



# Enabling Fast Bayesian Exoplanet Atmospheric Retrievals Using Amazon Web Services

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



235



## 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, data analysis software package and web application focused primarily on the atmospheric characterization of exoplanets and time-series observation planning. 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)<sup>1</sup>. Users can choose to performing retrievals on a local machine, or using Amazon Web Services (AWS) Elastic Computing (EC2) instances by properly configuring an EC2 instance and invoking the `use_aws()` method. A simple example using WASP-19b is provided below.

```
from exoctx.atmospheric_retrievals.platon_wrapper import PlatonWrapper
from exoctx.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)
```

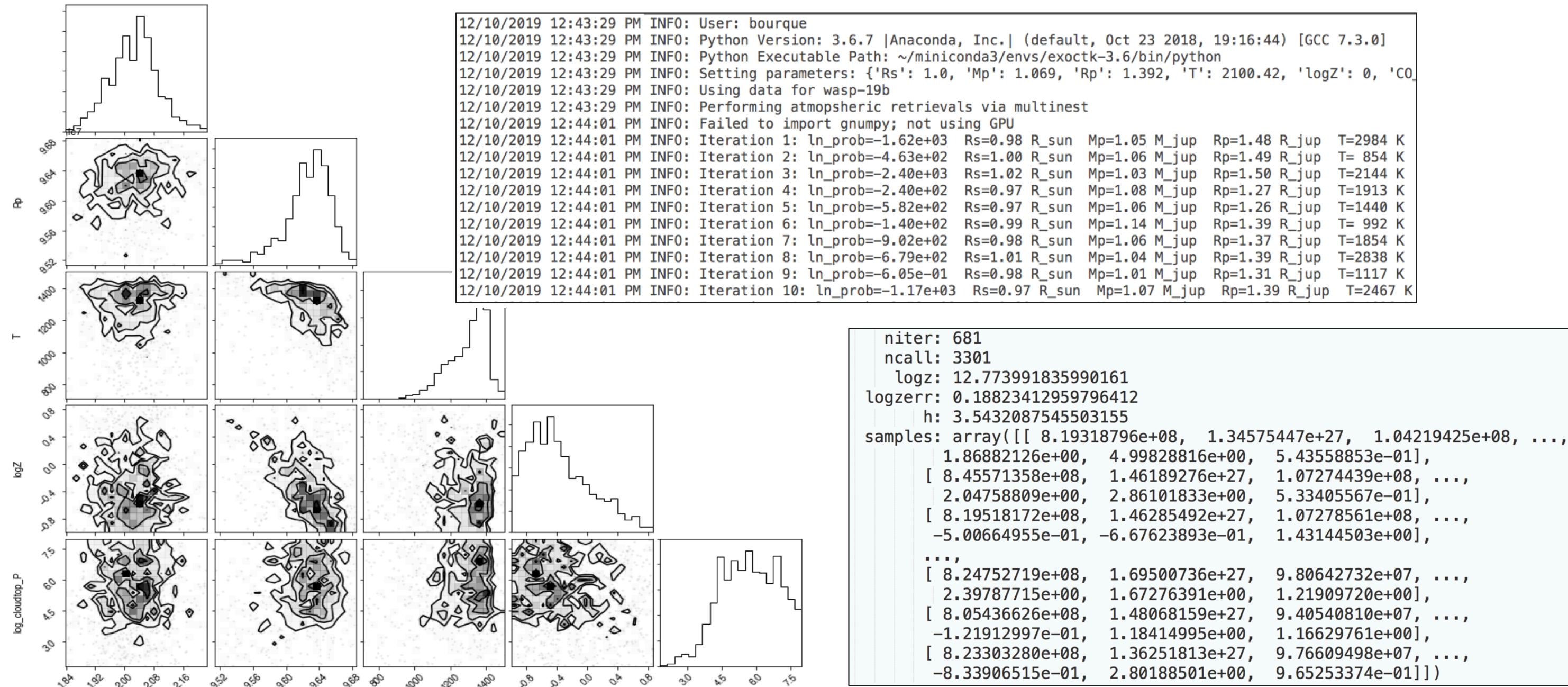
```
# Define wavelength bins, transit depths, and their uncertainties
pw.bins, pw.depths, pw.errors = get_example_data('wasp-19b')
```

