Additional file 1

<u>Variant conversion</u>

We coined the term *variant conversion* for cases where a true positive variant is phased with a nearby true or false positive variant, leading to miscalls. We have observed significant variant conversion when the variant calling algorithm is not constrained in the merging of nearby mismatches into primary MNP variant calls (see <u>Considerations for variant calling algorithm design</u>). Erroneous variant calls due to conversion are dependent on the product of 1) the probability of a true variant occurring at a given position, and 2) the probability of error in a window surrounding the position. For satmut_utils, this window is 3 nt. However, other variant callers may not impose any window constraint when merging mismatches into MNP; in this case, variant conversion can be a significant source of both false positives and false negatives.

<u>Variant frequencies in saturation mutagenesis libraries</u>

Variant frequencies are primarily dictated by the number of species in the variant library. As the size of the mutagenized target region increases, frequencies are driven lower as each variant-containing read contributes to depth at other target positions covered by the read. We recommend generating variant libraries that do not exceed approximately 5000 species, so that SNP variant frequencies are appreciably higher than typical PCR error rates (1×10^{-6} to 1×10^{-4}) and so that libraries do not need to be sequenced with coverage depths greater than $\sim 1 \times 10^{6}$ per target position. Another consideration is that MNPs are generally represented at lower frequencies than SNPs in variant libraries, due to lower mutagenesis efficiency by virtue of unfavorable thermodynamics of primer annealing. Nonetheless, MNPs are more likely to represent true variants over SNPs, as the probability of observing two and three mismatches to the reference is significantly less than the probability of observing a single mismatch in near reference-space. Thus, when designing experiments, one should consider not only the number of variants to mutagenize, but also the relative composition of SNPs and MNPs in the library.

<u>Comment on existing simulation tools</u>

Several solutions exist for simulating sequencing reads with variants [95–98]. These have successfully been used for simulating cancer genomes to assess germline and somatic variant calling tools [19,20,99–103]. Most of the simulators learn an error model from real alignments and then generate reads and variants starting from a reference sequence. In contrast, BAMSurgeon [24] and satmut_utils edit variants into real alignments. This approach captures the native error profile attributable to the library preparation and sequencing methods.

Simulation algorithm design

Using read sampling and a heuristic, satmut_utils ‘sim’ addresses the problem of simulation of many low-frequency variants at the same (and nearby) positions, while enforcing that no read pair is edited more than once. To avoid variant conversion, ‘sim’ provides the –edit_buffer argument to enable the user to only simulate variants in read pairs when the read matches the reference sequence +/- the edit buffer about the variant coordinate(s). This allows the user to control how “idealized” simulation is, which may affect interpretation of downstream variant calling performance. For example, if a low-frequency variant is simulated with only one count, and the randomly selected read pair to edit has an error nearby within the edit buffer, the variant may not be called (depending on the variant calling algorithm). In this case, increasing the edit buffer ensures the variant is simulated in a read pair without creating a higher order variant (e.g. SNP to a di-nt MNP composed of one true variant and one error).

Simulation requirements for assessing performance of consensus deduplication

The satmut_utils ‘sim’ workflow currently does not meet the requirements to measure the sensitivity-specificity tradeoff of consensus deduplication. To do this, reads must be grouped by unique molecular index (UMI) prior to simulation, and then the ‘sim’ workflow would need to make use of this information to ensure that all duplicates within a UMI group are edited. (Otherwise, the variant is not retained in the consensus). Additionally, the number of read pairs to edit- determined from the configured variant frequency- would be different for deduplicated and non-deduplicated alignments and would not be comparable between the two; this is due to deduplication effects on local coverage depth, which modulate the frequency of the variant. Thus, the requirements to properly simulate variants to measure the improvement of consensus deduplication pose challenges that preclude its support. The use of biological standards of known variant composition are better suited to measure performance gains from deduplication.

Considerations for variant calling algorithm design

A primary consideration for the algorithmic design of a MAVE variant caller is how to implement MNP calling. Specifically, should MNP calls be made across the entire read or confined to a smaller window? The downside of imposing no window constraint is that mismatches spanning greater than a small window are more likely to be false positives due to variant conversion. When simulating MNPs with large spans (haplotypes), and calling variants with a prototype algorithm that does not require mismatches to be within a span/window of 3 nt, we observed that the number of false positive variants scaled rapidly as the window size increased. Even with a relatively short window of 10 nt, the number of false positive variants exploded due to variant conversion and merging of errors.

Because mutagenesis primer lengths are almost always >10 nt, it is unexpected to observe phased variants shorter than a typical primer length (20 nt) if mutagenesis is carried out by a single primer extension reaction with pooled mutagenesis oligos [4]. Thus, we opted to only merge mismatches within a 3 nt window, which implies true haplotypes with mismatches spanning 4 nt or greater will not be called together by satmut_utils. We conclude that without improved variant calling algorithms and/or error correction methods, long-range, low-frequency haplotypes can not be reliably called without incurring a high cost of false positives.

