RLOS Benchmarks and Competitions

Benchmarking is hard

- Benchmarking is hard
- Benchmarking is tedious
- Benchmarking is questioned

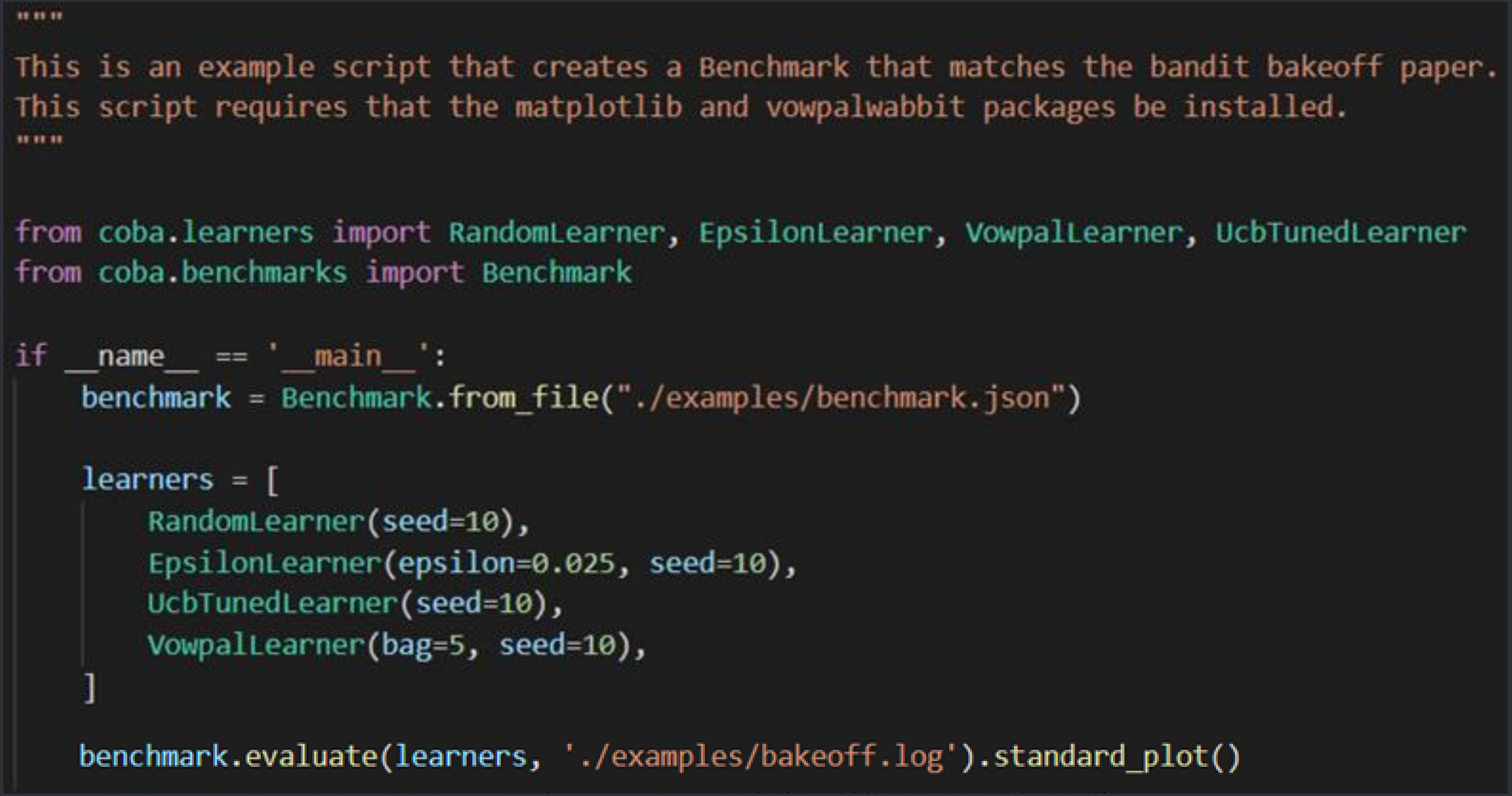
COBA (<u>CO</u>ntextual <u>BA</u>ndit)

“It’s got good bones”

Coba benchmarking is:

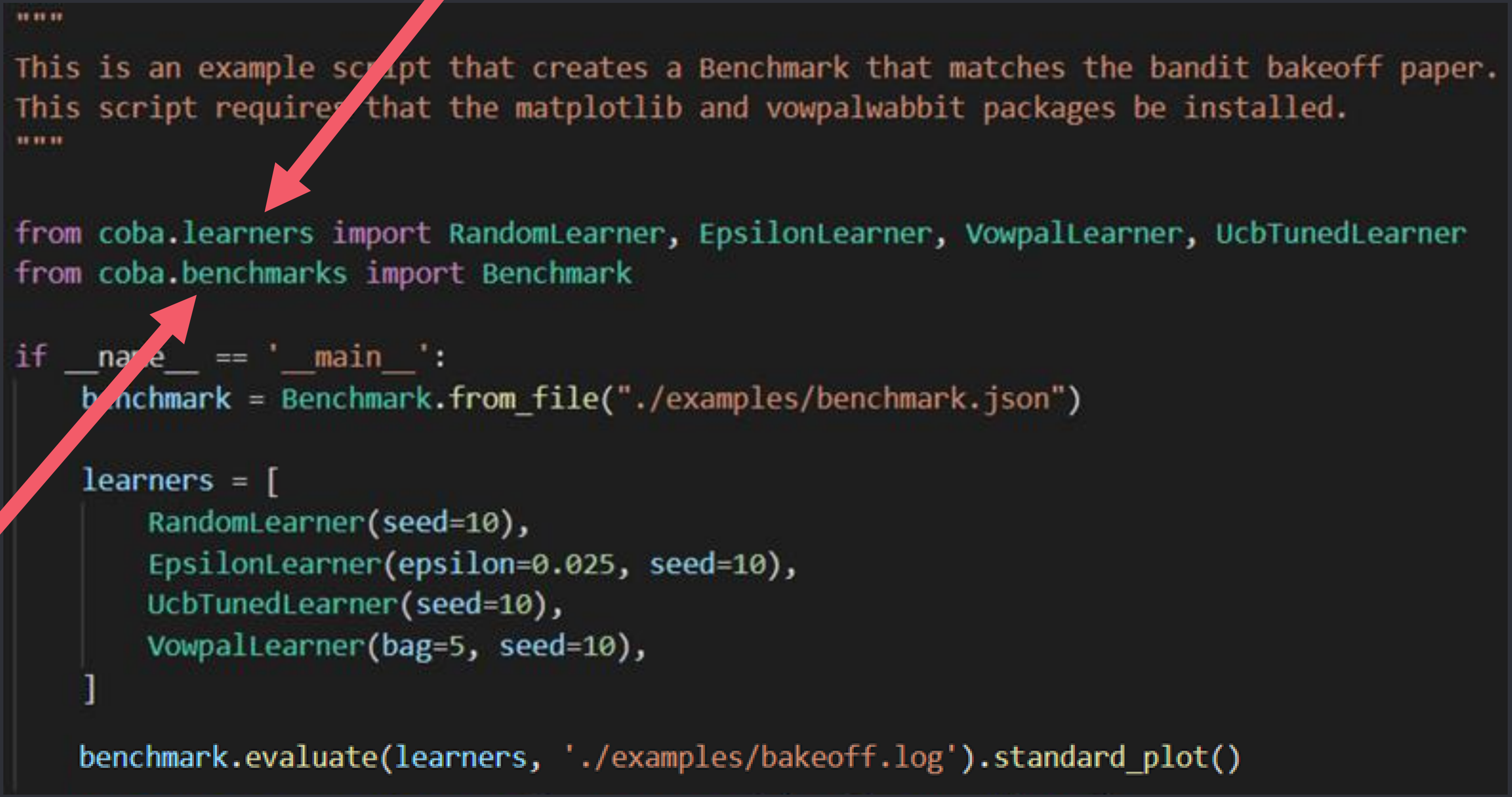
- Easy to add new data sets
- Easy to add new algorithms
- Easy to create benchmarks
- Easy to share benchmarks

