Supporting Information Appendix

Hoang et al., PNAS 2016

SI Materials and Methods

Dilution of Y-adapter-ligated molecules. TruSeq adapter-ligated molecules were in a total volume of 20 µL. Five ten-fold serial dilutions were performed in 96-well PCR plates starting with 2 µL of adapter-ligated molecules (prior to PCR) in 18 µL of dilution buffer (TE containing 1 ng/µL pBlueScript). Samples were mixed by gently pipetting with a multichannel pipette. Two µL of each sample was then transferred into 18 µL of fresh dilution buffer using a multichannel pipette. The mixing and transferring was repeated for a total of five serial dilutions. Only 2 µL of each dilution (1/10 total volume) was used as template for each PCR. A 10^3-fold dilution was accomplished as follows: (i) use of 2 µL of the total 20 µL of adapter-ligated molecules (10-fold dilution); (ii) mixing 2 µL of adapter-ligated molecules with dilution buffer in a total volume of 20 µL (10-fold dilution); and (iii) use of 2 µL of diluted adapter-ligated molecules from the total 20 µL volume in the PCR reaction (10-fold dilution, see below). The five serial dilutions resulted in final dilution factors of 10^3, 10^4, 10^5, 10^6, and 10^7. The minimum amount of input DNA per library was 3.4 ng.

PCR amplification of diluted Y-adapter-ligated molecules. Custom HPLC-purified PCR primers (IDT), TS-PCR Oligo1 (5'-AATGATACGGCGACCACCGAG*A) and TS-PCR Oligo2 (5'-CAAGCAGAAGACGGCATACGA*G), were designed with one phosphorothioated bond (*) at the 3' end. PCR was performed in 50 µL total volume with 0.5 µM TS-PCR Oligo1, 0.5 µM TS-PCR Oligo2, Q5 2X HotStart High-Fidelity Master Mix (NEB) at 1X final concentration, and 2 µL of diluted adapter-ligated molecules as template. PCR was performed in Thermo HyBaid PCR Express HBPX Thermal Cycler. The following PCR program was used: 1) 98°C for 30 s 2) 98°C for 10 s, 69°C for 30 s, 72°C for 30 s for 18 or 20 cycles (see Table S1), and 3) 72° for 2 min. PCR reactions were purified with AMPure XP (Agilent) at 1.0X bead-to-sample ratio according to the manufacturer's protocol.

MiSeq run and analysis. A subset of amplified BotSeqS sequencing libraries was evaluated on an Illumina MiSeq instrument (~5 M clusters passed filter per library) to empirically deduce the optimal dilution. The "optimal dilution" was determined to result in 5 to 10 PCR duplicates per molecule when scaled to 1/2 lane of a HiSeq 2500 instrument (~70 M clusters passed filter per library in rapid run mode). For example, for an input of 500 ng gDNA into the TruSeq PCR-free library prep (selecting for 350 bp insert size), amplified libraries from the 10^4-, 10^5-, 10^6-fold dilutions were sequenced at 2 x 50 bp depth on MiSeq. Three different well-barcoded samples (which were also molecularly barcoded) were multiplexed in one MiSeq lane to test three dilutions of each sample. The .bam output files were uploaded into Galaxy, and Picard's Estimate Library Complexity Tool (Galaxy Tool Version 1.56.0) was executed using the default parameters. Optimal dilutions showed distributions ranging from one to four members per family with singletons comprising ~60-80% of total counts. In general, with an input of 500 ng of gDNA into the TruSeq PCR-free library prep, the 10^5-fold dilution yielded

~10 members per family on a subsequent HiSeq run used for BotSeqS. From our sequencing data, we estimate the average number of high quality clusters required to identify one rare mutation in colonic tissues was (1) 30 M in a normal child, (2) 12 M in a normal young adult, and (3) 5.8 M in a normal old adult.

Whole-genome sequencing. Thirty-two whole-genome sequencing (WGS) libraries were generated from the 34 individuals in this study. In the remaining two individuals without WGS, COL238 and COL239, Sanger sequence was performed to exclude clonal variants in the BotSeqS data. Of the final 20 µL of adapter-ligated molecules used to prepare BotSeqS libraries (prior to dilution), 10 µL was used to amplify a library for whole-genome sequencing using TruSeq PCR Primer Cocktail (Illumina) and TruSeq PCR Master Mix (Illumina) according to TruSeq PCR protocol. PCR reactions were purified with AMPure XP (Agilent) at 1.0X bead-to-sample ratio according to the manufacturer's instructions. The libraries were PE sequenced 2x100 bp on Illumina HiSeq 2500 at >30x coverage.

Characterization of BotSeqS sensitivity. Two DNA mixtures were prepared from the DNA of normal spleen samples PEN93 and PEN95. Whole genome sequence data was available from these two samples (1) and SNPs in PEN93 that were not present in PEN95 could be identified. Both mixtures contained the same amount of PEN95 DNA, but the low spike-in mix contained only 10% of the PEN93 DNA contained in the high spike-in mix. BotSeqS libraries from these samples were first analyzed using the normal BotSeqS pipeline to minimize clonal and germline mutations. Only a total of two mutations were detected among the two libraries; these two mutations likely represented rare mutations in the PEN95 sample, and suggest a mutation prevalence of ~8×10^{-7} mutation/bp. Next, the data were processed through the BotSeqS pipeline *without* filtering out mutations that were present in dbSNP (build 130 and 142). Seven PEN93-specific SNPs in the low spike-in and 89 PEN93-specific SNPs in the high spike-in mixtures were identified. After normalizing for the number of sequenced bases, the "mutation prevalence" (number of PEN93-specific SNPs/bp) was 2.7×10^{-6} for the low spike-in and 2.0×10^{-5} for the high spike-in samples. The difference between the low spike-in and the high spike-in was 7.4-fold, within the range expected from the 10-fold dilution given the relatively low number of mutations identified in the low spike-in sample.

Sensitivity is one to a few cells, as follows: one human cell has ~1.3e7 nuclear DNA "molecules", estimated based on each molecule being 500 bp in size (approximate library insert size). Based on the amount of DNA originally purified from each tissue (corresponding to a median of 1.2 M cells), the BotSeqS dilution process resulted in the assessment of ~60,000 nuclear DNA molecules per library. One way to assess sensitivity in terms of cells is to calculate the probability that, given we have picked a molecule from a specific cell, at least one other DNA molecule in the BotSeqS library was from the same cell. This can be modeled as a conditional binomial probability (using the *pbinom* functions of the *R* software): (1- pbinom(1,60000,1/1200000))/(1-pbinom(0,60000,1/1200000)) = 0.025. The probability that at least 3 molecules were from the same cell is then (1- pbinom(2,60000,1/1200000))/(1-

pbinom(0,60000,1/1200000)) = 0.00041. The probability that at least 4 molecules were from that same cell is (1- pbinom(3,60000,1/1200000))/(1- dbinom(0,60000,1/1200000)) = 5.1e-06. And the probability that at least 5 molecules will be from that same cell is (1- pbinom(4,60000,1/1200000))/(1- dbinom(0,60000,1/1200000)) = 5.1e-08.

As with the nuclear genome, we used a conditional binomial distribution to estimate the probability that at least two mtDNA molecules were from the same cell. We evaluated ~3,400 mtDNA molecules per BotSeqS library. The probability that, given we have picked a mtDNA molecule from a specific cell, at least another one of the 3,400 mtDNA molecules was from the same cell is (1- pbinom(1,3400,1/1200000))/(1- pbinom(0,3400,1/1200000)) = 0.0014. The probability that at least 3 molecules were from the same cell is then (1- pbinom(2,3400,1/1200000))/(1- pbinom(0,3400,1/1200000)) = 1.3e-06. The probability that at least 4 molecules were from that same cell is (1- pbinom(3,3400,1/1200000))/(1- dbinom(0,3400,1/1200000)) = 9.5e-10. And the probability that at least 5 molecules were from that same cell is (1- pbinom(4,3400,1/1200000))/(1- dbinom(0,3400,1/1200000)) = 5.5e-13.

Characterization of BotSeqS specificity. As one measure of specificity, we identified *rare* mutations as usual except that we used mutations that were present in only one strand rather than in both. Specifically, mutations were present in ≥ 90% of the Watson family members and the reference sequence was present in ≥ 90% of the Crick family members, or vice versa, but satisfied our other criteria for being "rare". We then created false Watson and Crick pairings, where the Watson strand had overlapping but different coordinates than the Crick strand, and vice versa, to determine if they contained the same mutation by chance. BotSeqS works by having low coverage throughout the genome, generated through the bottleneck dilution step, and precluded this analysis in the nuclear DNA. Instead, we used mtDNA because of the multiple copies of mtDNA per cell. The coverage of mtDNA with BotSeqS is much higher than that of nuclear DNA and facilitated the identification of overlapping molecules. We processed 30 BotSeqS control libraries this way and identified a total of 146 mtDNA mutations present in one strand only. Using this dataset, we then searched within each sample for overlapping molecules and identified 27 examples. None of the 27 false Watson and Crick pairs shared the same artifactual mutation.

Non-random shearing could produce another type of artifact, falsely suggesting that the Watson and Crick strands of a family were actually derived from two different molecules that coincidentally had the same genomic coordinate. To test for such artifacts, we identified Watson and Crick family pairs that contained the variant in the Watson strand and the reference sequence in the Crick strand, or vice versa, but this time included heterozygous germline variants rather than just the rare variants, and in nuclear DNA rather than in mtDNA. There are many more heterozygous variants in nuclear DNA than in mtDNA because the mtDNA is derived only from the oocyte. The discordances of interest could arise as a result of mispairing of a Watson strand with a Crick strand derived from a different template molecule - i.e., non-random shearing. Alternatively, discordances could result from an amplification error in one of the two strands during an early PCR cycle. Using our WGS data, we first identified 8,535,891 nuclear heterozygous variants observed among the 30 BotSeqS control libraries (median of 268,180 variants per library with range 121,850 to 529,920, with the same common variants present in many libraries). From the 8,535,891 nuclear heterozygous variants, we identified a total of 3,960,818 families (median of 123,130 families per library with range 65,832 to 222,140) for which both strands could be evaluated. Of these, 3,960,807 families had the concordant sequence at the variant position in both strands; only 11 heterozygous variants were discordant (i.e., the variant was present in ≥ 90% of the Watson family members and the reference sequence was present in ≥ 90% of the Crick family members, or vice versa). The rate of discordant germline heterozygous variants was thus 2.8×10^{-6} (11 out of 3,960,818) per bp. This rate is compatible with the known error rate of high fidelity DNA polymerases and could easily represent an amplification error that occurred in one of the two strands during the first PCR cycle, so represents an overestimate of shearing artifacts. Furthermore, it is important to note that BotSeqS eliminates such amplification errors by requiring mutations to be observed on both strands. Because BotSeqS requires mutations to be observed on both strands, the actual false positive rate can be estimated to be $\sim(1/3)(2.8 \times 10^{-6})(2.8 \times 10^{-6}) = 2.6 \times 10^{-12}$.

Generation of BotSeqS change and molecule tables. Sequence alignments and variant calling were performed with the Illumina secondary analysis package (CASAVA 1.8) using ELANDv2 matching to the GRCh37/hg19 human reference genome. High-quality reads were selected for further analysis only if they satisfied all of the following criteria: (i) passed chastity filter, (ii) read mapped in a proper pair, (iii) ≤ 5 mismatches to reference sequence (for 100 bp read), and (iv) perfect identity to reference sequence within the first and last five bases of each read. Sequencing reads were grouped into families based on identical paired-end endogenous barcodes. From our 44 BotSeqS libraries, we calculated that unfiltered mtDNA families represent a median of 0.4% (range 0.08 – 2%) of total unfiltered families (mtDNA+nuclear) per library. This is the expected range in sequencing data based on no enrichment of mtDNA. The members of a family were further subdivided into the two possible sequencing orientations to determine the number of Watson and Crick-derived family members. Watson and Crick families had identical genomic coordinates with each end sequenced in opposite reads. Quality scores of identical changes within a family were calculated as the average among the family members. The output for each BotSeqS library was two annotated tables of changes and template molecules (i.e., families). Details outlined in Bottleneck Sequencing System Pipeline Supplemental Document.

Selection of high quality changes and molecules

Custom algorithms were written in Microsoft SQL Server Management Studio to query the changes and molecules tables for each BotSeqS library. Selection criteria detailed in Table S2-S6. Selection was based on quality, clonality, and mappability of single-base pair substitutions. For example, it is known that one of the major sources of errors facing all short read alignment and variant callers are artifacts that arise when variants map to repetitive regions in the genome, including low complexity regions and copy number variants (2). The BotSeqS pipeline eliminates this universal error in a downstream step by filtering out the genomic noise from repetitive DNA and structural variants (detailed in Table S6). Indels were excluded because they are prone to alignment artifacts and are about 10 times less frequent than spontaneous point mutations. High quality single-base substitutions were defined as those with average quality scores (within the family) of ≥ Q30 and with ≥ 2 reads and ≥ 90% mutation fraction in both the Watson and Crick strands. Variants were considered to be clonal if the variant position was present in the WGS data from that sample or observed in > 1 template molecules (i.e., both strands of more than one UID). We also excluded any positions present in dbSNP130 or dbSNP142. We noticed that the dbSNP filtering drastically minimized recurrent sequencing or mapping artifacts and highly mutable regions. For example, homopolymer tracts (≥8 bp) are mutation hotspots that flood the mutation list. We observed that nearly all were filtered out with dbSNP142. Finally, families that harbored > 1 mutation were excluded as possible mapping artifacts. Details outlined in Bottleneck Sequencing System Pipeline Supplemental Document.

Calculation of mutation prevalence. Mutation prevalences were determined for each BotSeqS library (see Table S9) by dividing the total number of rare mutations by the total bp sequenced. The total bp sequenced was defined by number of families x 2 x read length of each family. The average length of the libraries was ~ 500 bp such that the 100 bp paired-end reads were unlikely to overlap. Only templates with perfect identity to the reference sequence in the first and last 5 bp of every read were considered. We further trimmed the reads by excluding cycle 6 and 7 to ensure quality. Therefore, the actual read length was 88 bases (100-7-5 = 88). For the samples from which technical replicate BotSeqS libraries were generated, the average mutation prevalence of the technical replicates was considered the mutation prevalence for the sample.

Validation of somatic mutations. All rare mutations from the nuclear and mtDNA genome passed visual inspection of the sequencing reads. For rare nuclear mutations, Sanger sequencing was performed on a representative set (514 out of 876 mutations). Of these, 514 of 514 (100%) were confirmed to be invisible by Sanger sequencing (excluding the COL238 and COL239 samples that did not have a matched WGS). This demonstrated that these mutations were neither present in the germline nor present in a highly clonal fashion. Mutations confirmed to be absent upon Sanger sequencing are indicated in Table S8.

Comparison to cancer genomes. MAF files representing nuclear somatic mutations from TCGA tumor types COADREAD and KIRC were downloaded at https://www.synapse.org/#!Synapse:syn1729383 (3). From the TCGA data, only single-base substitutions were considered and somatic mutations from ultra-mutated tumors were excluded. Mitochondrial DNA somatic mutations from colorectal and renal tumors were derived from supplementary file 2 of a previous report (4).

Study design and statistical analysis. For study design, no prior power analysis or randomization was performed because the variance was initially unknown. The goal of the study was to find major, biologically meaningful differences between the cohorts. To find major differences, sample sizes can be small. The initial rationale for the sample size for colon and brain was to acquire at least three individuals in each age group in order to understand the average trend of somatic mutational patterns for each age group. Age groups for colon and brain were selected based on human body growth and maintenance: early body development at < 10 years, fully grown young adult body at ~20-40 years, and old, maintained adult body at > 90 years. For colon, one tissue from the young child age group (SIN230) was later determined to be duodenum, leaving only two individuals representing the young child age group for colon epithelium. For normal kidney, criteria for kidney acquisition were an age-matched and non-smoking control group for the kidneys of smokers and aristolochic acid-exposed samples. All normal kidney controls were Caucasian and therefore less likely to originate from a high risk AA-exposed population (e.g. Asia). From the same kidney tissue source, three aliquots of flash frozen, post-mortem normal kidney from a five month old individual were available as technical replicates and to further test an age-trend for non-carcinogen exposed normal kidneys. All analyzed samples were reported in the manuscript.

Even with the small sample size, no violations of the assumptions of the tests were detected, including violations about the homogeneity of variances. T-test and ANOVA analyses were performed using GraphPad Prism 5.0f. Fisher's exact test and principal component analysis was performed using R version 3.2.2.

1. Jiao Y, et al. (2011) DAXX/ATRX, MEN1, and mTOR pathway genes are frequently altered in pancreatic neuroendocrine tumors. *Science* 331(6021):1199-1203.
2. Li H (2014) Toward better understanding of artifacts in variant calling from high-coverage samples. *Bioinformatics* 30(20):2843-2851.
3. Kandoth C, et al. (2013) Mutational landscape and significance across 12 major cancer types. *Nature* 502(7471):333-339.
4. Ju YS, et al. (2014) Origins and functional consequences of somatic mitochondrial DNA mutations in human cancer. *eLife* 3.

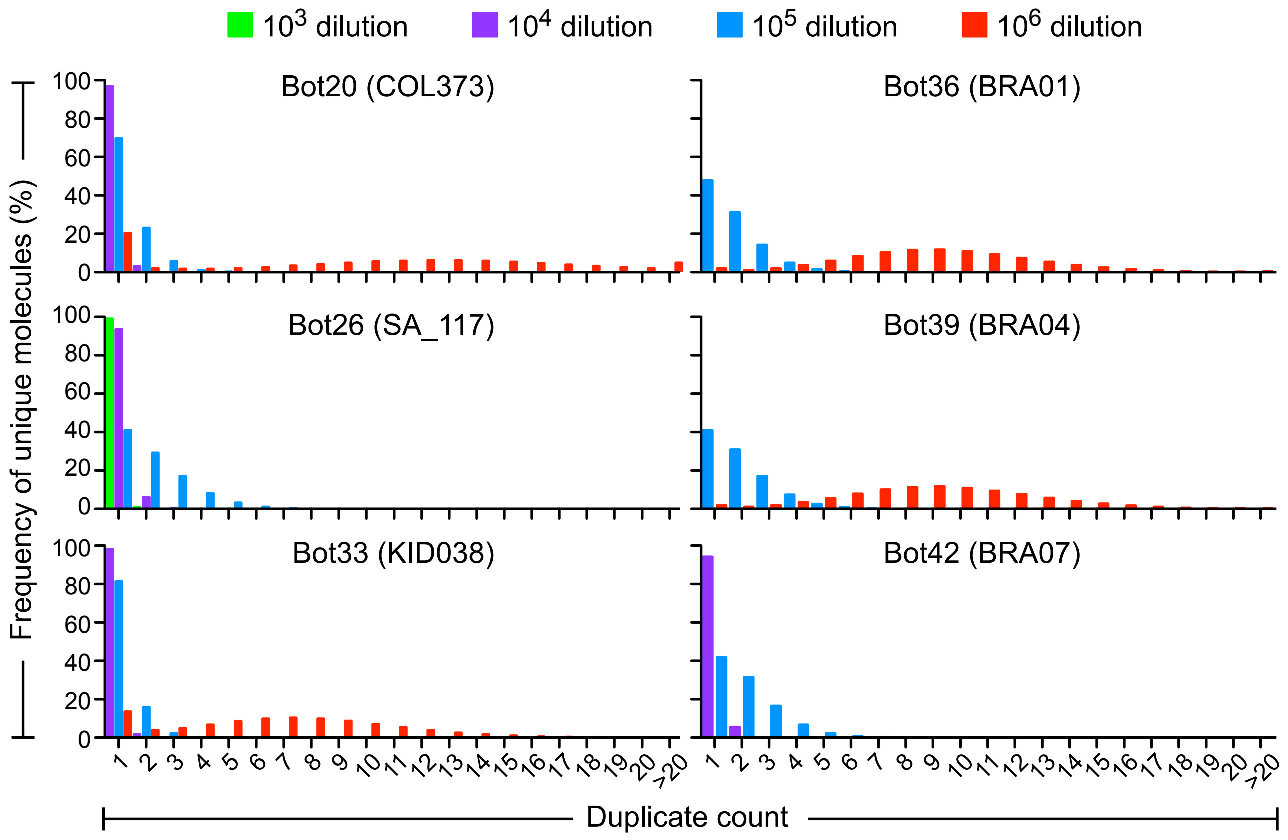


Fig. S1. Assessment of duplicate counts with MiSeq prerun. Histograms showing the distribution of UID family members (PCR duplicates from individual template molecules, shown on the x-axis). Either two or three serial dilutions (10^3, 10^4, 10^5, or 10^6) were evaluated on the MiSeq for six BotSeqS libraries (Bot20, Bot26, Bot33, Bot36, Bot39, Bot42) to generate ~5 M properly paired reads per library. Family member counts were determined here using Picard's Estimate Library Complexity program. Libraries generated from the 10^5 dilution (blue) were subsequently used for the final HiSeq run reported in this study. Note that the HiSeq distribution is expected to shift to the right compared to the MiSeq distribution due to the increase of clusters sequenced per library (~5 M clusters scaled to ~70 M clusters). For example, the BotSeqS libraries from the 10^6 dilution (red) were not used because the members per UID family would be too high on a HiSeq run, limiting the number of different families that could be evaluated with a given amount of sequencing.

