First VPLanet Developers Workshop

Lesson 2
Best Practices and Pro Tips

The VPLanet Lexicon

VPLanet is an executable that accepts a single “primary input file”

The primary input file contains “options” and their “arguments”

It also contains a list of “body files”, which contain their own options

The primary input file and body files are known as “infiles”

Body files include an option to specify the modules to be applied

They also contain an option for “outputs” that are printed to “forward files”

VPLanet can also write a “log file” that contains the initial and final conditions of a simulation

More information is available in the Quick Start guide

Command Line Options

-v: verbose. VPLanet will print everything to the screen,
this overrides iVerbose in the in files

-q: quiet. VPLanet will print nothing to the screen,
this overrifes iVerbose in the in files

-h: short help. Display brief info about options and outputs

-H: long help. Display formatted help, with long descriptions
(we use this output to generate online documentation)

Example Primary Input File

```
# General Options
sSystemName   solarsystem      # System Name
iVerbose      5                # Verbosity level
bOverwrite    1                # Allow file overwrites?
saBodyFiles   sun.in $         # List of all bodies files for the system
              venus.in         # The $ tells VPLanet to continue to the next line

# Input/Output Units
sUnitMass     solar            # Options: gram, kg, Earth, Neptune, Jupiter, solar
sUnitLength   AU               # Options: cm, m, km, Earth, Jupiter, solar, AU
sUnitTime     YEARS            # Options: sec, day, year, Myr, Gyr
sUnitAngle    d                # Options: deg, rad

# Input/Output
bDoLog        1          # Write a log file?
iDigits       6          # Maximum number of digits to right of decimal

# Evolution Parameters
bDoForward    1          # Perform a forward evolution?
bVarDt        1          # Use variable timestepping?
dEta          0.01       # Coefficient for variable timestepping
dStopTime     4.6e9      # Stop time for evolution
dOutputTime   1e6        # Output interval for forward file
```

Example Primary Input File

```
# General Options
```

All text after a # is considered a comment

All white space is ignored

Example Primary Input File

```
# General Options
sSystemName   solarsystem      # System Name
iVerbose      5                # Verbosity level
bOverwrite    1                # Allow file overwrites?
saBodyFiles   sun.in $         # List of all bodies files for the system
              venus.in         # The $ tells VPLanet to continue to the next line
```

- The name of an option must be the first string on a line
- Option names are unique, and exact spelling and case are required
- The leading lower cases letter(s) denote the type of argument
  b = Boolean (0 or 1)
  i = integer
  d = double precision
  s = string

- If one of those letters is followed by an "a" it means array and multiple arguments are permitted
- The $ means continue to the next line to obtain the next argument
- The $ and # are the only special characters in the infiles.

Example Primary Input File

- VPLanet allows you to input arguments in the most convenient units for your simulation
- If you include these units in the primary input file, the arguments propagate to the body files

```
# Input/Output Units
sUnitMass       solar          # Options: gram, kg, Earth, Neptune, Jupiter, solar
sUnitLength     AU              # Options: cm, m, km, Earth, Jupiter, solar, AU
sUnitTime       YEARS          # Options: sec, day, year, Myr, Gyr
sUnitAngle      d              # Options: deg, rad
```

- If you add these options in a body file, they supersede the arguments in the primary input file

Example Primary Input File

- You can write a log file that contains all the initial and final conditions
- The units of the log file are system units, which are SI (mks)
- We recommend writing a log file for each simulation

```
# Input/Output
bDoLog      1           # Write a log file?
iDigits     6           # Maximum number of digits to right of decimal
```

- You can also specify the precision of the output: 0 to 16 decimal places

Example Primary Input File

- Finally, you can specify the details of your simulation
- Here, we run a forward simulation (backwards is also available)
- We use variable timestepping (highly recommended)
- dEta is a coefficient (<1) that controls accuracy
  - Smaller dEta means more accuracy and a slower simulation
- The simulation will run for 4.6 Gyr (sUnitTime = years)
- The output interval will be 1 million years
- Note that you can not simulate evolution in time, and then the log file will contain all the conditions implied by your body files

```
# Evolution Parameters
bDoForward   1          # Perform a forward evolution?
bVarDt       1          # Use variable timestepping?
dEta         0.01       # Coefficient for variable timestepping
dStopTime    4.6e9      # Stop time for evolution
dOutputTime  1e6        # Output interval for forward file
```

Example Body File

```
# Planet a parameters
sName           venus                 # Body's name
saModules     atmesc   eqtide   # Modules to apply, exact spelling required


# Physical Properties
dMass          -0.815                  # Mass, negative -> Earth masses
dRadius        -0.9499                 # Radius, negative -> Earth radii
dRotPeriod  -243.                      # Rotation period, negative -> days
dObliquity    180.                     # Retrograde rotation
dRadGyra    0.5                       # Radius of gyration (moment of inertia constant)

# Orbital Properties
dSemi         -0.723                  # Semi-major axis, negative -> AU
dEcc          0.006772                # Eccentricity

# Output
saOutputOrder Time -SurfWaterMass -RGLimit -OxygenMantleMass
```

Example Body File

```
# Planet a parameters
sName          venus                # Body's name
saModules      atmesc   eqtide      # Modules to apply, exact spelling required
```

