

michalc /

OpenTTDLab

<> Code

Issues

Pull requests

Discussions

Actions

Security

Insights

Settings

Python framework for running reproducible experiments using OpenTTD

GPL-2.0 license

16 stars

1 fork

2 watching

1 Branch

52 Tags

Activity

Public repository

main

1 Branch

52 Tags

Go to file

Go to file

+

Add file

Code

michalc

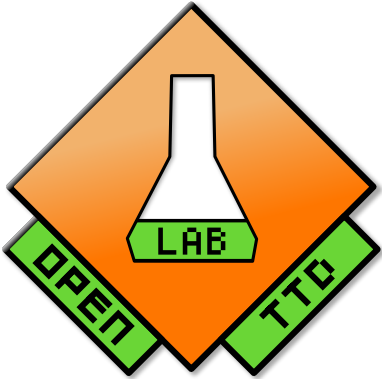
Merge pull request #156 from michalc/docs/update-examples

2 days ago

| | | |
|--------------------|---|--------------|
| .github/workflows | feat: avoid windows popping up during... | last month |
| docs/assets | docs: example for plotting | 4 months ago |
| examples | docs: update Example 2 for latest Ope... | 2 days ago |
| fixtures | feat: ability to download AI libraries fro... | last month |
| .coveragerc | ci: test with code coverage | 4 months ago |
| .gitignore | devex: more recent Python .gitignore | 4 months ago |
| LICENSE | Add: first version of savegame reader | 3 years ago |
| README.md | docs: document experiments parameter | 5 days ago |
| codecov.yml | ci: test with code coverage | 4 months ago |
| openttdlab.py | feat: expose experiment dictionary in r... | 5 days ago |
| pyproject.toml | feat: show basic progress bar when ru... | 3 weeks ago |
| test_openttdlab.py | feat: expose experiment dictionary in r... | 5 days ago |

README

License



OpenTTDLab - Run reproducible experiments using OpenTTD

PyPI package

v0.0.52

Test suite

passing

Code coverage

86%

OpenTTDLab is a Python framework for using [OpenTTD](#) to run reproducible experiments and extracting results from them, with as few manual steps as possible.

OpenTTDLab is based on [Patric Stout's OpenTTD Savegame Reader](#).

⚠ Caution

OpenTTDLab currently does not work with OpenTTD 14.0 or later. The latest version of OpenTTD known to work is 13.4.

Contents

- [Features](#)
- [Installation](#)
- [Running an experiment](#)
- [Plotting results](#)
- [Examples](#)
- [API](#)
- [Compatibility](#)
- [Licenses and attributions](#)

Features

- Allows you to easily run OpenTTD in a headless mode (i.e. without a graphical interface) over a variety of configurations.
- And allows you to do this from Python code - for example from a Jupyter Notebook.
- As is typical from Python code, it is cross platform - allowing to share code snippets between macOS, Windows, and Linux, even though details like how to install and start OpenTTD are different on each platform.
- Downloads (and caches) OpenTTD, OpenGFX, and AIs - no need to download these separately or through OpenTTD's built-in content browser.
- Transparently parallelises runs of OpenTTD, by default up to the number of CPUs. (Although with [fairly poor scaling properties](#).)
- Results are extracted from OpenTTD savegames as plain Python dictionaries and lists - reasonably convenient for importing into tools such as pandas for analysis or visualisation.

Installation

OpenTTDLab is distributed via [PyPI](#), and so can usually be installed using pip.

```
python -m pip install OpenTTDLab
```



When run on macOS, OpenTTDLab has a dependency that pip does not install: [7-zip](#). To install 7-zip, first install [Homebrew](#), and then use Homebrew to install the p7zip package that contains 7-zip.

```
brew install p7zip
```



You do not need to separately download or install OpenTTD (or [OpenGFX](#)) in order to use OpenTTDLab. OpenTTDLab itself handles downloading them.

