We thank the reviewers for the detailed comments and suggested improvements.

Reviewer 1: Estimate of OPT. In all our algorithms, it suffices to know the optimum value up to a constant (say 2). Thus if we know a range for the value of OPT, one can perform a binary search. For instance, if we know that it lies in the interval $(1/n^{10}, n^{10})$ (a fairly large range), the search takes $O(\log n)$ time. Two early (arbitrarily chosen) examples of guessing the optimum in clustering problems are: *Clustering to Minimize the Sum of Cluster Diameters* (Charikar, Panigrahy, 2001), *A fast k-means implementation using coresets* (Frahling, Sohler, 2005). We will include more details about this step in the final version (possibly in the supplement).

Reviewer 1: Typos. We thank the reviewer for pointing these out. We will correct (2) and (3). As for (1), the number of centers needs to be $(1+c)k$ as opposed to ck. We will correct this in the statements of theorems 1.2 and 1.4. This is why the theorems do not subsume theorems 1.1 and 1.3.

Reviewer 2: Comparison to prior work. We will compare and reference the works suggested by the reviewer. Indeed the works cited, as well as other "data reduction" approaches have been crucial to the development of algorithms for clustering. As our focus was on adaptive sampling approaches, we had not referred to those works earlier.

Algorithm of (*) is better than Theorem 1.3: This is indeed the case if "nearly linear time" is the main goal. However, note that the algorithm of (*) is based on iteratively reducing the size of the data, and is much more involved to describe. Meanwhile, our focus is to show that a simple variant of k-means++ itself achieves similar (though slightly worse guarantees). This is analogous to the case of vanilla (without outliers) k-means. Further, the bounds in Theorem 1.4 improve the approximation factor, albeit using more centers.

Reviewer 2: Analysis of k-center vs k-means. We will highlight at least some of the ideas involved in the k-means analysis in the body of the paper. The analysis is much more challenging because in k-means, it is no longer simply a matter of "covering" a cluster (i.e., choosing different points in a cluster lead to significantly different objective values for the other points). The lemmas in sections A.2 and A.3 of the supplement address this challenge.

Reviewer 2: Running time analysis. We will add this in the final version. The run time is $O(nk)$, the same as that for k-means++, as long as we have an estimate for the optimum value. Guessing that adds an extra logarithmic factor.

Reviewer 2: Experiments. In the final version (possibly in the supplement), we will add the details about the hyperparamters used in the synthetic experiments and in the noise addition step for real data. We will also perform experiments on the kdd-cup dataset (using only the numeric features and normalization as suggested in (**)).

Reviewer 2: Other comments. We will clarify the statements in the second paragraph of the introduction (latter line should say that a polynomial time approximation scheme is ruled out). The use of ℓ is because it is set to $(1+c)k$ in the bi-criteria algorithms.

Reviewer 3: Comparisons. As discussed above, we will include more comparisons (in both running time and approximation factors (a, b, c)) with prior works. A short summary is as follows: if one is only concerned with *polynomial* running times, one can achieve $a = c = 1$ and $b = O(1)$ (Krishnaswamy, Li, Sandeep, STOC 2018). Using iterative "data reduction" approaches (cited above), one can achieve $c = 1$ while having $a = b = O(1)$, with the $O(1)$ term having a trade-off with the running time. Our algorithms (i) avoid such tradeoffs, and (ii) are simple modifications of well-studied greedy update procedures.

Reviewer 3: Lower bounds. The result of Krishnaswamy et al. (above) shows that $a = c = 1$ and $b = O(1)$ is indeed achievable. It is an interesting open question if the constant b is worse for the outlier version of the problem. As for our algorithms, there are examples (based on the tight examples for k-means++) that indeed show that our *analysis* is tight. Thus improvements must come from more involved algorithms.