- All bodies must have a unique name
- saModules is critical! The arguments are all the physical modules to apply

Example Body File

- Next come option that describe the body, including module-specific options
- Option names are intended to be self-explanatory
- You can always learn more about options by checking the online documentation, or running VPLanet with the -h or -H flags

```
# Physical Properties
dMass          -0.815              # Mass, negative -> Earth masses
dRadius        -0.9499             # Radius, negative -> Earth radii
dRotPeriod     -243.               # Rotation period, negative -> days
dObliquity     180.                # Retrograde rotation
dRadGyra       0.5                 # Radius of gyration (moment of inertia constant)
```

- Note that some arguments are negative, suggesting unphysical values!
- Actually, VPLanet allows negative signs for positive-definite parameters to force a specific unit
- These “custom units” are generally typical for a star-terrestrial planet system
- The custom units are also documented online and with the help flags

Example Body File

- saOutputOrder tells VPLanet what to write in the forward files
- The order of the outputs is arbitrary
  - But we recommend including Time!
- Arguments only need to be unique
- Note the negative signs again; these force custom units
- If no negative sign is prepended, then the output units are those selected from the sUnit options, e.g. sUnitMass
- You can omit this option, and then no forward file is written

```
# Output
saOutputOrder Time -SurfWaterMass -RGLimit -OxygenMantleMass
```

Getting the Most Out of the Onboard Help

Looking for the name of that options/output? -h + grep is your friend!

> vplanet -h | grep XUV

[-]FXUV -- XUV flux. [Negative = W/m^2]
[-]LXUVFlare -- XUV Luminosity from flares. [Negative = LSUN]
LXUVFrac -- Fraction of luminosity in XUV.
[-]LXUVStellar -- Base X-ray/XUV Luminosity. [Negative = LSUN]
[-]LXUVTot -- Total XUV Luminosity. [Negative = LSUN]
[-]PresXUV -- Pressure at base of thermosphere. [Negative = Pa]
[-]RadXUV -- XUV radius separating hydro. dyn. escape and equilibrium. [Negative = Rearth]
[-]RRCriticalFlux -- Critical XUV Flux that separates RR and energy-limited escape. [Negative = W/m^2]