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

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

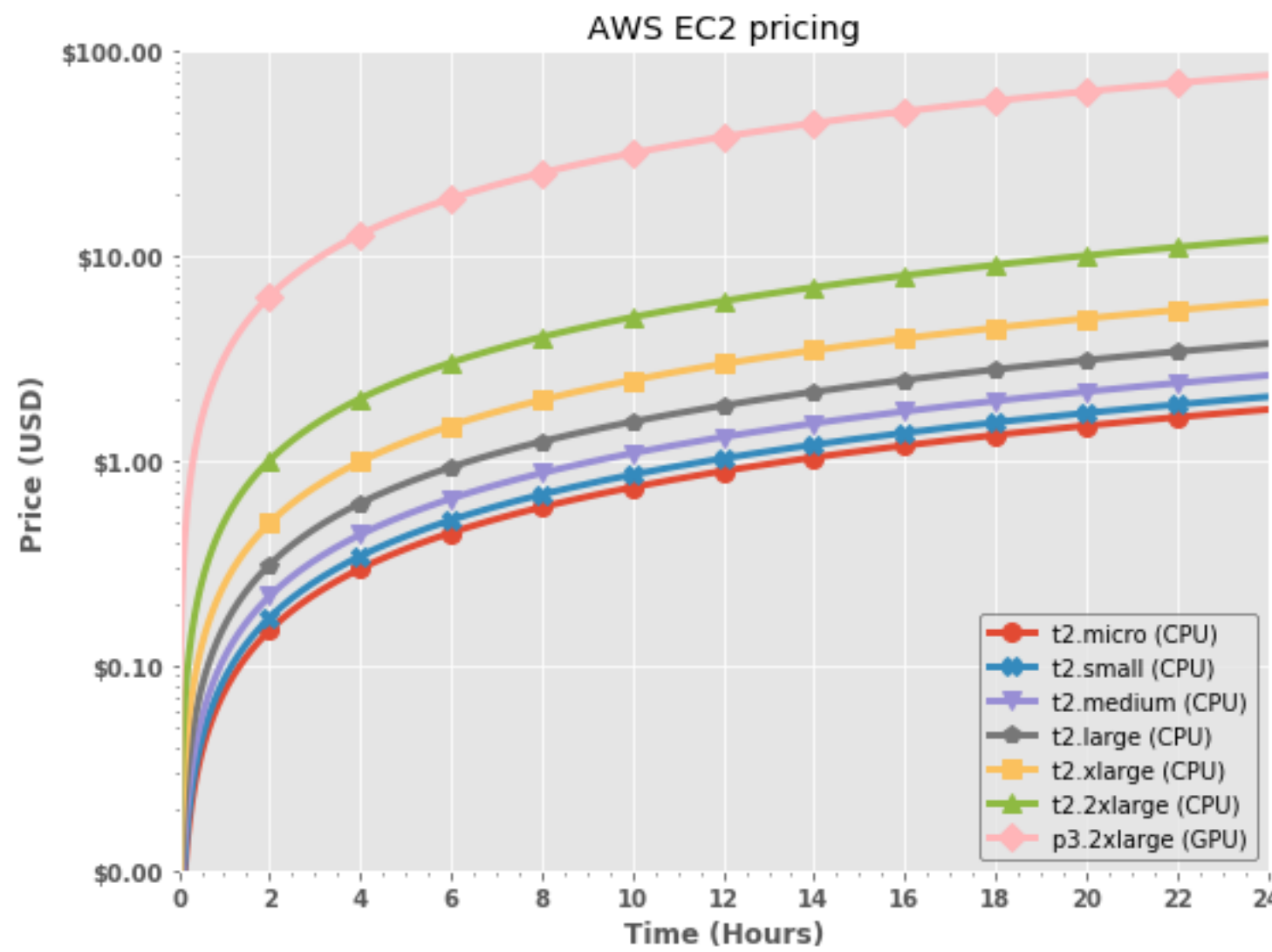
```
pw.results # Object containing results
pw.save_results() # Save results to file
pw.make_plot() # Create and save corner plot
```

## 3. Outputs



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.

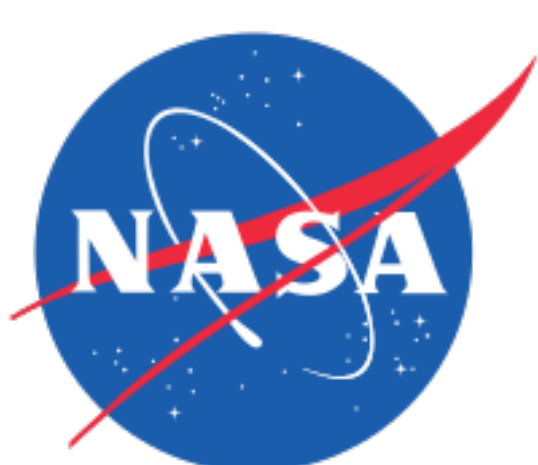
## 4. Amazon Web Services



	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

## 5. Future Work

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



### Acknowledgements and References:

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

1. For more information on PLATON, see <https://platon.readthedocs.io/>.