inverse Modeling and Retrieval Algorithms)
- Easier AWS EC2 configuration process via build scripts

Acknowledgements and References:

Support for this project was provided by the STScI Director's Research Fund and the 2019 STScI Data Science Innovation Initiative.

1. For more information on PLATON, see https://platon.readthedocs.io/.
2. Transmission spectra were acquired from ExoMAST, available at https://exo.mast.stsci.edu/.