Repo Overview

VPLanet: The Virtual Planet Simulator

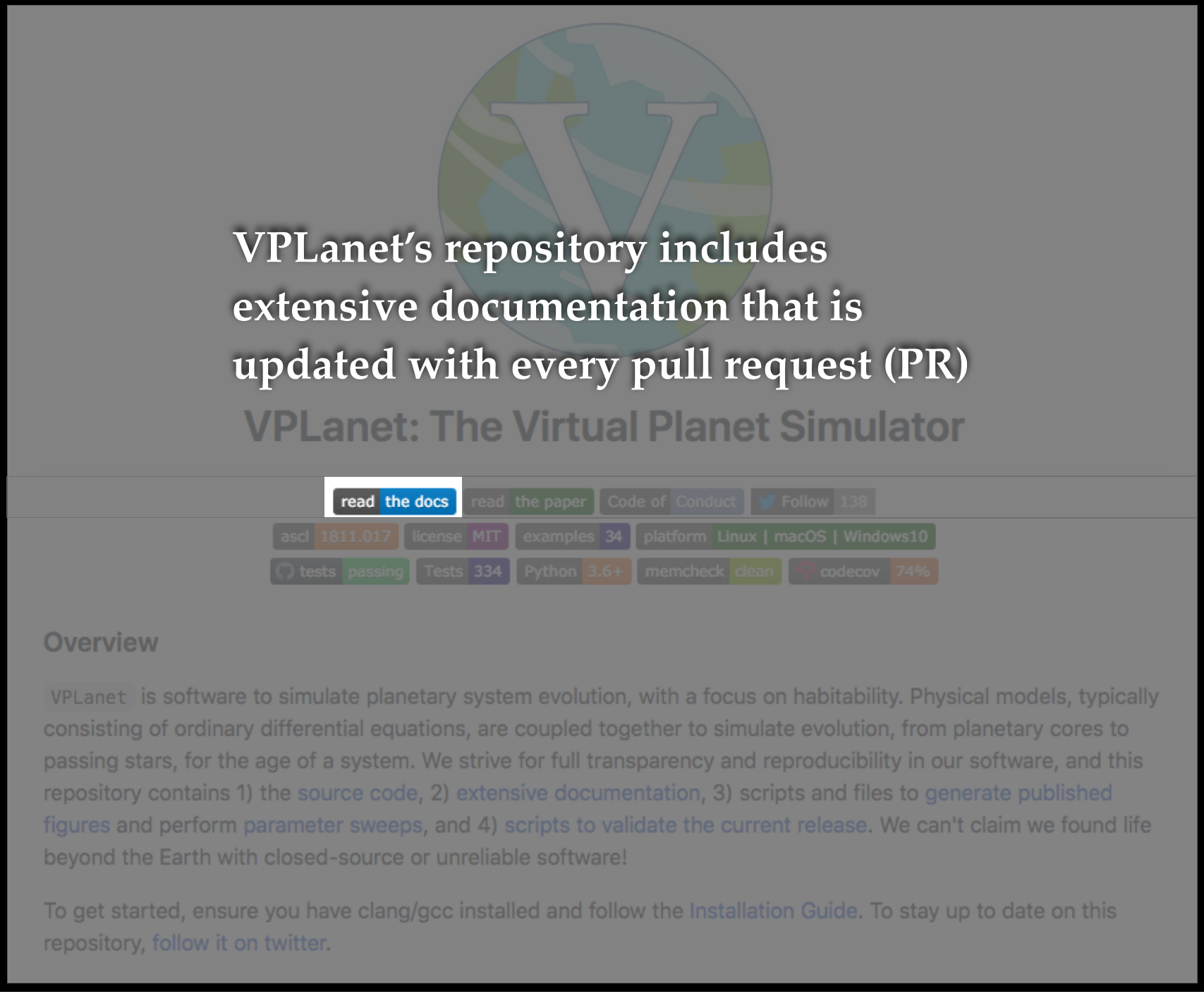
Overview

`VPLanet` is software to simulate planetary system evolution, with a focus on habitability. Physical models, typically consisting of ordinary differential equations, are coupled together to simulate evolution, from planetary cores to passing stars, for the age of a system. We strive for full transparency and reproducibility in our software, and this repository contains 1) the source code, 2) extensive documentation, 3) scripts and files to generate published figures and perform parameter sweeps, and 4) scripts to validate the current release. We can't claim we found life beyond the Earth with closed-source or unreliable software!

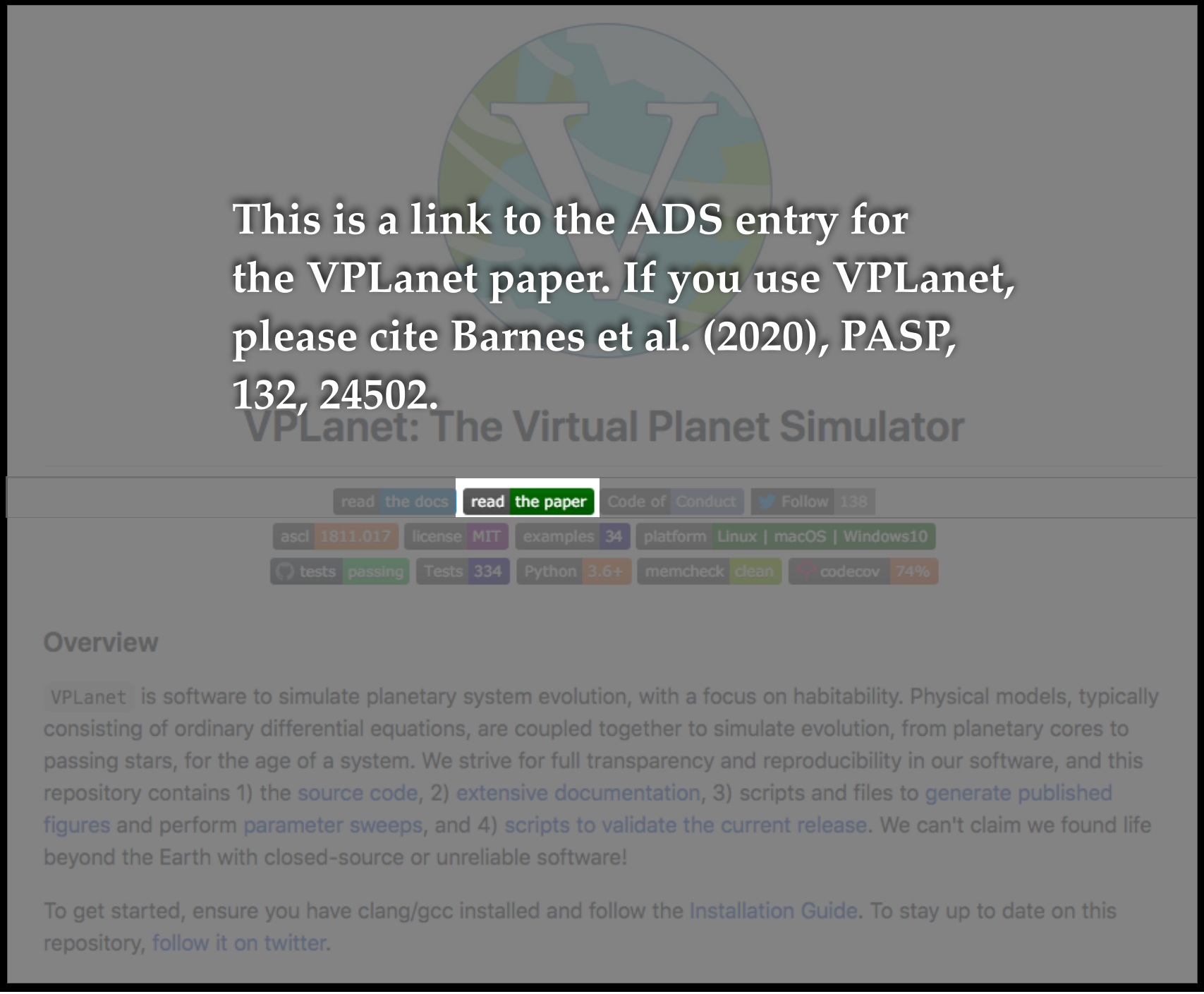
To get started, ensure you have clang/gcc installed and follow the Installation Guide. To stay up to date on this repository, follow it on twitter.