Code Walkthrough



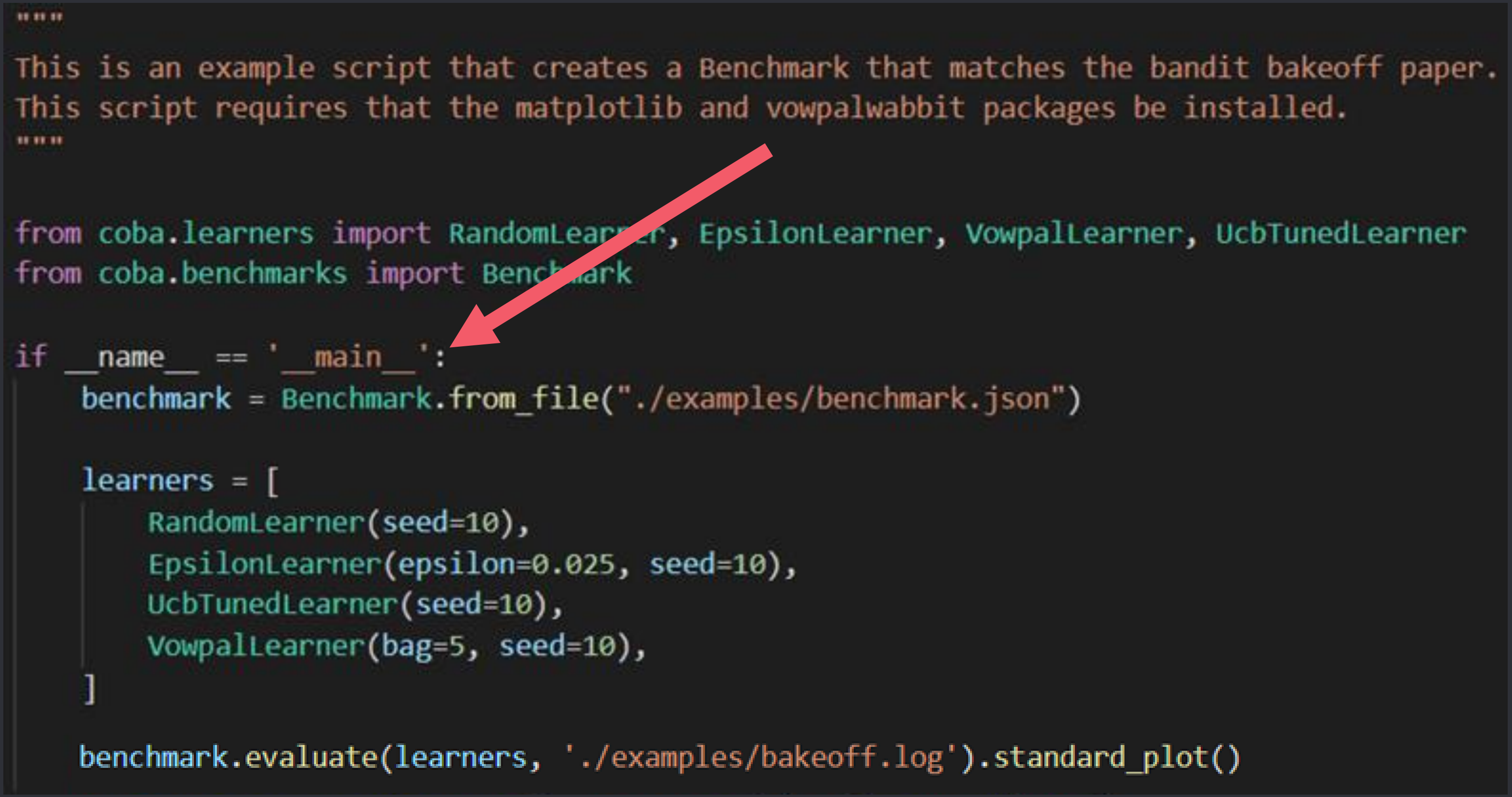
```python
"""
This is an example script that creates a Benchmark that matches the bandit bakeoff paper.
This script requires that the matplotlib and vowpalwabbit packages be installed.
"""

from coba.learners import RandomLearner, EpsilonLearner, VowpalLearner, UcbTunedLearner
from coba.benchmarks import Benchmark

if __name__ == '__main__':
    benchmark = Benchmark.from_file("./examples/benchmark.json")

    learners = [
        RandomLearner(seed=10),
        EpsilonLearner(epsilon=0.025, seed=10),
        UcbTunedLearner(seed=10),
        VowpalLearner(bag=5, seed=10),
    ]

    benchmark.evaluate(learners, './examples/bakeoff.log').standard_plot()
```

Code Walkthrough



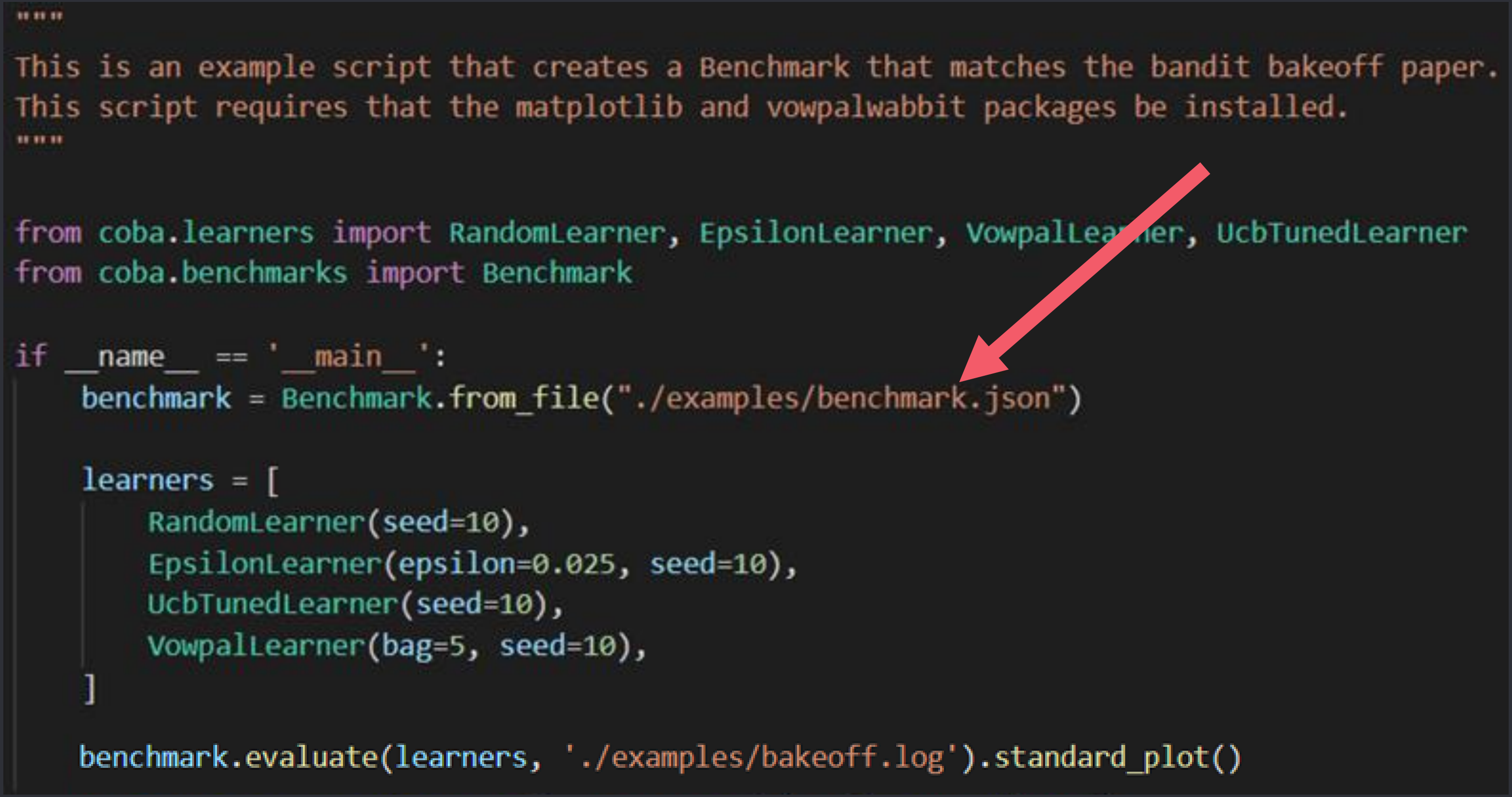
```python
"""
This is an example script that creates a Benchmark that matches the bandit bakeoff paper.
This script requires that the matplotlib and vowpalwabbit packages be installed.
"""

from coba.learners import RandomLearner, EpsilonLearner, VowpalLearner, UcbTunedLearner
from coba.benchmarks import Benchmark

if __name__ == '__main__':
    benchmark = Benchmark.from_file("./examples/benchmark.json")

    learners = [
        RandomLearner(seed=10),
        EpsilonLearner(epsilon=0.025, seed=10),
        UcbTunedLearner(seed=10),
        VowpalLearner(bag=5, seed=10),
    ]

    benchmark.evaluate(learners, './examples/bakeoff.log').standard_plot()
```

Code Walkthrough



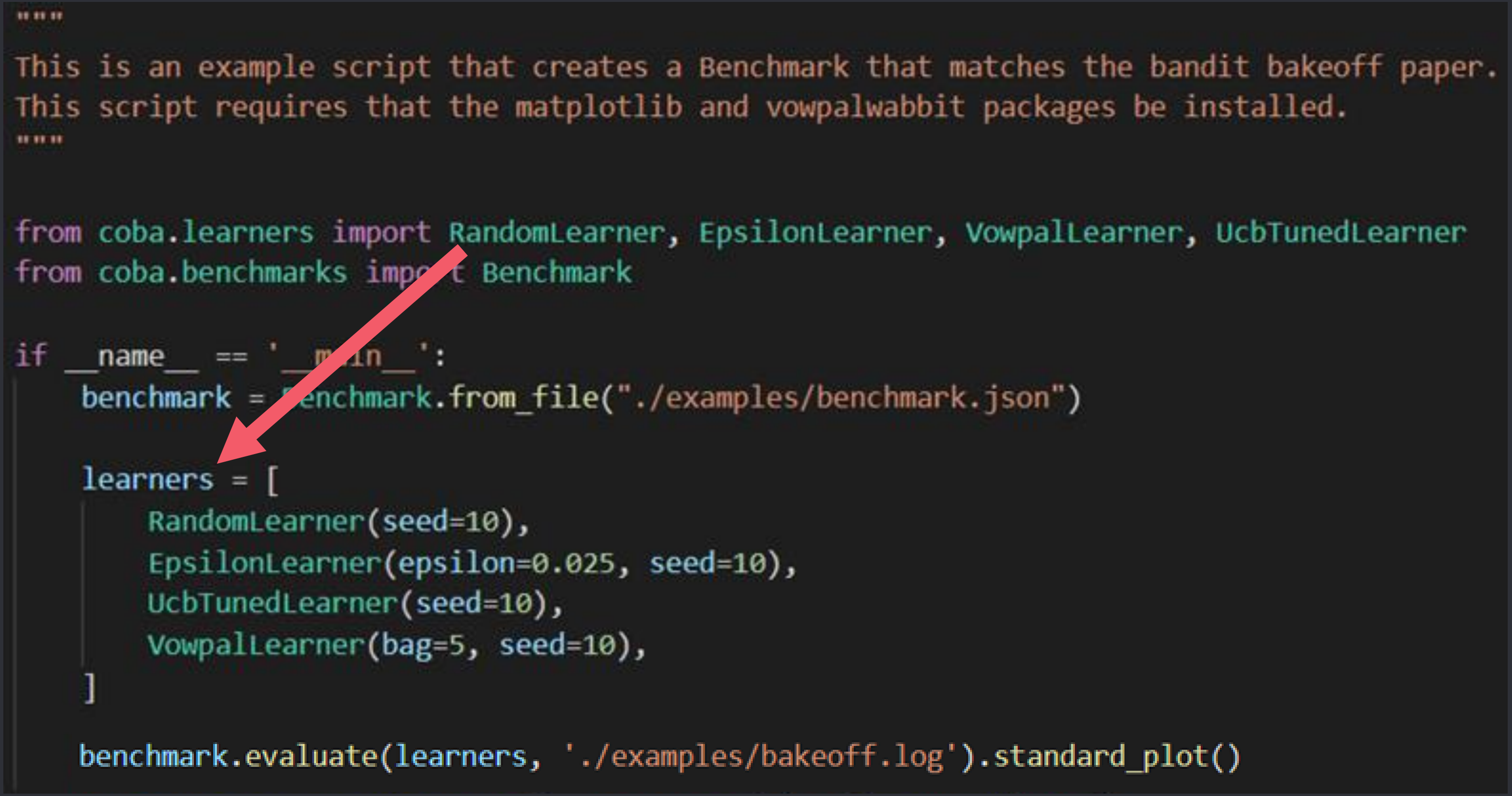
```python
"""
This is an example script that creates a Benchmark that matches the bandit bakeoff paper.
This script requires that the matplotlib and vowpalwabbit packages be installed.
"""

from coba.learners import RandomLearner, EpsilonLearner, VowpalLearner, UcbTunedLearner
from coba.benchmarks import Benchmark

if __name__ == '__main__':
    benchmark = Benchmark.from_file("./examples/benchmark.json")

    learners = [
        RandomLearner(seed=10),
        EpsilonLearner(epsilon=0.025, seed=10),
        UcbTunedLearner(seed=10),
        VowpalLearner(bag=5, seed=10),
    ]

    benchmark.evaluate(learners, './examples/bakeoff.log').standard_plot()
```