Benchmarking considerations

We expect that the lack of benchmarking from prior MAVE variant callers is due to difficulty of configuration and specific constraints on input data. For example, to successfully benchmark against Enrich2 and dms_tools2, we wrote a script to 1) filter reads containing InDels, 2) make reads flush with codons by trimming and/or appending reference sequence, and 3) add barcodes/UMIs to the 5' end of each read. Although an alignment strategy (DiMSum, satmut_utils) is more computationally expensive, it imposes minimal constraints on primer design or library preparation method.

While nucleotide strings are available in DiMSum output, to benchmark DiMSum nucleotide-level calls, we would need to write post-processing code to identify the mismatches in the nucleotide string that define the variant. This would make DiMSum performance partially based on our own logic of variant calling. Thus, we decided to compare only amino acid changes for DiMSum to ensure an unbiased analysis.

One explanation for relatively poorer precision of Enrich2 in benchmarking was the use of its Basic mode, which does not take into account support from both reads. We attempted to use Enrich2 Overlap mode, but found a high proportion of variant calls were unresolvable by this mode (https://github.com/FowlerLab/Enrich2/issues/45), precluding analysis of most simulated variants. Another reason Enrich2 and DimSum may have poorer precision than satmut_utils is that the former variant calling methods do not put a constraint on the merging of mismatches into MNP calls (see Considerations for variant calling algorithm design).

Lastly, one other possible explanation for differences in performance is that analysis parameters for read filtering (e.g. base quality threshold) are implemented in different ways for each variant caller. Nonetheless, in benchmarking, efforts were made to make analysis parameters as similar as possible between variant callers.

Time and memory consumption
6,359,057 read pairs from two amplicons analyzed with satmut_utils 'call' (with primer base quality masking, no consensus deduplication, –nthreads 10) took 42 min and consumed a maximum of 142.2 Mb memory.

42,900,903 read pairs from fifteen tiles (RACE-like method) analyzed with satmut_utils 'call' (with primer base quality masking and consensus deduplication, –nthreads 10) took 16 h and consumed a maximum of 165 Mb memory.

We have successfully analyzed up to 67 million read pairs (RACE-like method, primer base quality masking, consensus deduplication, –nthreads 10), which took nearly 47 h. Consensus deduplication can take significant time. We attempted to use gencore [29], which is written in C++, however gencore does not handle RACE-like library preparation methods and yielded much lower unique fragments counts after deduplication compared to umi_tools 'dedup' and satmut_utils consensus deduplication (data not shown). We opted to use umi_tools for UMI grouping and satmut_utils as a flexible implementation for consensus deduplication of grouped reads.

Error correction model filtering
satmut_utils call extracts numerous quality features for the mismatches comprising both SNP and MNP variant calls. That is, for MNPs, quality features are provided for each mismatch in a primary variant call. This allows filtering of MNPs composed of a true variant and an error, which should help remediate variant conversion. After application of ML models for filtering individual mismatches, the user must appropriately filter MNP variants with both true and false positive predictions. That is, if a component mismatch of a MNP call is determined to be false by the error correction model, the user may manually filter the encompassing variant call, using prior

knowledge of the original variant type (di-nt MNP, tri-nt MNP) and how many mismatches were retained in the variant call after filtering.

Caveats of error correction models

While our results suggest binary classifiers can learn error signatures in each experiment, there is a risk for bias when applying models to real data. Models trained on simulated data are unlikely to generalize on real datasets if the ‘sim’ parameters (variant frequencies, SNP/MNP composition) are not representative of the true probability distributions for real mutagenesis libraries. Furthermore, regarding the high performance of error correction models, we caution that the design rules of satmut_utils ‘sim’ impose constraints on read editing (mate pair overlap, edit distance), that may be subsequently learned by downstream models. We also did not include ‘sim’ hyperparameters under the scope of cross-validation. Together, these limitations may lead to optimistic performance measures [104].

Potential bias of variants in terminal amplicons by the RACE-like method

Position-specific biases have been noted before, as well as methods developed to remediate these biases [61]. We chose not to normalize for position-specific biases in RACE-like data primarily because normalization procedures can mask the impact of true biological variation and technical artifacts that may arise in different amplicons and protein domains. That is, using position-specific normalization removes the impact of global variant effects manifested in larger domains of the target. For instance, nonsense variants that occur in the linker region of CBS between the catalytic and regulatory domain still produce a functional “core” CBS protein with higher stability and activity than full-length CBS [4,32]. Altogether, use of amplicon- and/or position-specific normalization may remove true biological variation, which is a caveat noted by other normalization methods [105].