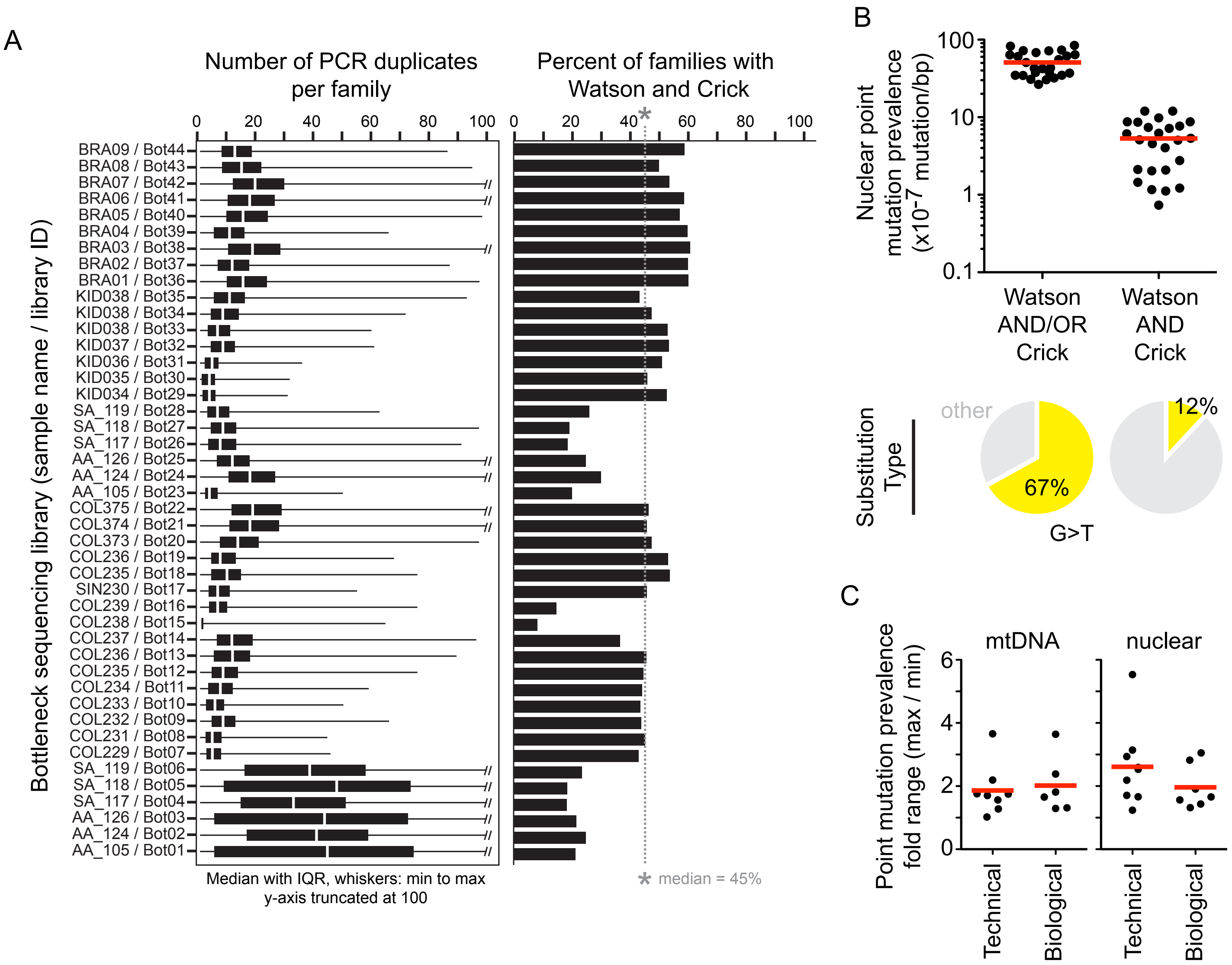


Fig. S2. Methodological parameters in BotSeqS (A) Left graph: horizontal box and whisker plots for 44 BotSeqS libraries (y-axis) and number of members per family (PCR duplicate count, x-axis). Filled boxes represent the first to third quartile range with hash mark indicating median. Whiskers represent minimum to maximum. Double slash of whisker indicates that maximum was truncated at 100. An average of 4 M (range 0.4 to 10 M) unfiltered families per library were assessed. Right graph: percent of unfiltered families with both Watson and Crick duplicate families present in each BotSeqS library. Grey dashed line indicates median. (B) Top graph: Nuclear point mutation prevalences (y-axis) considering mutations observed in “Watson AND/OR Crick” or “Watson AND Crick” families in normal tissues of 25 control individuals. “OR” mutations represent ≥ 90% mutation fraction in Watson family with a minimum of two Watson reads or ≥ 90% mutation fraction Crick family with a minimum of two Crick reads. “AND” mutations represent ≥ 90% mutation fraction in Watson family with a minimum of two Watson reads and ≥ 90% mutation fraction Crick family with a minimum of two Crick reads. “AND” mutations are an internal subset of the “AND/OR” dataset, which is a modified version of the BotSeqS pipeline. Red line indicates average. Bottom pie charts: frequency of G>T substitutions out of all possible substitution types (labeled as ‘other’) considering Watson AND/OR Crick (left pie chart) or Watson AND Crick (right pie chart) of 25 control individuals shown in top graph. Number of nuclear mutations generating mutational spectra for Watson AND/OR Crick was N=4415 and for Watson AND Crick was N=206. (C) Fold-range of point mutation prevalence of technical and biological BotSeqS replicates (details in Table S1) from mtDNA (left graph) and nuclear genome (right graph). Each point represents the ratio of the maximum to minimum of the replicate group. Red line indicates average.

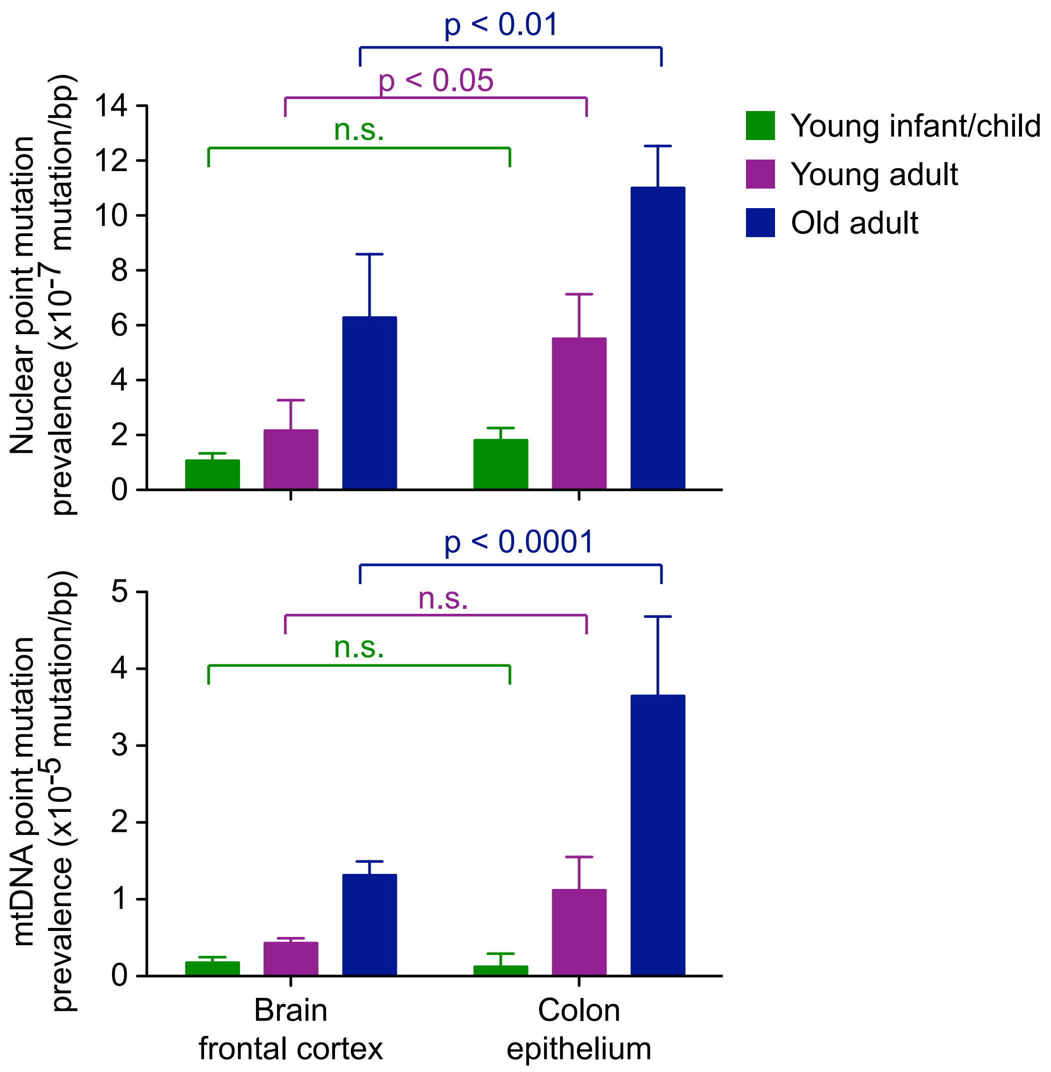


Fig. S3. Rare point mutations accumulate in normal tissues of the colon more than in brain. Point mutation prevalence (y-axis) in nuclear (top graph) and mitochondrial (bottom graph) genome in normal brain frontal cortex (left side) and normal colon epithelium (right side) grouped by age (young infant/child in green, young adult in purple, old adult in blue). Average +/- SD of each cohort shown. Two-way ANOVA with Bonferroni multiple comparison post-test was performed using GraphPad Prism 5.0f software with P values reported above bars. n.s. indicates P > 0.05. For brain, the number of individuals and average age of group are as follows- infant/child: N=3 (BRA01, BRA02, BRA03), 3.5 years old (y/o); young adult: N=3 (BRA04, BRA05, BRA06), 22 y/o; and old adult: N=3 (BRA07, BRA08, BRA09), 93 y/o. For colon, infant/child: N=2 (COL229, COL231), 5.5 y/o ; young adult: N=6 (COL235, COL236, COL237, COL373, COL374, COL375), 28 y/o; old adult: N=3 (COL232, COL233, COL234), 96 y/o.

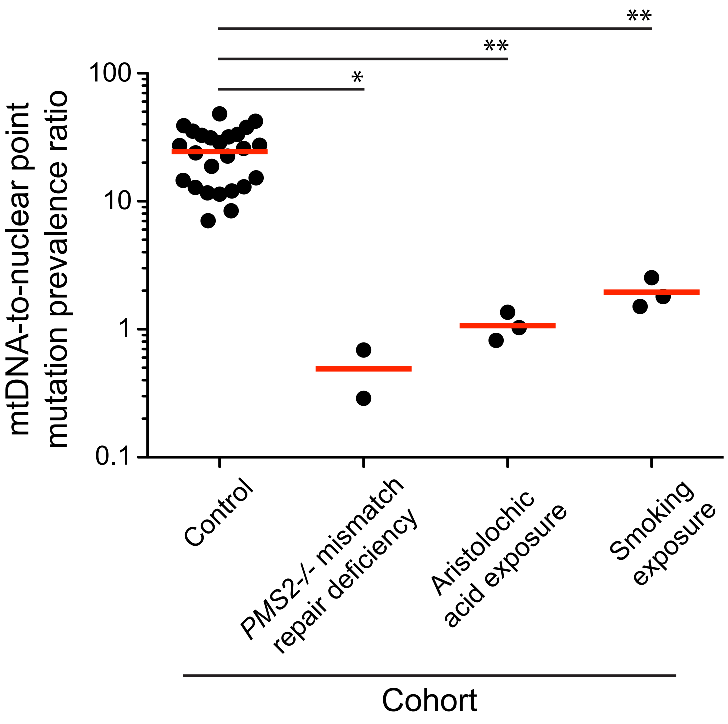
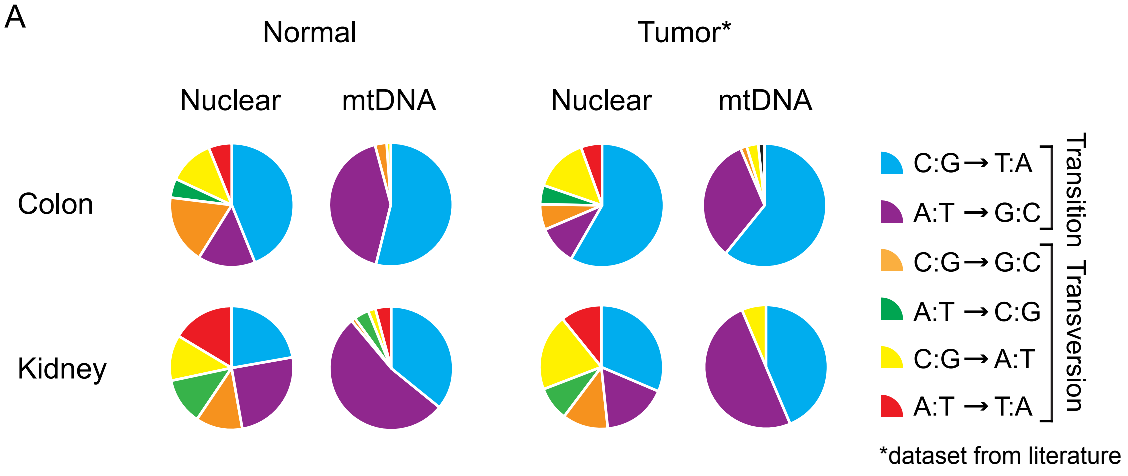


Fig. S4. Mitochondrial and nuclear point mutation prevalences in normal tissues from the same individual. Data points represent the ratio between mitochondrial to nuclear point mutation prevalences (y-axis) within the normal tissue of each individual. Individuals were grouped into four cohorts (x-axis) with N=24 for Control (see Table S9), N=2 (COL238, COL239) for *PMS2-/-* mismatch repair deficiency, N=3 (AA_105, AA_124, AA_126) for aristolochic acid exposure, and N=3 individuals (SA_117, SA_118, SA_119) for smoking exposure. One ratio from the control cohort (COL229) was zero and omitted from this analysis. Average (red line) ratio for each cohort is 25 for Control, 0.5 for DNA repair defect PMS2-/-, 1.1 for aristolochic acid exposure, and 2.0 for smoking exposure. *P < 0.05, **P < 0.01, one-way ANOVA with Bonferroni multiple comparison post-test.

A



B

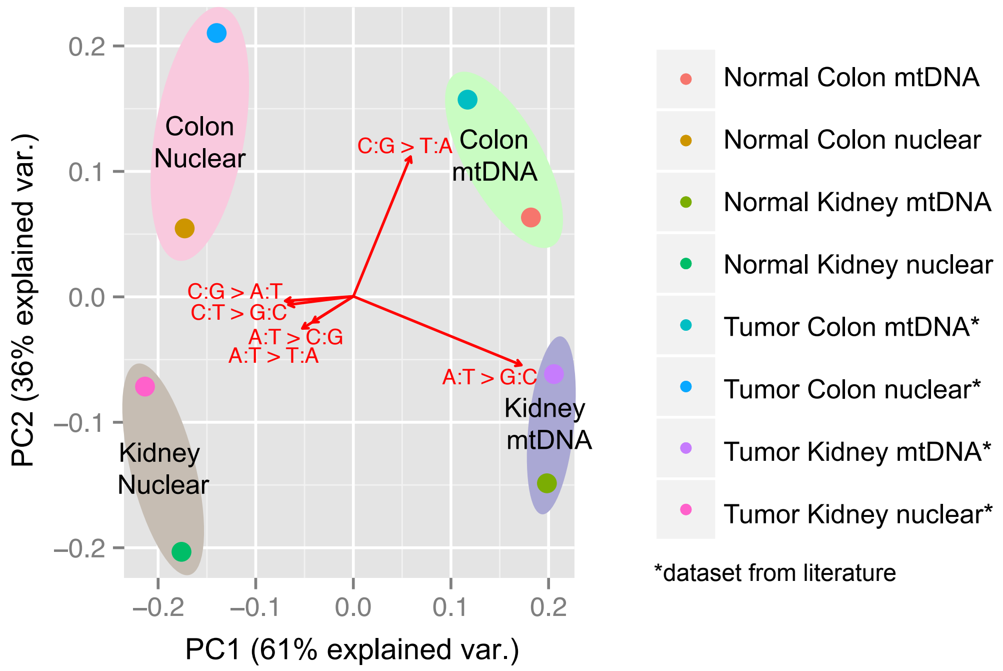


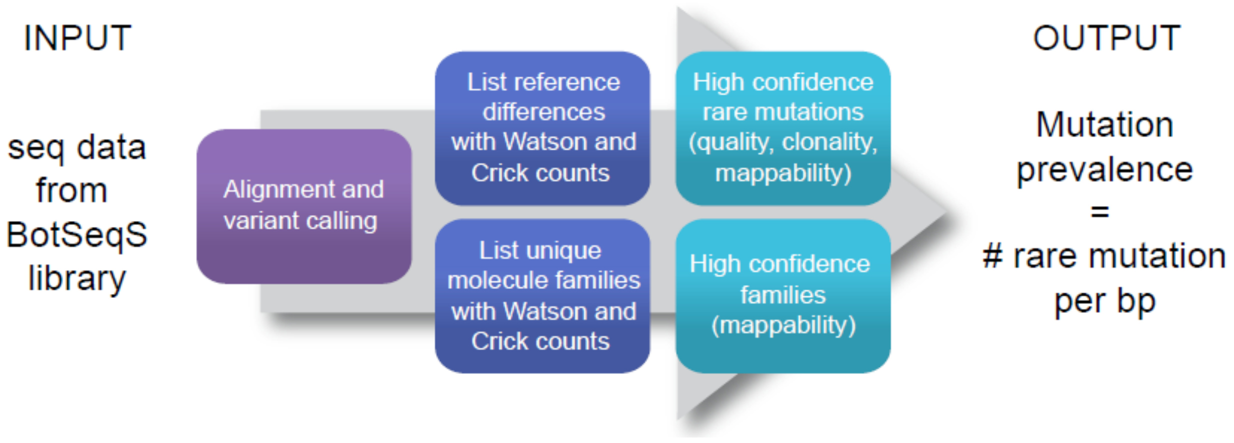
Fig. S5. Normal tissues and tumors derived from the same tissue type have similar mutation spectra (A) Pie charts of nuclear and mitochondrial substitution types out of six possible (see legend) comparing normal (left side) and tumors (right side) derived from colon (top) and kidney (bottom). "Normal" represents the rare mutational spectra data derived from normal tissues shown in Fig. 3. "Tumor" nuclear mutations represent clonal mutation data from colorectal carcinomas (COAD/READ) and clear cell renal carcinoma (KIRC) from the TCGA dataset acquired from www.synapse.org/#!Synapse:syn1729383. “Tumor” mtDNA mutations from colon and kidney were acquired from “colorectal” and “renal” tumor types in supplementary file 2 of Ju et al., 2014. For normal tissues, the number of substitutions assessed was as follows: colon nuclear N=94 from 13 individuals, colon mtDNA N=116 from 12 individuals, kidney nuclear N=73 from 7 individuals, and kidney mtDNA N=299 from five individuals. For tumor tissue, the number of substitutions assessed was as follows: colorectal carcinoma nuclear N=18,538 from 193 individuals, colorectal carcinoma mtDNA N=64 from 76 individuals, clear cell renal cell carcinoma nuclear N=24,559 from 417 individuals, and renal carcinoma mtDNA N=16 from 23 individuals. (B) Principal component analysis (PCA) of mutational spectra from the cohorts indicated in (A). PCA performed and graphed using R software.