Code Walkthrough



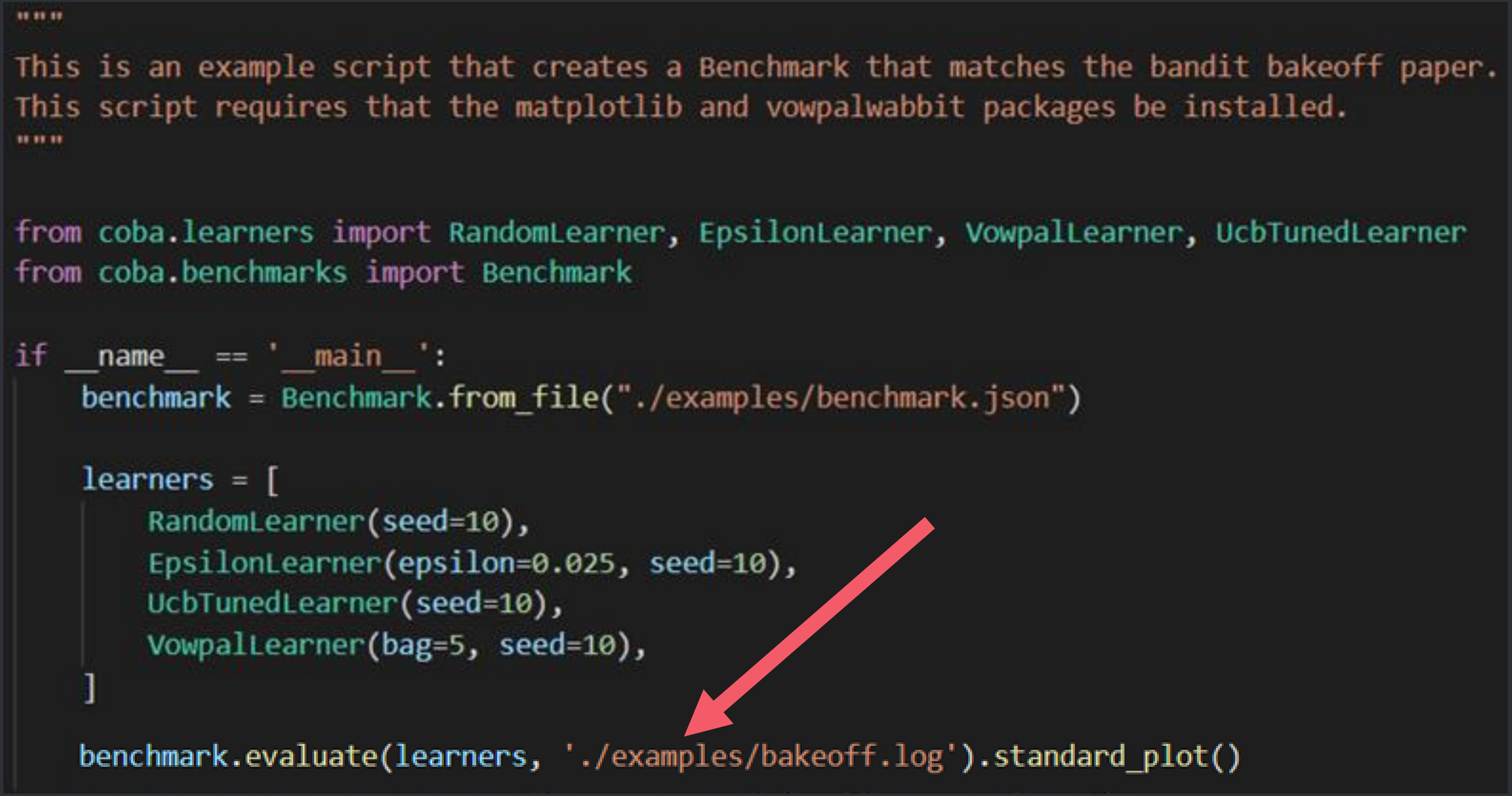
```python
"""
This is an example script that creates a Benchmark that matches the bandit bakeoff paper.
This script requires that the matplotlib and vowpalwabbit packages be installed.
"""

from coba.learners import RandomLearner, EpsilonLearner, VowpalLearner, UcbTunedLearner
from coba.benchmarks import Benchmark

if __name__ == '__main__':
    benchmark = Benchmark.from_file("./examples/benchmark.json")

    learners = [
        RandomLearner(seed=10),
        EpsilonLearner(epsilon=0.025, seed=10),
        UcbTunedLearner(seed=10),
        VowpalLearner(bag=5, seed=10),
    ]

    benchmark.evaluate(learners, './examples/bakeoff.log').standard_plot()
```

Code Walkthrough



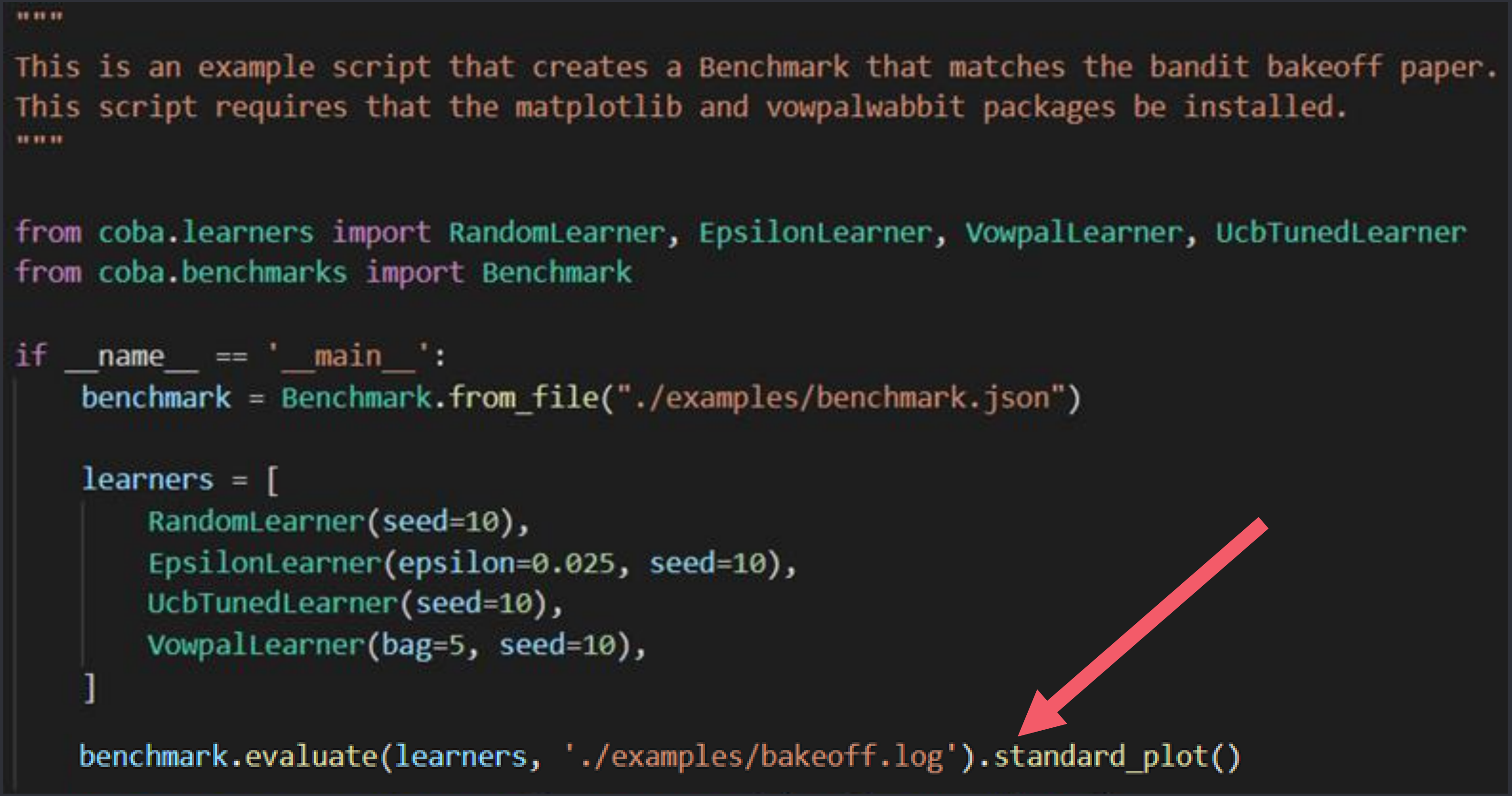
```python
"""
This is an example script that creates a Benchmark that matches the bandit bakeoff paper.
This script requires that the matplotlib and vowpalwabbit packages be installed.
"""

from coba.learners import RandomLearner, EpsilonLearner, VowpalLearner, UcbTunedLearner
from coba.benchmarks import Benchmark

if __name__ == '__main__':
    benchmark = Benchmark.from_file("./examples/benchmark.json")

    learners = [
        RandomLearner(seed=10),
        EpsilonLearner(epsilon=0.025, seed=10),
        UcbTunedLearner(seed=10),
        VowpalLearner(bag=5, seed=10),
    ]

    benchmark.evaluate(learners, './examples/bakeoff.log').standard_plot()
```

