A What Is Inside the Datasets?

Every dataset is repacked into HDF5 files similar to Fu et al. (2020). The data keys are described in Table 3; along to the typical (s_t, a_t, r_t, d_t) tuples, the metadata is also provided as the datasets' attributes with a comprehensive information about specific trajectories similar to Hambro et al. (2022b). The re-packing script is provided at `https://github.com/tinkoff-ai/katakomba/scripts/generate_small_dataset.py`.

Table 3: The re-packed datasets constitute of transformed data from Hambro et al. (2022b). Dissimilar the the large scale dataset, the repacked data is now in the format familiar to the ORL practitioners. We also save the metadata for each trajectory, for a comprehensive description, please, see Appendix F in Hambro et al. (2022b).

Name	Type	Shape	Description
`tty_chars`	`np.uint8`	[B, T, H, W]	s_t: The on-screen characters (default screen size 80 x 24).
`tty_colors`	`np.int8`	[B, T, H, W]	s_t: The on-screen colors for each character.
`tty_cursor`	`np.int16`	[B, T, 2]	s_t: The coordinates of the on-screen cursor.
`actions`	`np.uint8`	[B, T]	a_t: The NLE actions the player made in response to the s_t.
`rewards`	`np.int32`	[B, T]	r_t: The difference between in-game scores at states s_t and s_{t-1}. Note that this was used in all implementations of the algorithms provided in Hambro et al. (2022b). We also found that without this reward-shaping, all offline RL algorithms failed completely.
`dones`	`np.uint8`	[B, T]	d_t: An indicator whether the current state is the last one in the trajectory.

B License

C General Ethic Conduct and Potential Negative Societal Impact

To the best of our knowledge, our work does not present any direct potential negative societal impact.

As of the general ethic conduct, we believe that the most relevant issue to be discussed is the "Consent to use or share the data". Our work is largely built upon both the NetHack Learning Environment (Küttler et al., 2020) and the coresponding large-scale dataset (Hambro et al., 2022b), and as already described in the Appendix B both are distributed under the NETHACK GENERAL PUBLIC LICENSE that explicitly allows for re-usage and re-distribution.

D Resources and Statistics

We used 64 separated computational nodes with 1xA100, 14CPU, 128GB RAM, and the NVMe as long-term storage for all our experiments. All the values reported in the paper were also obtained under this configuration. One can also find more detailed information inside the Weights&Biases logs in the code repository.

Table 4: Scores used for Normalization. You can also find them at `https://github.com/tinkoff-ai/katakomba/katakomba/utils/scores.py`. For other statistics, please, see Table 2 in the main text.

Tasks	Minimum Score	Maximum Score	Mean Score
Base (Role-Centric)	-	-	-
arc-hum-neu	0.0	138103.0	6636.44
bar-hum-neu	0.0	292342.0	17836.68
cav-hum-neu	0.0	258978.0	12113.87
hea-hum-neu	0.0	64337.0	4068.27
kni-hum-law	0.0	419154.0	14137.06
mon-hum-neu	0.0	171224.0	17456.05
pri-hum-neu	0.0	114269.0	7732.69
ran-hum-neu	0.0	54874.0	8067.99
rog-hum-cha	0.0	68628.0	4818.20
sam-hum-law	0.0	155163.0	11009.36
tou-hum-neu	0.0	59484.0	4211.47
val-hum-neu	16.0	313858.0	18624.77
wiz-hum-neu	0.0	71709.0	5323.48
Extended (Race-Centric)	-	-	-
pri-elf-cha	0.0	83744.0	7109.35
ran-elf-cha	0.0	66690.0	9014.18
wiz-elf-cha	0.0	71664.0	5005.16
arc-dwa-law	0.0	83496.00	5445.69
cav-dwa-law	0.0	161682.0	11893.48
val-dwa-law	0.0	1136591.0	23473.61
arc-gno-neu	0.0	110054.0	5316.57
cav-gno-neu	0.0	142460.0	10083.06
hea-gno-neu	0.0	69566.0	3783.93
ran-gno-neu	0.0	58137.0	6965.04
wiz-gno-neu	0.0	37376.0	4317.51
bar-orc-cha	0.0	164296.0	17594.38
ran-orc-cha	3.0	69244.0	7608.48
rog-orc-cha	0.0	54892.0	4897.69
wiz-orc-cha	0.0	40871.0	5016.74
Complete (Alignment-Centric)	-	-	-
arc-hum-law	2.0	84823.0	5826.35
cav-hum-law	0.0	156966.0	12462.82
mon-hum-law	7.0	190783.0	16091.57
pri-hum-law	0.0	99250.0	6847.99
val-hum-law	0.0	428274.0	26103.03
bar-hum-cha	0.0	164446.0	18228.11
mon-hum-cha	0.0	223997.0	18353.30
pri-hum-cha	0.0	58367.0	8262.56
ran-hum-cha	3.0	62599.0	8378.50
wiz-hum-cha	0.0	55185.0	5316.82