SI BotSeqS Pipeline Supplement

1 Summary

Bottleneck Sequencing System (BotSeqS) is an unbiased next-generation sequencing (NGS) method designed to accurately quantitate extremely rare, or “nonclonal”, somatic mutations from a population of cells. Our initial application of BotSeqS was on human tissue samples using the Illumina sequencing platform. Therefore the BotSeqS pipeline described here is specific to human samples and Illumina sequencing. However, in principal, BotSeqS may be developed with any type of cell population, organism, and/or high throughput sequencing platform.

The goal of BotSeqS data processing is to accurately calculate the genome-wide mutation prevalence for nuclear DNA and mitochondrial DNA (mtDNA) per sample. The pipeline can be broken down into three phases as shown in Figure 1: (1) Alignment and variant calling, (2) Listing reference differences (or “changes”) and unique molecules annotated with counts and quality parameters, and (3) selection of high confidence rare mutations and molecules.


INPUT
seq data from BotSeqS library
Alignment and variant calling
List reference differences with Watson and Crick counts
List unique molecule families with Watson and Crick counts
High confidence rare mutations (quality, clonality, mappability)
High confidence families (mappability)
OUTPUT
Mutation prevalence = # rare mutation per bp

2 Illumina sequencing data outputs

Two libraries were generated for each sample, whole-genome sequencing (WGS) and bottleneck sequencing library (BOT).

1. WGS libraries were sequenced to >30x average coverage. Variants were aligned and called using Illumina’s Isaac algorithm. BotSeqS pipeline worked with variants tabulated in the genome variant call format (gVCF). This information facilitated the identification of somatic mutations in the BOT libraries. For example, about 90% of reference differences in BOT libraries were germline or clonal variants present in the WGS, and therefore filtered out.

2. BOT libraries were pair-ended sequenced at 2 x 100 bp with an average of 73 M clusters pass filter (about one-half lane of Illumina HiSeq 2500 on rapid run mode). Sequencing data were processed through Illumina's on machine secondary analysis CASAVA 1.8. This program included ELANDv2 alignment to hg19 and variant calling. The output of BOT data was an Illumina export txt file with 22 columns (for details of Illumina column information, please refer to *http://support.illumina.com/help/SequencingAnalysisWorkflow/Content/Vault/Informatics/Sequencing_Analysis/CASAVA/swSEQ_mCA_Exporttxt.htm*).

3 Data pre-processing and structure

Microsoft Sequel (SQL) Server Management Studio software was used for the majority of the data processing steps. BOT export files were first processed through our custom pipeline used for Illumina DNA sequencing studies. Because this pre-processing pipeline was developed specifically for our in-house hardware structure, it is not useful for off-site reproduction and therefore not included here. However, below we describe the output of our custom pipeline pre-processing. The output data structure is important to understand as it was the basis for the following SQL queries we provide further below.

SQL outputted the following 885 tables for each BOT library. Tables 1-6 include experimental information, processing criteria, and summary sheets. The remaining tables were organized into chromosomal regions, with each region associated with three tables: changes, coverage, and tags. The human genome was arbitrarily split into a number of different regions to increase processing speed.

1. ConsolidatedChanges
2. Experiment
3. ExperimentSummary
4. ImportedFiles
5. RoiCoverage
6. Sample
7. Zchr1_10Changes
8. Zchr1_10Coverage
9. Zchr1_10Tags
10. Zchr1_11Changes
11. Zchr1_11Coverage
12. Zchr1_11Tags
13. Zchr1_12Changes
14. Zchr1_12Coverage
15. Zchr1_12Tags
16. Zchr1_13Changes
17. Zchr1_13Coverage
18. Zchr1_13Tags
19. Zchr1_14Changes
20. Zchr1_14Coverage
21. Zchr1_14Tags
22. Zchr1_15Changes
23. Zchr1_15Coverage
24. Zchr1_15Tags
25. Zchr1_16Changes

26. Zchr1_16Coverage
27. Zchr1_16Tags
28. Zchr1_17Changes
29. Zchr1_17Coverage
30. Zchr1_17Tags
31. Zchr1_18Changes
32. Zchr1_18Coverage
33. Zchr1_18Tags
34. Zchr1_19Changes
35. Zchr1_19Coverage
36. Zchr1_19Tags
37. Zchr1_1Changes
38. Zchr1_1Coverage
39. Zchr1_1Tags
40. Zchr1_20Changes
41. Zchr1_20Coverage
42. Zchr1_20Tags
43. Zchr1_21Changes
44. Zchr1_21Coverage
45. Zchr1_21Tags
46. Zchr1_22Changes
47. Zchr1_22Coverage
48. Zchr1_22Tags
49. Zchr1_23Changes
50. Zchr1_23Coverage
51. Zchr1_23Tags
52. Zchr1_24Changes
53. Zchr1_24Coverage
54. Zchr1_24Tags
55. Zchr1_25Changes
56. Zchr1_25Coverage
57. Zchr1_25Tags
58. Zchr1_26Changes
59. Zchr1_26Coverage
60. Zchr1_26Tags
61. Zchr1_27Changes
62. Zchr1_27Coverage
63. Zchr1_27Tags
64. Zchr1_28Changes
65. Zchr1_28Coverage
66. Zchr1_28Tags
67. Zchr1_29Changes
68. Zchr1_29Coverage
69. Zchr1_29Tags
70. Zchr1_2Changes
71. Zchr1_2Coverage
72. Zchr1_2Tags
73. Zchr1_30Changes
74. Zchr1_30Coverage
75. Zchr1_30Tags
76. Zchr1_31Changes
77. Zchr1_31Coverage
78. Zchr1_31Tags
79. Zchr1_32Changes
80. Zchr1_32Coverage
81. Zchr1_32Tags
82. Zchr1_33Changes
83. Zchr1_33Coverage
84. Zchr1_33Tags
85. Zchr1_34Changes
86. Zchr1_34Coverage
87. Zchr1_34Tags

88. Zchr1_35Changes
89. Zchr1_35Coverage
90. Zchr1_35Tags
91. Zchr1_36Changes
92. Zchr1_36Coverage
93. Zchr1_36Tags
94. Zchr1_37Changes
95. Zchr1_37Coverage
96. Zchr1_37Tags
97. Zchr1_38Changes
98. Zchr1_38Coverage
99. Zchr1_38Tags
100. Zchr1_3Changes
101. Zchr1_3Coverage
102. Zchr1_3Tags
103. Zchr1_4Changes
104. Zchr1_4Coverage
105. Zchr1_4Tags
106. Zchr1_5Changes
107. Zchr1_5Coverage
108. Zchr1_5Tags
109. Zchr1_6Changes
110. Zchr1_6Coverage
111. Zchr1_6Tags
112. Zchr1_7Changes
113. Zchr1_7Coverage
114. Zchr1_7Tags
115. Zchr1_8Changes
116. Zchr1_8Coverage
117. Zchr1_8Tags
118. Zchr1_9Changes
119. Zchr1_9Coverage
120. Zchr1_9Tags
121. Zchr10_10Changes
122. Zchr10_10Coverage
123. Zchr10_10Tags
124. Zchr10_11Changes
125. Zchr10_11Coverage
126. Zchr10_11Tags
127. Zchr10_12Changes
128. Zchr10_12Coverage
129. Zchr10_12Tags
130. Zchr10_13Changes
131. Zchr10_13Coverage
132. Zchr10_13Tags
133. Zchr10_14Changes
134. Zchr10_14Coverage
135. Zchr10_14Tags
136. Zchr10_15Changes
137. Zchr10_15Coverage
138. Zchr10_15Tags
139. Zchr10_16Changes
140. Zchr10_16Coverage
141. Zchr10_16Tags
142. Zchr10_1Changes
143. Zchr10_1Coverage
144. Zchr10_1Tags
145. Zchr10_2Changes
146. Zchr10_2Coverage
147. Zchr10_2Tags
148. Zchr10_3Changes
149. Zchr10_3Coverage

150. Zchr10_3Tags
151. Zchr10_4Changes
152. Zchr10_4Coverage
153. Zchr10_4Tags
154. Zchr10_5Changes
155. Zchr10_5Coverage
156. Zchr10_5Tags
157. Zchr10_6Changes
158. Zchr10_6Coverage
159. Zchr10_6Tags
160. Zchr10_7Changes
161. Zchr10_7Coverage
162. Zchr10_7Tags
163. Zchr10_8Changes
164. Zchr10_8Coverage
165. Zchr10_8Tags
166. Zchr10_9Changes
167. Zchr10_9Coverage
168. Zchr10_9Tags
169. Zchr11_1Changes
170. Zchr11_1Coverage
171. Zchr11_1Tags
172. Zchr11_2Changes
173. Zchr11_2Coverage
174. Zchr11_2Tags
175. Zchr11_3Changes
176. Zchr11_3Coverage
177. Zchr11_3Tags
178. Zchr11_4Changes
179. Zchr11_4Coverage
180. Zchr11_4Tags
181. Zchr11_5Changes
182. Zchr11_5Coverage
183. Zchr11_5Tags
184. Zchr11_6Changes
185. Zchr11_6Coverage
186. Zchr11_6Tags
187. Zchr11_7Changes
188. Zchr11_7Coverage
189. Zchr11_7Tags
190. Zchr11_8Changes
191. Zchr11_8Coverage
192. Zchr11_8Tags
193. Zchr12_10Changes
194. Zchr12_10Coverage
195. Zchr12_10Tags
196. Zchr12_1Changes
197. Zchr12_1Coverage
198. Zchr12_1Tags
199. Zchr12_2Changes
200. Zchr12_2Coverage
201. Zchr12_2Tags
202. Zchr12_3Changes
203. Zchr12_3Coverage
204. Zchr12_3Tags
205. Zchr12_4Changes
206. Zchr12_4Coverage
207. Zchr12_4Tags
208. Zchr12_5Changes
209. Zchr12_5Coverage
210. Zchr12_5Tags
211. Zchr12_6Changes

212. Zchr12_6Coverage
213. Zchr12_6Tags
214. Zchr12_7Changes
215. Zchr12_7Coverage
216. Zchr12_7Tags
217. Zchr12_8Changes
218. Zchr12_8Coverage
219. Zchr12_8Tags
220. Zchr12_9Changes
221. Zchr12_9Coverage
222. Zchr12_9Tags
223. Zchr13_1Changes
224. Zchr13_1Coverage
225. Zchr13_1Tags
226. Zchr13_2Changes
227. Zchr13_2Coverage
228. Zchr13_2Tags
229. Zchr13_3Changes
230. Zchr13_3Coverage
231. Zchr13_3Tags
232. Zchr13_4Changes
233. Zchr13_4Coverage
234. Zchr13_4Tags
235. Zchr13_5Changes
236. Zchr13_5Coverage
237. Zchr13_5Tags
238. Zchr14_1Changes
239. Zchr14_1Coverage
240. Zchr14_1Tags
241. Zchr15_1Changes
242. Zchr15_1Coverage
243. Zchr15_1Tags
244. Zchr15_2Changes
245. Zchr15_2Coverage
246. Zchr15_2Tags
247. Zchr15_3Changes
248. Zchr15_3Coverage
249. Zchr15_3Tags
250. Zchr15_4Changes
251. Zchr15_4Coverage
252. Zchr15_4Tags
253. Zchr15_5Changes
254. Zchr15_5Coverage
255. Zchr15_5Tags
256. Zchr15_6Changes
257. Zchr15_6Coverage
258. Zchr15_6Tags
259. Zchr15_7Changes
260. Zchr15_7Coverage
261. Zchr15_7Tags
262. Zchr15_8Changes
263. Zchr15_8Coverage
264. Zchr15_8Tags
265. Zchr15_9Changes
266. Zchr15_9Coverage
267. Zchr15_9Tags
268. Zchr16_1Changes
269. Zchr16_1Coverage
270. Zchr16_1Tags
271. Zchr16_2Changes
272. Zchr16_2Coverage
273. Zchr16_2Tags

274. Zchr16_3Changes
275. Zchr16_3Coverage
276. Zchr16_3Tags
277. Zchr15_4Changes
278. Zchr15_4Coverage
279. Zchr15_4Tags
280. Zchr15_5Changes
281. Zchr15_5Coverage
282. Zchr15_5Tags
283. Zchr15_6Changes
284. Zchr15_6Coverage
285. Zchr15_6Tags
286. Zchr15_7Changes
287. Zchr15_7Coverage
288. Zchr15_7Tags
289. Zchr15_8Changes
290. Zchr15_8Coverage
291. Zchr15_8Tags
292. Zchr15_9Changes
293. Zchr15_9Coverage
294. Zchr15_9Tags
295. Zchr16_1Changes
296. Zchr16_1Coverage
297. Zchr16_1Tags
298. Zchr16_2Changes
299. Zchr16_2Coverage
300. Zchr16_2Tags
301. Zchr16_3Changes
302. Zchr16_3Coverage
303. Zchr16_3Tags
304. Zchr16_4Changes
305. Zchr16_4Coverage
306. Zchr16_4Tags
307. Zchr16_5Changes
308. Zchr16_5Coverage
309. Zchr16_5Tags
310. Zchr17_1Changes
311. Zchr17_1Coverage
312. Zchr17_1Tags
313. Zchr17_2Changes
314. Zchr17_2Coverage
315. Zchr17_2Tags
316. Zchr17_3Changes
317. Zchr17_3Coverage
318. Zchr17_3Tags
319. Zchr17_4Changes
320. Zchr17_4Coverage
321. Zchr17_4Tags
322. Zchr17_5Changes
323. Zchr17_5Coverage
324. Zchr17_5Tags
325. Zchr17_6Changes
326. Zchr17_6Coverage
327. Zchr17_6Tags
328. Zchr17_7Changes
329. Zchr17_7Coverage
330. Zchr17_7Tags
331. Zchr17_8Changes
332. Zchr17_8Coverage
333. Zchr17_8Tags
334. Zchr18_1Changes
335. Zchr18_1Coverage

336. Zchr18_1Tags
337. Zchr18_2Changes
338. Zchr18_2Coverage
339. Zchr18_2Tags
340. Zchr18_3Changes
341. Zchr18_3Coverage
342. Zchr18_3Tags
343. Zchr18_4Changes
344. Zchr18_4Coverage
345. Zchr18_4Tags
346. Zchr18_5Changes
347. Zchr18_5Coverage
348. Zchr18_5Tags
349. Zchr19_1Changes
350. Zchr19_1Coverage
351. Zchr19_1Tags
352. Zchr19_2Changes
353. Zchr19_2Coverage
354. Zchr19_2Tags
355. Zchr19_3Changes
356. Zchr19_3Coverage
357. Zchr19_3Tags
358. Zchr19_4Changes
359. Zchr19_4Coverage
360. Zchr19_4Tags
361. Zchr19_5Changes
362. Zchr19_5Coverage
363. Zchr19_5Tags
364. Zchr2_10Changes
365. Zchr2_10Coverage
366. Zchr2_10Tags
367. Zchr2_11Changes
368. Zchr2_11Coverage
369. Zchr2_11Tags
370. Zchr2_12Changes
371. Zchr2_12Coverage
372. Zchr2_12Tags
373. Zchr2_13Changes
374. Zchr2_13Coverage
375. Zchr2_13Tags
376. Zchr2_14Changes
377. Zchr2_14Coverage
378. Zchr2_14Tags
379. Zchr2_15Changes
380. Zchr2_15Coverage
381. Zchr2_15Tags
382. Zchr2_16Changes
383. Zchr2_16Coverage
384. Zchr2_16Tags
385. Zchr2_17Changes
386. Zchr2_17Coverage
387. Zchr2_17Tags
388. Zchr2_18Changes
389. Zchr2_18Coverage
390. Zchr2_18Tags
391. Zchr2_19Changes
392. Zchr2_19Coverage
393. Zchr2_19Tags
394. Zchr2_1Changes
395. Zchr2_1Coverage
396. Zchr2_1Tags
397. Zchr2_2Changes