Code Walkthrough



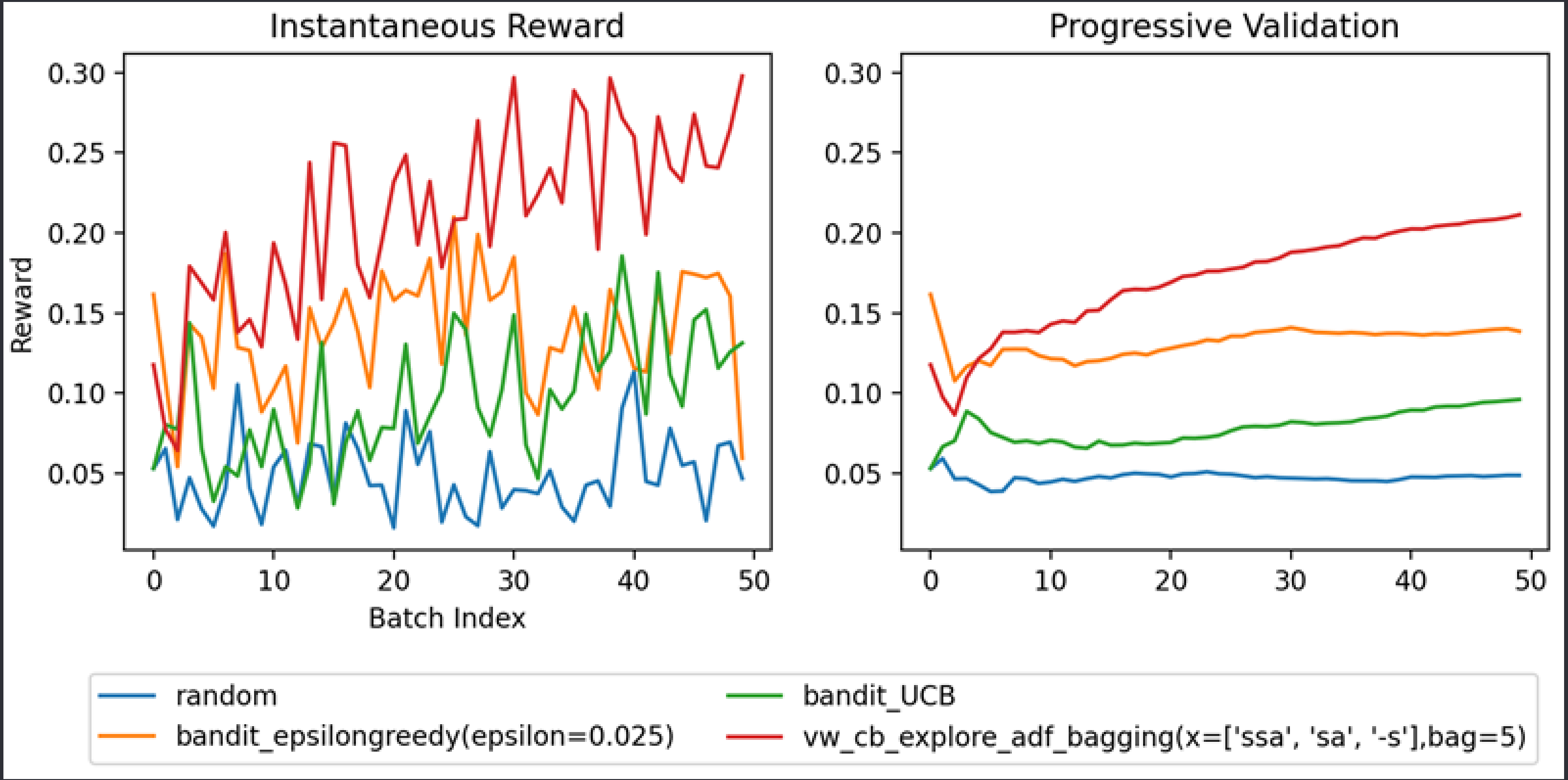
```
"""
This is an example script that creates a Benchmark that matches the bandit bakeoff paper.
This script requires that the matplotlib and vowpalwabbit packages be installed.
"""

from coba.learners import RandomLearner, EpsilonLearner, VowpalLearner, UcbTunedLearner
from coba.benchmarks import Benchmark

if __name__ == '__main__':
    benchmark = Benchmark.from_file("./examples/benchmark.json")

    learners = [
        RandomLearner(seed=10),
        EpsilonLearner(epsilon=0.025, seed=10),
        UcbTunedLearner(seed=10),
        VowpalLearner(bag=5, seed=10),
    ]

    benchmark.evaluate(learners, './examples/bakeoff.log').standard_plot()
```

Code Walkthrough

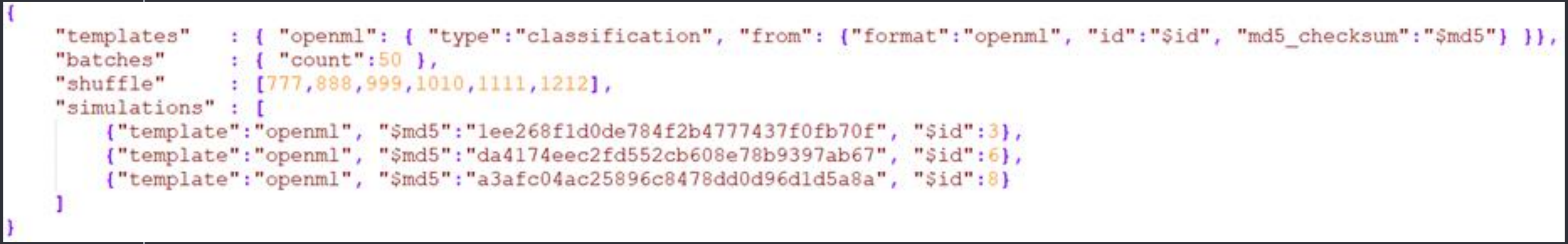


```
"""
This is an example script that creates a Benchmark that matches the bandit bakeoff paper.
This script requires that the matplotlib and vowpalwabbit packages be installed.
"""

from coba.learners import RandomLearner, EpsilonLearner, VowpalLearner, UcbTunedLearner
from coba.benchmarks import Benchmark

if __name__ == '__main__':
    benchmark = Benchmark.from_file("./examples/benchmark.json")

    learners = [
        RandomLearner(seed=10),
        EpsilonLearner(epsilon=0.025, seed=10),
        UcbTunedLearner(seed=10),
        VowpalLearner(bag=5, seed=10),
    ]

    benchmark.evaluate(learners, './examples/bakeoff.log').standard_plot()
```