E Hyperparameters

For all algorithms, hyperparameters have been reused from previous work whenever possible. For BC, CQL, and IQL reference values, see Appendix I.4 in the Hambro et al. (2022b). For AWAC, hyperparameters from IQL were reused due to the very similar policy updating scheme. For REM, hyperparameters were taken from the original work (see Agarwal et al. (2020)).

As in Hambro et al. (2022b), and in contrast to the original CQL implementation, we multiply the TD loss by the α coefficient instead of the CQL loss, as we observed better results with such a scheme. We performed a search for $\alpha \in [0.0001, 0.0005, 0.001, 0.05, 0.01, 0.05, 0.1, 0.5, 1.0]$ with best value $\alpha = 0.0001$.

Table 5: BC hyperparameters.

Parameter	**Value**
optimizer	AdamW (Kingma & Ba, 2014; Loshchilov & Hutter, 2017)
training iterations	500000
batch size	64
sequence length	16
learning rate	3e-4
weight decay	0.0
state encoder	Chaotic-Dwarven-GPT-5(Hambro et al., 2022a,b)
LSTM hidden dim	2048
LSTM layers	2
LSTM dropout	0.0
use previous action	True

Table 6: CQL hyperparameters. Note that in our implementation, the α coefficient multiplies the TD loss.

Parameter	**Value**
optimizer	AdamW (Kingma & Ba, 2014; Loshchilov & Hutter, 2017)
training iterations	500000
batch size	64
sequence length	16
learning rate	3e-4
weight decay	0.0
state encoder	Chaotic-Dwarven-GPT-5(Hambro et al., 2022a,b)
LSTM hidden dim	2048
LSTM layers	2
LSTM dropout	0.0
use previous action	True
tau (τ)	5e-3
gamma (γ)	0.999
reward clip range	[-10.0, 10.0]
alpha (α)	1e-4

Table 7: IQL hyperparameters.

Parameter	Value
optimizer	AdamW (Kingma & Ba, 2014; Loshchilov & Hutter, 2017)
training iterations	500000
batch size	64
sequence length	16
learning rate	3e-4
weight decay	0.0
state encoder	Chaotic-Dwarven-GPT-5(Hambro et al., 2022a,b)
LSTM hidden dim	2048
LSTM layers	2
LSTM dropout	0.0
use previous action	True
tau (τ)	5e-3
gamma (γ)	0.999
reward clip range	[-10.0, 10.0]
expectile	0.8
temperature	1.0
advantage clip max	100

Table 8: AWAC hyperparameters.

Parameter	Value
optimizer	AdamW (Kingma & Ba, 2014; Loshchilov & Hutter, 2017)
training iterations	500000
batch size	64
sequence length	16
learning rate	3e-4
weight decay	0.0
state encoder	Chaotic-Dwarven-GPT-5(Hambro et al., 2022a,b)
LSTM hidden dim	2048
LSTM layers	2
LSTM dropout	0.0
use previous action	True
tau (τ)	5e-3
gamma (γ)	0.999
reward clip range	[-10.0, 10.0]
temperature	1.0
advantage clip max	100

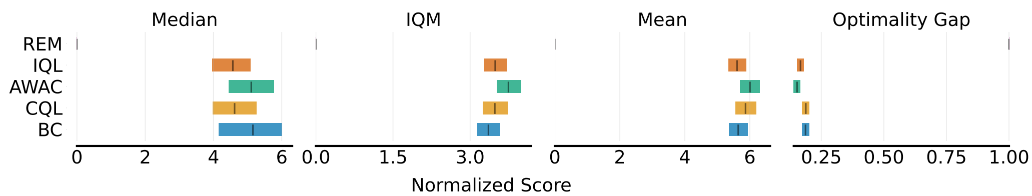
Table 9: REM hyperparameters.