398. Zchr2_2Coverage
399. Zchr2_2Tags
400. Zchr2_3Changes
401. Zchr2_3Coverage
402. Zchr2_3Tags
403. Zchr2_4Changes
404. Zchr2_4Coverage
405. Zchr2_4Tags
406. Zchr2_5Changes
407. Zchr2_5Coverage
408. Zchr2_5Tags
409. Zchr2_6Changes
410. Zchr2_6Coverage
411. Zchr2_6Tags
412. Zchr2_7Changes
413. Zchr2_7Coverage
414. Zchr2_7Tags
415. Zchr2_8Changes
416. Zchr2_8Coverage
417. Zchr2_8Tags
418. Zchr2_9Changes
419. Zchr2_9Coverage
420. Zchr2_9Tags
421. Zchr20_1Changes
422. Zchr20_1Coverage
423. Zchr20_1Tags
424. Zchr20_2Changes
425. Zchr20_2Coverage
426. Zchr20_2Tags
427. Zchr20_3Changes
428. Zchr20_3Coverage
429. Zchr20_3Tags
430. Zchr20_4Changes
431. Zchr20_4Coverage
432. Zchr20_4Tags
433. Zchr20_5Changes
434. Zchr20_5Coverage
435. Zchr20_5Tags
436. Zchr20_6Changes
437. Zchr20_6Coverage
438. Zchr20_6Tags
439. Zchr21_10Changes
440. Zchr21_10Coverage
441. Zchr21_10Tags
442. Zchr21_11Changes
443. Zchr21_11Coverage
444. Zchr21_11Tags
445. Zchr21_1Changes
446. Zchr21_1Coverage
447. Zchr21_1Tags
448. Zchr21_2Changes
449. Zchr21_2Coverage
450. Zchr21_2Tags
451. Zchr21_3Changes
452. Zchr21_3Coverage
453. Zchr21_3Tags
454. Zchr21_4Changes
455. Zchr21_4Coverage
456. Zchr21_4Tags
457. Zchr21_5Changes
458. Zchr21_5Coverage
459. Zchr21_5Tags

460. Zchr21_6Changes
461. Zchr21_6Coverage
462. Zchr21_6Tags
463. Zchr21_7Changes
464. Zchr21_7Coverage
465. Zchr21_7Tags
466. Zchr21_8Changes
467. Zchr21_8Coverage
468. Zchr21_8Tags
469. Zchr21_9Changes
470. Zchr21_9Coverage
471. Zchr21_9Tags
472. Zchr22_1Changes
473. Zchr22_1Coverage
474. Zchr22_1Tags
475. Zchr22_2Changes
476. Zchr22_2Coverage
477. Zchr22_2Tags
478. Zchr22_3Changes
479. Zchr22_3Coverage
480. Zchr22_3Tags
481. Zchr22_4Changes
482. Zchr22_4Coverage
483. Zchr22_4Tags
484. Zchr3_1Changes
485. Zchr3_1Coverage
486. Zchr3_1Tags
487. Zchr3_2Changes
488. Zchr3_2Coverage
489. Zchr3_2Tags
490. Zchr3_3Changes
491. Zchr3_3Coverage
492. Zchr3_3Tags
493. Zchr3_4Changes
494. Zchr3_4Coverage
495. Zchr3_4Tags
496. Zchr4_10Changes
497. Zchr4_10Coverage
498. Zchr4_10Tags
499. Zchr4_11Changes
500. Zchr4_11Coverage
501. Zchr4_11Tags
502. Zchr4_1Changes
503. Zchr4_1Coverage
504. Zchr4_1Tags
505. Zchr4_2Changes
506. Zchr4_2Coverage
507. Zchr4_2Tags
508. Zchr4_3Changes
509. Zchr4_3Coverage
510. Zchr4_3Tags
511. Zchr4_4Changes
512. Zchr4_4Coverage
513. Zchr4_4Tags
514. Zchr4_5Changes
515. Zchr4_5Coverage
516. Zchr4_5Tags
517. Zchr4_6Changes
518. Zchr4_6Coverage
519. Zchr4_6Tags
520. Zchr4_7Changes
521. Zchr4_7Coverage

522. Zchr4_7Tags
523. Zchr4_8Changes
524. Zchr4_8Coverage
525. Zchr4_8Tags
526. Zchr4_9Changes
527. Zchr4_9Coverage
528. Zchr4_9Tags
529. Zchr5_1Changes
530. Zchr5_1Coverage
531. Zchr5_1Tags
532. Zchr5_2Changes
533. Zchr5_2Coverage
534. Zchr5_2Tags
535. Zchr5_3Changes
536. Zchr5_3Coverage
537. Zchr5_3Tags
538. Zchr5_4Changes
539. Zchr5_4Coverage
540. Zchr5_4Tags
541. Zchr5_5Changes
542. Zchr5_5Coverage
543. Zchr5_5Tags
544. Zchr5_6Changes
545. Zchr5_6Coverage
546. Zchr5_6Tags
547. Zchr6_1Changes
548. Zchr6_1Coverage
549. Zchr6_1Tags
550. Zchr6_2Changes
551. Zchr6_2Coverage
552. Zchr6_2Tags
553. Zchr6_3Changes
554. Zchr6_3Coverage
555. Zchr6_3Tags
556. Zchr6_4Changes
557. Zchr6_4Coverage
558. Zchr6_4Tags
559. Zchr6_5Changes
560. Zchr6_5Coverage
561. Zchr6_5Tags
562. Zchr6_6Changes
563. Zchr6_6Coverage
564. Zchr6_6Tags
565. Zchr6_7Changes
566. Zchr6_7Coverage
567. Zchr6_7Tags
568. Zchr6_8Changes
569. Zchr6_8Coverage
570. Zchr6_8Tags
571. Zchr6_9Changes
572. Zchr6_9Coverage
573. Zchr6_9Tags
574. Zchr7_10Changes
575. Zchr7_10Coverage
576. Zchr7_10Tags
577. Zchr7_11Changes
578. Zchr7_11Coverage
579. Zchr7_11Tags
580. Zchr7_12Changes
581. Zchr7_12Coverage
582. Zchr7_12Tags
583. Zchr7_13Changes

584. Zchr7_13Coverage
585. Zchr7_13Tags
586. Zchr7_14Changes
587. Zchr7_14Coverage
588. Zchr7_14Tags
589. Zchr7_15Changes
590. Zchr7_15Coverage
591. Zchr7_15Tags
592. Zchr7_16Changes
593. Zchr7_16Coverage
594. Zchr7_16Tags
595. Zchr7_1Changes
596. Zchr7_1Coverage
597. Zchr7_1Tags
598. Zchr7_2Changes
599. Zchr7_2Coverage
600. Zchr7_2Tags
601. Zchr7_3Changes
602. Zchr7_3Coverage
603. Zchr7_3Tags
604. Zchr7_4Changes
605. Zchr7_4Coverage
606. Zchr7_4Tags
607. Zchr7_5Changes
608. Zchr7_5Coverage
609. Zchr7_5Tags
610. Zchr7_6Changes
611. Zchr7_6Coverage
612. Zchr7_6Tags
613. Zchr7_7Changes
614. Zchr7_7Coverage
615. Zchr7_7Tags
616. Zchr7_8Changes
617. Zchr7_8Coverage
618. Zchr7_8Tags
619. Zchr7_9Changes
620. Zchr7_9Coverage
621. Zchr7_9Tags
622. Zchr8_1Changes
623. Zchr8_1Coverage
624. Zchr8_1Tags
625. Zchr8_2Changes
626. Zchr8_2Coverage
627. Zchr8_2Tags
628. Zchr8_3Changes
629. Zchr8_3Coverage
630. Zchr8_3Tags
631. Zchr8_4Changes
632. Zchr8_4Coverage
633. Zchr8_4Tags
634. Zchr8_5Changes
635. Zchr8_5Coverage
636. Zchr8_5Tags
637. Zchr8_6Changes
638. Zchr8_6Coverage
639. Zchr8_6Tags
640. Zchr8_7Changes
641. Zchr8_7Coverage
642. Zchr8_7Tags
643. Zchr8_8Changes
644. Zchr8_8Coverage
645. Zchr8_8Tags

646. Zchr9_10Changes
647. Zchr9_10Coverage
648. Zchr9_10Tags
649. Zchr9_11Changes
650. Zchr9_11Coverage
651. Zchr9_11Tags
652. Zchr9_12Changes
653. Zchr9_12Coverage
654. Zchr9_12Tags
655. Zchr9_13Changes
656. Zchr9_13Coverage
657. Zchr9_13Tags
658. Zchr9_14Changes
659. Zchr9_14Coverage
660. Zchr9_14Tags
661. Zchr9_15Changes
662. Zchr9_15Coverage
663. Zchr9_15Tags
664. Zchr9_16Changes
665. Zchr9_16Coverage
666. Zchr9_16Tags
667. Zchr9_17Changes
668. Zchr9_17Coverage
669. Zchr9_17Tags
670. Zchr9_18Changes
671. Zchr9_18Coverage
672. Zchr9_18Tags
673. Zchr9_19Changes
674. Zchr9_19Coverage
675. Zchr9_19Tags
676. Zchr9_1Changes
677. Zchr9_1Coverage
678. Zchr9_1Tags
679. Zchr9_20Changes
680. Zchr9_20Coverage
681. Zchr9_20Tags
682. Zchr9_21Changes
683. Zchr9_21Coverage
684. Zchr9_21Tags
685. Zchr9_22Changes
686. Zchr9_22Coverage
687. Zchr9_22Tags
688. Zchr9_23Changes
689. Zchr9_23Coverage
690. Zchr9_23Tags
691. Zchr9_24Changes
692. Zchr9_24Coverage
693. Zchr9_24Tags
694. Zchr9_25Changes
695. Zchr9_25Coverage
696. Zchr9_25Tags
697. Zchr9_26Changes
698. Zchr9_26Coverage
699. Zchr9_26Tags
700. Zchr9_27Changes
701. Zchr9_27Coverage
702. Zchr9_27Tags
703. Zchr9_28Changes
704. Zchr9_28Coverage
705. Zchr9_28Tags
706. Zchr9_29Changes
707. Zchr9_29Coverage

708. Zchr9_29Tags
709. Zchr9_2Changes
710. Zchr9_2Coverage
711. Zchr9_2Tags
712. Zchr9_30Changes
713. Zchr9_30Coverage
714. Zchr9_30Tags
715. Zchr9_31Changes
716. Zchr9_31Coverage
717. Zchr9_31Tags
718. Zchr9_32Changes
719. Zchr9_32Coverage
720. Zchr9_32Tags
721. Zchr9_33Changes
722. Zchr9_33Coverage
723. Zchr9_33Tags
724. Zchr9_34Changes
725. Zchr9_34Coverage
726. Zchr9_34Tags
727. Zchr9_35Changes
728. Zchr9_35Coverage
729. Zchr9_35Tags
730. Zchr9_36Changes
731. Zchr9_36Coverage
732. Zchr9_36Tags
733. Zchr9_37Changes
734. Zchr9_37Coverage
735. Zchr9_37Tags
736. Zchr9_38Changes
737. Zchr9_38Coverage
738. Zchr9_38Tags
739. Zchr9_39Changes
740. Zchr9_39Coverage
741. Zchr9_39Tags
742. Zchr9_3Changes
743. Zchr9_3Coverage
744. Zchr9_3Tags
745. Zchr9_4Changes
746. Zchr9_4Coverage
747. Zchr9_4Tags
748. Zchr9_5Changes
749. Zchr9_5Coverage
750. Zchr9_5Tags
751. Zchr9_6Changes
752. Zchr9_6Coverage
753. Zchr9_6Tags
754. Zchr9_7Changes
755. Zchr9_7Coverage
756. Zchr9_7Tags
757. Zchr9_8Changes
758. Zchr9_8Coverage
759. Zchr9_8Tags
760. Zchr9_9Changes
761. Zchr9_9Coverage
762. Zchr9_9Tags
763. ZchrM_1Changes
764. ZchrM_1Coverage
765. ZchrM_1Tags
766. ZchrX_10Changes
767. ZchrX_10Coverage
768. ZchrX_10Tags
769. ZchrX_11Changes

770. ZchrX_11Coverage
771. ZchrX_11Tags
772. ZchrX_12Changes
773. ZchrX_12Coverage
774. ZchrX_12Tags
775. ZchrX_13Changes
776. ZchrX_13Coverage
777. ZchrX_13Tags
778. ZchrX_14Changes
779. ZchrX_14Coverage
780. ZchrX_14Tags
781. ZchrX_15Changes
782. ZchrX_15Coverage
783. ZchrX_15Tags
784. ZchrX_16Changes
785. ZchrX_16Coverage
786. ZchrX_16Tags
787. ZchrX_17Changes
788. ZchrX_17Coverage
789. ZchrX_17Tags
790. ZchrX_18Changes
791. ZchrX_18Coverage
792. ZchrX_18Tags
793. ZchrX_19Changes
794. ZchrX_19Coverage
795. ZchrX_19Tags
796. ZchrX_1Changes
797. ZchrX_1Coverage
798. ZchrX_1Tags
799. ZchrX_20Changes
800. ZchrX_20Coverage
801. ZchrX_20Tags
802. ZchrX_21Changes
803. ZchrX_21Coverage
804. ZchrX_21Tags
805. ZchrX_22Changes
806. ZchrX_22Coverage
807. ZchrX_22Tags
808. ZchrX_2Changes
809. ZchrX_2Coverage
810. ZchrX_2Tags
811. ZchrX_3Changes
812. ZchrX_3Coverage
813. ZchrX_3Tags
814. ZchrX_4Changes
815. ZchrX_4Coverage
816. ZchrX_4Tags
817. ZchrX_5Changes
818. ZchrX_5Coverage
819. ZchrX_5Tags
820. ZchrX_6Changes
821. ZchrX_6Coverage
822. ZchrX_6Tags
823. ZchrX_7Changes
824. ZchrX_7Coverage
825. ZchrX_7Tags
826. ZchrX_8Changes
827. ZchrX_8Coverage
828. ZchrX_8Tags
829. ZchrX_9Changes
830. ZchrX_9Coverage
831. ZchrX_9Tags

832. ZchrY_10Changes
833. ZchrY_10Coverage
834. ZchrY_10Tags
835. ZchrY_11Changes
836. ZchrY_11Coverage
837. ZchrY_11Tags
838. ZchrY_12Changes
839. ZchrY_12Coverage
840. ZchrY_12Tags
841. ZchrY_13Changes
842. ZchrY_13Coverage
843. ZchrY_13Tags
844. ZchrY_14Changes
845. ZchrY_14Coverage
846. ZchrY_14Tags
847. ZchrY_15Changes
848. ZchrY_15Coverage
849. ZchrY_15Tags
850. ZchrY_16Changes
851. ZchrY_16Coverage
852. ZchrY_16Tags
853. ZchrY_17Changes
854. ZchrY_17Coverage
855. ZchrY_17Tags
856. ZchrY_1Changes
857. ZchrY_1Coverage
858. ZchrY_1Tags
859. ZchrY_2Changes
860. ZchrY_2Coverage
861. ZchrY_2Tags
862. ZchrY_3Changes
863. ZchrY_3Coverage
864. ZchrY_3Tags
865. ZchrY_4Changes
866. ZchrY_4Coverage
867. ZchrY_4Tags
868. ZchrY_5Changes
869. ZchrY_5Coverage
870. ZchrY_5Tags
871. ZchrY_6Changes
872. ZchrY_6Coverage
873. ZchrY_6Tags
874. ZchrY_7Changes
875. ZchrY_7Coverage
876. ZchrY_7Tags
877. ZchrY_8Changes
878. ZchrY_8Coverage
879. ZchrY_8Tags
880. ZchrY_9Changes
881. ZchrY_9Coverage
882. ZchrY_9Tags
883. ZNMChanges
884. ZNMCoverage
885. ZNMTags

Below we describe the following tables from the output above:

(1) Consolidated Changes
(2) Zchr*_*Changes
(3) Zchr*_*Coverage
(4) Zchr*_*Tags

(1) ConsolidatedChanges- summarizes all variants called

Column header	**Example value**
UID	59173F25-9FFE-42B6-96F7-0006CF05D366
IndexString	0
Coverage	8
MutCount	8
ACount	0
CCount	0
GCount	0
TCount	8
SumQualityScore	311
MinQualityScore	30
MaxQualityScore	41
AverageQualityScore	38
Forward	3
Reverse	5
DistinctCoverage	2
DistinctTags	2
DistinctPairs	2
MutType	Homo-SBS
Chrom	chr4
Start	181198628
End	181198628
BaseFrom	C
BaseTo	T
Transcript	NULL
TxChangeStart	NULL
TxChangeEnd	NULL
TxBaseFrom	NULL
TxBaseTo	NULL
AminoStart	NULL
AminoEnd	NULL
AAFrom	NULL
AATo	NULL
SNP	rs2309255
Deleted	1
Comment	NULL

(2) Zchr*_*Changes

Column header	**Example value**
ClusterID	0_HISEQ_150_2_2109_10547_89283
IndexString	0
ReadNumber	1
Cycle	34

Filtered	Y
TotalMutationsInTag	1
QualityScore	9
MutType	SBS
Chrom	chr1
MatchOrientation	F
Start	29401957
End	29401957
BaseFrom	T
BaseTo	A
Deleted	1
PairCoordinates	chr1F29401924chr1R29402354

(3) Zchr*_*Coverage

Column header	**Example value**
IndexString	0
Position	17185422
RawCoverage	13
PhredCoverage	11
EffectiveCoverage	0.994141
DistinctRawCoverage	2
DistinctPhredCoverage	2
DistinctEffectiveCoverage	0.25

(4) Zchr*_*Tags

Column	**Example value**
ClusterID	0_HISEQ_150_2_2109_10045_85816
IndexString	0
ReadNumber	1
Sequence	CCCACCACCACACCCAGTTAATTTTTTGTATTTTTAGTAGAGGCAG
Quality	___eeeeegggfeffgffhghfhhihhhfhhiihihhfhefefdgfif_e`fcbbb\db^`dddeec
Match	chr1.fa
Chrom	chr1
MatchOrientat	F
LeftPosition	22162961
RightPosition	22163060
MatchContig	NULL
MatchDescript	100
MinQS	7
MaxQS	41
AverageQS	35
TotalMuations	0
AlignmentSco	11
PairedEndRe	888
PartnerChrom	NULL

PartnerContig	NULL
PartnerOffset	314
PartnerOrient	R
Filtered	Y
Exclude	0
PairCoordinat	chr1F22162961chr1R22163275

4 Organize data into change and molecule table with Watson and Crick family counts

The goal of this section is to use the data structure detailed above to reorganize the data into two tables per BOT library: changes and molecules. The outputs of QUERY 1 were two intermediate tables that are the input to QUERY 2. The outputs of QUERY 2 were the final tables with example names as follows:

Final molecule output of QUERY 2 (molecules table): *NgsNCS_Brain01NormalP30MM6F5Genome100D031615_SupermutantMoleculesSummaryFinal_022213v6*

Final changes output of QUERY 2 (changes table): *NgsNCS_Brain01NormalP30MM6F5Genome100D031615_SuperMutantsSummaryFinal_022213v6*

Below are Query 1 and Query 2. After the queries, examples of the **Final molecules output of QUERY 2** and **Final changes output of QUERY 2** are given.