Resulting Plot

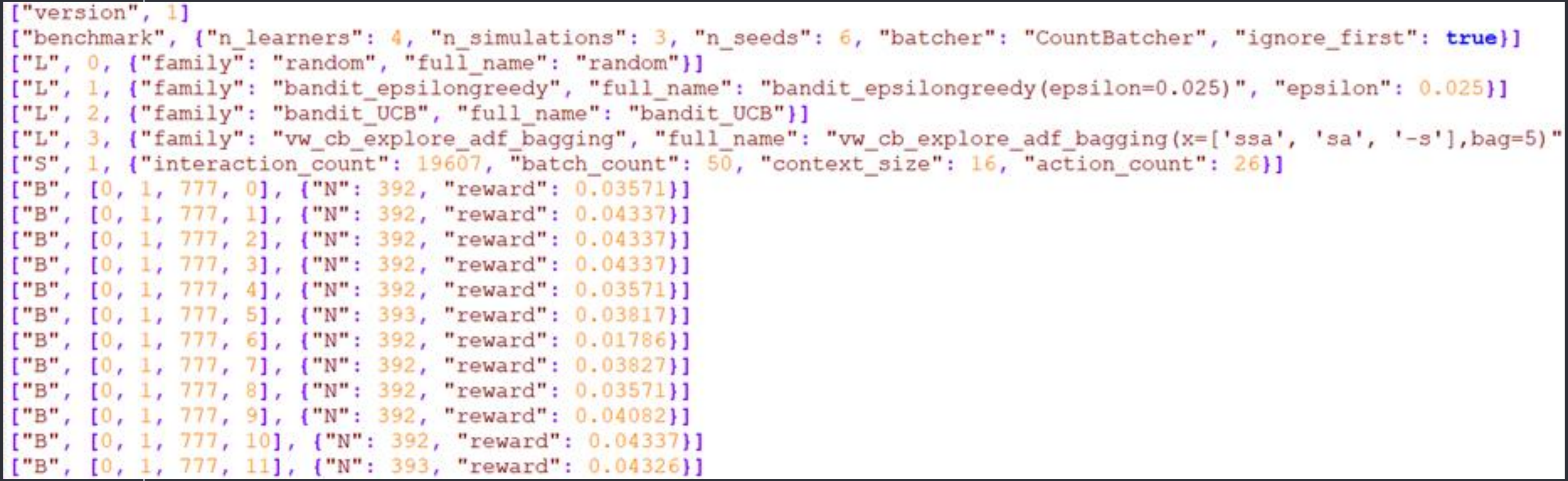


Benchmark Configuration File



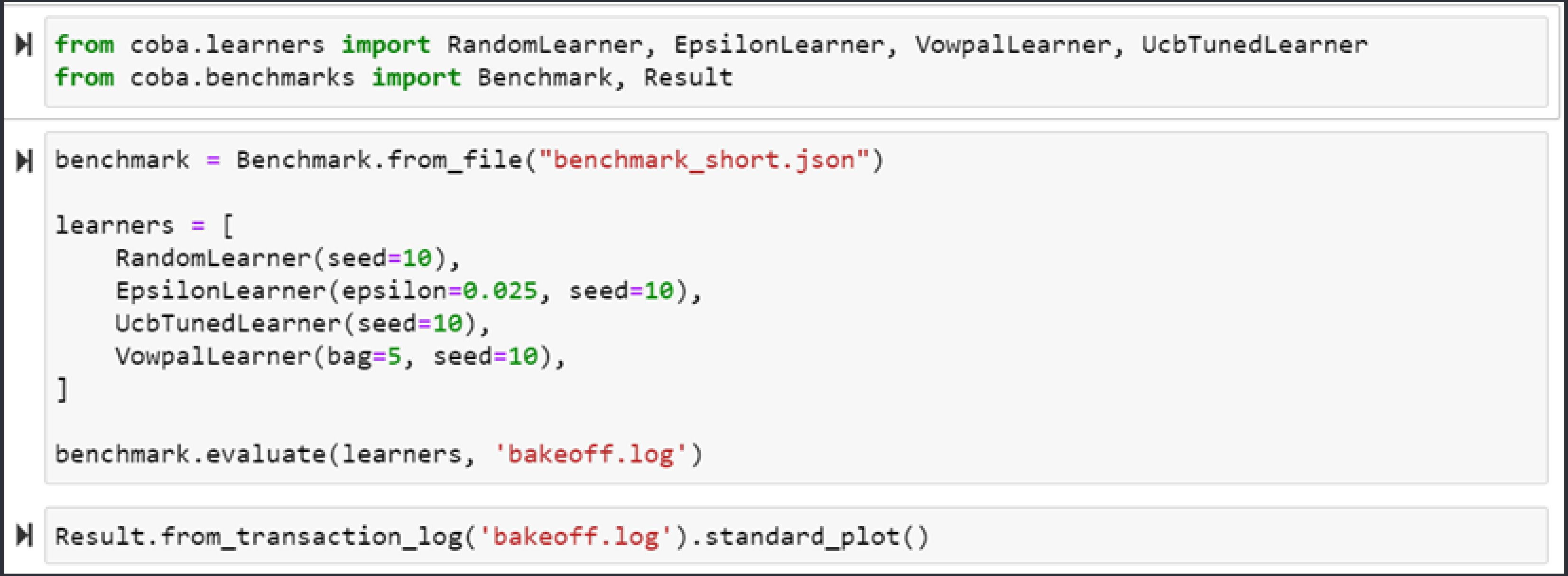
```
{
    "templates"   : { "openml": { "type":"classification", "from": {"format":"openml", "id":"$id", "md5_checksum":"$md5"} }},
    "batches"     : { "count":50 },
    "shuffle"     : [777,888,999,1010,1111,1212],
    "simulations" : [
        {"template":"openml", "$md5":"1ee268f1d0de784f2b4777437f0fb70f", "$id":3},
        {"template":"openml", "$md5":"da4174eec2fd552cb608e78b9397ab67", "$id":6},
        {"template":"openml", "$md5":"a3afc04ac25896c8478dd0d96d1d5a8a", "$id":8}
    ]
}
```

Benchmark Transaction File



```
["version", 1]
["benchmark", {"n_learners": 4, "n_simulations": 3, "n_seeds": 6, "batcher": "CountBatcher", "ignore_first": true}]
["L", 0, {"family": "random", "full_name": "random"}]
["L", 1, {"family": "bandit_epsilongreedy", "full_name": "bandit_epsilongreedy(epsilon=0.025)", "epsilon": 0.025}]
["L", 2, {"family": "bandit_UCB", "full_name": "bandit_UCB"}]
["L", 3, {"family": "vw_cb_explore_adf_bagging", "full_name": "vw_cb_explore_adf_bagging(x=['ssa', 'sa', '-s'],bag=5)"
["S", 1, {"interaction_count": 19607, "batch_count": 50, "context_size": 16, "action_count": 26}]
["B", [0, 1, 777, 0], {"N": 392, "reward": 0.03571}]
["B", [0, 1, 777, 1], {"N": 392, "reward": 0.04337}]
["B", [0, 1, 777, 2], {"N": 392, "reward": 0.04337}]
["B", [0, 1, 777, 3], {"N": 392, "reward": 0.04337}]
["B", [0, 1, 777, 4], {"N": 392, "reward": 0.03571}]
["B", [0, 1, 777, 5], {"N": 393, "reward": 0.03817}]
["B", [0, 1, 777, 6], {"N": 392, "reward": 0.01786}]
["B", [0, 1, 777, 7], {"N": 392, "reward": 0.03827}]
["B", [0, 1, 777, 8], {"N": 392, "reward": 0.03571}]
["B", [0, 1, 777, 9], {"N": 392, "reward": 0.04082}]
["B", [0, 1, 777, 10], {"N": 392, "reward": 0.04337}]
["B", [0, 1, 777, 11], {"N": 393, "reward": 0.04326}]
```

Jupyter Notebook Too

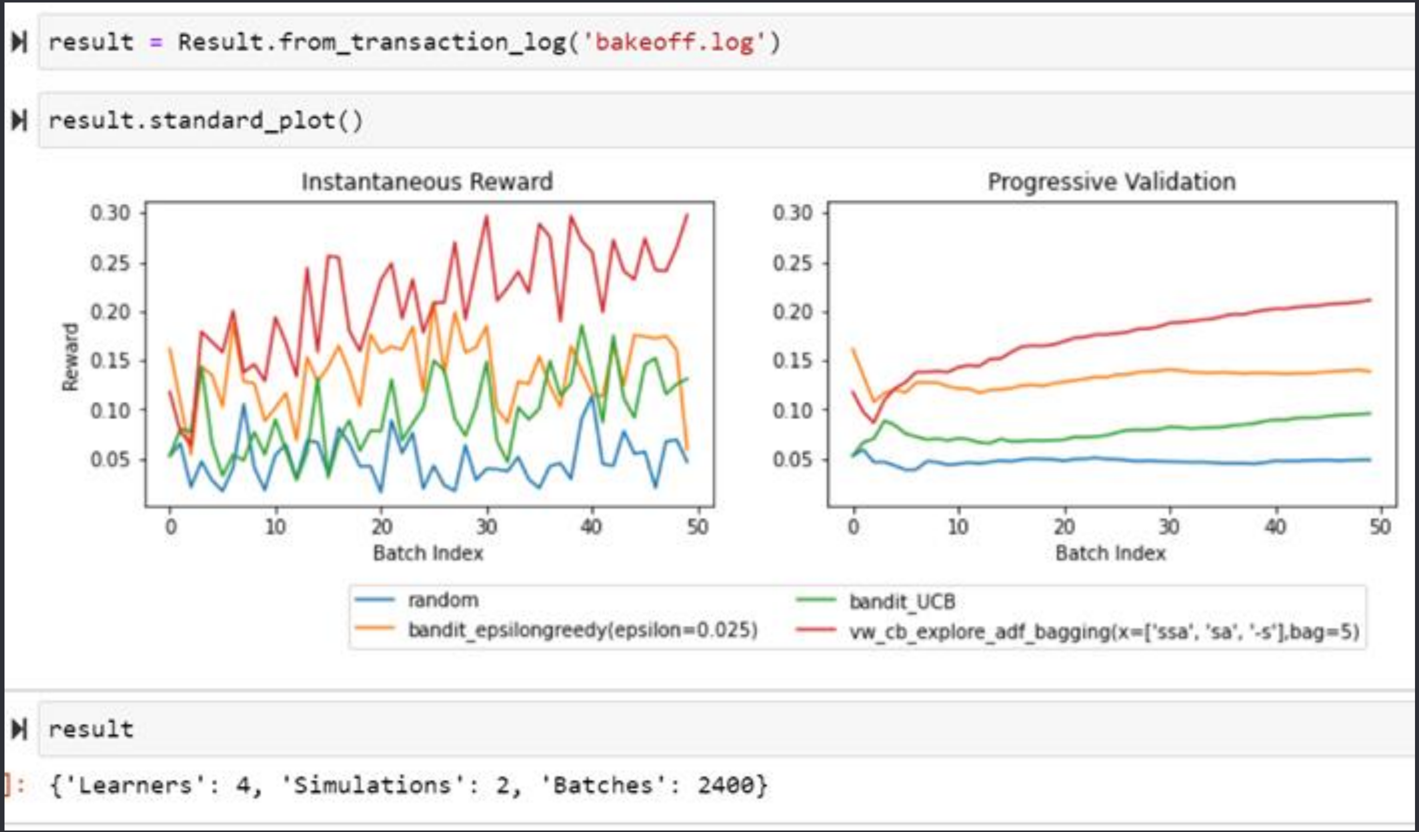


```
from coba.learners import RandomLearner, EpsilonLearner, VowpalLearner, UcbTunedLearner
from coba.benchmarks import Benchmark, Result
```

```
benchmark = Benchmark.from_file("benchmark_short.json")

learners = [
    RandomLearner(seed=10),
    EpsilonLearner(epsilon=0.025, seed=10),
    UcbTunedLearner(seed=10),
    VowpalLearner(bag=5, seed=10),
]

benchmark.evaluate(learners, 'bakeoff.log')
```