Parameter	Value
optimizer	AdamW (Kingma & Ba, 2014; Loshchilov & Hutter, 2017)
training iterations	500000
batch size	64
sequence length	16
learning rate	3e-4
weight decay	0.0
state encoder	Chaotic-Dwarven-GPT-5(Hambro et al., 2022a,b)
LSTM hidden dim	2048
LSTM layers	2
LSTM dropout	0.0
use previous action	True
tau (τ)	5e-3
gamma (γ)	0.999
reward clip range	[-10.0, 10.0]
ensemble heads	200.0

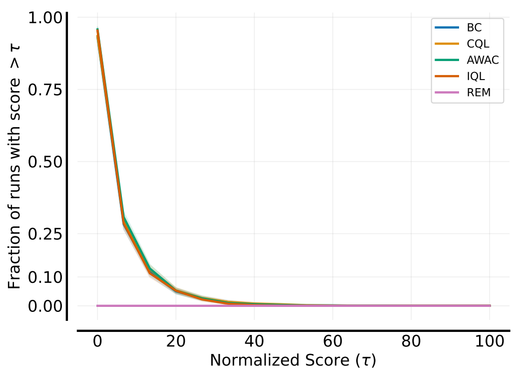
F Results per Benchmark Categories

In this section, we report the results stratified by the introduced categories. If one is willing to inspect specific datasets, we organized all training logs into Weights&Biases public reports, found at `https://wandb.ai/tlab/NetHack/reports`.

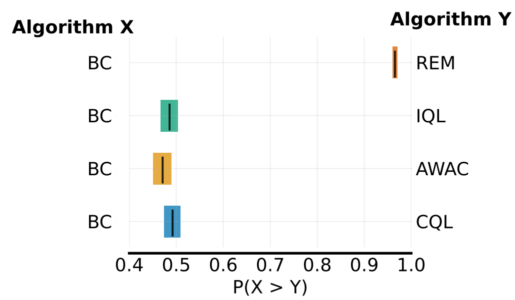
Note that one can find all the evaluation scores (for more than one checkpoint) within the runs and use them for any evaluation tools of interest. Also, we provide convenient scripts for constructing RLiable (Agarwal et al., 2021) graphs based on the provided runs that can be configured for one's purposes as well (see `https://github.com/tinkoff-ai/katakomba/scripts/rliable_report.py`).



(a) Bootstrapped point estimates.

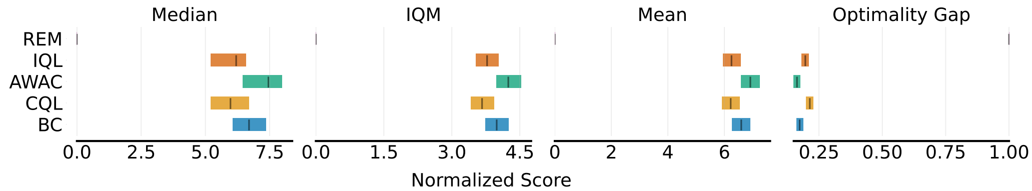


(b) Performance profiles.

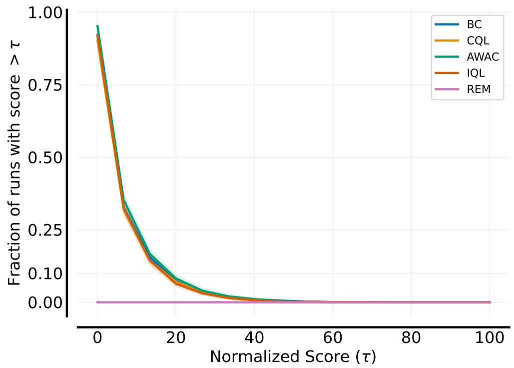


(c) Probability of improvement of BC to other algorithms.