Running experiments

The core function of OpenTTD is the `run_experiments` function.

```
from openttdlab import run_experiments, bananas_ai

# Run experiments...
results = run_experiments(
    openttd_version='13.4', # ... for a specific versions of OpenTTD
    opengfx_version='7.1', # ... and a specific versions of OpenGFX
    experiments=(
        {
            # ... for random seeds
            'seed': seed,
            # ... running specific AIs. In this case a single AI, with no
            # parameters, fetching it from https://bananas.openttd.org/package/ai
            'ais': (
                bananas_ai('54524149', 'trAIns', ai_params=()),
            ),
            # ... each for a number of (in game) days
            'days': 365 * 4 + 1,
        }
        for seed in range(0, 10)
    ),
)
```



Plotting results

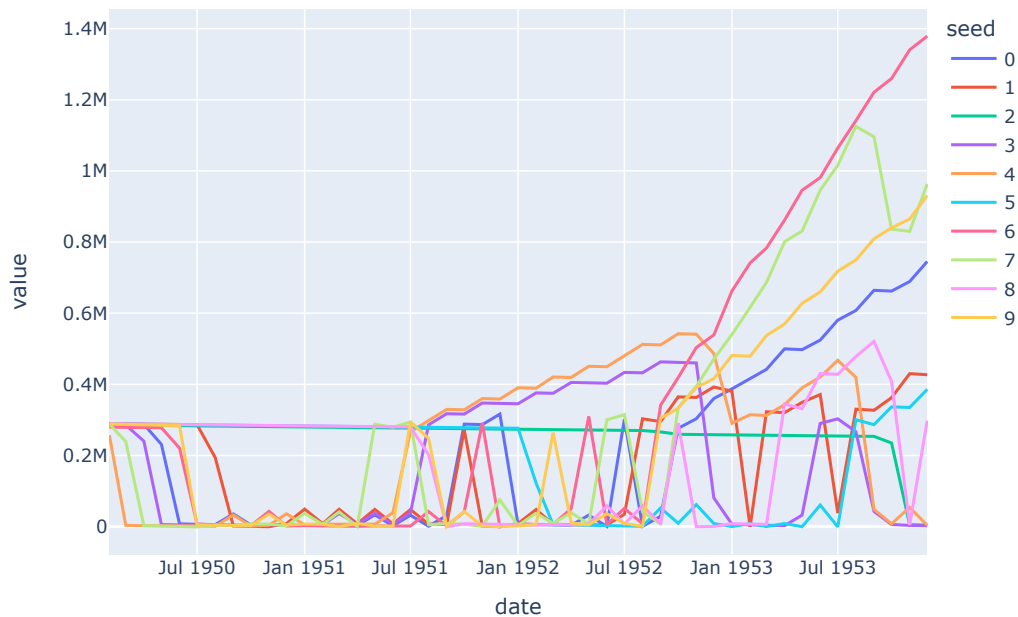
OpenTTD does not require any particular library for plotting results. However, [pandas](#) and [Plotly Express](#) are common options for plotting from Python. For example if you have a `results` object from `run_experiments` as in the above example, the following code

```
import pandas as pd
import plotly.express as px

df = pd.DataFrame(
    {
        'seed': row['experiment']['seed'],
        'date': row['date'],
        'money': row['chunks']['PLYR']['0']['money'],
    }
    for row in results
)
df = df.pivot(index='date', columns='seed', values='money')
fig = px.line(df)
fig.show()
```



should output a plot much like this one.



Examples

A notebook of the above example and an example measuring the performance of OpenTTDLab are in the [examples](#) folder.

API

Running experiments

`run_experiments(...)`

The core function of OpenTTDLab is the `run_experiments` function, used to run an experiment and return results extracted from the savegame files that OpenTTD produces. It has the following parameters and defaults.

- `experiments=()`

An iterable of the experiments to run. Each experiment should be a dictionary with the (string) keys:

- `'ais'`

The list of AIs to run in this experiment. See the [Fetching AIs](#) section for details on this parameter.

- `'seed'`

The integer seed of the random number generator for this experiment.

- `'days'`

The integer number of in-game days that this experiment will run for.

- `ais_libraries=()`

The list of AI libraries to have available to AI code. See the [Fetching AI libraries](#) section for details on this parameter.

- `base_openttd_config=''`

OpenTTD config to run each experiment under. This must be in the [openttd.cfg format](#). This is added to by OpenTTDLab before being passed to OpenTTD.

- `final_screenshot_directory=None`

The directory to save a PNG screenshot of the entire map at the end of each run. Each is named in the format `<seed>.png`, where `<seed>` is the experiment's seed of the random number generator. If `None`, then no screenshots are saved.