Repo Overview

VPLanet's repository includes extensive documentation that is updated with every pull request (PR)



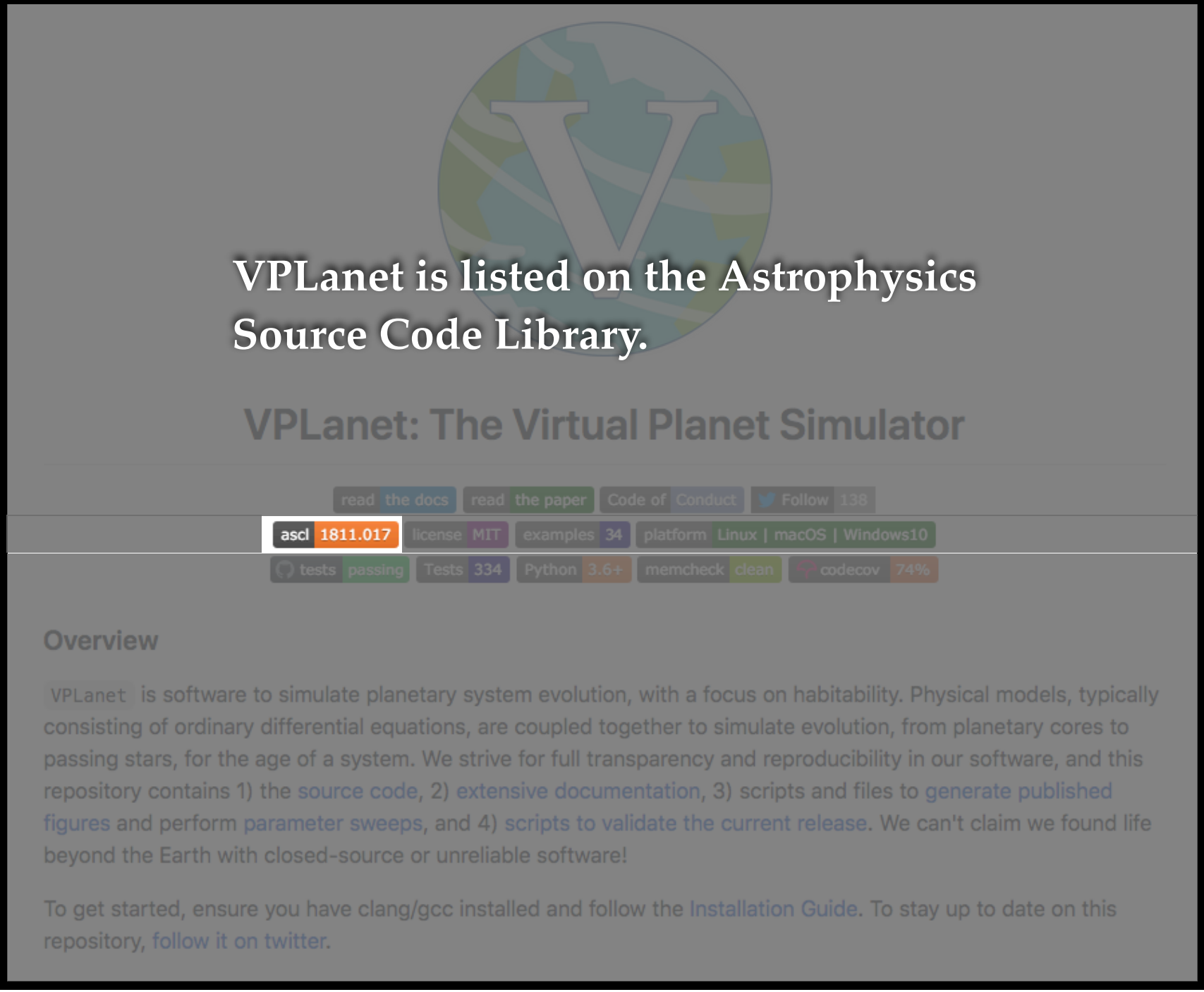
Repo Overview