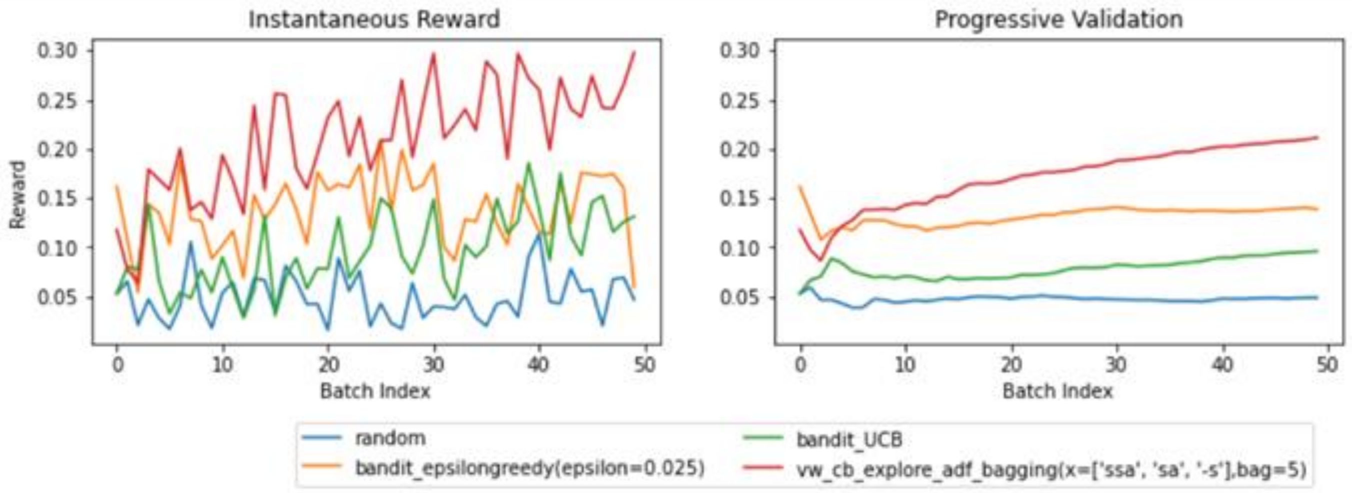
```
Result.from_transaction_log('bakeoff.log').standard_plot()
```

In Depth Analysis



```
result = Result.from_transaction_log('bakeoff.log')
```

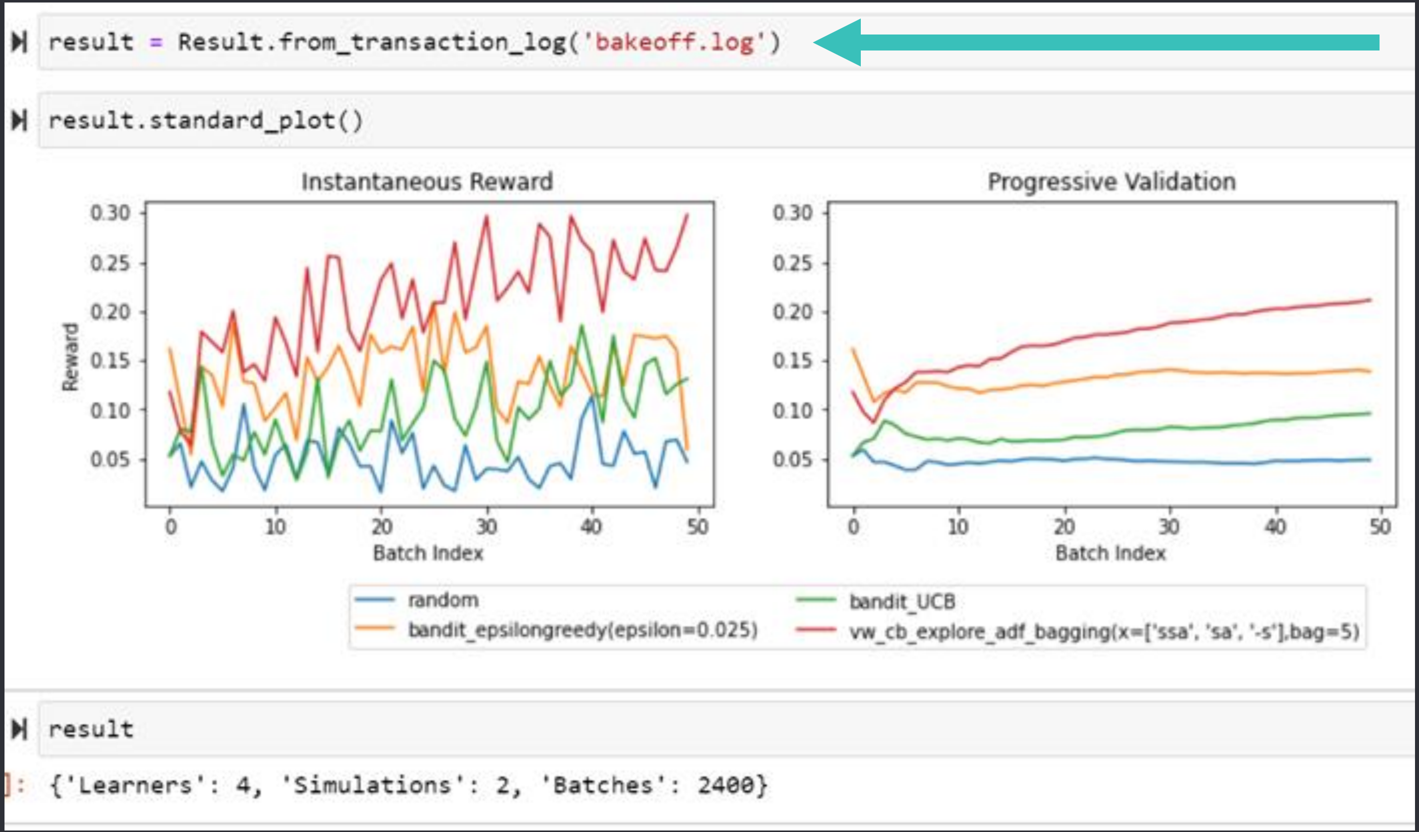
```
result.standard_plot()
```



```
result
```

```
]: {'Learners': 4, 'Simulations': 2, 'Batches': 2400}
```

In Depth Analysis



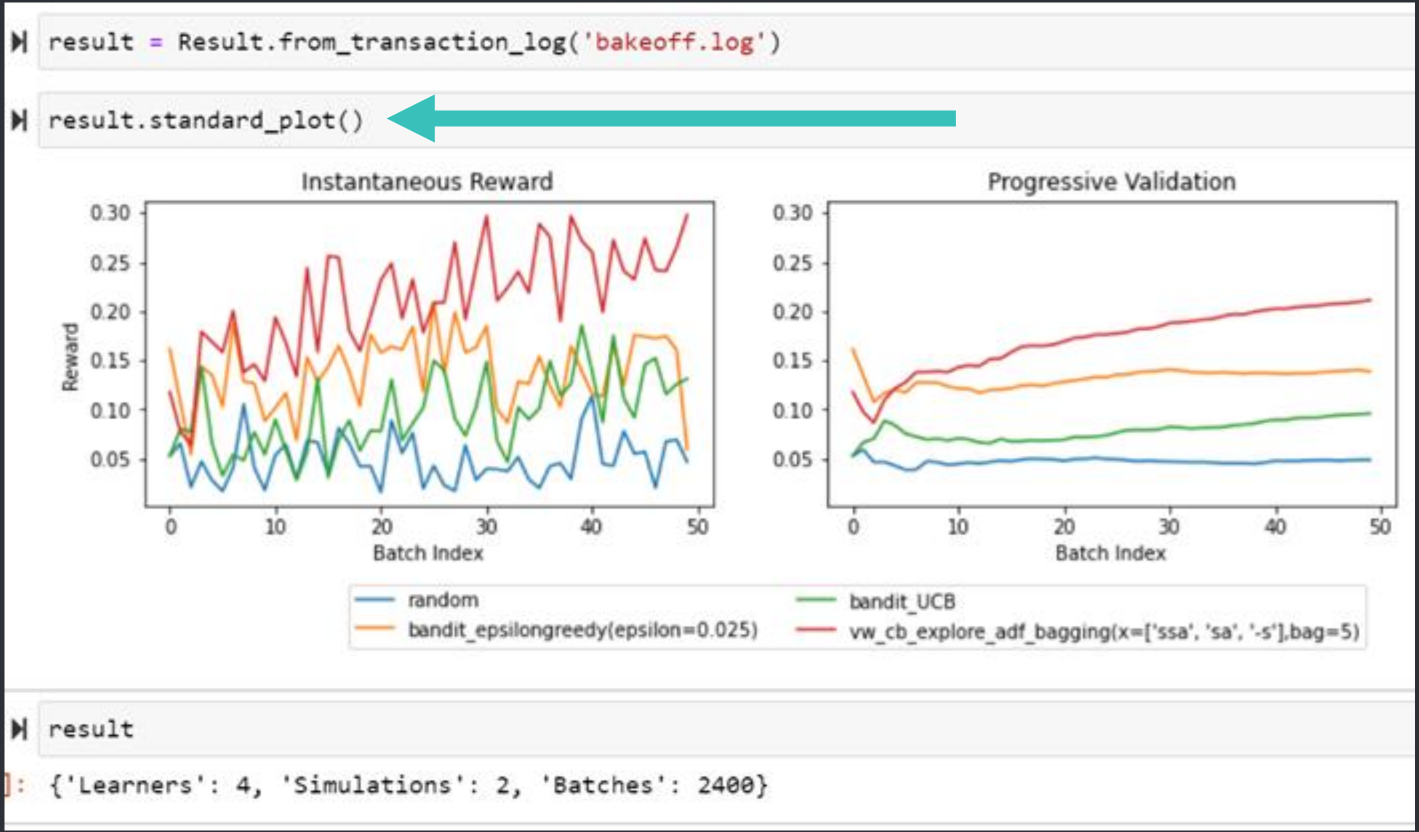
```
result = Result.from_transaction_log('bakeoff.log')
```

```
result.standard_plot()
```

```
result
```

```
{'Learners': 4, 'Simulations': 2, 'Batches': 2400}
```