For technical reasons, a window will briefly appear while each screenshot is being saved. This can be avoided when running on Linux if `xvfb-run` is installed and available in the path.

- `max_workers=None`

The maximum number of workers to use to run OpenTTD in parallel. If `None`, then `os.cpu_count()` defined how many workers run.

- `openttd_version=None`

The version of OpenTTD to use. If `None`, the latest version available at `openttd_base_url` is used.

[!CAUTION] OpenTTDLab currently does not work with OpenTTD 14.0 or later. The latest version of OpenTTD known to work is 13.4.

- `opengfx_version=None`

The version of OpenGFX to use. If `None`, the latest version available at `opengfx_base_url` is used.

- `openttd_base_url='https://cdn.openttd.org/openttd-releases/'`

The base URL used to fetch the list of OpenTTD versions, and OpenTTD binaries.

- `opengfx_base_url='https://cdn.openttd.org/opengfx-releases/'`

The URL used to fetch the list of OpenGFX versions, and OpenGFX binaries.

- `get_http_client=lambda: httpx.Client(transport=httpx.HTTPTransport(retries=3))`

The HTTP client used to make HTTP requests when fetching OpenTTD, OpenGFX, or AIs. Note that the `bananas_ai` function uses a raw TCP connection in addition to HTTP requests, and so not all outgoing connections use the client specified by this.

Fetching AIs

The value of the `ais` key of each dictionary in the `experiments` parameter configures which AIs will run, how their code will be located, their names, and what parameters will be passed to each of them when they start. In more detail, the `ais` parameter must be an iterable of the return value of any of the the following 4 functions.

Important

The `ai_name` argument passed to each of the following functions must exactly match the name of the corresponding AI as published. If it does not match, the AI will not be started.

Important

The return value of each of the following is opaque: it should not be used in client code, other than by passing into `run_experiments` as part of the `ais` parameter.

`bananas_ai(unique_id, ai_name, ai_params=())`

Defines an AI by the `unique_id` and `ai_name` of an AI published through OpenTTD's content service at <https://bananas.openttd.org/package/ai>. This allows you to quickly run OpenTTDLab with a published AI. The `ai_params` parameter is an optional parameter of an iterable of (key, value) parameters passed to the AI on startup.

The `unique_id` is sometimes surfaced as the "Content Id", but it should not include its `ai/` prefix.

local_folder(folder_path, ai_name, ai_params=())

Defines an AI by the `folder_path` to a local folder that contains the AI code of an AI with name `ai_name`. The `ai_params` parameter is an optional parameter of an iterable of (key, value) parameters passed to the AI on startup.

local_file(path, ai_name, ai_params=())

Defines an AI by the local path to a .tar AI file that contains the AI code. The `ai_params` parameter is an optional parameter of an iterable of (key, value) parameters passed to the AI on startup.

remote_file(url, ai_name, ai_params=())

Fetches the AI by the URL of a tar.gz file that contains the AI code. For example, a specific GitHub tag of a repository that contains its code. The `ai_params` parameter is an optional parameter of an iterable of (key, value) parameters passed to the AI on startup.

Fetching AI libraries

The `ai_libraries` parameter of `run_experiments` ensures that AI libraries are available to the AIs running. In more detail, the `ais_libraries` parameter must be an iterable, where each item the the return value of the following function.

bananas_ai_library(unique_id, ai_library_name)

Fetches the AI library defined by `unique_id` and `ai_name` of a library published through OpenTTD's content service at <https://bananas.openttd.org/package/ai-library>.

The `unique_id` is sometimes surfaced as the "Content Id", but it should not include its `ai-library/` prefix.

Compatibility

- Linux (tested on Ubuntu 20.04), Windows (tested on Windows Server 2019), or macOS (tested on macOS 11)
- Python >= 3.8.0 (tested on 3.8.0 and 3.12.0)

Licenses and attributions

TL;DR



Releases 52

 **v0.0.52** Latest
5 days ago

[+ 51 releases](#)

Packages

No packages published
[Publish your first package](#)

Contributors 3



michalc Michal Charemza



TrueBrain Patric Stout



BasicBeluga Ian Earle

Deployments 52

✅ pypi 5 days ago

[+ 51 deployments](#)

Languages

● Python 100.0%