QUERY 1

```
declare
@SQL varchar(8000),
@ChromSub varchar(50),
@exp varchar(255),
@dbTo varchar(255),
@serverFrom varchar(255),
@Version varchar(255),
@TagMutationsMax tinyint,
@TagLen_i int,
@TagLen_s varchar(3),
@Trim tinyint,
@lowlim_search varchar(3),
@maxgap varchar(3),
@lowlim_total varchar(3)

set @serverFrom = 'LUDSEQ7'
set @exp = 'NgsNCS_Brain09NormalP30MM6F5Genome100D031615'
set @dbTo = '[MH_Nonclonal_NCS]'
set @Version='022213v6'
set @TagMutationsMax = 5 -- Maximum number of mutations in both tags,
inclusive
set @TagLen_i = 100 -- Length of tag (integer)
set @TagLen_s = 100 -- Length of tag (string)
set @Trim = 5 -- Number of bases on each end of tag that must be perfect,
inclusive

set @lowlim_search = 2
set @maxgap = 1
set @lowlim_total = 8

create table #TempTag (
      [ClusterID] [varchar](250) not null,
      [IndexString] [varchar](25) null,
      [ReadNumber] [tinyint] not null,
      [Sequence] [varchar](100) null,
      [Quality] [varchar](100) null,
      [Match] [varchar](50) null,
      [Chrom] [varchar](50) null,
      [MatchOrientation] [varchar](1) null,
      [LeftPosition] [int] null,
      [RightPosition] [int] null,
      [MatchContig] [varchar](150) null,
      [MatchDescription] [varchar](150) null,
      [MinQS] [tinyint] null,
      [MaxQS] [tinyint] null,
      [AverageQS] [tinyint] null,
      [TotalMutationsInTag] [tinyint] null,
      [AlignmentScore] [int] null,
      [PairedEndReadAligmentScore] [int] null,
      [PartnerChrom] [varchar](150) null,
      [PartnerContig] [varchar](150) null,
      [PartnerOffset] [int] null,
      [PartnerOrientation] [char](1) null,
      [Filtered] [char](1) null,
      [Exclude] [bit] null,
      [PairCoordinates] [varchar](120) null,
      [GenePair] [varchar](50)
)

create table #TempChanges (
      [ClusterID] [varchar](250) not null,
      [IndexString] [varchar](25) null,
      [ReadNumber] [tinyint] not null,
      [Cycle] [tinyint] not null,
      [Filtered] [char](1) null,
      [TotalMutationsInTag] [tinyint] null,
      [QualityScore] [tinyint] null,
      [MutType] [varchar](25) null,
      [Chrom] [varchar](50) null,
      [MatchOrientation] [char](1) null,
      [Start] [int] null,
      [end] [int] null,
      [BaseFrom] [varchar](60) null,
      [BaseTo] [varchar](60) null,
      [Deleted] [bit] null,
      [PairCoordinates] [varchar](120) null
)

create table #TempTagsChangesInDels (
	[ClusterID] [varchar](250) not null,
	[ReadNumber] [tinyint] not null,
	[Sequence] [varchar](100) null,
	[Quality] [varchar](100) null,
	[Match] [varchar](50) null,
	[Chrom] [varchar](50) null,
	[MatchOrientation] [varchar](1) null,
	[LeftPosition] [int] null,
	[RightPosition] [int] null,
	[MinQS] [tinyint] null,
	[MaxQS] [tinyint] null,
	[AverageQS] [tinyint] null,
	[TotalMutationsInTag] [tinyint] null,
	[AlignmentScore] [int] null,
	[PairedEndReadAligmentScore] [int] null,
	[PartnerChrom] [varchar](150) null,
	[PartnerOrientation] [char](1) null,
	[Filtered] [char](1) null,
	[Exclude] [bit] null,
	[PairCoordinates] [varchar](120) null,
	[GenePair] [varchar](50) null,

	[Cycle] [tinyint] not null,
	[QualityScore] [tinyint] null,
	[MutType] [varchar](25) null,
	[Start] [int] null,
	[end] [int] null,
	[BaseFrom] [varchar](60) null,
	[BaseTo] [varchar](60) null,
	[Deleted] [bit] null,

	[HomoPolymerString] [varchar](100) null
)

create table #TempChangesWithHT (
	[ClusterID] [varchar](250) not null,
	[IndexString] [varchar](25) null,
	[ReadNumber] [tinyint] not null,
	[Cycle] [tinyint] not null,
	[Filtered] [char](1) null,
	[TotalMutationsInTag] [tinyint] null,
	[QualityScore] [tinyint] null,
	[MutType] [varchar](25) null,
	[Chrom] [varchar](50) null,
	[MatchOrientation] [char](1) null,
	[Start] [int] null,
	[end] [int] null,
	[BaseFrom] [varchar](60) null,
	[BaseTo] [varchar](60) null,
	[Deleted] [bit] null,
	[PairCoordinates] [varchar](120) null,

	[HomoPolymerString] [varchar](100) null
)
```

```
create table #TempMoleculesTable (
	[ClusterID] [varchar](250) not null,
	[ReadNumber] [tinyint] not null,
	[PairCoordinates] [varchar](120) null,
)

create table #TempMoleculesSummaryTable	(
	[PairCoordinates] [varchar](120) null,
	[ForRead1] [int] not null,
	[ForRead2] [int] not null,
	[TotalCount] [int]
)

set @SQL = 'create table
'+@dbTo+'.dbo.['+@exp+'_SupermutantMoleculesSummary'+@Version+'] '

set @SQL = @SQL+'([PairCoordinates] [varchar](120) null, ForRead1 [int] not
null, ForRead2 [int] not null, TotalCount [int])  '
exec(@SQL)

set @SQL = 'create table
'+@dbTo+'.dbo.['+@exp+'_SuperMutantsSummary'+@Version+'] '
set @SQL = @SQL+'([PairCoordinates] [varchar](120) null, [Chrom] [varchar](50)
null, [Start] [int] null, [end] [int] null, '
set @SQL = @SQL+'[BaseFrom] [varchar](60) null, [BaseTo] [varchar](60) null,
[Cycle] [tinyint] not null, [MatchOrientation] [char](1) null, [InHT]
[CHAR](1) null, '
set @SQL = @SQL+'[AverageQualityScore] [int] null, [MutRead1] [int] null,
[MutRead2] [int] null, [ForRead1s] [int] null, [ForRead2s] [int] null,
[TotalMutCount] [int] null ) '
exec(@SQL)

declare mycursor cursor forward_only for
select [ChromosomeSub] from [LudwigMasterGenes].[dbo].[Genome_hg19cForGenome]
order by [Chromosome], [CutOn]

open mycursor

fetch next from mycursor into @ChromSub

while @@fetch_status = 0
begin

set @SQL= '
insert into #TempTag
select *
from '+@serverFrom+'.['+@exp+'].[dbo].[Z'+@ChromSub+'Tags]
'
exec(@SQL)
update #TempTag set [ClusterID] = substring([ClusterID],3,len(ClusterID)-2)

set @SQL='
insert into #TempChanges
select *
from '+@serverFrom+'.['+@exp+'].dbo.[Z'+@ChromSub+'Changes]
'
```

```
exec(@SQL)
update #TempChanges set [ClusterID] = substring([ClusterID],3,len(ClusterID)-
2)

insert into #TempTagsChangesInDels (
	[ClusterID],
	[ReadNumber],
	[Sequence],
	[Quality],
	[Match],
	[Chrom],
	[MatchOrientation],
	[LeftPosition],
	[RightPosition],
	[MinQS],
	[MaxQS],
	[AverageQS],
	[TotalMutationsInTag],
	[AlignmentScore],
	[PairedEndReadAligmentScore],
	[PartnerChrom],
	[PartnerOrientation],
	[Filtered],
	[Exclude],
	[PairCoordinates],
	[GenePair],
	[Cycle],
	[QualityScore],
	[MutType],
	[Start],
	[end],
	[BaseFrom],
	[BaseTo],
	[Deleted]
)
select
	a.[ClusterID],
	a.[ReadNumber],
	a.[Sequence],
	a.[Quality],
	a.[Match],
	a.[Chrom],
	a.[MatchOrientation],
	a.[LeftPosition],
	a.[RightPosition],
	a.[MinQS],
	a.[MaxQS],
	a.[AverageQS],
	a.[TotalMutationsInTag],
	a.[AlignmentScore],
	a.[PairedEndReadAligmentScore],
	a.[PartnerChrom],
	a.[PartnerOrientation],
	a.[Filtered],
	a.[Exclude],
	a.[PairCoordinates],
	a.[GenePair],
	b.[Cycle],
	b.[QualityScore],
	b.[MutType],
	b.[Start],
	b.[end],
	b.[BaseFrom],
	b.[BaseTo],
	b.[Deleted]

from #TempTag a inner join #TempChanges b
on a.[ClusterID] = b.[ClusterID] AND a.[ReadNumber] = b.[ReadNumber]
where b.[MutType] = 'INS' OR b.[MutType] = 'DEL'

set @SQL = '
	use '+@dbTo+'
	update #TempTagsChangesInDels
	set [HomoPolymerString] = dbo.HomopolymerTractPosforSBS( [Sequence],
'+@lowlim_search+', '+@maxgap+', '+@lowlim_total+' )
	where [MutType] = ''INS'' or [MutType] = ''DEL''
	'
exec(@SQL)

insert into #TempChangesWithHT
select
	a.[ClusterID],
	a.[IndexString],
	a.[ReadNumber],
	a.[Cycle],
	a.[Filtered],
	a.[TotalMutationsInTag],
	a.[QualityScore],
	a.[MutType],
	a.[Chrom],
	a.[MatchOrientation],
	a.[Start],
	a.[end],
	a.[BaseFrom],
	a.[BaseTo],
	a.[Deleted],
	a.[PairCoordinates],
	b.[HomoPolymerString]
from #TempChanges a left outer join #TempTagsChangesInDels b
on a.[ClusterID] = b.[ClusterID] AND a.[ReadNumber] = b.[ReadNumber] AND
a.[Cycle] = b.[Cycle]

insert into #TempMoleculesTable
select
	FR.[ClusterID],
	FR.[ReadNumber],
	FR.[PairCoordinates]
from #TempTag FR
inner join #TempTag RR on FR.[ClusterID]=RR.[ClusterID] and FR.ReadNumber !=
RR.ReadNumber
left join #TempChanges C on C.ClusterID=FR.ClusterID and (C.Cycle <=@Trim or
C.Cycle >(@TagLen_i-@Trim))
where FR.MatchOrientation='F' and FR.PartnerOrientation ='R' and
FR.[PartnerChrom] is null
```

```
and FR.Exclude=0 and RR.Exclude=0 and
FR.TotalMutationsInTag<=@TagMutationsMax and
RR.TotalMutationsInTag<=@TagMutationsMax and C.ClusterID is null and
(RR.RightPosition-FR.LeftPosition)>(@TagLen_i*2-2)

insert into #TempMoleculesSummaryTable
select [PairCoordinates],
      sum(case ReadNumber when 1 then 1 else 0 end) as  ForRead1,
/*!*/ sum(case ReadNumber when 2 then 1 else 0 end) as  ForRead2,
      count(*) as TotalCount
from #TempMoleculesTable
group by [PairCoordinates]

set @SQL = 'insert into
'+@dbTo+'.dbo.['+@exp+'_SupermutantMoleculesSummary'+@Version+'] '
set @SQL = @SQL+'select * from #TempMoleculesSummaryTable '
exec(@SQL)

set @SQL = '
insert into '+@dbTo+'.dbo.['+@exp+'_SuperMutantsSummary'+@Version+']
select
      C2.PairCoordinates,
      C2.Chrom,
      C2.Start,
      C2.[end],
      C2.BaseFrom,
      C2.BaseTo,
      C2.Cycle,
      C2.MatchOrientation,
case
when ((C2.[MutType] = ''INS'' OR C2.[MutType] = ''DEL'') AND
C2.[MatchOrientation] = ''F'' AND substring(C2.[HomoPolymerString], C2.Cycle,
1) = ''X'') then ''Y''
when ((C2.[MutType] = ''INS'' OR C2.[MutType] = ''DEL'') AND
C2.[MatchOrientation] = ''R'' AND substring(C2.[HomoPolymerString],
('+@TagLen_s+'-C2.Cycle)+1, 1) = ''X'') then ''Y''
else ''N''
end as InHT,
      sum(C2.QualityScore)/count(*) as AverageQualityScore,
      sum(case C2.ReadNumber when 1 then 1 else 0 end) as  MutRead1,
/*!*/ sum(case C2.ReadNumber when 2 then 1 else 0 end) as  MutRead2,
      Min(MS.ForRead1) as ForRead1s,
      Min(MS.ForRead2) as  ForRead2s,
      count(*) as TotalMutCount
from #TempChangesWithHT C2 inner join #TempMoleculesTable M
on M.ClusterID=C2.ClusterID and C2.MatchOrientation=''F'' inner join
#TempMoleculesSummaryTable MS on MS.PairCoordinates=C2.PairCoordinates
group by
C2.PairCoordinates, C2.Chrom, C2.Start, C2.[end], C2.BaseFrom, C2.BaseTo,
C2.MutType, C2.Cycle, C2.MatchOrientation, C2.[HomoPolymerString]
order by C2.Chrom, C2.Start, C2.[end], C2.BaseFrom, C2.BaseTo, C2.Cycle,
C2.MatchOrientation, C2.PairCoordinates
'
exec(@SQL)

set @SQL = '
insert into '+@dbTo+'.dbo.['+@exp+'_SuperMutantsSummary'+@Version+']
```

```
select
C2.PairCoordinates, C2.Chrom, C2.Start, C2.[end], C2.BaseFrom, C2.BaseTo,
C2.Cycle, C2.MatchOrientation,
case
when ((C2.[MutType] = ''INS'' OR C2.[MutType] = ''DEL'') AND
C2.[MatchOrientation] = ''F'' AND substring(C2.[HomoPolymerString], C2.Cycle,
1) = ''X'') then ''Y''
when ((C2.[MutType] = ''INS'' OR C2.[MutType] = ''DEL'') AND
C2.[MatchOrientation] = ''R'' AND substring(C2.[HomoPolymerString],
('+@TagLen_s+'-C2.Cycle)+1, 1) = ''X'') then ''Y''
else ''N''
end as InHT,
      sum(C2.QualityScore)/count(*) as AverageQualityScore,
      sum(case C2.ReadNumber when 1 then 1 else 0 end) as  MutRead1,
/*!*/ sum(case C2.ReadNumber when 2 then 1 else 0 end) as  MutRead2,
      Min(MS.ForRead2) as ForRead1s,
      Min(MS.ForRead1) as  ForRead2s,
      count(*) as TotalMutCount
from #TempChangesWithHT C2 inner join #TempMoleculesTable M
on M.ClusterID=C2.ClusterID and C2.MatchOrientation=''R'' inner join
#TempMoleculesSummaryTable MS
on MS.PairCoordinates=C2.PairCoordinates
group by
C2.PairCoordinates, C2.Chrom, C2.Start, C2.[end], C2.BaseFrom, C2.BaseTo,
C2.MutType, C2.Cycle, C2.MatchOrientation, C2.[HomoPolymerString]
order by C2.Chrom, C2.Start, C2.[end], C2.BaseFrom, C2.BaseTo, C2.Cycle,
C2.MatchOrientation, C2.PairCoordinates
'
exec(@SQL)

delete from #TempTag
delete from #TempChanges
delete from #TempTagsChangesInDels
delete from #TempChangesWithHT
delete from #TempMoleculesTable
delete from #TempMoleculesSummaryTable

fetch next from mycursor into @ChromSub
end

close mycursor
deallocate mycursor

drop table #TempTag
drop table #TempChanges
drop table #TempMoleculesTable
drop table #TempMoleculesSummaryTable
drop table #TempTagsChangesInDels
drop table #TempChangesWithHT


----Remove NULLs from SuperMutantsSummary table (InDels)
set @SQL = '
update '+@dbTo+'.dbo.['+@exp+'_SuperMutantsSummary'+@Version+']
set
      BaseFrom = isnull(BaseFrom,''''),
      BaseTo = isnull(BaseTo,'''')
```

```
'
exec(@SQL)
```