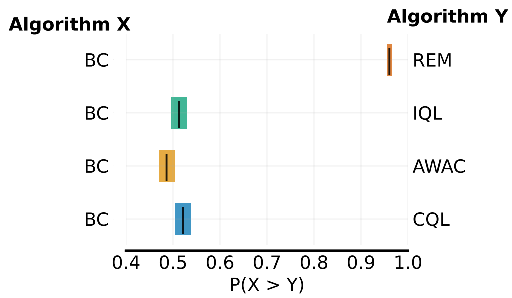
Figure 3: Normalized performance under the Katakomba benchmark for Base datasets. Each algorithm was run for three seeds and evaluated over 50 episodes resulting in 1950 points for constructing these graphs.



(a) Bootstrapped point estimates.

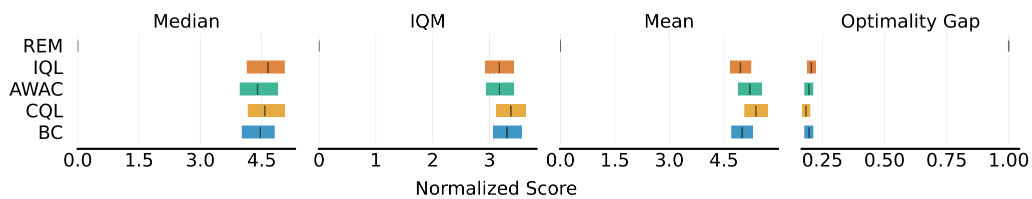


(b) Performance profiles.

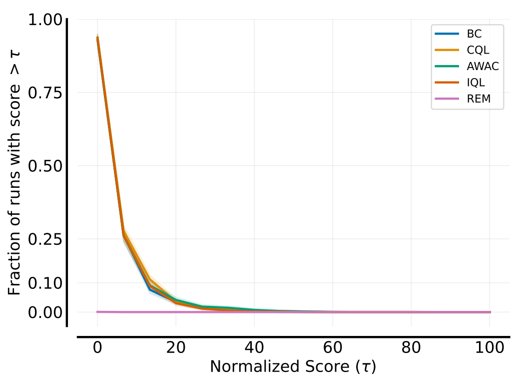


(c) Probability of improvement of BC to other algorithms.

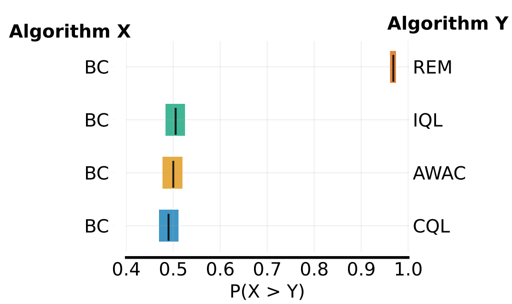
Figure 4: Normalized performance under the Katakomba benchmark for Extended datasets. Each algorithm was run for three seeds and evaluated over 50 episodes resulting in 2250 points for constructing these graphs.



(a) Bootstrapped point estimates.

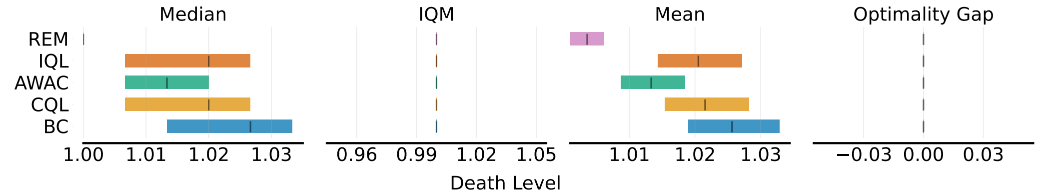


(b) Performance profiles.

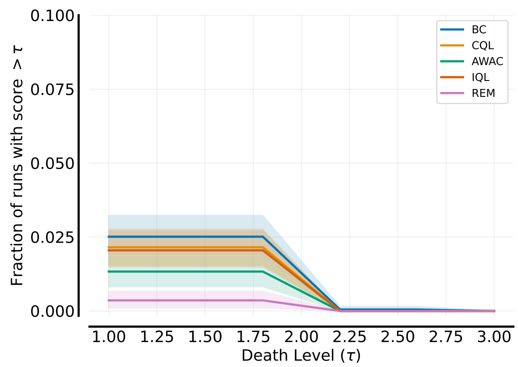


(c) Probability of improvement of BC to other algorithms.