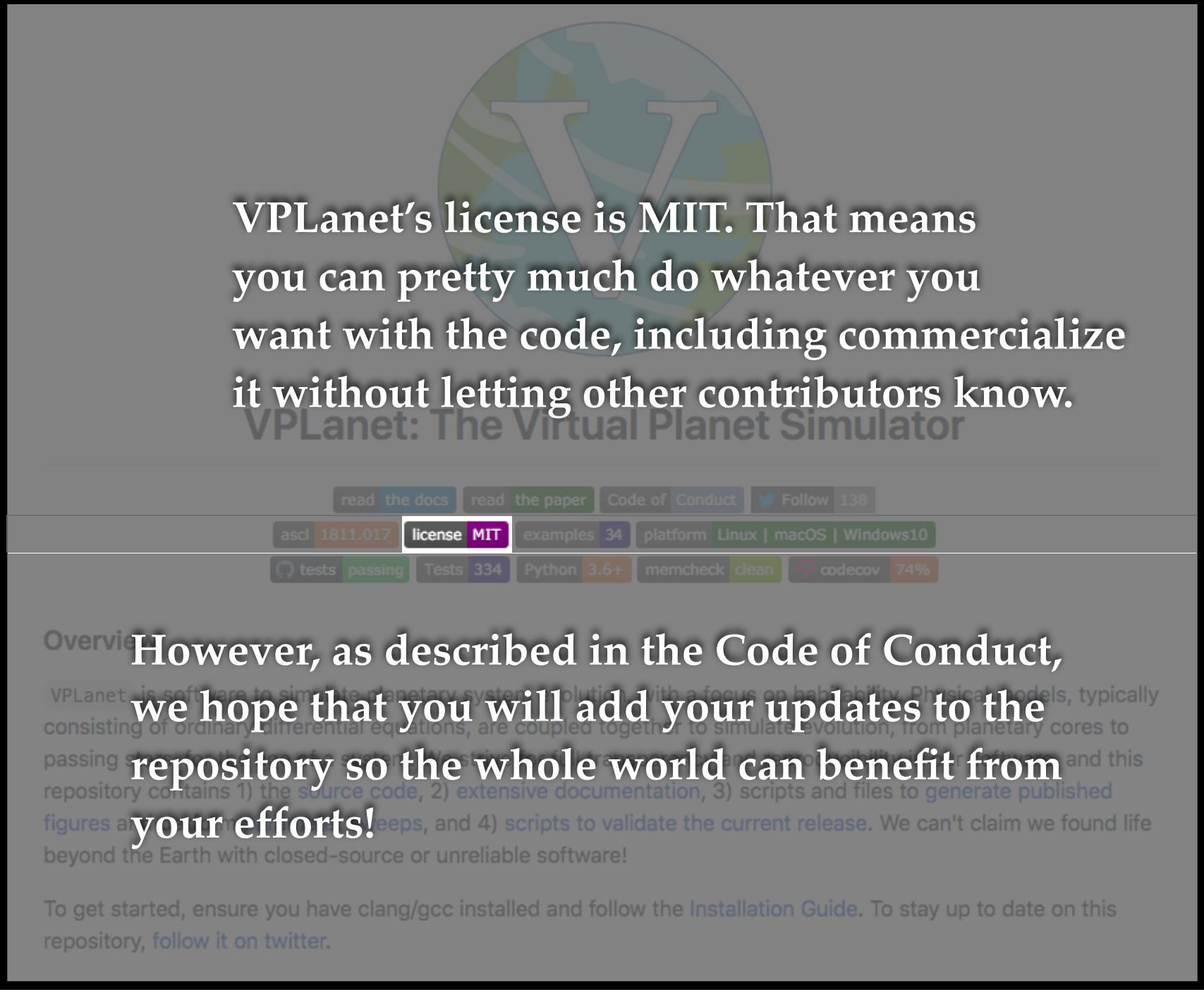
Repo Overview

Repo Overview

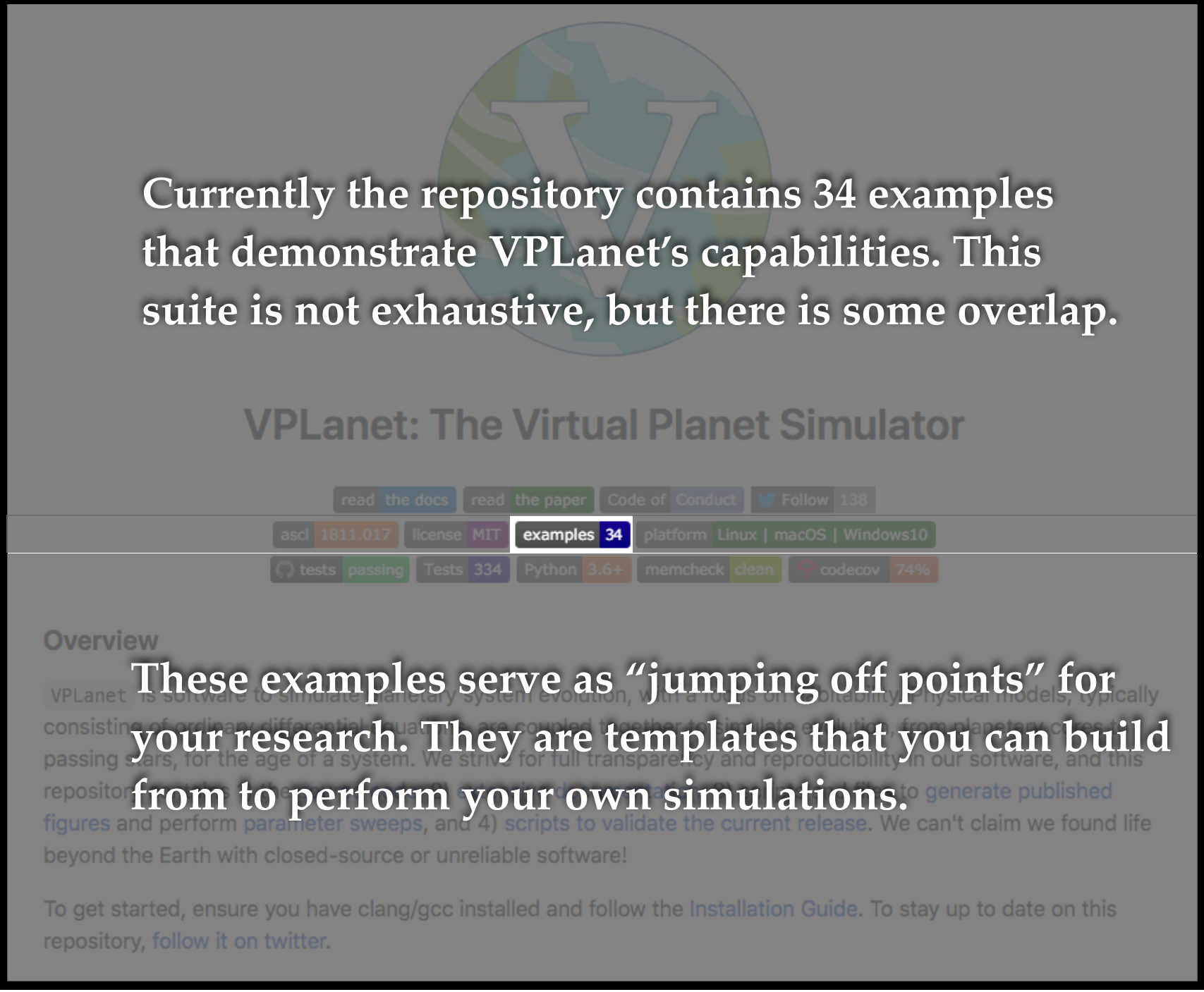
Repo Overview



Repo Overview



Repo Overview