QUERY 2

```
declare
@SQL varchar(8000),
@chromSub varchar(50),
@exp varchar(255),
@dbData varchar(255),
@dbFilter varchar(255),
@tableBins varchar(255),
@tableStructVars varchar(255),
@tableIntRepeats varchar(255),
@tableRM varchar(255),
@tableSimRepeats varchar(255),
@tableSuperDups varchar(255),
@version varchar(255),
@tagLen varchar(3)

set @exp = 'NgsNCS_Brain09NormalP30MM6F5Genome100D031615'
set @dbData = '[MH_NonClonal_NCS]'
set @dbFilter = '[MH_Filtering]'
set @tableBins = '[hg18_RepDNA_Bins_Master]'
set @tableStructVars = '[hg19__RepDNA_dgvMerged_042814]'
set @tableIntRepeats = '[hg19__RepDNA_nestedRepeats_042814]'
set @tableRM = '[hg19__RepDNA_rmsk_042814]'
set @tableSimRepeats = '[hg19_RepDNA_simpleRepeat_042814]'
set @tableSuperDups = '[hg19__RepDNA_genomicSuperDups_042814]'
set @version='022213v6'
set @tagLen = '100'
--MH revision 5/8/14: imported hg19 repetitive tables and replaced old hg18
files above

--Calculate Molecule Span for Filtering
set @SQL = 'alter table
'+@dbData+'.dbo.['+@exp+'_SupermutantMoleculesSummary'+ @version +']
      add
            chrom varchar(50),
            MoleculeLeft int,
            MoleculeRight int
'
exec(@SQL)

set @SQL = 'update '+@dbData+'.dbo.['+@exp+'_SupermutantMoleculesSummary'+
@version +']
      set
      chrom =
            substring(PairCoordinates,1,charindex( ''F'', PairCoordinates
            collate Latin1_General_CS_AS ) - 1),
      MoleculeLeft =
            substring(PairCoordinates,charindex( ''F'', PairCoordinates
            collate Latin1_General_CS_AS ) + 1, charindex( ''c'',
            PairCoordinates collate Latin1_General_CS_AS, 2) -
            charindex( ''F'', PairCoordinates collate Latin1_General_CS_AS )
            - 1),  --note: 1-based start
      MoleculeRight =
            substring(PairCoordinates,charindex( ''R'', PairCoordinates
            collate Latin1_General_CS_AS ) + 1 ,( len(PairCoordinates) -
            charindex( ''R'', PairCoordinates collate
            Latin1_General_CS_AS ) )) + '+@tagLen+' - 1 -- note: 1-based end
'
exec(@SQL)

--Assign Bins to Molecules
set @SQL = '
create table '+@dbData+'.dbo.['+@exp+'_SupermutantMoleculesSummaryFinal_'+
@version +'] (
      [PairCoordinates] [varchar](120) null, [ForRead1] [int] not null,
[ForRead2] [int] not null, [TotalCount] [int], [chrom] [varchar](50) not null,
[MoleculeLeft] [int] not null, [MoleculeRight] [int] not null, [#bin] bigint
null
)
'
exec(@SQL)

set @SQL = '
insert into '+@dbData+'.dbo.['+@exp+'_SupermutantMoleculesSummaryFinal_'+
@version +']
select
      a.[PairCoordinates],
      a.[ForRead1],
      a.[ForRead2],
      a.[TotalCount],
      a.[chrom],
      a.[MoleculeLeft],
      a.[MoleculeRight],
      b.[#bin]

from '+@dbData+'.dbo.['+@exp+'_SupermutantMoleculesSummary'+ @version +'] a
      left outer join '+@dbFilter+'.dbo.'+@tableBins+' b
on a.chrom = b.chrom
      and(a.MoleculeLeft between b.#bin_MIN + 1 and b.#bin_MAX and
      a.MoleculeRight between b.#bin_MIN + 1 and b.#bin_MAX
)
'
exec(@SQL)

--Begin Filtering
set @SQL = ' alter table
'+@dbData+'.dbo.['+@exp+'_SupermutantMoleculesSummaryFinal_'+ @version +']
add RepDNA varchar(25) ' exec(@SQL)

--Filter Molecules that DON'T have a bin
set @SQL = '
update SM
set SM.RepDNA =
case
-- Filter against Segmental Duplications (SuperDups)
when exists (
      select *
```

```
       from '+@dbFilter+'.dbo.'+@tableSuperDups+' SD
       where (
             SM.chrom = SD.chrom
             and SM.MoleculeLeft between SD.chromStart + 1 and SD.chromEnd
             ) or (
             SM.chrom = SD.otherchrom
             and SM.MoleculeLeft between SD.otherStart + 1 and SD.otherEnd
             ) or (
             SM.chrom = SD.chrom
             and SM.MoleculeRight between SD.chromStart + 1 and SD.chromEnd
             ) or (
             SM.chrom = SD.otherchrom
             and SM.MoleculeRight between SD.otherStart + 1 and SD.otherEnd
             )
             ) then ''SuperDup''

--Exclude Copy Number variants
when exists(
       select *
       from '+@dbFilter+'.dbo.'+@tableStructVars+' CN
       where (
             SM.chrom = CN.chrom
             and   (
             (SM.MoleculeLeft  between CN.chromStart + 1 and CN.chromEnd)
             or   (SM.MoleculeRight between CN.chromStart + 1 and CN.chromEnd)
             )
             and CN.varType = ''CopyNumber''
             )
             ) then ''StructVar''

--Exclude interrupted Repeats
when exists(
       select *
       from '+@dbFilter+'.dbo.'+@tableIntRepeats+' IR
       where (
             SM.chrom = IR.chrom
             and   (
             (SM.MoleculeLeft  between IR.chromStart + 1 and IR.chromEnd)
             or
             (SM.MoleculeRight between IR.chromStart + 1 and IR.chromEnd)
             )
             )
             ) then ''InterruptedRepeat''

--Exclude Simple Repeats
when exists(
       select *
       from '+@dbFilter+'.dbo.'+@tableSimRepeats+' SR
       where (
             SM.chrom = SR.chrom
             and   (
             (SM.MoleculeLeft  between SR.chromStart + 1 and SR.chromEnd)
             or
             (SM.MoleculeRight between SR.chromStart + 1 and SR.chromEnd)
             )
             )
             ) then ''SimpleRepeat''
```

```
-- Filter against Repeat Masker 3.27
when exists (
      select *
      from '+@dbFilter+'.dbo.'+@tableRM+' RM
      where
            SM.chrom = RM.genoName
            and (
            (SM.MoleculeLeft  between RM.genoStart + 1 and RM.genoEnd)
            or
            (SM.MoleculeRight between RM.genoStart + 1 and RM.genoEnd)
            )
            ) then ''RM327''

else ''Unique''
end
from '+@dbData+'.dbo.['+@exp+'_SupermutantMoleculesSummaryFinal_'+ @version
+'] SM
where SM.[#bin] is null
'
exec(@SQL)

--Filter Molecules that DO have an assigned bin
set @SQL = '
update SM
set SM.RepDNA =
case
-- Filter against Segmental Duplications (SuperDups)
      when exists (
      select *
      from '+@dbFilter+'.dbo.'+@tableSuperDups+' SD
      where SM.[#bin] = SD.[#bin] and (
            SM.chrom = SD.chrom
            and SM.MoleculeLeft between SD.chromStart + 1 and SD.chromEnd
            ) or (
            SM.chrom = SD.otherchrom
            and SM.MoleculeLeft between SD.otherStart + 1 and SD.otherEnd
            ) or (
            SM.chrom = SD.chrom
            and SM.MoleculeRight between SD.chromStart + 1 and SD.chromEnd
            ) or (
            SM.chrom = SD.otherchrom
            and SM.MoleculeRight between SD.otherStart + 1 and SD.otherEnd
            )
            ) then ''SuperDup''

--Exclude Copy Number variants
      when exists(
      select *
      from '+@dbFilter+'.dbo.'+@tableStructVars+' CN
      where SM.[#bin] = CN.[#bin] and (
            SM.chrom = CN.chrom
            and (
            (SM.MoleculeLeft  between CN.chromStart + 1 and CN.chromEnd)
            or
            (SM.MoleculeRight between CN.chromStart + 1 and CN.chromEnd)
            )
            and CN.varType = ''CopyNumber''
            )
            ) then ''StructVar''

--Exclude interrupted Repeats
      when exists(
      select *
      from '+@dbFilter+'.dbo.'+@tableIntRepeats+' IR
      where SM.[#bin] = IR.[#bin] and (
            SM.chrom = IR.chrom
            and (
            (SM.MoleculeLeft  between IR.chromStart + 1 and IR.chromEnd)
            or
            (SM.MoleculeRight between IR.chromStart + 1 and IR.chromEnd)
            )
            )
            ) then ''InterruptedRepeat''

--Exclude Simple Repeats
      when exists(
      select *
      from '+@dbFilter+'.dbo.'+@tableSimRepeats+' SR
      where SM.[#bin] = SR.[#bin] and (
            SM.chrom = SR.chrom
            and   (
            (SM.MoleculeLeft  between SR.chromStart + 1 and SR.chromEnd)
            or
            (SM.MoleculeRight between SR.chromStart + 1 and SR.chromEnd)
            )
            )
            ) then ''SimpleRepeat''

-- Filter against Repeat Masker 3.27
      when exists (
      select *
      from '+@dbFilter+'.dbo.'+@tableRM+' RM
      where SM.[#bin] = RM.[#bin] and (
            SM.chrom = RM.genoName
            and (
            (SM.MoleculeLeft  between RM.genoStart + 1 and RM.genoEnd)
            or
            (SM.MoleculeRight between RM.genoStart + 1 and RM.genoEnd)
            )
            )
            ) then ''RM327''

else ''Unique''
end
from '+@dbData+'.dbo.['+@exp+'_SupermutantMoleculesSummaryFinal_'+ @version
+'] SM
where SM.[#bin] is not null
'
exec(@SQL)

--add RepDNA filtering info to SuperMutantsSummary table
set @SQL = '
```

```
create table '+@dbData+'.dbo.['+@exp+'_SuperMutantsSummaryFinal_'+ @version
+'] (
      [PairCoordinates] varchar(250), [RepDNA] varchar(25), [chrom]
varchar(50), [Start] int, [end] int, [BaseFrom] varchar(60), [Baseto]
varchar(60), [Cycle] TINYint, [MatchOrientation] varchar(1), [InHT]
[CHAR](1),[AverageQualityScore] TINYint, [MutRead1] int, [MutRead2] int,
[ForRead1s] int, [ForRead2s] int, [TotalMutCount] int
)
'
exec(@SQL)

set @SQL = '
insert into '+@dbData+'.dbo.['+@exp+'_SuperMutantsSummaryFinal_'+ @version +']
      select
            a.[PairCoordinates], b.[RepDNA], a.[chrom], a.[Start], a.[end],
            a.[BaseFrom], a.[Baseto], a.[Cycle], a.[MatchOrientation],
            a.[InHT], a.[AverageQualityScore], a.[MutRead1], a.[MutRead2],
            a.[ForRead1s], a.[ForRead2s], a.[TotalMutCount]
from '+@dbData+'.dbo.['+@exp+'_SuperMutantsSummary'+ @version +'] a
left outer join (
--Subquery below prevents duplicating rows when two molecules share the same
pair coordinates
      select c.PairCoordinates, c.RepDNA
      from '+@dbData+'.dbo.['+@exp+'_SupermutantMoleculesSummaryFinal_'+
@version +'] c
      group by c.PairCoordinates, c.RepDNA
      ) b
      on a.[PairCoordinates] = b.[PairCoordinates]
'
exec(@SQL)
```

Final molecule output of QUERY 2

Column header	Description	Example value
Paircoordinates	Leftmost coordinate of each paired-end read of the template; used as Unique Identifier (UID)	chr21F28219909chr21R28220135
ForRead1	“Watson” family read counts	19
ForRead2	“Crick” family read counts	22
TotalCount	Total sum of counts (Watson + Crick)	41
Chrom	chromosome	chr21
MoleculeLeft	hg19 leftmost coordinate of DNA molecule	28219909
MoleculeRight	hg19 rightmost coordinate of DNA molecule	28220234
#bin	UCSC Human browser hg19 coordinate bin system	800

RepDNA	Unique or Repetitive DNA; values for RepDNA are ‘InterruptedRepeat’, ‘SimpleRepeat’, ‘SuperDup’, ‘StructVar’, ‘RM327’	Unique

Final changes output of QUERY 2

Column header	**Description**	**Example value**
PairCoordinates	Leftmost coordinate of each paired-end read of the template; used as Unique Identifier (UID)	chr1F20528671chr1R20529090
RepDNA	Annotated ‘Unique’ or Repetitive DNA	RM327
chrom	Chromosome	chr1
Start	hg19 Start coordinate, 1-start	20528702
end	hg19 end coordinate, 1-end	20528702
BaseFrom	Reference Base	C
Baseto	Non-reference Base	G
Cycle	Cycle number in read	32
MatchOrientation	Reference strand orientation	F
InHT	In Homopolymer with >= 8 ntd	N
AverageQualityScore	Average Quality Score of all family duplicates	41
MutRead1	“Watson” family read counts with mutation	1
MutRead2	“Crick” family read counts with mutation	0
ForRead1s	Total “Watson” family read counts	8
ForRead2s	Total “Crick” family read counts	4
TotalMutCount	Sum total of read counts with mutation (Watson + Crick mutation read counts)	1

5 Filter and select high confidence rare mutations and high confidence molecules

The goal of this section is to:

(1) Use **Final molecules output of QUERY 2** to filter and select for high confidence DNA molecules based on mappability. To do this, we used QUERY 3 for nuclear genome and QUERY 4 for mtDNA.

(2) Use **Final changes output of QUERY 2** to filter and select for high confidence rare somatic mutations based on quality, clonality, and mappability. To do this, we used QUERY 5 for nuclear genome and QUERY 6 for mtDNA.

QUERY 3 (for nuclear molecules)

```
create table #sm_select (
      pairCoordinates varchar(250), repDNA varchar(25), chrom varchar(50),
start bigint, [end] bigint, baseFrom varchar(60), baseTo varchar(60), cycle
tinyint, matchOrientation varchar(1), inHT char(1), averageQualityScore
tinyint, mutRead1 int, mutRead2 int, forRead1s int, forRead2s int,
totalMutCount int
)

insert #sm_select
select
      sm.pairCoordinates, sm.repDNA, sm.chrom, sm.start, sm.[end],
sm.baseFrom, sm.baseTo, sm.cycle, sm.matchOrientation, sm.inHT,
sm.averageQualityScore,sm.mutRead1, sm.mutRead2, sm.forRead1s, sm.forRead2s,
sm.totalMutCount

from[MH_Nonclonal_NCS].[dbo].[NgsNCS_SB118NN2NormalP30MM6F5Genome100D063014_1
20714_SuperMutantsSummaryFinal_022213v6] sm
where sm.averageQualityScore >= 20
      and sm.averageQualityScore != 100
      and sm.forRead1s >= 1
      and sm.forRead2s >= 1
      and sm.inHT = 'N'
      and sm.chrom != 'chrM'
      and cast(sm.mutRead1 as float)/cast(sm.forRead1s as float) > 0.5
      and cast(sm.mutRead2 as float)/cast(sm.forRead2s as float) > 0.5

--WGS snp option 1:
      and not exists (
            select *
            from [MH_Nonclonal_WGS].[dbo].[LP6005897-DNA_F02.SNPs] wgs1_snp
            where sm.chrom = wgs1_snp.#chrom
                  and sm.start = wgs1_snp.pos
      )
--WGS snp option 2:
      --and not exists (
      --select *
      --from [MH_Nonclonal_WGS].[dbo].[Brain06_variants] wgs2_snp
      --where len(wgs2_snp.ref) = 1
      --    and   (
      --    len(wgs2_snp.alt) = 1
      --    or (wgs2_snp.alt like '%,%' and len(wgs2_snp.alt) = 3)
      --    )
      --    and sm.chrom = wgs2_snp.#chrom
      --    and (sm.start = wgs2_snp.pos or sm.[end] = wgs2_snp.pos)
      --)

      and not exists (
            select *
            from [LudwigMasterGenes].[dbo].[SNP_SNP130_hg19] snp130
            where sm.[chrom] = snp130.chromosome
                  and   sm.[start] = snp130.position
                  and snp130.posReferenceAllele is not null
                  and snp130.posOtherAllele is not null
      )

create table #mutCountPairCoo (
      pairCoordinates varchar(250), mutPerPairCoo int
)
insert #mutCountPairCoo
select paircoordinates, count(*) as mutPerPairCoo
from #sm_select
group by pairCoordinates

create table #uid_select (
pairCoordinates varchar(120), forRead1 int, forRead2 int, totalCount int,
chrom varchar(50), moleculeLeft int, moleculeRight int, #bin bigint, repDNA
varchar(25)
)
insert #uid_select
select pairCoordinates, forRead1, forRead2, totalCount, chrom, moleculeLeft,
moleculeRight,#bin, repDNA
from
[MH_Nonclonal_NCS].[dbo].[NgsNCS_SB118NN2NormalP30MM6F5Genome100D063014_12071
4_SupermutantMoleculesSummaryFinal_022213v6] uids
where uids.chrom != 'chrM'
      and uids.forRead1 >= 2
      and uids.forRead2 >= 2
      and not exists (
            select *
            from
[MH_Nonclonal_NCS].[dbo].[NgsNCS_SB118NN2NormalP30MM6F5Genome100D063014_12071
4_SuperMutantsSummaryFinal_022213v6] hp
            where hp.inHT = 'Y' and
                  hp.pairCoordinates = uids.pairCoordinates
            )
      and not exists (
            select *
            from #mutCountPairCoo mc
            where mutPerPairCoo > 1
                  and mc.pairCoordinates = uids.PairCoordinates
      )

create table #uid_oldrepFilter (
      pairCoordinates varchar(120), forRead1 int, forRead2 int, totalCount
int, chrom varchar(50), moleculeLeft int, moleculeRight int, #bin bigint,
repDNA varchar(25)
)
insert #uid_oldrepFilter
select pairCoordinates, forRead1, forRead2, totalCount, chrom, moleculeLeft,
moleculeRight, #bin, repDNA
from #uid_select uids

where not exists (
      select *
```

```
from [MH_Filtering].[dbo].[hg19__RepDNA_dgvMerged_042814] dgv
where uids.chrom = dgv.chrom
and (
uids.moleculeLeft between dgv.chromStart + 1 and dgv.chromEnd
or
uids.moleculeRight between dgv.chromStart + 1 and dgv.chromEnd
)
)
and not exists (
select *
from [MH_Filtering].[dbo].[hg19__RepDNA_genomicSuperDups_042814] sd
where (
uids.chrom = sd.chrom
and   (
uids.moleculeLeft between sd.chromStart + 1 and sd.chromEnd
or
uids.moleculeRight between sd.chromStart + 1 and sd.chromEnd
)
) or (
uids.chrom = sd.otherchrom
and (
uids.moleculeLeft between sd.otherStart + 1 and sd.otherEnd
or
uids.moleculeRight between sd.otherStart + 1 and sd.otherEnd
)
)
)
and not exists (
select *
from [MH_Filtering].[dbo].[hg19__RepDNA_nestedRepeats_042814] nr
where uids.chrom = nr.chrom
and   (
      uids.moleculeLeft  between nr.chromStart + 1 and nr.chromEnd
      or
      uids.moleculeRight  between nr.chromStart + 1 and nr.chromEnd
      )
)
and not exists (
select *
from [MH_Filtering].[dbo].[hg19__RepDNA_rmsk_042814] rm
where uids.chrom = rm.genoName
and (
uids.moleculeLeft between rm.genoStart + 1 and rm.genoEnd
or
uids.moleculeRight between rm.genoStart + 1 and rm.genoEnd
)
)
and not exists (
select *
from [MH_Filtering].[dbo].[hg19_RepDNA_simpleRepeat_042814] sr
where uids.chrom = sr.chrom
and (
uids.moleculeLeft between sr.chromStart + 1 and sr.chromEnd
or
uids.moleculeRight between sr.chromStart + 1 and sr.chromEnd
)
)
```

```
create table #uid_newrepFilter (
pairCoordinates varchar(120), forRead1 int, forRead2 int, totalCount int,
chrom varchar(50),moleculeLeft int, moleculeRight int, #bin bigint, repDNA
varchar(25)
)
insert #uid_newrepFilter
select pairCoordinates, forRead1, forRead2, totalCount, chrom, moleculeLeft,
moleculeRight,#bin, repDNA
from #uid_oldRepFilter uids
where
      not exists (
      select *
      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_dgvMerged_c1toc14]
dgvM
      where uids.chrom = dgvM.chrom
      and (
      uids.moleculeLeft between dgvM.chromStart + 1 and dgvM.chromEnd
      or
      uids.moleculeRight between dgvM.chromStart + 1 and dgvM.chromEnd
      )
      )
      and not exists (
      select *
     from  [MH_Filtering].[dbo].[031015_goldenPath_hg19_dgvSupporting_c1toc14]
dgvS
      where uids.chrom = dgvS.chrom
      and (
      uids.moleculeLeft between dgvS.chromStart + 1 and dgvS.chromEnd
      or
      uids.moleculeRight between dgvS.chromStart + 1 and dgvS.chromEnd
      )
      )
      and not exists (
      select *
      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_genomicSuperDups] sd
      where (     uids.chrom = sd.chrom
      and   uids.moleculeLeft between sd.chromStart + 1 and sd.chromEnd
      ) or (
      uids.chrom = sd.otherChrom
      and uids.moleculeLeft between sd.otherStart + 1 and sd.otherEnd
      ) or (
      uids.chrom = sd.chrom
      and uids.moleculeRight between sd.chromStart + 1 and sd.chromEnd
      ) or (
      uids.chrom = sd.otherChrom
      and uids.moleculeRight between sd.otherStart + 1 and sd.otherEnd
      )
      )
      and not exists (
      select *
      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_nestedRepeats] nr
      where uids.chrom = nr.chrom
      and (
      uids.moleculeLeft between nr.chromStart + 1 and nr.chromEnd
      or
      uids.moleculeRight between nr.chromStart + 1 and nr.chromEnd
      )
      )
      and not exists (
      select *
      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_simpleRepeat] sr
      where uids.chrom = sr.chrom
      and   (
      uids.moleculeLeft between sr.chromStart - 99  and sr.chromEnd + 100 --
MH UPDATED 5/14/15
      or
      uids.moleculeRight between sr.chromStart - 99  and sr.chromEnd + 100 --
MH UPDATED 5/14/15
      )
      )
      and not exists (
      select *
      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm
      where uids.chrom = rm.genoName
      and (
      uids.moleculeLeft between rm.genoStart + 1 and rm.genoEnd
      or
      uids.moleculeRight between rm.genoStart + 1 and rm.genoEnd
      )
      )
      and not exists (
      select *
      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_sine
      where rm_sine.repClass = 'SINE'
      and uids.chrom = rm_sine.genoName
      and (
      uids.moleculeLeft between rm_sine.genoStart - 99 and rm_sine.genoEnd +
100
      or
      uids.moleculeRight between rm_sine.genoStart - 99 and rm_sine.genoEnd +
100
      )
      )
      and not exists (
      select *
      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_lc
      where rm_lc.repClass = 'low_complexity'
      and   uids.chrom = rm_lc.genoName
      and (
      uids.moleculeLeft between rm_lc.genoStart - 99 and rm_lc.genoEnd + 100
      or
      uids.moleculeRight between rm_lc.genoStart - 99 and rm_lc.genoEnd + 100
      )
      )
      and not exists (
      select *
      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_sr
      where rm_sr.repClass = 'simple_repeat'
      and   uids.chrom = rm_sr.genoName
      and (
      uids.moleculeLeft between rm_sr.genoStart - 99 and rm_sr.genoEnd + 100
      or
      uids.moleculeRight between rm_sr.genoStart - 99 and rm_sr.genoEnd + 100
      )
      )

select COUNT(*)
from #sm_select
select COUNT(*)
from #mutCountPairCoo
select COUNT(*)
from #uid_select
select COUNT(*)
from #uid_oldRepFilter
select COUNT(*)
from #uid_newRepFilter

drop table #sm_select
drop table #mutCountPairCoo
drop table #uid_select
drop table #uid_oldRepFilter
drop table #uid_newRepFilter
```