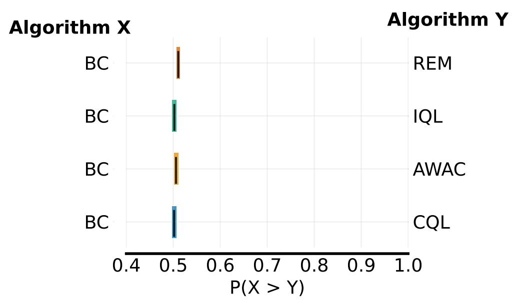
Figure 5: Normalized performance under the Katakomba benchmark for Complete datasets. Each algorithm was run for three seeds and evaluated over 50 episodes resulting in 1500 points for constructing these graphs.



(a) Bootstrapped point estimates.

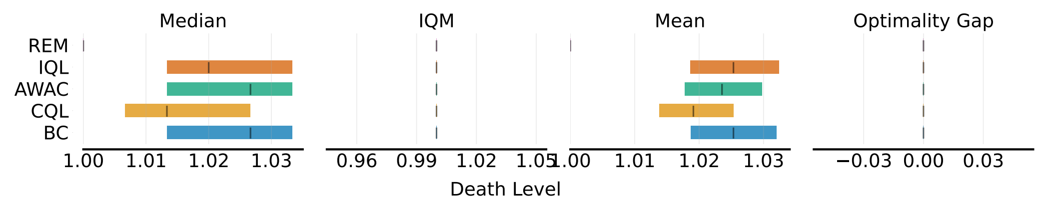


(b) Performance profiles.

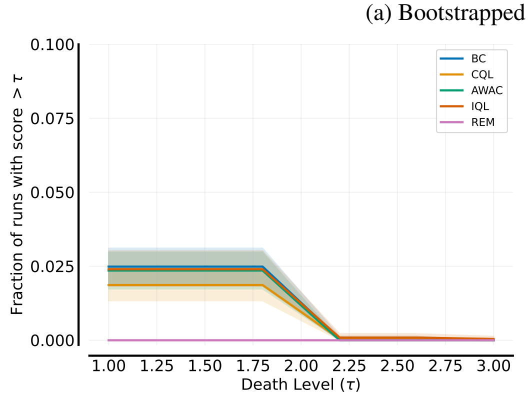


(c) Probability of improvement of BC to other algorithms.

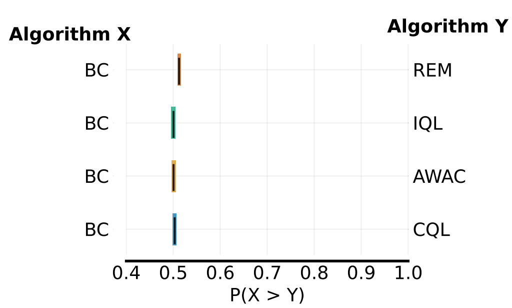
Figure 6: Death levels under the Katakomba benchmark for Base datasets. Each algorithm was run for three seeds and evaluated over 50 episodes resulting in 1950 points for constructing these graphs.



(a) Bootstrapped point estimates.

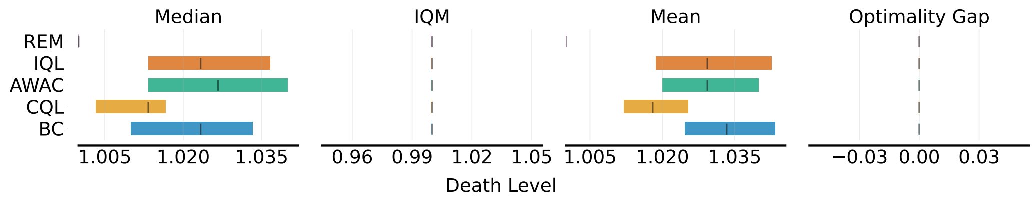


(b) Performance profiles.