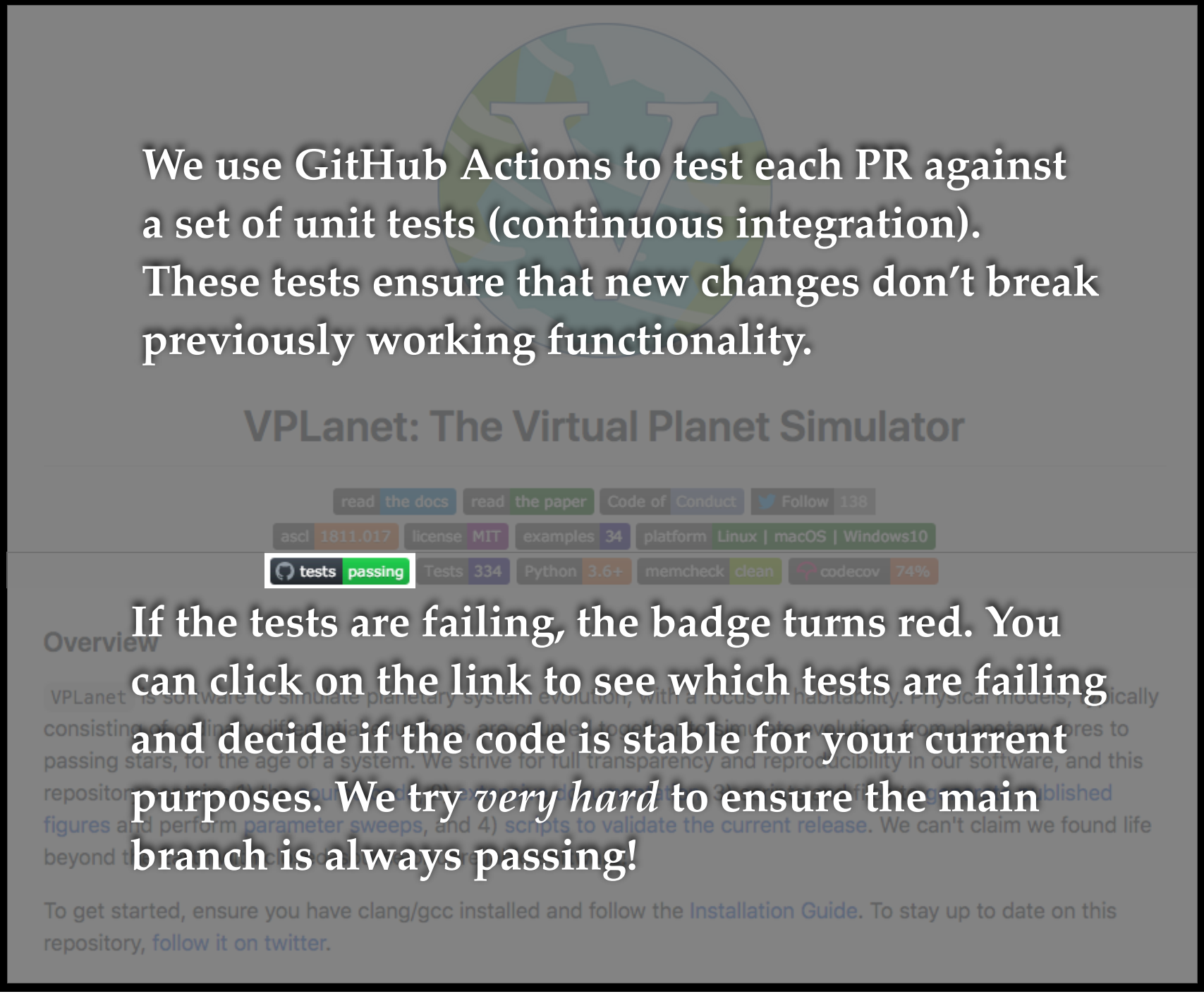
Currently the repository contains 34 examples that demonstrate VPLanet's capabilities. This suite is not exhaustive, but there is some overlap.