QUERY 4 (for mtDNA molecules)

```
create table #sm_select (
pairCoordinates varchar(250), repDNA varchar(25), chrom varchar(50), start
bigint, [end] bigint, baseFrom varchar(60), baseTo varchar(60), cycle tinyint,
matchOrientation varchar(1), inHT char(1), averageQualityScore tinyint,
mutRead1 int, mutRead2 int, forRead1s int, forRead2s int, totalMutCount int
)
insert #sm_select
select sm.pairCoordinates, sm.repDNA, sm.chrom, sm.start, sm.[end],
sm.baseFrom, sm.baseTo, sm.cycle, sm.matchOrientation, sm.inHT,
sm.averageQualityScore,sm.mutRead1, sm.mutRead2, sm.forRead1s, sm.forRead2s,
sm.totalMutCount
from
[MH_Nonclonal_NCS].[dbo].[NgsNCS_CRC238N1NormalP30MM6F5Genome100D042814_Super
MutantsSummaryFinal_022213v6] sm
where sm.averageQualityScore >= 20
      and sm.averageQualityScore != 100
      and   sm.forRead1s >= 1
      and sm.forRead2s >= 1
      and sm.inHT = 'N'
      and sm.chrom = 'chrM' --for mtDNA
      and cast(sm.mutRead1 as float)/cast(sm.forRead1s as float) > 0.5
      and cast(sm.mutRead2 as float)/cast(sm.forRead2s as float) > 0.5

--WGS snp option 1:
      --and not exists (
      --    select *
      --    from [MH_Nonclonal_WGS].[dbo].[LP6005897-DNA_H01.SNPs] wgs1_snp
      --    where wgs1_snp.#chrom = 'chrM' --for mtDNA
      --          and sm.start = wgs1_snp.pos
      --)

--WGS snp option 2:
      and not exists (
	select *
	from [MH_Nonclonal_WGS].[dbo].[CRC238N_variants] wgs2_snp
	where wgs2_snp.#chrom = 'chrM'  --for mtDNA
		and   len(wgs2_snp.ref) = 1
		and   (
		len(wgs2_snp.alt) = 1
		or (wgs2_snp.alt like '%,%' and len(wgs2_snp.alt) = 3)
		)
		and (sm.start = wgs2_snp.pos or sm.[end] = wgs2_snp.pos)
	)

	--and not exists (
	--	select *
	--	from [LudwigMasterGenes].[dbo].[SNP_SNP130_hg19] snp130
	--	where sm.[chrom] = snp130.chromosome
	--		and   sm.[start] = snp130.position
	--		and snp130.posReferenceAllele is not null
	--		and snp130.posOtherAllele is not null
	--)  --dbSNP does not have mitoSNP


create table #mutCountPairCoo (
	pairCoordinates varchar(250), mutPerPairCoo int
)
insert #mutCountPairCoo
select paircoordinates, count(*) as mutPerPairCoo
from #sm_select
group by pairCoordinates


create table #uid_select (
	pairCoordinates varchar(120), forRead1 int, forRead2 int, totalCount
int, chrom varchar(50), moleculeLeft int, moleculeRight int, #bin bigint,
repDNA varchar(25)
	)
insert #uid_select
select pairCoordinates, forRead1, forRead2, totalCount, chrom, moleculeLeft,
moleculeRight, #bin, repDNA
from
[MH_Nonclonal_NCS].[dbo].[NgsNCS_CRC238N1NormalP30MM6F5Genome100D042814_Super
mutantMoleculesSummaryFinal_022213v6] uids
where uids.chrom = 'chrM' --for mtDNA
	and uids.forRead1 >= 2
	and uids.forRead2 >= 2
	and not exists (
	select *
	from
[MH_Nonclonal_NCS].[dbo].[NgsNCS_CRC238N1NormalP30MM6F5Genome100D042814_Super
MutantsSummaryFinal_022213v6] hp
	where hp.inHT = 'Y' and
		hp.pairCoordinates = uids.pairCoordinates
		)
	and not exists (
		select *
		from #mutCountPairCoo mc
		where mutPerPairCoo > 1
	and mc.pairCoordinates = uids.PairCoordinates
```

```
      )

--ALL oldRepDNA tables do not include chrM
--create table #uid_oldrepFilter (
--    pairCoordinates varchar(120), forRead1 int, forRead2 int,
--    totalCount int, chrom varchar(50),
--    moleculeLeft int, moleculeRight int, #bin bigint, repDNA varchar(25)
--)
--insert #uid_oldrepFilter
--select
--    pairCoordinates, forRead1, forRead2, totalCount, chrom,
--    moleculeLeft, moleculeRight,
--    #bin, repDNA
--from #uid_select uids
--where not exists (
--    select *
--    from [MH_Filtering].[dbo].[hg19__RepDNA_dgvMerged_042814] dgv
--    where uids.chrom = dgv.chrom
--    and (
--    uids.moleculeLeft between dgv.chromStart + 1 and dgv.chromEnd
--    or
--    uids.moleculeRight between dgv.chromStart + 1 and dgv.chromEnd
--                      )
--    )
--    and not exists (
--    select *
--    from [MH_Filtering].[dbo].[hg19__RepDNA_genomicSuperDups_042814] sd
--          where (
--    uids.chrom = sd.chrom
--    and   (
--    uids.moleculeLeft between sd.chromStart + 1 and sd.chromEnd
--    or
--    uids.moleculeRight between sd.chromStart + 1 and sd.chromEnd
--    )
--    ) or (
--    uids.chrom = sd.otherchrom
--    and (
--    uids.moleculeLeft between sd.otherStart + 1 and sd.otherEnd
--    or
--    uids.moleculeRight between sd.otherStart + 1 and sd.otherEnd
--    )
--    )
--    )
--    and not exists (
--    select *
--    from [MH_Filtering].[dbo].[hg19__RepDNA_nestedRepeats_042814] nr
--    where uids.chrom = nr.chrom
--    and   (
--    uids.moleculeLeft  between nr.chromStart + 1 and nr.chromEnd
--    or
--    uids.moleculeRight  between nr.chromStart + 1 and nr.chromEnd
--    )
--    )
--    and not exists (
--    select *
--    from [MH_Filtering].[dbo].[hg19__RepDNA_rmsk_042814] rm
--    where uids.chrom = rm.genoName
--    and (
--    uids.moleculeLeft between rm.genoStart + 1 and rm.genoEnd
--    or
--    uids.moleculeRight between rm.genoStart + 1 and rm.genoEnd
--    )
--    )
--    and not exists (
--    select *
--    from [MH_Filtering].[dbo].[hg19_RepDNA_simpleRepeat_042814] sr
--    where uids.chrom = sr.chrom
--    and (
--    uids.moleculeLeft between sr.chromStart + 1 and sr.chromEnd
--    or
--    uids.moleculeRight between sr.chromStart + 1 and sr.chromEnd
--    )
--    )

--FOUR tRNA sites in RepeatMasker; all others do not have chrM
create table #uid_newrepFilter (
      pairCoordinates varchar(120), forRead1 int, forRead2 int, totalCount
int, chrom varchar(50), moleculeLeft int, moleculeRight int, #bin bigint,
repDNA varchar(25)
)
insert #uid_newrepFilter
select pairCoordinates, forRead1, forRead2, totalCount, chrom, moleculeLeft,
moleculeRight,#bin, repDNA
from #uid_select uids
where
--not exists (
--    select *
--    from
--    [MH_Filtering].[dbo].[031015_goldenPath_hg19_dgvMerged_c1toc14] dgvM
--    where uids.chrom = dgvM.chrom
--    and (
--    uids.moleculeLeft between dgvM.chromStart + 1 and dgvM.chromEnd
--    or
--    uids.moleculeRight between dgvM.chromStart + 1 and dgvM.chromEnd
--                )
--)
--and not exists (
--    select *
--    from
--[MH_Filtering].[dbo].[031015_goldenPath_hg19_dgvSupporting_c1toc14] dgvS
--    where uids.chrom = dgvS.chrom
--    and (
--    uids.moleculeLeft between dgvS.chromStart + 1 and dgvS.chromEnd
--    or
--    uids.moleculeRight between dgvS.chromStart + 1 and dgvS.chromEnd
--                )
--)
--and not exists (
--    select *
--    from [MH_Filtering].[dbo].[031015_goldenPath_hg19_genomicSuperDups] sd
--    where (     uids.chrom = sd.chrom
--    and   uids.moleculeLeft between sd.chromStart + 1 and sd.chromEnd
--    ) or (
--    uids.chrom = sd.otherChrom
--    and uids.moleculeLeft between sd.otherStart + 1 and sd.otherEnd
--    ) or (
--    uids.chrom = sd.chrom
--    and uids.moleculeRight between sd.chromStart + 1 and sd.chromEnd
--    ) or (
--    uids.chrom = sd.otherChrom
--    and uids.moleculeRight between sd.otherStart + 1 and sd.otherEnd
--                )
--)
--and not exists (
--    select *
--    from [MH_Filtering].[dbo].[031015_goldenPath_hg19_nestedRepeats] nr
--    where uids.chrom = nr.chrom
--    and (
--    uids.moleculeLeft between nr.chromStart + 1 and nr.chromEnd
--    or
--    uids.moleculeRight between nr.chromStart + 1 and nr.chromEnd
--          )
--)
--and not exists (
--    select *
--    from [MH_Filtering].[dbo].[031015_goldenPath_hg19_simpleRepeat] sr
--    where uids.chrom = sr.chrom
--    and   (
--    uids.moleculeLeft between sr.chromStart + 1 and sr.chromEnd
--    or
--    uids.moleculeRight between sr.chromStart + 1 and sr.chromEnd
--          )
--)
not exists (
      select *
      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm
      where rm.genoName = 'chrM'
      and (
      uids.moleculeLeft between rm.genoStart + 1 and rm.genoEnd
      or
      uids.moleculeRight between rm.genoStart + 1 and rm.genoEnd
      )
      )
--and not exists (
--    select *
--    from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_sine
--    where rm_sine.repClass = 'SINE'
--    and uids.chrom = rm_sine.genoName
--    and (
--    uids.moleculeLeft between rm_sine.genoStart - 99 and
--    rm_sine.genoEnd + 100
--    or
--    uids.moleculeRight between rm_sine.genoStart - 99
--    and rm_sine.genoEnd + 100
--          )
--)
--and not exists (
--    select *
--    from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_lc
--    where rm_lc.repClass = 'low_complexity'
--    and   uids.chrom = rm_lc.genoName
--      and (
--      uids.moleculeLeft between rm_lc.genoStart - 99 and rm_lc.genoEnd + 100
--      or
--      uids.moleculeRight between rm_lc.genoStart - 99 and rm_lc.genoEnd + 100
--            )
--)
--and not exists (
--      select *
--      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_sr
--      where rm_sr.repClass = 'simple_repeat'
--      and   uids.chrom = rm_sr.genoName
--      and (
--      uids.moleculeLeft between rm_sr.genoStart - 99 and rm_sr.genoEnd + 100
--      or
--      uids.moleculeRight between rm_sr.genoStart - 99 and rm_sr.genoEnd + 100
--)
--)

select COUNT(*)
from #sm_select
select COUNT(*)
from #mutCountPairCoo
select COUNT(*)
from #uid_select
--select COUNT(*)
--from #uid_oldRepFilter
select COUNT(*)
from #uid_newRepFilter

drop table #sm_select
drop table #mutCountPairCoo
drop table #uid_select
--drop table #uid_oldRepFilter
drop table #uid_newRepFilter
```

QUERY 5 (nuclear mutations)

```
create table #sm_select (pairCoordinates varchar(250), repDNA varchar(25),
chrom varchar(50), start bigint, [end] bigint, baseFrom varchar(60), baseTo
varchar(60), cycle tinyint, matchOrientation varchar(1), inHT char(1),
averageQualityScore tinyint, mutRead1 int, mutRead2 int, forRead1s int,
forRead2s int, totalMutCount int
)
insert #sm_select
select sm.pairCoordinates, sm.repDNA, sm.chrom, sm.start, sm.[end],
sm.baseFrom, sm.baseTo, sm.cycle, sm.matchOrientation, sm.inHT,
sm.averageQualityScore,sm.mutRead1, sm.mutRead2, sm.forRead1s, sm.forRead2s,
sm.totalMutCount
from
[MH_Nonclonal_NCS].[dbo].[NgsNCS_Barin07NormalP30MM6F5Genome100D031615_SuperM
utantsSummaryFinal_022213v6] sm
where sm.averageQualityScore >= 20
      and sm.averageQualityScore != 100
      and   sm.forRead1s >= 1
      and sm.forRead2s >= 1
      and sm.inHT = 'N'
      and sm.chrom != 'chrM'
      and cast(sm.mutRead1 as float)/cast(sm.forRead1s as float) > 0.5
      and cast(sm.mutRead2 as float)/cast(sm.forRead2s as float) > 0.5

--WGS snp option 1:
      --and not exists (
      --    select *
      --    from [MH_Nonclonal_WGS].[dbo].[LP6005897-DNA_H01.SNPs] wgs1_snp
      --    where sm.chrom = wgs1_snp.#chrom
      --          and sm.start = wgs1_snp.pos
      --)
--WGS snp option 2:
      and not exists (
      select *
      from [MH_Nonclonal_WGS].[dbo].[Brain07_variants] wgs2_snp
      where len(wgs2_snp.ref) = 1
      and   (
      len(wgs2_snp.alt) = 1
      or (wgs2_snp.alt like '%,%' and len(wgs2_snp.alt) = 3)
      )
      and sm.chrom = wgs2_snp.#chrom
      and (sm.start = wgs2_snp.pos or sm.[end] = wgs2_snp.pos)
      )

      and not exists (
      select *
      from [LudwigMasterGenes].[dbo].[SNP_SNP130_hg19] snp130
      where sm.[chrom] = snp130.chromosome
      and   sm.[start] = snp130.position
      and snp130.posReferenceAllele is not null
      and snp130.posOtherAllele is not null
      )


create table #mutCountPairCoo (
      pairCoordinates varchar(250), mutPerPairCoo int
)
insert #mutCountPairCoo
select paircoordinates, count(*) as mutPerPairCoo
from #sm_select
group by pairCoordinates


create table #sm_select_filtered (
      pairCoordinates varchar(250), repDNA varchar(25), chrom varchar(50),
start bigint, [end] bigint, baseFrom varchar(60), baseTo varchar(60), cycle
tinyint, matchOrientation varchar(1), inHT char(1), averageQualityScore
tinyint, mutRead1 int, mutRead2 int, forRead1s int, forRead2s int,
totalMutCount int, mutPerPairCoo int
)
insert #sm_select_filtered
select
sm.pairCoordinates, sm.repDNA, sm.chrom, sm.start, sm.[end], sm.baseFrom,
sm.baseTo, sm.cycle, sm.matchOrientation, sm.inHT,
sm.averageQualityScore,sm.mutRead1, sm.mutRead2, sm.forRead1s, sm.forRead2s,
sm.totalMutCount, mc.MutPerPairCoo
```

```
from #sm_select sm left outer join #mutCountPairCoo mc
on sm.pairCoordinates = mc.pairCoordinates
where sm.averageQualityScore >= 30
      and cast(sm.mutRead1 as float)/cast(sm.forRead1s as float) >= 0.9
      and cast(sm.mutRead2 as float)/cast(sm.forRead2s as float) >= 0.9
      and   sm.cycle != 6
      and sm.cycle != 7
      and mc.mutPerPairCoo = 1
      and not exists (
            select *
            from [MH_Filtering].[dbo].[hg19__RepDNA_dgvMerged_042814] dgv
            where sm.chrom = dgv.chrom
      and sm.start between dgv.chromStart + 1 and dgv.chromEnd
      )
      and not exists (
      select *
      from [MH_Filtering].[dbo].[hg19__RepDNA_genomicSuperDups_042814] sd
      where (
      sm.chrom = sd.chrom
      and   sm.start between sd.chromStart + 1 and sd.chromEnd
      ) or (
      sm.chrom = sd.otherchrom
      and sm.start between sd.otherStart + 1 and sd.otherEnd
      )
      )
      and not exists (
            select *
            from [MH_Filtering].[dbo].[hg19__RepDNA_nestedRepeats_042814] nr
            where sm.chrom = nr.chrom
                  and   sm.start  between nr.chromStart + 1 and nr.chromEnd
      )
      and not exists (
            select *
            from [MH_Filtering].[dbo].[hg19__RepDNA_rmsk_042814] rm
            where sm.chrom = rm.genoName
                  and sm.start between rm.genoStart + 1 and rm.genoEnd
      )
      and not exists (
            select *
            from [MH_Filtering].[dbo].[hg19_RepDNA_simpleRepeat_042814] sr
            where sm.chrom = sr.chrom
                  and sm.start between sr.chromStart + 1 and sr.chromEnd
      )


select count(*)
from #sm_select
select count(*)
from #mutCountPairCoo
select count(*)
from #sm_select_filtered


select *
from #sm_select_filtered sm
where forRead1s >= 2 and forRead2s >= 2
```

```
--WGS indel option 1:
--and not exists (
--    select *
--    from [MH_Nonclonal_WGS].[dbo].[LP6005897-DNA_H01.Indels] wgs1_indel
--    where sm.chrom = wgs1_indel.#chrom
--    and ( sm.start between wgs1_indel.pos + 1 and wgs1_indel.pos + len(ref)
--    or
--    sm.[end] between wgs1_indel.pos + 1 and wgs1_indel.pos + len(ref)
--          )
--)
--WGS indel option 2:
      and not exists (
      select *
      from [MH_Nonclonal_WGS].[dbo].[Brain07_variants] wgs2_indel
      where (     len(wgs2_indel.ref) > 1
      or
      (           len(wgs2_indel.ref) = 1
      and len(wgs2_indel.ref) != len(wgs2_indel.alt)
      and wgs2_indel.alt not like 'A,%'
      and wgs2_indel.alt not like 'T,%'
      and wgs2_indel.alt not like 'C,%'
      and wgs2_indel.alt not like 'G,%'
      )
      )
      and sm.chrom = wgs2_indel.#chrom
      and (
      sm.start between wgs2_indel.pos + 1 and wgs2_indel.pos + len(ref)
      or    sm.[end] between wgs2_indel.pos + 1 and wgs2_indel.pos + len(ref)
      )
      )

      and not exists (
      select *
      from [MH_filtering].[dbo].[031015_goldenPath_hg19_snp142_snps] snp142
      where sm.chrom = snp142.chrom
      and sm.start = snp142.posEnd
      )
      and not exists (
      select *
      from [MH_filtering].[dbo].[031015_goldenPath_hg19_snp142_other]
      other142
      where sm.chrom = other142.chrom
      and sm.Start between other142.posStart and other142.posEnd
      )
      and not exists (
      select *
      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_dgvMerged_c1toc14]
      dgvM
      where sm.chrom = dgvM.chrom
      and sm.start between dgvM.chromStart + 1 and dgvM.chromEnd
      )
      and not exists (
      select *
     from  [MH_Filtering].[dbo].[031015_goldenPath_hg19_dgvSupporting_c1toc14]
dgvS
      where sm.chrom = dgvS.chrom
      and sm.start between dgvS.chromStart + 1 and dgvS.chromEnd
```

```
        )
        and not exists (
        select *
        from [MH_Filtering].[dbo].[031015_goldenPath_hg19_genomicSuperDups] sd
        where (
        sm.chrom = sd.chrom
        and sm.start between sd.chromStart + 1 and sd.chromEnd
        ) or (
        sm.chrom = sd.otherchrom
        and sm.start between sd.otherStart + 1 and sd.otherEnd
        )
        )
        and not exists (
        select *
        from [MH_Filtering].[dbo].[031015_goldenPath_hg19_nestedRepeats] nr
        where sm.chrom = nr.chrom
        and   sm.start between nr.chromStart + 1 and nr.chromEnd
        )
        and not exists (
        select *
        from [MH_Filtering].[dbo].[031015_goldenPath_hg19_simpleRepeat] sr
        where sm.chrom = sr.chrom
        and   sm.start between sr.chromStart - 99 and sr.chromEnd + 100
        )
        and not exists (
        select *
        from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm
        where       sm.chrom = rm.genoName
        and sm.start between rm.genoStart + 1 and rm.genoEnd
        )
        and not exists (
        select *
        from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_sine
        where rm_sine.repClass = 'SINE'
        and sm.chrom = rm_sine.genoName
        and sm.start between rm_sine.genoStart - 99 and rm_sine.genoEnd + 100
        )
        and not exists (
        select *
        from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_lc
        where rm_lc.repClass = 'low_complexity'
        and   sm.chrom = rm_lc.genoName
        and sm.start between rm_lc.genoStart - 99 and rm_lc.genoEnd + 100
        )
        and not exists (
        select *
        from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_sr
        where rm_sr.repClass = 'simple_repeat'
        and   sm.chrom = rm_sr.genoName
        and sm.start between rm_sr.genoStart - 99 and rm_sr.genoEnd + 100
        )
order by chrom, start


drop table #sm_select
drop table #mutCountPairCoo
drop table #sm_select_filtered
```