In Depth Analysis



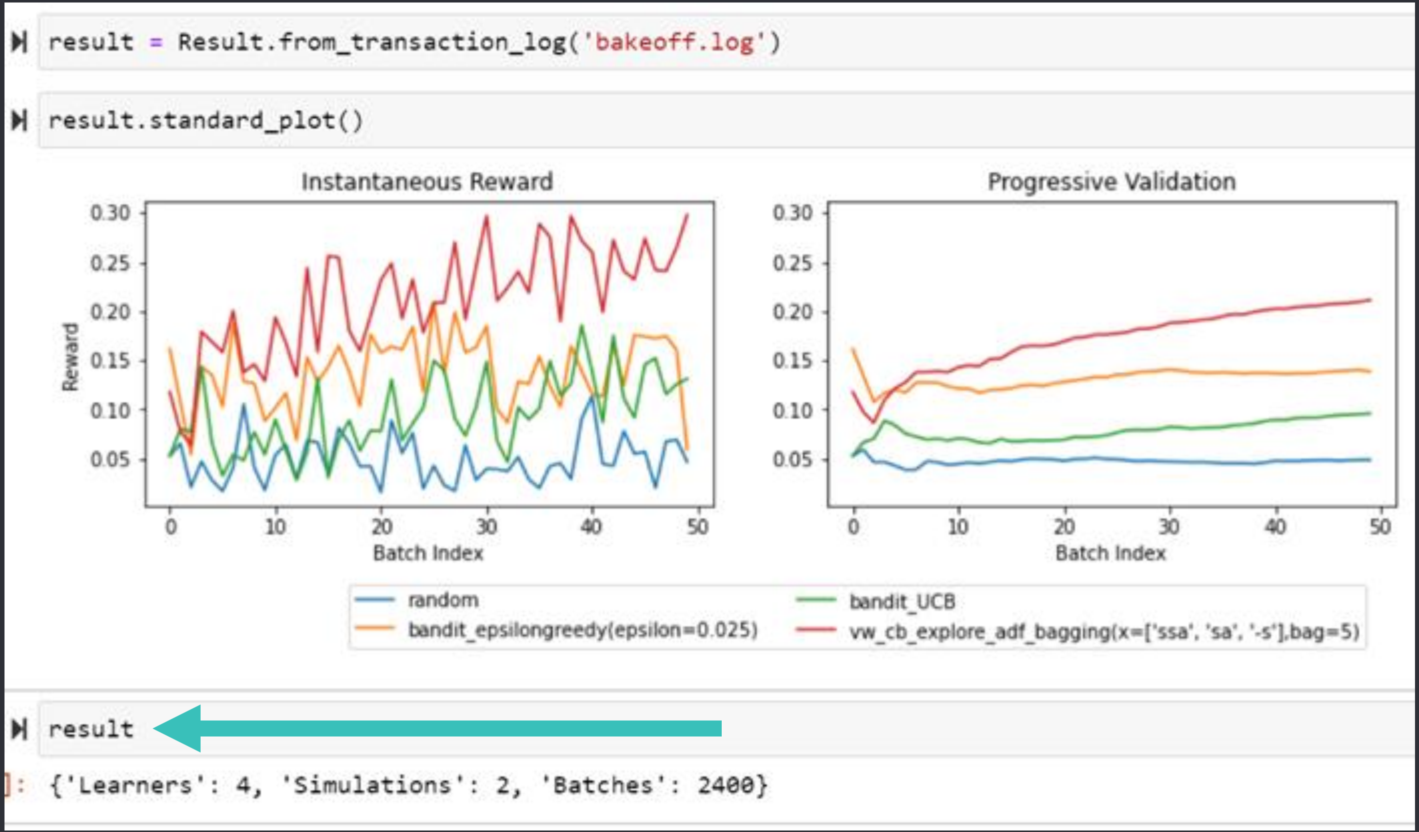
```
result = Result.from_transaction_log('bakeoff.log')
```

```
result.standard_plot()
```

```
result
```

```
]: {'Learners': 4, 'Simulations': 2, 'Batches': 2400}
```

In Depth Analysis



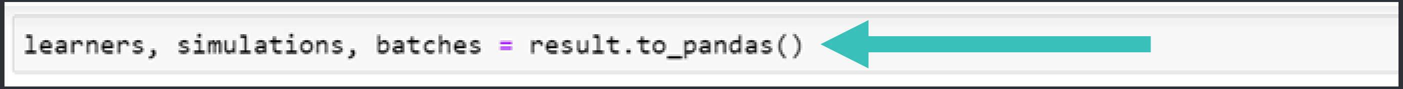
```
result = Result.from_transaction_log('bakeoff.log')
```

```
result.standard_plot()
```

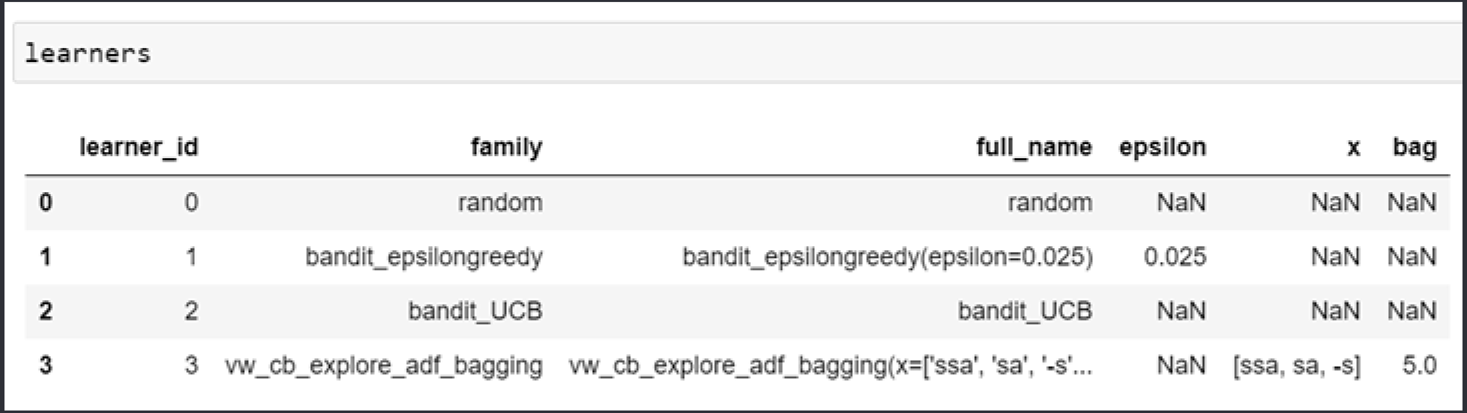
```
result
```

```
]: {'Learners': 4, 'Simulations': 2, 'Batches': 2400}
```

In Depth Analysis



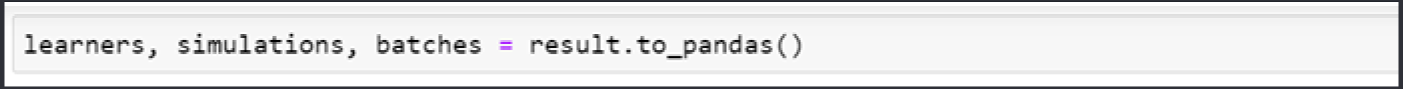
```
learners, simulations, batches = result.to_pandas()
```



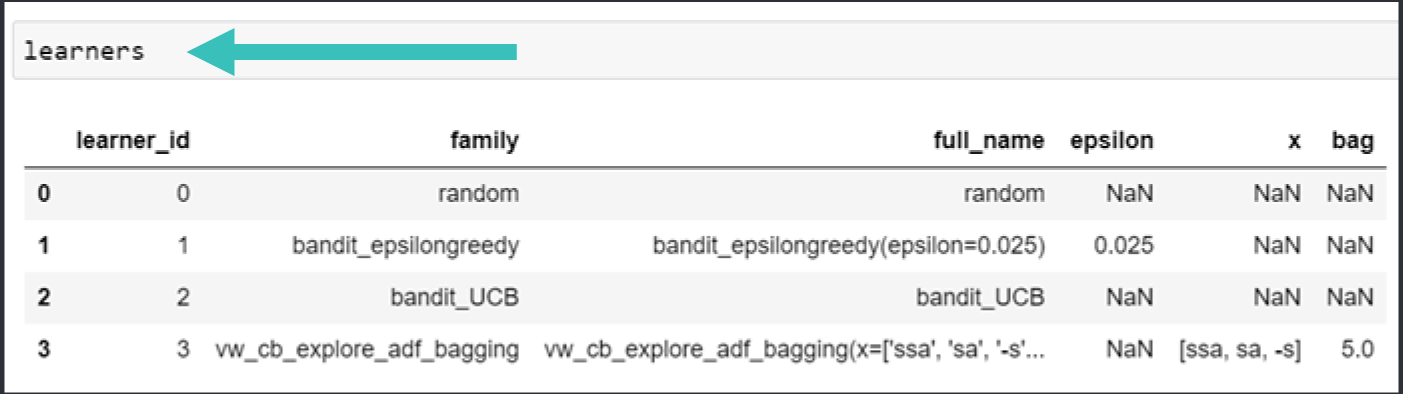
```
learners
```

	learner_id	family	full_name	epsilon	x	bag
0	0	random	random	NaN	NaN	NaN
1	1	bandit_epsilongreedy	bandit_epsilongreedy(epsilon=0.025)	0.025	NaN	NaN
2	2	bandit_UCB	bandit_UCB	NaN	NaN	NaN
3	3	vw_cb_explore_adf_bagging	vw_cb_explore_adf_bagging(x=['ssa', 'sa', '-s'...	NaN	[ssa, sa, -s]	5.0