These examples serve as "jumping off points" for your research. They are templates that you can build from to perform your own simulations.

Repo Overview

Repo Overview

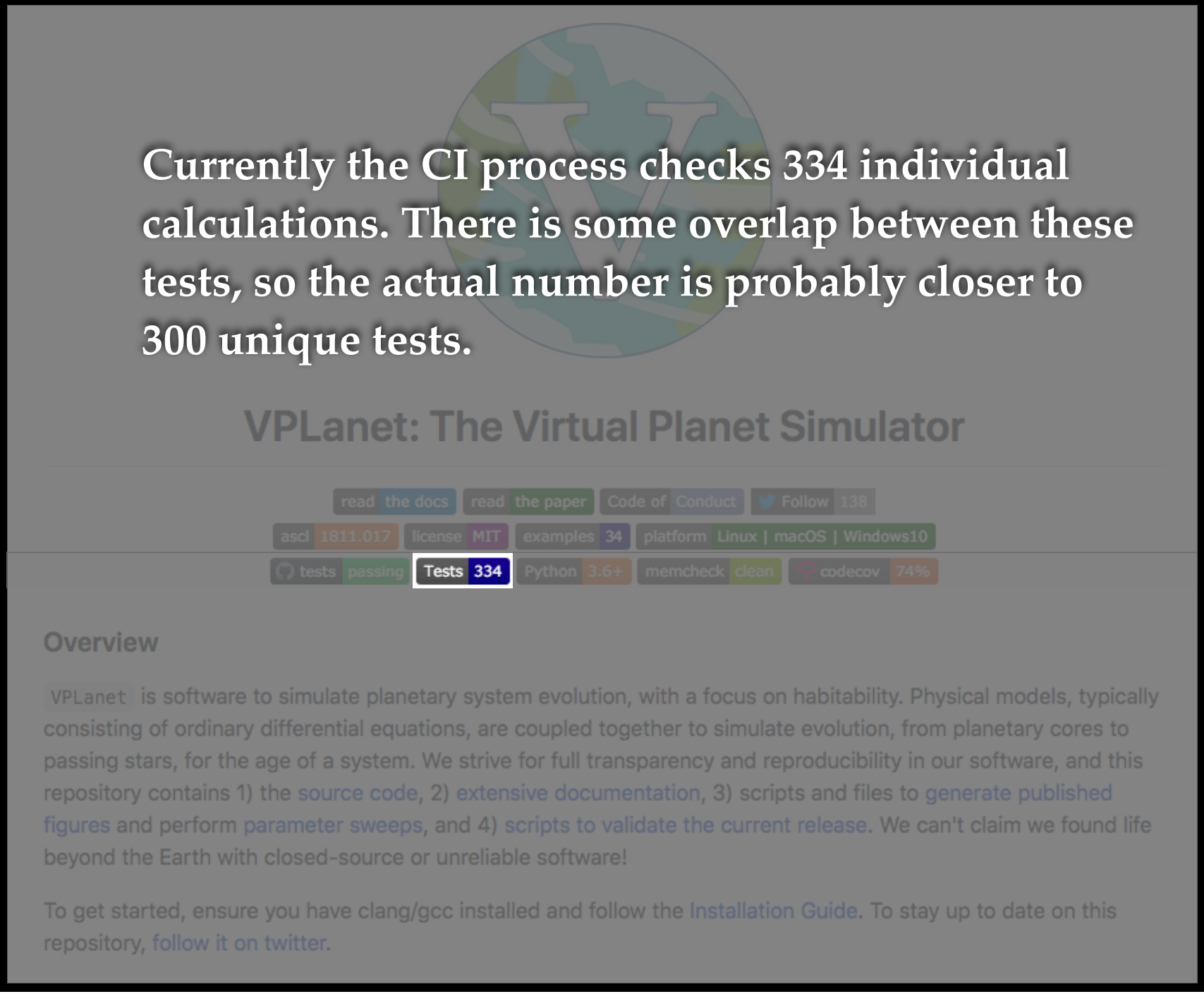
We use GitHub Actions to test each PR against a set of unit tests (continuous integration). These tests ensure that new changes don't break previously working functionality.



If the tests are failing, the badge turns red. You can click on the link to see which tests are failing and decide if the code is stable for your current purposes. We try *very hard* to ensure the main branch is always passing!

Repo Overview

Currently the CI process checks 334 individual calculations. There is some overlap between these tests, so the actual number is probably closer to 300 unique tests.

