We would like to thank the reviewers for your thoughtful feedback and comments which would undoubtedly make the paper better. We will update our paper to reflect your comments, fix typos and include missing references. Here, we aim to address the concerns of each reviewer.

R4 & R5 – main contribution / insights: Our goal was to devise an effective algorithm that works well on a diverse set of offline RL problems. We consider this to be the main contribution of our paper: a simple algorithm which achieves good results on a large variety of environments/dataset, including challenging high-dimensional, partially observed problems from pixels. Although some components of our algorithm have been considered in related recent or concurrent work, this has usually been the case in the context of more complicated formulations. A secondary contribution of our work is thus a careful analysis and ablation that sheds light on the important attributes, and to identify a particularly simple but effective combination. We will update the paper to make this more overt.

R2 & R3 – motivation behind Eq. 3 and 4: One intuition for Eq. 4 (as explained in the subsection titled regularized policy iteration) works as follows. Through exponential weighting, we approximately regularize the policy by how much it diverges from the behavioral policy as measured by KL. The regularization would constrain the policy to not deviate too much from the support of the dataset thus ensuring effective offline learning. Eq. 4 is therefore chosen because of its connection to KL divergence. Both Eq. 3 and 4 are motivated by the policy improvement theorem. Whereas Eq. 3 seeks to improve the policy by choosing a better action to copy, Eq. 4 does this in a soft manner. We chose Eq. 3 because of its direct link to the policy improvement theorem. There indeed are many other choices of filtering functions and finding the best is an interesting direction of future research.

R2 – reproducibility: We have open-sourced the code for CRR on Github and the link will be made available.

R2 – network architecture: Due to the complexity of the locomotion datasets, we use relatively large networks. For consistency reasons, we adopt the same network architecture for the control suite dataset though smaller networks would suffice. We contend that other than being large in size, the networks used are not very unconventional in the RL community. The convolutional nets are actually borrowed from Impala and ACME where we only modified the activation functions. The residual MLPs are chosen for their expressive power.

R3 – bootstrapping: In the offline setting, the problem with bootstrapping arises when the current policy selects actions not in the support of the dataset. When this happens, the bootstrapping target can be inaccurate due to the lack of data. Since CRR's policies are learned by copying the actions of the dataset, the actions selected by CRR would be close to that of the dataset thereby making the problem much less severe. The fact that CRR does better than BC supports the claim the Q function is learned well. D4PG's poor performance (despite the same value learning rule) suggests naive strategies in learning the policy is not sufficient since it is not constrained to the support of the dataset.

R3 – weighting functions: It is indeed the case that the two weighting functions produce different results. This finding makes it clear that not all weighting functions are created equal and thus opens new doors to future research. As offline RL algorithms cannot escape the existence of hyper-parameters, the choice of weighting functions is treated as one additional hyper-parameter.

R4 – advantage estimation: We agree that adding constants which depend only on s_t to f in Eq. 2 (i.e. $f + g(s_t)$) does not make a difference. However, f is defined as a nonlinear transformation of the advantage estimates. A_{max} underestimates the advantage where as A_{mean} does not. Thus, using A_{mean} vs. A_{max} leads to different outcomes.

R4 – convergence of Q: Our algorithm performs standard policy evaluation to learn Q for the current policy, effectively as an inner loop (Eq. 1). Thus, if our algorithm converges to a pair (π, Q), Q will estimate the value of the policy, i.e. Q^π (restricted to the data in the batch and subject to the usual caveats of RL with nonlinear function approximation). In the tabular setting, under some mild regularity conditions, CRR constitutes policy improvements in a restricted MDP similar to that defined in BCQ [10]. Therefore each iteration of CRR will result in a better policy. Via similar arguments for the convergence of policy iteration, Q of CRR would converge to one corresponding to an optimal policy for the restricted MDP. The theoretical results would be included in an updated version of the paper.

R4 – additional baselines: Thanks for raising the question. We would like to point out that our experiments already include several baselines that are considered to be the state of the art at the moment of writing. In our experiments, we also examine elements of the baselines and compare it to the equivalent bits in our method. For example, in Appendix A.2 we evaluate the effect of K-step returns used by AWR and MARWIL.

R4 – novelty: Our main contribution is a simple and effective algorithm that identifies important ingredients of offline learning algorithms (see response "Main contribution" above). We do not claim that the use of distributional Q learning is a contribution per se. We will make this clearer in the text. CRR uses a different advantages estimate compared to [35]. We include a toy example (in subsection titled "CRR vs. return-based methods.") that shows how the approach of [35] can fail. We also show empirically (in appendix A.2) that our advantage estimation methods significantly outperforms that adopted by [35].