QUERY 6 (mtDNA mutations)

```
create table #sm_select (
pairCoordinates varchar(250), repDNA varchar(25), chrom varchar(50), start
bigint, [end] bigint, baseFrom varchar(60), baseTo varchar(60), cycle tinyint,
matchOrientation varchar(1), inHT char(1), averageQualityScore tinyint,
mutRead1 int, mutRead2 int, forRead1s int, forRead2s int, totalMutCount int
)
insert #sm_select
select
sm.pairCoordinates, sm.repDNA, sm.chrom, sm.start, sm.[end], sm.baseFrom,
sm.baseTo, sm.cycle, sm.matchOrientation, sm.inHT,
sm.averageQualityScore,sm.mutRead1, sm.mutRead2, sm.forRead1s, sm.forRead2s,
sm.totalMutCount
from
[MH_Nonclonal_NCS].[dbo].NgsNCS_CRC238N1NormalP30MM6F5Genome100D042814_SuperM
utantsSummaryFinal_022213v6 sm
where sm.averageQualityScore >= 20
      and sm.averageQualityScore != 100
      and   sm.forRead1s >= 1
      and sm.forRead2s >= 1
      and sm.inHT = 'N'
      and sm.chrom = 'chrM' --for mtDNA
      and cast(sm.mutRead1 as float)/cast(sm.forRead1s as float) > 0.5
      and cast(sm.mutRead2 as float)/cast(sm.forRead2s as float) > 0.5

--WGS snp option 1:
      --and not exists (
      --    select *
      --    from [MH_Nonclonal_WGS].[dbo].[LP6005897-DNA_B03.SNPs] wgs1_snp
      --    where wgs1_snp.#chrom = 'chrM'
      --          and sm.start = wgs1_snp.pos
      --)

--WGS snp option 2:
      and not exists (
      select *
      from [MH_Nonclonal_WGS].[dbo].[CRC238N_variants] wgs2_snp
      where wgs2_snp.#chrom = 'chrM' --for chrM
            and len(wgs2_snp.ref) = 1
            and   (
            len(wgs2_snp.alt) = 1
            or (wgs2_snp.alt like '%,%' and len(wgs2_snp.alt) = 3)
            )
            and (sm.start = wgs2_snp.pos or sm.[end] = wgs2_snp.pos)
      )
--and not exists (
--    select *
--    from [LudwigMasterGenes].[dbo].[SNP_SNP130_hg19] snp130
--    where sm.[chrom] = snp130.chromosome
--          and   sm.[start] = snp130.position
--          and snp130.posReferenceAllele is not null
--    and snp130.posOtherAllele is not null
--)
```

```
create table #mutCountPairCoo (
      pairCoordinates varchar(250), mutPerPairCoo int
)
insert #mutCountPairCoo
select paircoordinates, count(*) as mutPerPairCoo
from #sm_select
group by pairCoordinates


create table #sm_select_filtered (
pairCoordinates varchar(250), repDNA varchar(25), chrom varchar(50), start
bigint, [end] bigint, baseFrom varchar(60), baseTo varchar(60), cycle tinyint,
matchOrientation varchar(1), inHT char(1), averageQualityScore tinyint,
mutRead1 int, mutRead2 int, forRead1s int, forRead2s int, totalMutCount int,
mutPerPairCoo int
)
insert #sm_select_filtered
select
sm.pairCoordinates, sm.repDNA, sm.chrom, sm.start, sm.[end], sm.baseFrom,
sm.baseTo, sm.cycle, sm.matchOrientation, sm.inHT,
sm.averageQualityScore,sm.mutRead1, sm.mutRead2, sm.forRead1s, sm.forRead2s,
sm.totalMutCount, mc.MutPerPairCoo
from #sm_select sm left outer join #mutCountPairCoo mc
on sm.pairCoordinates = mc.pairCoordinates
where sm.averageQualityScore >= 30
      and cast(sm.mutRead1 as float)/cast(sm.forRead1s as float) >= 0.9
      and cast(sm.mutRead2 as float)/cast(sm.forRead2s as float) >= 0.9
      and   sm.cycle != 6
      and sm.cycle != 7
      and mc.mutPerPairCoo = 1
--and not exists (
--    select *
--    from [MH_Filtering].[dbo].[hg19__RepDNA_dgvMerged_042814] dgv
--    where sm.chrom = dgv.chrom
--          and sm.start between dgv.chromStart + 1 and dgv.chromEnd
--)
--and not exists (
--    select *
--    from [MH_Filtering].[dbo].[hg19__RepDNA_genomicSuperDups_042814] sd
--    where (
--          sm.chrom = sd.chrom
--          and   sm.start between sd.chromStart + 1 and sd.chromEnd
--    ) or (
--          sm.chrom = sd.otherchrom
--          and sm.start between sd.otherStart + 1 and sd.otherEnd
--    )
--)
--and not exists (
--    select *
--    from [MH_Filtering].[dbo].[hg19__RepDNA_nestedRepeats_042814] nr
--    where sm.chrom = nr.chrom
--          and   sm.start  between nr.chromStart + 1 and nr.chromEnd
--)
--    and not exists (
--    select *
--      from [MH_Filtering].[dbo].[hg19__RepDNA_rmsk_042814] rm
--      where sm.chrom = rm.genoName
--            and sm.start between rm.genoStart + 1 and rm.genoEnd
--)
--      and not exists (
--      select *
--      from [MH_Filtering].[dbo].[hg19_RepDNA_simpleRepeat_042814] sr
--      where sm.chrom = sr.chrom
--            and sm.start between sr.chromStart + 1 and sr.chromEnd
--)


select count(*)
from #sm_select
select count(*)
from #mutCountPairCoo
select count(*)
from #sm_select_filtered


select *
from #sm_select_filtered sm
where forRead1s >= 2 and forRead2s >= 2

--WGS indel option 1:
--and not exists (
--      select *
--      from [MH_Nonclonal_WGS].[dbo].[LP6005897-DNA_B03.Indels] wgs1_indel
--      where wgs1_indel.#chrom = 'chrM' --for mtDNA
--      and ( sm.start between wgs1_indel.pos + 1 and wgs1_indel.pos + len(ref)
--      or    sm.[end] between wgs1_indel.pos + 1 and wgs1_indel.pos + len(ref)
--            )
--)

--WGS indel option 2:
      and not exists (
      select *
      from [MH_Nonclonal_WGS].[dbo].[CRC238N_variants] wgs2_indel
      where wgs2_indel.#chrom = 'chrM' --for mtDNA
      and ( len(wgs2_indel.ref) > 1
      or
      (     len(wgs2_indel.ref) = 1
            and len(wgs2_indel.ref) != len(wgs2_indel.alt)
            and wgs2_indel.alt not like 'A,%'
            and wgs2_indel.alt not like 'T,%'
            and wgs2_indel.alt not like 'C,%'
            and wgs2_indel.alt not like 'G,%'
      )
      )
      and (
      sm.start between wgs2_indel.pos + 1 and wgs2_indel.pos + len(ref)
      or    sm.[end] between wgs2_indel.pos + 1 and wgs2_indel.pos + len(ref)
      )
      )

      and not exists (
      select *
```

```
        from [MH_filtering].[dbo].[031015_goldenPath_hg19_snp142_snps] snp142
        where snp142.chrom = 'chrM' --for mtDNA
              and sm.start = snp142.posEnd
        )
        and not exists (
        select *
        from
        [MH_filtering].[dbo].[031015_goldenPath_hg19_snp142_other] other142
        where other142.chrom = 'chrM' --for mtDNA
        and sm.Start between other142.posStart and other142.posEnd
        )
--and not exists (
--      select *
--      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_dgvMerged_c1toc14]
dgvM
--      where sm.chrom = dgvM.chrom
--            and sm.start between dgvM.chromStart + 1 and dgvM.chromEnd
--)
--and not exists (
--      select *
--      from
--      [MH_Filtering].[dbo].[031015_goldenPath_hg19_dgvSupporting_c1toc14]dgvS
--      where sm.chrom = dgvS.chrom
--            and sm.start between dgvS.chromStart + 1 and dgvS.chromEnd
--)
--and not exists (
--      select *
--      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_genomicSuperDups] sd
--      where (
--                  sm.chrom = sd.chrom
--            and sm.start between sd.chromStart + 1 and sd.chromEnd
--            ) or (
--                  sm.chrom = sd.otherchrom
--            and sm.start between sd.otherStart + 1 and sd.otherEnd
--            )
--)
--and not exists (
--      select *
--      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_nestedRepeats] nr
--      where sm.chrom = nr.chrom
--            and   sm.start between nr.chromStart + 1 and nr.chromEnd
--)
--and not exists (
--      select *
--      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_simpleRepeat] sr
--      where sm.chrom = sr.chrom
--            and   sm.start between sr.chromStart - 99 and sr.chromEnd + 100
--)
        and not exists (
        select *
        from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm
        where       rm.genoName = 'chrM'
                    and sm.start between rm.genoStart + 1 and rm.genoEnd
        )
--and not exists (
--      select *
--      from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_sine
--    where rm_sine.repClass = 'SINE'
--    and sm.chrom = rm_sine.genoName
--    and sm.start between rm_sine.genoStart - 99 and rm_sine.genoEnd + 100
--)
--and not exists (
--    select *
--    from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_lc
--    where rm_lc.repClass = 'low_complexity'
--          and   sm.chrom = rm_lc.genoName
--          and sm.start between rm_lc.genoStart - 99 and rm_lc.genoEnd + 100
--)
--and not exists (
--    select *
--    from [MH_Filtering].[dbo].[031015_goldenPath_hg19_rmsk] rm_sr
--    where rm_sr.repClass = 'simple_repeat'
--          and   sm.chrom = rm_sr.genoName
--          and sm.start between rm_sr.genoStart - 99 and rm_sr.genoEnd + 100
--)
order by chrom, start


drop table #sm_select
drop table #mutCountPairCoo
drop table #sm_select_filtered
```

6 Validation of mutation calls

From QUERY 5 and QUERY 6, we get a candidate mutation list for nuclear DNA and mtDNA, respectively.

For nuclear DNA mutations candidates:

1. All candidate calls were visually inspected in a mutation viewer to verify call.
2. Subset of candidate mutations were Sanger sequenced from the original gDNA sample to verify that they were not present at clonal frequencies (e.g. germline).

For mtDNA mutation candidates:

1. All candidate calls were visually inspected in a mutation viewer to verify call.
2. Any candidate observed in the mutation viewer to be present in more than one UID family was eliminated.

7 Addendum to QUERY 1

Integrated into QUERY 1 is the following homopolymer code written in C#. This code identifies mutations that occur within homopolymers that are 8 or more nucleotides in length.

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.Text.RegularExpressions;
using System.Collections.Generic;

public static partial class UserDefinedFunctions
{

    [SqlFunction(IsDeterministic = true, IsPrecise = true)]
    public static string HomopolymerTractPosforSBS(string matchString, int
lowlim_search, int maxgap, int lowlim_total)
    {
           /* Define search patterns for homopolymers */
        string ht_pos_pat =     "([A]{" + lowlim_search + ",})|" +
                                "([C]{" + lowlim_search + ",})|" +
                                "([T]{" + lowlim_search + ",})|" +
                                "([G]{" + lowlim_search + ",})";
        Regex ht_pos = new Regex(ht_pos_pat, RegexOptions.IgnoreCase |
RegexOptions.Compiled);

           /* Search for homopolymers */
           MatchCollection ht_pos_matches = ht_pos.Matches(matchString);
        int[,] ht_pos_key = new int[ht_pos_matches.Count, 3];
        int ht_pos_key_cursor = 0;
        int nuc_num;
        foreach (Match hpMatch in ht_pos_matches)
        {
            if      (hpMatch.ToString().Substring(0, 1) == "A")
            {
                nuc_num = 1;
            }
            else if (hpMatch.ToString().Substring(0, 1) == "C")
            {
                nuc_num = 2;
            }
            else if (hpMatch.ToString().Substring(0, 1) == "T")
            {
                nuc_num = 3;
            }
            else if (hpMatch.ToString().Substring(0, 1) == "G")
            {
                nuc_num = 4;
            }
            else
            {
                nuc_num = 0;
            }

            ht_pos_key[ht_pos_key_cursor, 0] = nuc_num;
            ht_pos_key[ht_pos_key_cursor, 1] = hpMatch.Index;
            ht_pos_key[ht_pos_key_cursor, 2] = hpMatch.Length;
            if (ht_pos_key_cursor > 0 && ht_pos_key_cursor <
ht_pos_matches.Count - 1) /*not first and not last?*/
            {
                if (ht_pos_key[ht_pos_key_cursor - 1, 2] < lowlim_total &&
ht_pos_key[ht_pos_key_cursor - 1, 0] != nuc_num)
                {
                    ht_pos_key[ht_pos_key_cursor - 1, 2] = 0;
                }
                else if (ht_pos_key[ht_pos_key_cursor - 1, 0] == nuc_num)
                {
                    if (hpMatch.Index - (ht_pos_key[ht_pos_key_cursor - 1,
1] + ht_pos_key[ht_pos_key_cursor - 1, 2]) <= maxgap)
                    {/* "merge" the items */
                        ht_pos_key[ht_pos_key_cursor - 1, 2] = 0;
                        ht_pos_key[ht_pos_key_cursor, 1] =
ht_pos_key[ht_pos_key_cursor - 1, 1];
                        ht_pos_key[ht_pos_key_cursor, 2] = (hpMatch.Index
- ht_pos_key[ht_pos_key_cursor - 1, 1]) + hpMatch.Length;
                    }
                    else if (ht_pos_key[ht_pos_key_cursor - 1, 2] >=
lowlim_total)
                    {
                        ht_pos_key_cursor++;
                        continue;
                    }
                    else
                    {
                        ht_pos_key[ht_pos_key_cursor - 1, 2] = 0;
                    }
                }

            }
            else if (ht_pos_matches.Count == 1 && ht_pos_key_cursor == 0)
/*first and only?*/
            {
                if (hpMatch.Length < lowlim_total)
                {
                    ht_pos_key[ht_pos_key_cursor, 2] = 0;
                }
            }
            else if (ht_pos_key_cursor > 0 && ht_pos_key_cursor ==
ht_pos_matches.Count - 1)
            {
                if (ht_pos_key[ht_pos_key_cursor - 1, 2] < lowlim_total &&
ht_pos_key[ht_pos_key_cursor - 1, 0] != nuc_num)
                {
                    ht_pos_key[ht_pos_key_cursor - 1, 2] = 0;
                }
                else if (ht_pos_key[ht_pos_key_cursor - 1, 0] == nuc_num)
                {
                    if (hpMatch.Index - (ht_pos_key[ht_pos_key_cursor - 1,
1] + ht_pos_key[ht_pos_key_cursor - 1, 2]) <= maxgap)
                    {/* "merge" the items */
                        ht_pos_key[ht_pos_key_cursor - 1, 2] = 0;
                        ht_pos_key[ht_pos_key_cursor, 1] =
ht_pos_key[ht_pos_key_cursor - 1, 1];
                        ht_pos_key[ht_pos_key_cursor, 2] = (hpMatch.Index
- ht_pos_key[ht_pos_key_cursor - 1, 1]) + hpMatch.Length;
                        if (ht_pos_key[ht_pos_key_cursor, 2] <
lowlim_total)     /*new block unique for last item only*/
                        {
                            ht_pos_key[ht_pos_key_cursor, 2] = 0;
                        }

                    }
                    else if (ht_pos_key[ht_pos_key_cursor - 1, 2] >=
lowlim_total)
                    {
                        ht_pos_key_cursor++;
                        continue;
                    }
                    else
                    {
                        ht_pos_key[ht_pos_key_cursor - 1, 2] = 0;
                    }
                }
            }

            ht_pos_key_cursor++;
        }


        /* Build output string */
        string ht_pos_out = new string(' ', matchString.Length);
        int j;
        for (j = 0; j < ht_pos_matches.Count; j++)
        {
            if (ht_pos_key[j, 2] >= lowlim_total)
            {
                ht_pos_out = ht_pos_out.Insert(ht_pos_key[j, 1], new
string(' ', ht_pos_key[j, 2]).Replace(" ", "X"));
                ht_pos_out = ht_pos_out.Remove(matchString.Length);
            }
        }

        return ht_pos_out;
    }

}
```