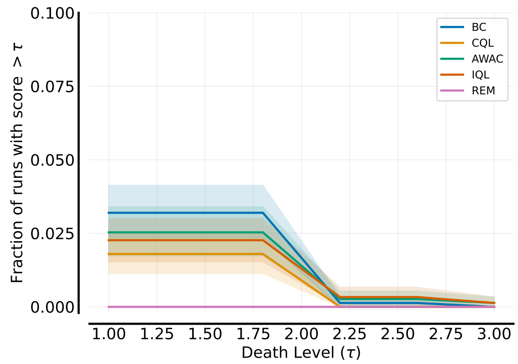


(c) Probability of improvement of BC to other algorithms.

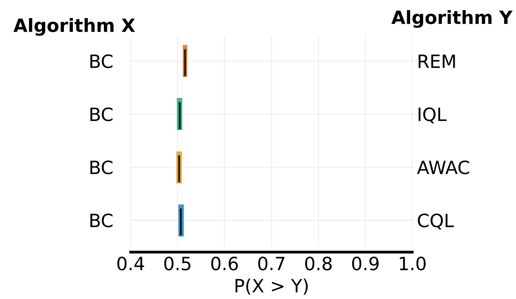
Figure 7: Death level under the Katakomba benchmark for Extended datasets. Each algorithm was run for three seeds and evaluated over 50 episodes resulting in 2250 points for constructing these graphs.



(a) Bootstrapped point estimates.

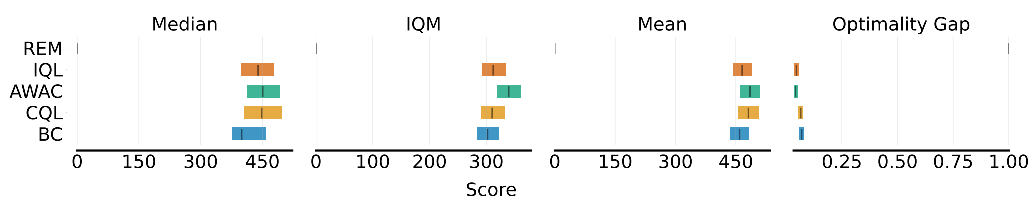


(b) Performance profiles.

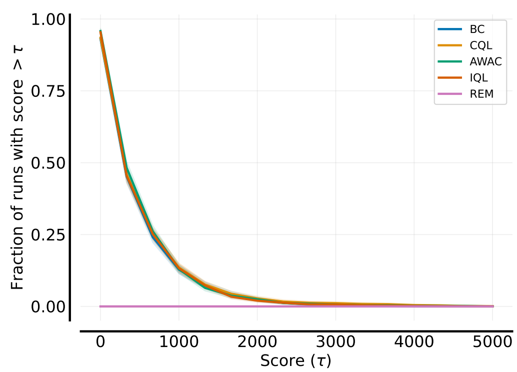


(c) Probability of improvement of BC to other algorithms.

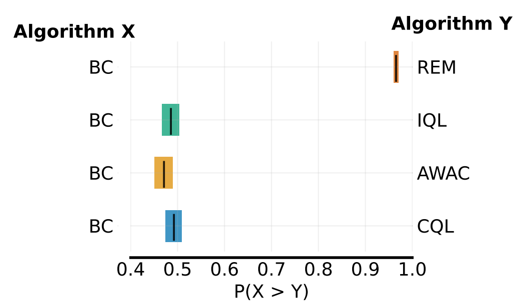
Figure 8: Death levels under the Katakomba benchmark for Complete datasets. Each algorithm was run for three seeds and evaluated over 50 episodes resulting in 1500 points for constructing these graphs.



(a) Bootstrapped point estimates.

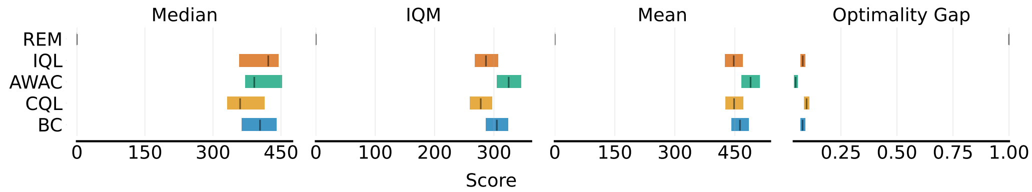


(b) Performance profiles.