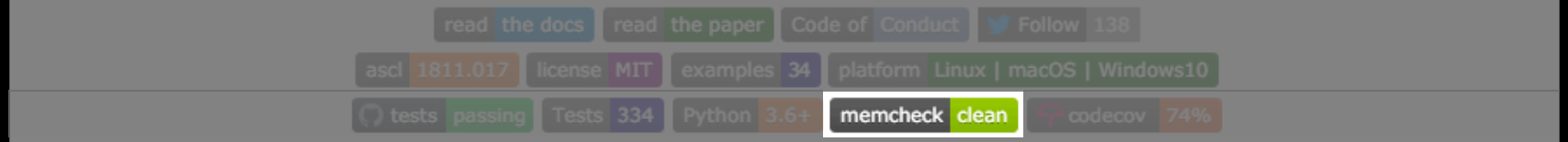


Repo Overview

Although VPLanet is written in C, it is designed to seamlessly connect with Python. In addition, the examples all use Python for generating plots. Currently VPLanet, and its support scripts, are verified for Python distributions 3.6 - 3.9.

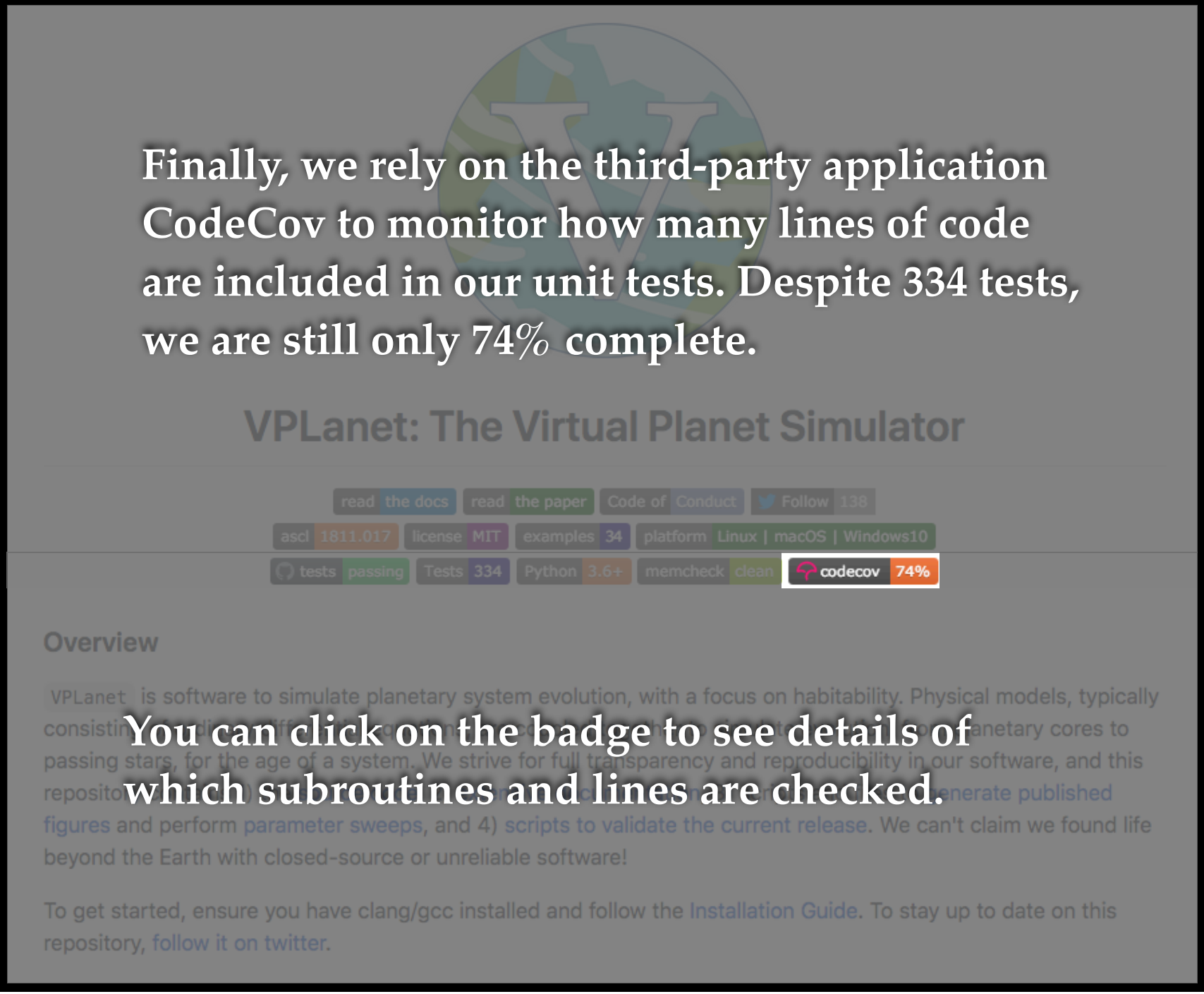
Repo Overview

The VPLanet team uses Valgrind's memcheck feature to test the code for memory errors. For each unit test, Valgrind checks for issues such as memory writes/reads beyond an array, conditional expressions that rely on uninitialized memory, blocks of memory that become locked, etc. If all tests pass, this badge is green, and VPLanet is "memcheck-clean".



When memcheck is clean, that means we can say that for every unit test, we have tracked every single bit for the duration of the execution. But there may still be bugs due to programming mistakes!

Repo Overview



Quick Tour of the Repo