In Depth Analysis



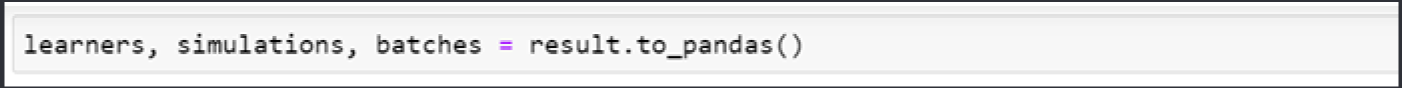
```
learners, simulations, batches = result.to_pandas()
```



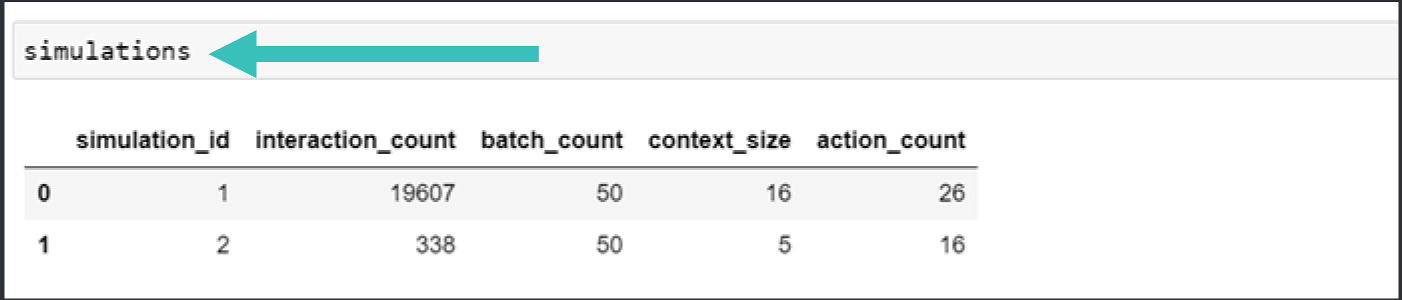
```
learners
```

	learner_id	family	full_name	epsilon	x	bag
0	0	random	random	NaN	NaN	NaN
1	1	bandit_epsilongreedy	bandit_epsilongreedy(epsilon=0.025)	0.025	NaN	NaN
2	2	bandit_UCB	bandit_UCB	NaN	NaN	NaN
3	3	vw_cb_explore_adf_bagging	vw_cb_explore_adf_bagging(x=['ssa', 'sa', '-s'...	NaN	[ssa, sa, -s]	5.0

In Depth Analysis



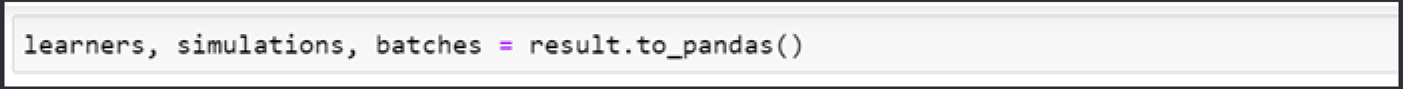
```
learners, simulations, batches = result.to_pandas()
```



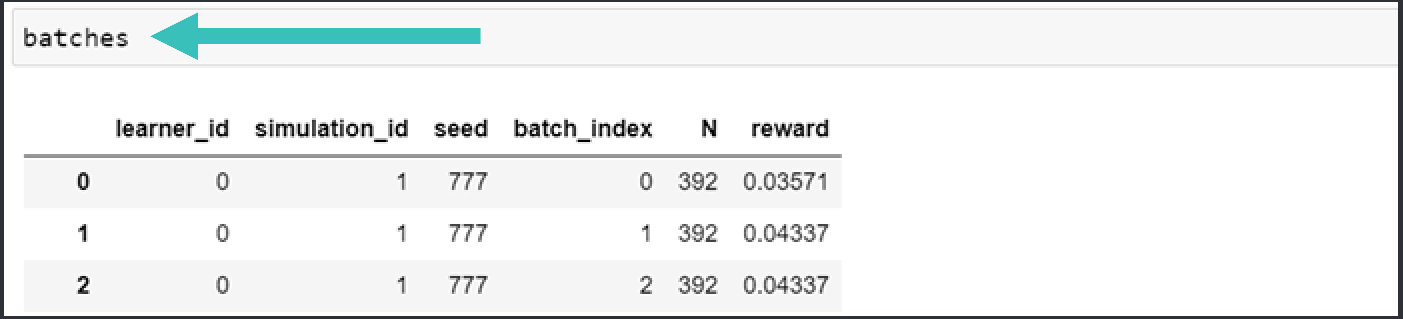
```
simulations
```

	simulation_id	interaction_count	batch_count	context_size	action_count
0	1	19607	50	16	26
1	2	338	50	5	16

In Depth Analysis



```
learners, simulations, batches = result.to_pandas()
```

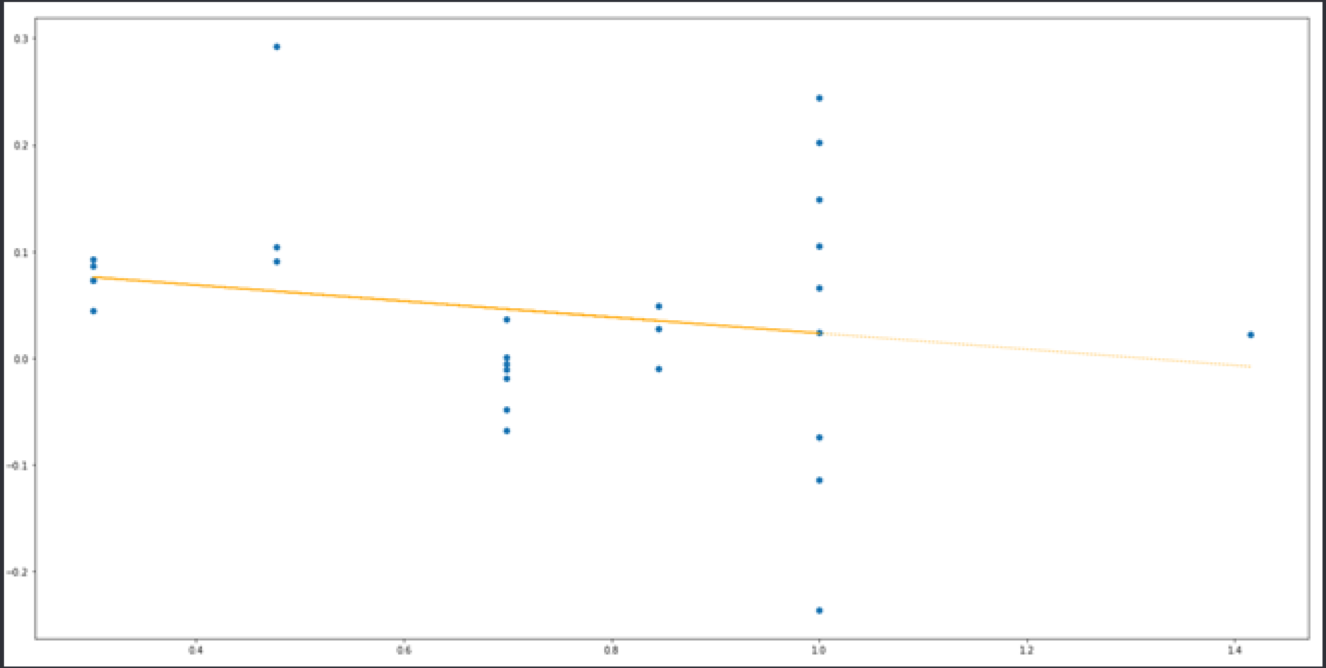


```
batches
```

	learner_id	simulation_id	seed	batch_index	N	reward
0	0	1	777	0	392	0.03571
1	0	1	777	1	392	0.04337
2	0	1	777	2	392	0.04337

In Depth Analysis

Baseline - Candidate

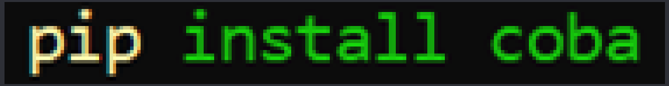


Log(|Actions|)

Other features

1. Efficient Multi-processing
2. Local caching of remote calls
3. Detailed Logs and Errors

Installing Coba



```
pip install coba
```

Required Dependencies

- Requests (for remote download)

Installing Coba

```
pip install coba
```

Optional Dependencies

- Matplotlib
- Pandas
- Vowpal Wabbit

What Can You Do?

1. Implement your own Contextual Bandit learner and compare it to Vowpal Wabbit.
2. Upload your own data set and see which Coba learner solves it best.
3. Get involved and contribute your own source code to the project

What’s Next for COBA

1. Improved Benchmark Configuration file notation
2. Improved onboarding documentation
3. Creation of a reference benchmark for contextual bandit research
4. More reference implementations of contextual bandit algorithms

Thank You!