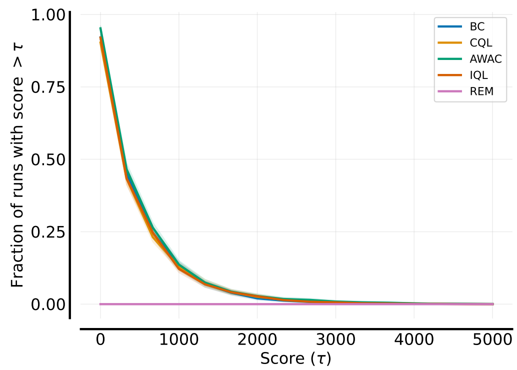


(c) Probability of improvement of BC to other algorithms.

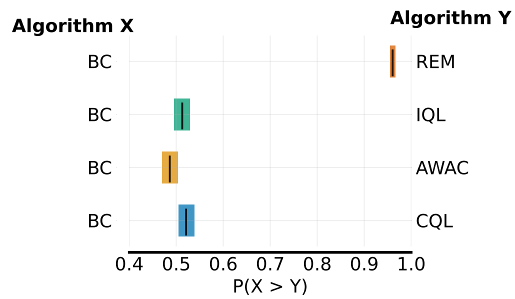
Figure 9: Unnormalized in-game score under the Katakomba benchmark for Base datasets. Each algorithm was run for three seeds and evaluated over 50 episodes resulting in 1950 points for constructing these graphs.



(a) Bootstrapped point estimates.

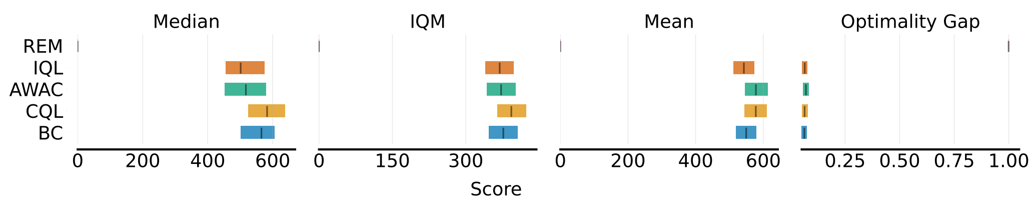


(b) Performance profiles.

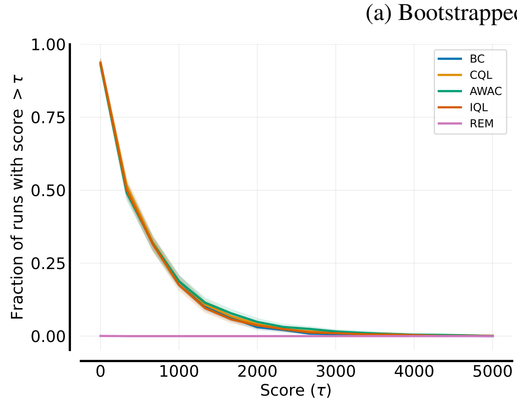


(c) Probability of improvement of BC to other algorithms.

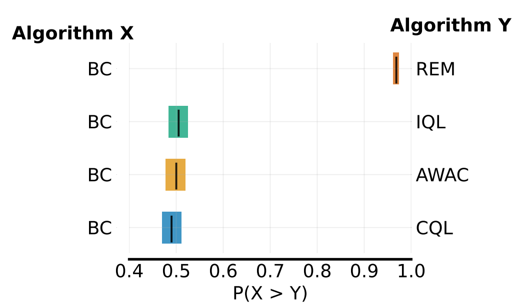
Figure 10: Unnormalized in-game score under the Katakomba benchmark for Extended datasets. Each algorithm was run for three seeds and evaluated over 50 episodes resulting in 2250 points for constructing these graphs.



(a) Bootstrapped point estimates.



(b) Performance profiles.



(c) Probability of improvement of BC to other algorithms.

Figure 11: Unnormalized in-game score under the Katakomba benchmark for Complete datasets. Each algorithm was run for three seeds and evaluated over 50 episodes resulting in 1500 points for constructing these graphs.